FY 2011 US OMB PMM
DOE SC OASCR Software Metric SC GG 3.1/2.5.2: Improve Computational Science Capabilities

October 26, 2011

APPLICATION CREDITS

LAMMPS
Paul Crozier, Steve Plimpton

OMEN, NEMO 5
Gerhard Klimeck, Mathieu Luisier, Sebastian Steiger, Michael Povolotskyi, Hong-Hyun Park, Tillman Kubis

OSIRIS
Warren Mori, Ricardo Fonseca, Asher Davidson, Frank Tsung, Jorge Vieira, Frederico Fiuza, Panagiotis Spentzouris

eSTOMP
Steve Yabusaki, Yilin Fang, Bruce Palmer, Manojkumar Krishnan

Special Thanks and Active Participants
Entire ORNL NCCS and NERSC staff.
Rebecca Hartman-Baker, ORNL
Ricky Kendall, ORNL
Don Maxwell, ORNL
Bronson Messer, ORNL
Viraj Paropkari, LBL
David Skinner, LBL
Cary Whitney, LBL

Contacts
Christine Chalk, US DOE ASCR
Barbara Helland, US DOE ASCR
Daniel Hitchcock, US DOE ASCR
Lucy Nowell, US DOE ASCR[1]
Kenneth J. Roche, PNNL, U. Washington[2] [3]

[1] `Lucy.Nowell@science.doe.gov`
[2] High Performance Computing, Computational Sciences and Mathematics Division, Pacific Northwest National Laboratory, `kenneth.roche@pnl.gov`
[3] Nuclear Theory Group, Department of Physics, University of Washington, `k8r@uw.edu`

Contents

Metric Statement for Computational Effectiveness **7**
0.0.1 GPRA-PMM Metrics 7
0.0.2 FY11 GPRA-PMM Goals for the DOE ASCR Program 9
0.0.3 Quarterly Tasks Related to **SC GG 3.1/2.5.2** 10

Metric Results for Computational Effectiveness **11**
0.0.4 FY11 HPC Systems 11
0.0.5 Results Summary 13
0.1 **OMEN and NEMO 5** 18
0.1.1 Introduction 18
0.1.2 Background and Motivation 18
0.1.3 Capability Overview 20
0.1.4 Science Driver for Metric Problem 26
0.1.5 The Model and Algorithm 30
0.1.6 Q2 Baseline Problem Results 33
0.1.7 Developments, Enhancements, and Q4 Benchmark Results 43
0.1.8 Q4 Benchmark Results -OMEN 47
0.1.9 Q4 Benchmark Results -NEMO5 49
0.2 **LAMMPS** 59
0.2.1 Introduction 59
0.2.2 Background and Motivation 59
0.2.3 Capability Overview 60
0.2.4 Science Driver for Metric Problem 61
0.2.5 The Model and Algorithm 62
0.2.6 Q2 Baseline Problem Results 63
0.2.7 Developments, Enhancements, and Q4 Benchmark Results 66
0.2.8 Q4 Benchmark Results 70
0.3 **OSIRIS:** Plasma based acceleration 80

0.3.1 Introduction 80
0.3.2 Background and Motivation 81
0.3.3 Capability Overview 81
0.3.4 Science Driver for Metric Problem 86
0.3.5 The Model Algorithm 87
0.3.6 Q2 Baseline Problem 93
0.3.7 Developments, Enhancements, and Q4 Benchmark Results 100
0.3.8 Q4 Benchmark Results 109
0.4 **eSTOMP** 120
0.4.1 Introduction 120
0.4.2 Background and Motivation 120
0.4.3 Capability Overview 122
0.4.4 Science Driver for Metric Problem 122
0.4.5 The Model and Algorithm 123
0.4.6 Q2 Baseline Problem Results 125
0.4.7 Developments, Enhancements, and Q4 Benchmark Results 127
0.4.8 Q4 Benchmark Results 129

Bibliography **135**

Appendices **145**
0.5 A. Software Libraries 146
0.5.1 Modules Available on the Target Architectures 146
0.6 Application Inputs, Scripts, Environments 161
0.6.1 **STOMP** 161
0.6.2 **LAMMPS** 171
0.6.3 **OMEN, NEMO 5** 204
0.6.4 **OSIRIS** 248

List of Figures

1 OMEN: Device structures that OMEN can simulate. 21
2 OMEN: Flow chart. 23
3 OMEN: Quad-level parallel scheme of OMEN. 23
4 NEMO5: Strain simulation example of NEMO 5. 25
5 OMEN: Schematic of a double-gate ultra-thin-body transistor. 27
6 OMEN: InGaAs and Si UTB FET strong scaling. 34
7 OMEN: Time in single self-consistent step in Si and InGaAs devices. 35
8 NEMO5: Ballistic quantum transport through UTB. 40
9 NEMO5: Weak scaling in strain calculation. 42
10 NEMO5: Q4 strong scaling in quantum transport in UTB. . . 50
11 NEMO5: Q4 weak scaling in strain calculation. 54
12 NEMO5: Q4 weak scaling in eigenstate computation. 57
13 LAMMPS: Strong scaling of EAM metallic solid problem. . . . 64
14 LAMMPS: Weak scaling of the EAM metallic solid problem. . 64
15 LAMMPS: Q4 Strong scaling of 1/r force term for 1M atoms. 77
16 LAMMPS: Q4 Weak scaling of 1/r force term -increasing atoms and PEs. 78
17 LAMMPS: Performance versus fractional split of work to GPU. 79
18 OSIRIS:(L) Electron beam wakefield and (R) laser wakefield. . 82
19 OSIRIS: EM-PIC loop. 88
20 OSIRIS: Particle interpolation shapes. 89
21 OSIRIS: Position of grid quantities inside simulation cell. . . . 91
22 OSIRIS: Parallel domain decomposition. 92
23 OSIRIS: Laser Wakefield Accelerator Baseline Run 2. 95
24 OSIRIS: Multiple self-injection in run 3. 97
25 OSIRIS: Volume rendering of the pseudo-potential diagnostic. 98
26 OSIRIS: Particle phasespace density -particle trapping and acceleration. 98
27 OSIRIS: Particle distribution in parallel decomposition. 99
28 OSIRIS: Particles per PE as a function of time. 100

29 OSIRIS: Single code SSE speedups for interpolation levels. . . 103
30 OSIRIS: Code performance with number of threads per MPI process.106
31 OSIRIS: Detailed algorithm performance for shared memory parallelism.106
32 OSIRIS: Dynamic load balancing of parallel particle-in-cell simulations.107
33 OSIRIS: Particle density $x_2 - x_1$ slice for LWFA-01 run. . . . 108
34 OSIRIS: Q2 , Q4 Code performance for the 55k partition tests. 111
35 OSIRIS: Breakdown of performance speedup for the LWFA-01 run.111
36 OSIRIS: Code performance for the 220k partition tests. 112
37 OSIRIS: Floating point performance for the 220k partition tests.113
38 OSIRIS: Visualization from the strong scaling of the LWFA-01 run114
39 OSIRIS: 2D slice in Q4 of the electron density in first two buckets.115
40 OSIRIS: Comparison of the energy spectra of the beam in the first bucket.115
41 OSIRIS: P1 versus X1 phase space of injected beam. 117
42 OSIRIS: P2 versus P1 phase space of injected beam. 118
43 OSIRIS: P2 versus X2, P3 versus X3 phase spaces of injected beam.119
44 eSTOMP: Spider file system. 132
45 eSTOMP: Lustre-based I/O parameter study. 133

List of Tables

1 FY11 Q4 Results. 15
2 OMEN: Q2 Machine events for Si UTB benchmark. 38
3 OMEN: Q2 machine events for self-consistent step in Si UTB benchmark. 39
4 NEMO 5: Q2 Baseline Performance Results -Initialization. . . 41
5 NEMO 5: Q2 Baseline Performance Results -Solve and Shutdown. 41
6 OMEN: Q4 machine events for Si UTB benchmark. 47
7 OMEN: Q4 machine events for self-consistent step in Si UTB benchmark. 48
8 OMEN: Q4 42600 PE machine events for Si UTB benchmark. 49
9 NEMO5: Q4 machine events for strong scaling of UTB. 51
10 NEMO5: Q4 weak scaling machine events for strain computation. 52
11 NEMO5: Q4 weak scaling results of strain computation. . . . 53
12 NEMO5: Q4 eigenstate weak scaling improvements. 55
13 NEMO5: Q4 weak scaling table for eigenstates problem. . . . 56
14 LAMMPS: Q2 Performance Benchmark Results for PGI and Intel builds. 67
15 LAMMPS: Q2 Strong Scaling of Benchmark for PGI build. . . 67
16 LAMMPS: Computed Observables. 67
17 LAMMPS: Q4 raw data for weak scaling study. 78
18 LAMMPS: Raw data for Q4 fixed problem fractional GPU use study. 79
19 OSIRIS: Particle interpolation functions. 90
20 OSIRIS: Simulation parameter summary for runs on 55295 PEs. 96
21 OSIRIS: Q2 Simulation Performance. 96
22 OSIRIS: Summary of Machine Events for Q2 55295 PE baselines. 97
23 OSIRIS: Q4 single core SSE performance -interpolation levels. 103
24 OSIRIS: Single core SSE performance for multiple interpolation levels.110
25 OSIRIS: Parameter study of the electron beam in the first bucket.116
26 eSTOMP: Q2 158,400 PEs benchmark performance results. . . 127
27 eSTOMP: Q4 aggregated field file sizes -ascii vs. binary. . . . 128
28 eSTOMP: Summary of FY11 Performance Results. 129

29 eSTOMP: Lustre I/O parameter study -best parameters. . . . 134

Metric Statement for Computational Effectiveness

0.0.1 GPRA-PMM Metrics

The U.S. Department of Energy's Advanced Scientific Computing Research program's GPRA-PMM Software Metric for Computational Effectiveness is designed to comply with Public Authorizations PL 95-91, Department of Energy Organization Act, and PL 103-62, Government Performance and Results Act.

The U.S. Office of Management and Budget (OMB)[4] oversees the preparation and administration of the Presidents budget; evaluates the effectiveness of agency programs, policies, and procedures; assesses competing funding demands across agencies; and sets the funding priorities for the federal government. The OMB has the power of audit and exercises this right annually for each federal agency. According to the Government Performance and Results Act of 1993 (GPRA), federal agencies are required to develop three planning and performance documents:

1. Strategic Plan: a broad, 3 year outlook;
2. Annual Performance Plan: a focused, 1 year outlook of annual goals and objectives that is reflected in the annual budget request (What results can the agency deliver as part of its public funding?); and
3. Performance and Accountability Report: an annual report that details the previous fiscal year performance (What results did the agency produce in return for its public funding?).

OMB uses its Performance Assessment Rating Tool (PART) to perform evaluations. PART has seven worksheets for seven types of agency functions. The function of Research and Development (R&D) programs is included. R&D programs are assessed on the following criteria:

- Does the R&D program perform a clear role?
- Has the program set valid long term and annual goals?
- Is the program well managed?

[4] www.whitehouse.gov/omb

- Is the program achieving the results set forth in its GPRA documents?

In Fiscal Year (FY) 2003, the Department of Energy Office of Science (DOE SC-1) worked directly with OMB to come to a consensus on an appropriate set of performance measures consistent with PART requirements. The scientific performance expectations of these requirements reach the scope of work conducted at the DOE national laboratories. The Joule system emerged from this interaction. In FY 2008, the DOE transitioned from the Joule performance measure tracking system to OMBs Line of Business, Performance Measure Manager (PMM). The PMM is a performance-management database facilitated by the Treasury Department with the capability of uploading performance metrics directly into OMBs PARTWeb system. The PMM organizes annual performance measures into various hierarchical structures to show the relationship between individual performance targets and overall departmental performance. Departmental program and staff offices input performance measures and results directly into PMM on a periodic basis. This system is then used to produce the Performance Measure Details section of the DOE's Annual Performance Report that goes to Congress. In short, PMM enables the chief financial officer and senior DOE management to track annual performance on a quarterly basis. GPRA-PMM scores are reported as success, goal met (green light in PART), mixed results, goal partially met (yellow light in PART), and unsatisfactory, goal not met (red light in PART). GPRA-PMM links the DOE strategic plan[5] to the underlying base program targets.

[5] www.energy.gov/media/DOE_StrategicPlan_Draft.pdf

0.0.2 FY11 GPRA-PMM Goals for the DOE ASCR Program

The OASCR[6] has the following two annual performance measures as part of its PART requirements:

1. **SC GG 3.1/2.5.1** Focus usage of the primary supercomputer at the National Energy Research Scientific Computing Center (NERSC) on capability computing, defined as the percentage of the computing time used by computations that require at least 1/8 of the total resource. FY11 performance metric: *capability usage is at least 40%.*

2. **SC GG 3.1/2.5.2** Improve computational science capabilities, defined as the average annual percentage increase in the computational effectiveness (either by simulating the same problem in less time or simulating a larger problem in the same time) of a subset of application codes. FY11 performance metric: efficiency measure is $\geq$100%.

Ensuring compliance with these metrics, which are tracked on a quarterly basis, is an important milestone each fiscal year for the DOE ASCR Program Office as well as for the success of the overall DOE SC-1 open science computing effort. This document details the results of the effectiveness of the computational science capability (SC GG 3.1/2.5.2).

[6] www.sc.doe.gov/ascr/About/about.html

0.0.3 Quarterly Tasks Related to SC GG 3.1/2.5.2

The GPRA-PMM effort to improve computational science capabilities is a year long effort requiring quarterly updates. The quarterly sequence of tasks for exercising this software metric is as follows.

Quarter One (Q1) Tasks (deadline: December 31). Identify a subset of candidate applications (scientific software tools) to be investigated on DOE SC supercomputers. Management (at DOE SC and national laboratories) decides upon a short list of applications and computing platforms to be exercised. The Advanced Scientific Computing Advisory Committee (ASCAC) approves or rejects the list. The Q1 milestone is satisfied when a short list of target applications and machines (supercomputers) is approved.
Quarter Two (Q2) Tasks (deadline: March 31). Problems that each chosen application will simulate on the target machines are determined. The science capability (simulation result) and computational performance of the implementation are benchmarked and baselined (recorded) on the target machines for the defined problems and problem instances. The Q2 milestone is satisfied when benchmark data is collected and explained. If an application is striving to achieve a new science result in addition to demonstrating improved performance, then providing a detailed discussion of the existing capability, why it is insufficient and how the challenges / deficiencies are being addressed satisfy the Q2 milestone.
Quarter Three (Q3) Tasks (deadline: June 30). The application software (its models, algorithms, and implementation) is enhanced for efficiency, scalability, science capability, etc. The Q3 milestone is satisfied when the status of each application is reported at the Q3 deadline. Corrections to Q2 problem statements are normally submitted at this time.
Quarter Four (Q4) Tasks (deadline: September 30). Enhancements to the application software continue as in Q3. The enhancements are stated and demonstrated on the machines used to generate the Q2 baseline information. A comparative analysis of the Q2 and Q4 data is summarized and reported. The Q4 milestone is satisfied if the enhancements made to the application software are in accordance with the efficiency measure as defined in Q2 (run-time efficiency, scalability, or new result).

Metric Results for Computational Effectiveness

A brief description of the machines used is presented. A summary of measured results for each application as measured in Q4 and compared to the Q2 status is provided. The detailed individual sections are ordered according to the date the baselines were completed -there is no otherwise implied meaning for the section ordering. The details of the science capabilities and FY11 PMM baseline problems and benchmarking goals, and realized enhancements are stated in these detailed sections.

0.0.4 FY11 HPC Systems

Please note that a complete list of the software available to users in the form of modules available at the time the baseline problems were benchmarked is printed in the Appendix for both target platforms.

Target HPC System: *jaguarpf.ccs.ornl.gov*

The Cray XT5 high-performance computing (HPC) leadership system, Jaguar, at the Oak Ridge National Laboratory (ORNL) Leadership Computing Facility (OLCF) is again used to exercise the DOE ASCR FY11 GPRA-PMM software metric. The exercises have been greatly assisted by the staff at the center.

Jaguar has a total of 18,688 Cray XT5 compute nodes or 224,256 processing elements (PEs). These dual-socket compute nodes are six-core AMD Opteron® "Istanbul" chips operating at 2.6 GHz with 16 gigabytes (GB) of unbuffered memory per node, 6 megabytes (MB) of shared, 48-way associative L3 cache per chip, 512 kilobytes (KB) of 16-way associative L2 cache per core, and 64 KB instruction and 64 KB data two-way associative L1 caches per core. Each socket employs double data-rate two (DDR2) dual inline memory modules (DIMMs) at 800 MHz with, in the best case, 25.6 GB/s of local memory bandwidth per node.

Installed on the compute nodes is the Cray Linux Environment (CLE) version 2.2. CLE is a lightweight operating system (OS) designed to minimize the layers of OS between the application and the hardware. CLE supports many parallel programming models, including MPI 2.0 and OpenMP, the two models used by the applications in this report. Application developers

have access to a variety of compilers and libraries, including five different C, C++, and Fortran programming environments (Cray, GNU, Intel, Pathscale, and PGI); highly optimized computing libraries, such as BLAS, LAPACK, and the Cray Scientific Libraries; and internal instrumentation libraries, such as PAPI, which was used to collect the machine event data for the metrics described in this report.

Jaguar has 192 input/output (I/O) and login/service nodes. Each of these nodes consists of a 2.6 GHz dual-core AMD Opteron® chip with 8 GB of memory per node. The I/O and service nodes are running a variant of SuSE Linux. Approximately 10 petabytes (PB) of disk space are available in the scratch file systems that support massive I/O parallelism through the Lustre[7] file system software. HyperTransport links all nodes to Cray's proprietary SeaStar2+chips, which are used to construct a three- dimensional torus communication network between nodes. There are six switch ports per Cray SeaStar2+ chip, and each port has a bandwidth of 9.6 GB/s. The best-case bandwidth between the compute node and the SeaStar2+ interconnect chip is 6.4 GB/s. Thus, the injection bandwidth is half this, or 3.2 GB/s. For further information, the NCCS website[8] describes the system and its software stack and is sufficiently detailed for the purposes of this report. For information on the Cray XT5 platform, see the Cray website[9]. For more information on the AMD Istanbul chip set, see the AMD webpage AMD Six-Core Opteron .

Target HPC System: *carver.nersc.gov (dirac)*

Dirac is a 50 nodes GPU cluster[10], each node is composed of two Intel 5530 2.4 GHz, 8 MB cache, 5.86 GTps Intel QuickPath Interconnect (QuickPath, QPI) Quad-Core Nehalem® (8 cores per node) with 24 GB DDR3-1066 Reg ECC memory. 44 of the nodes each have one NVIDIA Tesla C2050 (code named Fermi[11]) having 3 GB of memory and 448 parallel CUDA cores. 4 of the nodes each have one C1060 NVIDIA Tesla GPU which has 4 GB of memory and 240 parallel CUDA cores. There is a single node with 4 NVIDIA Tesla C2050s having 3 GB of memory and 448 parallel CUDA processor cores,

[7] www.lustre.org
[8] www.nccs.gov/computing-resources/jaguar/
[9] www.cray.com/Assets/PDF/products/xt/CrayXT5Brochure.pdf
[10] newweb.nersc.gov/users/computational-systems/dirac
[11] NVIDIAFermiComputeArchitectureWhitepaper.pdf

and one with 4 C1060 NVIDIA Tesla GPUs having 4 GB of memory and 240 parallel CUDA cores each. PCIe2 (x16 2-way channels) connect the host processors to the GPUs within each node. The nodes are connected to each other by QDR Infiniband which is 10 Gbps for communication. It is noted that only 8 / 10 bits contain data and so the data transmission rate for QDR is 8 Gbps. The cluster utilizes a large (785 TB) GPFS[12] file system for temporary storage. The system boasts 15 GBps for I/O intensive production applications.

Briefly, the NVIDIA Tesla GPUs are comprised of 448 CUDA cores with a 1.15 GHz clock. This impressive device can theoretically achieve 515 GFlops of double precision floating point arithmetical performance. It can achieve double this in single precision. The CUDA cores share either 3 GB or 6 GB GDDR5 (Graphics Double Data Rate, version 5) SGRAM of high bandwidth, high performance dynamic random-access graphics card memory for the Telsa C2050 and C2070 devices respectively. The memory speed is 1.5 GHz with a 384 bit interface achieving a theoretical internal memory bandwidth of 144 GBps.

The Dirac GPU cluster is maintained by the the Computational Research Division at Berkeley Lab and is sponsored by US DOE SC ASCR's Computer Science Research Testbeds Program (DOE Contract Number DE-AC02-05CH11231). The objective of the multi-laboratory CS Research Testbeds program is to make emerging computing platforms available to facilitate the research and development of advanced systems software.

0.0.5 Results Summary

The FY11 studies aim to demonstrate strong scaling, where the problem complexity for an application is fixed and the time to execute the instance is reduced by demonstrating effective utilization of an increased hardware allocation, or weak scaling, where the goal is to compute in the same wall-clock time a more complex problem on an increased hardware allocation (i.e., maintaining fixed work per processing element). In lieu of or in addition to these modes of enhancement, the process of computing a particular algorithm may be enhanced for efficiency, where the time to execute a fixed problem is reduced on a fixed hardware allocation. Application developers often develop an efficiency metric that is relevant to their problem to report enhancements

[12] http://www-03.ibm.com/systems/software/gpfs

such as the rate of computing math operations on floating point numbers or the rate an external device such as the file system can be effectively utilized. The program binary (a compiled/loaded executable constructed from the application source code) is the instantiation of the problem on the target machine, and the computational complexity of each problem instance is deduced directly by monitoring the values of the various program counters for the various functional units (e.g., floating point operations) activated during program execution. In other words, the required resources define the complexity of the problem and the work conducted to actually execute it. This measure of work is fairly basic from the hardware perspective and can be derived from system observables such as the number of processing elements (PEs) dedicated to executing the program, execution time, total number of instructions executed, the magnitude of the memory demand, etc.

Table 1 provides a high level summary of the FY11 Q4 results. An exemplary measured enhancement is reported. Detailed descriptions of each problem are provided in the respective detailed sections in this document. There were multiple successful enhancements not reported in this table that are described in detail in the report detailed sections.

OMEN The same Q2 problem was executed in Q4. A Si UTB transistor was simulated on 85200 cores including electron-phonon scattering until convergence between the Green's Functions G and self-energies Σ was achieved. The key achievement is a reduction of the simulation time from 6400 sec down to 2950 sec corresponding to a speed up factor of 2.17, more than the required factor 2. The time reduction can mainly be attributed to a decrease of the total instruction counts which is a direct consequence of the reduction of the device unit cell and the corresponding Hamiltonian matrix and scattering self-energies.

The efficiency enhancement was verified as well with the enhanced Q4 code. The Q2 problem was executed on 42,600 processor cores instead of 85200. The times and operation counts are reported in Table 8. With the enhanced software, the exact same simulation takes 5900 sec on 42600 processor cores. This equates to a CPU Hrs savings of 151,467 CPU Hrs (Q2) - 69,817 CPU Hrs (Q4) = 81,650 CPU Hrs for the same problem, or better than 2X improvement in this scaling mode as well.

NEMO 5 The performance metrics of the Q4 run at 41472 cores are shown in table 10 and compared to the Q2 run. An efficiency result was obtained for the strain problem described in the text in that a constant matrix size describing the 3n = 994,121,664 DOFs for a 331,373,888 atom system in a

OMEN	NEMO 5	LAMMPS	OSIRIS	eSTOMP
Si UTB FET	InAs -QD strain	1M atom rapid melt	Laser Wakefield	Uranium Bioremediation
6068 atoms	N = 994,121,664 DOFs	Lennard-Jones (12,6) U	200TW (6 J) laser	$18m \times 20m \times 6.3m \sim$ long,wide,deep
dim(H)=60680	$(5nm)^3$ Qdot	1000 time steps	3.72×10^9 particles	2,268,000 grid cells, .1 m res
$N_k = 25$	331,373,888 atoms		41000 iterations	300 tsteps, 1 simulated day
$N_E(k) \in [1241, 1914]$			$1.5 \times 10^{18} cm^{-3}$ plasma	5 lithofacies
N_ω=10			$8064 \times 480 \times 480$ grid	102 biogeochemical species
			laser $a_0 = 4.0$	+ 7 minerals reaction network
				check point / 6 sim hrs
85200 PEs (jaguar)	41472 PEs (jaguar)	240 PEs + 30 GPUs (dirac)	55295 PEs (jaguar)	26400 PEs (jaguar)
$\sim 1.17 \times 10^{18}$ ins			4.58×10^{17} ins	2.619×10^{17} ins
$\sim 4.13 \times 10^{17}$ fp	$\sim 6.75 \times 10^{15}$ fp		7.16×10^{16} fp	416965910025780 fp
2950 s	2002 s	2.86258 s	13781 s	12279 s
2.17X speed-up	3.52X efficiency	strong	7.01X efficiency	15X efficiency

Table 1: FY11 Q4 Results.

$(5nm)^3$ cube of a InAs quantum dot was re-executed in Q4 nearly 3.52X faster on the same hardware allocation used in Q2. The initialization and in particular the solve time both exhibit significant reductions. The reduction in initialization time can be attributed to a faster set-up of the atomistic domain, and the reduction in solve time is a combination of better compiler options and improved preconditioning. The time required for writing the results stayed essentially the same, as no improvements were made. This time is substantially lower than the other two and can be controlled easily by increasing the number of created files.

LAMMPS A 1 million atom Lennard-Jones potential rapid melt benchmark was executed on 30 8 processor core nodes each equipped with a single Tesla 2050 GPU. In Q4, a hybrid algorithm was compared to the MPI specific algorithm on the same problem. The 240 processor core hybrid algorithm outperformed the host only version achieving a greater than 2X speedup, ie 5.80736 s (Q2) / 2.86258 s (Q4) = 2.028X speed-up. Furthermore, while both the MPI-only and hybrid algorithms exhibit outstanding weak scaling at the processor counts 8, 16, 64, 128 PEs for 500K, 1M, 4M, and 8M atoms respectively, the hybrid algorithm is more than 4X faster for each event. The Q4 algorithm was enabled to offload the computation of the interaction of pairs of neighboring atoms. The algorithm is available in the current LAMMPS release. It has the kind feature that a fractional split of work can be assigned by the informed user to be executed by the GPU, as well as the ability to allow multiple processes sharing the PCIe card to access the GPU in a single run.

OSIRIS The amount of meaningful work accomplished by OSIRIS during the FY11 exercises is impressive. The laser wakefield run described in the document demonstrated a 7.03X speedup over the Q2 counterpart. The enhancement efforts benefitted from both shared memory parallelism and an improved parallel partition, reducing the simulation load imbalance. This LWFA - 01 run can now be completed on the same hardware (55k cores) in 10.3 hours, rather than the original 2 days and 17.8 hours. The breakdown of the speedup can be seen in the figure 35. The speedup comes from a combination of 3 factors: i) a new parallel partition yields a 1.91 times improvement, ii) the SMP parallelism yields an additional speedup of 1.82 and iii) explicitly inlined SSE code yields an additional speedup of 2.02 leading to a total speedup of 7.03X. The behavior was similar for the other laser wakefield runs. In particular, one strong scaling study utilized 221,184 PEs

from the target pushing 3.39E+9 particles for 175,400 iterations demanding $\sim$ 8380 s of walltime for completion achieving a global performance of 3.21E+05 particles pushed per second and achieving nearly a 17X speedup over the same Q2 problem!

The reader is urged to read the entire enhancement section of this group -very impressive overall accomplishments.

eSTOMP Table 28 in the text summarizes the machine events measured in Q2 and Q4 for the Uranium bioremediation benchmark problem. The combination of the following optimizations resulted in over 2.5X reduction in total wall clock time and over 15X savings in the measure of CPU cost of the Q2 problem with the Q4 enhanced code: reduced per core memory requirements leads to fewer processors to execute the same problem, replacing the sequence of Global Array `put`, `sync`, and `get` commands with the `reduce` operation, packed binary instead of ASCII output files. eSTOMP was modified to use only distributed arrays for the chemical species, eliminating the temporary allocation of 102 field arrays in local memory from the Q2 algorithm. Each of these arrays is comprised of one double-precision word for each grid cell in the 2,268,000 grid cell problem domain. This modification resulted in 1.72 GB less memory usage per core. It is noted that in Q4 111,871 PETSc iterations over 755 STOMP iterations were completed versus 106,272 PETSc and 722 STOMP iterations in Q2. Thus, the gains per iteration are even greater than reported in the table, but quantifying the differences is an unnecessary detail here. However, it is noted that the added FP OPs in Q4 are partly accounted for by this fact, and the new reduction-based I/O write routine.

Additionally, a Lustre-based I/O system was designed explicitly for the eSTOMP major write phases. Using the parameters discovered through guided exhaustive search, the new algorithm is 54.5X faster than the ASCII version of the I/O write phase for 5 checkpoints, and 10.7X faster than the best binary version of the I/O write phase for 5 checkpoints!

0.1 OMEN and NEMO 5

0.1.1 Introduction

OMEN is a multi-dimensional, massively parallel quantum transport solver dedicated to the simulation of next generation nanoelectronic devices. It provides an atomistic resolution of the simulation domain and a full-band description of the material properties. Its implementation started in 2005 at ETH Zürich in Switzerland under the name GreenSolver and has continued since 2008 at Purdue University. Although OMEN is already a mature code, new physical models are periodically added to it which will benefit from the FY11 OMB PMM OASCR Software Effectiveness Metric program to become computationally more efficient.

NEMO 5 is a new massively parallel simulation tool whose development started at the end of 2009 at Purdue University. It can be considered as a new incarnation of OMEN, but with more functionalities and simulation capabilities. NEMO 5 is still in an early development phase and is based on an implementation philosophy different from OMEN. Currently it is not as efficient as OMEN (factor 2 to 6 slower), but participating to the FY11 OMB PMM OASCR Software Effectiveness Metric program will help it make considerable progress.

0.1.2 Background and Motivation

During the last four decades the functionality of the electronic systems such as audio players, televisions, personal computers, and cellular phones has kept improving due to breakthrough innovations, but also due to a continuous increase of the transistor density per integrated circuit (IC). The number of transistors per IC has doubled almost every 2 years for more than 40 years to follow Moore's scaling law [1]. This exponential increase has been responsible for an aggressive scaling of the transistor dimensions, which nowadays do not exceed a few nanometers and are composed of a countable number of atoms. Quantum mechanical effects such as energy quantization and electron tunneling are already playing a very important role in the currently manufactured transistors and will completely dominate the behavior of the next generation nanoelectronic devices. The greatest challenges that the semiconductor industry will face in the next couple of years is to demonstrate and fabricate devices that will (i) work "because" and not "in

spite" of quantum mechanical effects, (ii) keep Moore's law going as long as possible, (iii) outperform the conventional silicon metal-oxide-semiconductor field-effect transistors (MOSFETs), and (iv) reduce the IC power consumption.

A well-established way of addressing these issues and accelerating the development of any new technology consists in accompanying the experimental effort by computer aided design (CAD) simulations. A physics-based CAD tool is a powerful methodology to investigate the performances of novel electronic devices prior to their fabrication, to reveal unexpected device physics, and to explore new device concepts. Semi-classical approaches like the drift-diffusion model [2, 3, 4] and Monte Carlo [5, 6, 7, 8] device simulations have proved their utility in the past, but as the active region of the transistors is getting closer to the atomic scale, new modeling concepts are required to design ultra-scaled devices.

OMEN has been developed to address the accuracy problems of the DD and Monte Carlo methods at the nanometer scale [9]. It is the first multi-dimensional, atomistic, full-band quantum transport simulator capable of treating realistically extended device structures. OMEN was first designed to investigate 3-D nanowire transistors in the ballistic limit of transport [10], but was then enhanced to deal with 2-D structures such as ultra-thin-body (UTB) field-effect transistors [11]. The main purpose of OMEN is to accompany the development of future nanoelectronic devices, for which no reliable simulation approach exists.

Since 2008, the simulation capabilities of OMEN have been further extended to treat new materials such as graphene and wurtzite, to study novel devices such as band-to-band tunneling (BTBT) transistors [12], and to go beyond coherent transport and include electron-phonon scattering [13]. A simplified version of OMEN is available through the community web platform `nanoHUB.org` where users from all around the world can use it with a simple GUI. The simulations run locally on Purdue machines and the results are transferred to the users Internet browser. OMEN powers three applications with more than 4,000 annual users on `nanoHUB.org`.

NEMO 5 is a separate code that aims at extending the functionality of OMEN. While OMEN focuses on transport through nanostructures comprising some tens of thousands of atoms, NEMO 5 also aims at the simulation of structures containing tens of millions of atoms. Although it is not possible to investigate transport in structures of this size, other physical aspects, such as the computation of strain and quantum eigenstates, provide a fundamental

understanding of these devices which is relevant in many fields ranging from quantum computing [14] to optoelectronics [15]. Although its development started as recently as at the end of 2009, NEMO 5 already has the potential of becoming a community code where both industry and academia can benefit from its capabilities. This is reflected in its integration into `nanoHUB.org`, where NEMO 5 powers two online simulation tools and the migration of three others is under way. The lesser GPL license enables industrial and academic partners to contribute to the development.

The complexity of the physical models, the size of the considered electronic devices, and the desire to investigate new phenomena at the nanometer scale make OMEN and NEMO 5 the perfect candidates to run on the largest available supercomputers. Indeed, both codes include a quad-level parallelization scheme that allows them to scale up to several thousands of cores.

0.1.3 Capability Overview

In this section, the physical and numerical models, as well as the software implementation of OMEN and NEMO 5 are summarized.

OMEN physical model. OMEN solves multi-dimensional Schrödinger equations with open boundary conditions (OBCs) in the Non-Equilibrium Green's Function (NEGF) or Wave Function (WF) formalisms for electrons, holes, and phonons. It is based on different declinations of the nearest-neighbor tight-binding model [16] (single s-orbital, sp^3, sp^3s^*, or $sp^3d^5s^*$, with or without spin-orbit coupling) for electrons and holes and on an extended Keating model for phonons [17]. The carrier and current densities of the simulated nanotransistors are obtained by self-consistently coupling the quantum transport calculation to the solution of Poisson equation. OMEN can treat the ballistic limit of transport (no interaction of electrons/holes/phonons with their environment), but it can also include scattering mechanisms such as interface roughness or electron-phonon interactions [13].

OMEN has already demonstrated its usefulness in various applications such as studying the transport properties and the gate length scaling behavior of ideal and rough Si triple-gate nanowires with different crystal orientations [18, 19, 20], comparing the output characteristics of n- and p-doped

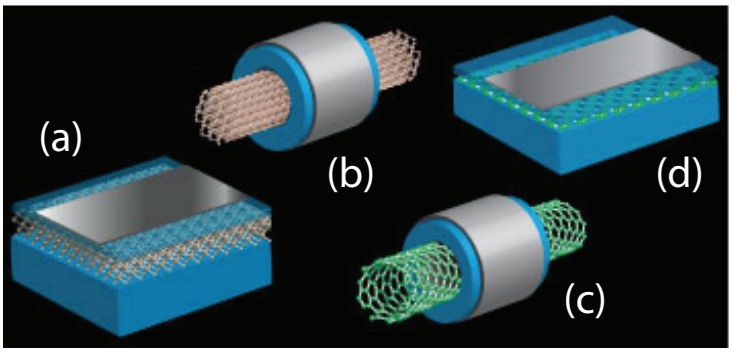

Figure 1: Example of device structures that OMEN can simulate: (a) Single-Gate and Double-Gate Ultra-Thin-Body field-effect transistor (FET) made of Si, Ge, or III-V semiconductors; (b) Gate-All-Around Nanowire FET; (c) Graphene Nanoribbon FET; and (d) Coaxially-Gated Carbon Nanotube FET.

Si double-gate ultra-thin-body FETs with different transport and confinement directions [21], reproducing the experimental data of existing InAs high-electron mobility transistors [22,23], investigating line-edge roughness in graphene nanoribbons [24,25], analyzing and optimizing the performances of different types of band-to-band tunneling transistors (TFETs) [26,27,28,29], or understanding the effect of electron-phonon scattering in nanowires [13,30]. Some of the nanoelectronic devices that OMEN can handle are sketched in Fig. 1.

OMEN numerical models. The most time-consuming task of OMEN is to solve the Schrödinger equation, either in the WF or NEGF formalism. This operation must be repeated up to 100,000 times since one Schrödinger equation must be solved for each electron/hole/phonon energy E and momentum k and for each externally applied bias conditions. In the WF formalism, which is applicable only to ballistic transport (75% of the simulations), the Schrödinger equation takes the form of a sparse linear systems of equations (LSE) $\mathbf{Ax} = \mathbf{b}$ with multiple right-hand sides $\mathbf{b}$. The best way to solve these LSE is to use direct sparse linear solvers such as *Umfpack* [31], *Pardiso* [32], *SuperLU_dist* [33], or *MUMPS* [34]. A parallel solver based on the block cyclic reduction of the matrix $\mathbf{A}$ has also been implemented in OMEN. For tight-binding problems it is usually faster than the other methods [35, 36].

Solving the Schrödinger equation with OBC using the NEGF formalism has been popularized in the nanodevice modeling community by Supriyo Datta [39]. This approach requires the inversion and multiplication of full matrices which is accomplished using the highly optimized *Blas* [37] and *Lapack* [38] numerical libraries. The NEGF formalism can also be applied to ballistic transport problems, where it is about 10 times slower than the WF formalism in 3-D structures, 3-4 times slower in 2-D, and slightly faster in 1-D devices. It is however the most powerful way of including inelastic scattering processes such as electron-phonon scattering in quantum transport simulations.

After the Schrödinger equation has been solved for each electron/hole energy and momentum, the different contributions are summed up (integrated) to give the total charge density at each position inside the simulation domain. The charge density is an input parameter to Poisson equation, which is solved in the finite element method (FEM) [40, 41]. The FEM mesh is generated with the *Qhull* library [42], while the sparse linear system of equations resulting from the application of the Newton-Raphson scheme is solved using the parallel iterative solver library *Aztec* [43].

OMEN software implementation. The quantum-mechanical simulator OMEN is written in C++ and is specifically designed for distributed memory computers using MPI as the communication protocol. The solution of the Schrödinger equation is completely transparent, i. e. it just takes a matrix $\mathbf{A}$ as an input and will return the charge and current densities as an output, no matter whether the NEGF or WF formalism are used or whether electron-phonon scattering is considered or not.

The task flow of OMEN is depicted in Fig. 2 where four loops can be identified, over the externally applied bias points (dashed blue), over the self-consistent calculation of the electrostatic potential with Poisson equation (solid orange), over the electron/hole momentum (dashed-dotted green), and over the electron/hole energy (red with circles). The core of the simulator is the computation of the Schrödinger equation with open-boundary conditions. In a typical nanoelectronic device simulation, the current flowing from one side to the other is computed for 10 to 20 different applied voltages V_{gs}. For each V_{gs} 5-10 Poisson iterations are required before the charge density $n(x, y, z)$ and the electrostatic potential $V(x, y, z)$ converge to their final value. The momentum and energy dependence of the electrons and holes

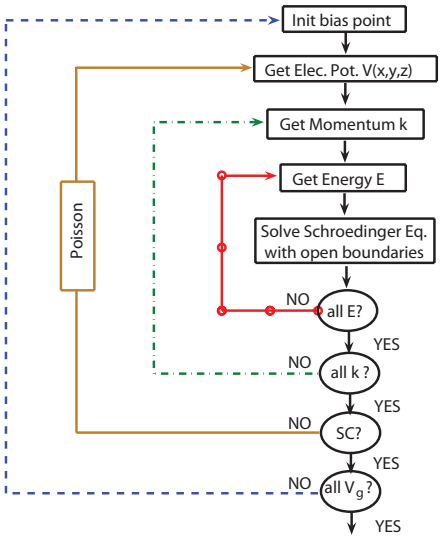

Figure 2: Flow chart of OMEN. For a given bias point (dashed blue loop) the Schrödinger equation has to be solved for each momentum-energy (k,E) configuration (solid red loop with circles and dashed-dotted green loop). The charge density $n(x, y, z)$ and the electrostatic potential $V(x, y, z)$ are updated by solving Poisson equation until self-consistency (SC, solid orange loop) is reached.

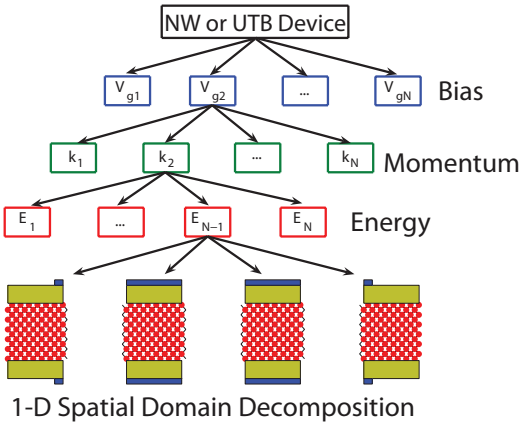

Figure 3: Quad-level parallel scheme of OMEN. In Fig. 2, the dashed blue (bias points), the dashed-dotted green (momentum), and the red with circle (energy) loops are parallelized and form the three highest levels of parallelism. A 1-D spatial domain decomposition forms the fourth level of parallelism.

are discretized using 15 to 30 and 500 to 2000 points, respectively. Hence, typically, the Schrödinger equation must be solved more 100,000 times per device simulation.

To reduce the simulation time, four natural levels of parallelism are implemented in OMEN, the distribution of (i) the bias points, (ii) the momentum points, (iii) the energy points, and (iv) a 1-D spatial domain decomposition of the Schrödinger equation as illustrated in Fig. 22. The highest level, the loop over the bias points, is embarrassingly parallel since there is not dependence between the different biases. The different momentum and energy points (second and third parallelization level) are independent for ballistic transport simulations, but become tightly coupled when electron-phonon scattering is turned-on. In effect, the absorption or emission of a phonon by an electron couples different momentum and energy points. The fourth parallelization level, the 1-D domain decomposition, consists in solving the Schrödinger equation in parallel, which is automatically achieved by *Pardiso*, *SuperLU_dist*, *MUMPS*, or our block cyclic reduction solver.

OMEN computational performance. OMEN has already been tested up to 59,904 cores on the SUN constellation star Ranger at TACC, up to 110,700 cores on the CRAY-XT5 Kraken at NICS, and up to 222,720 cores on the CRAY-XT5 Jaguar at NCCS. On Kraken and Jaguar, the ballistic simulation of an InAs high electron mobility transistor (HEMT) fabricated at MIT [44] has reached a sustained performance of 637 TFlop/s and 1.26 PFlop/s, respectively, corresponding to about 55% of the peak performance of these machines. A good match between the simulation and the experimental data could also be obtained. The electron-phonon scattering model, which will be optimized in this project, has also been tested on Jaguar and Kraken, up to 95,256 cores for ultra-thin-body transistors [45], reaching a sustained performance of only 142 TFlop/s, around 14% of the peak performance.

NEMO 5 physical models. NEMO 5 implements the same physical models for strain, phonons and quantum transport as OMEN, with the exception of electron-phonon scattering where NEMO 5 has so far adapted a simpler model [46]. Possible simulation examples and the task flow for this simulation type can therefore also be seen in Figs. 1 and 2. In addition, NEMO 5 can calculate strain relaxation (i. e. the relaxed atomic positions and the mesoscopic strain and polarization fields) in large structures that contain

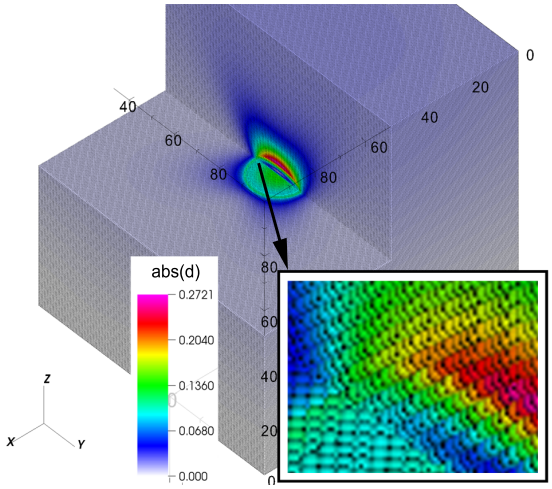

Figure 4: Strain simulation example of NEMO 5. Depicted is a 44-million-atom cube of $(100\text{nm})^3$ GaAs surrounding with an embedded dome-shaped InAs quantum dot. Shown is the magnitude of the displacement vector $\mathbf{d}$ which signifies the dislocation from the initial atomic positions of pseudomorphic GaAs. $\mathbf{d}$ is the solution of a nonlinear equation system of the size $3N \sim 1.3e8$. Individual atoms are visible in the magnified inset.

tens of millions of atoms. The current record is held by a simulation of an InAs dome-shaped quantum dot embedded in a $(100\text{nm})^3$ GaAs matrix consisting of about 44 million atoms (Fig. 4). It is possible to calculate interior eigenstates of the corresponding electronic tight-binding Hamiltonian using an iterative Lanczos algorithm. NEMO 5 implements a multiscale approach where a mix-and-match of physical models with varying complexity can be done in the same simulation. As an example, a combination of semiclassical, quantum equilibrium and quantum nonequilibrium charge density can be employed to obtain computationally inexpensive results that have been used successfully in the past for the calculation of resonant tunneling diodes characteristics [47].

NEMO 5 numerical models. NEMO 5 is designed as a general-purpose nanoelectronics simulator. Its modularity allows for rapid additions of new functionality by adding individual modules which can access simulator-wide numerics capabilities. The solution of linear, nonlinear and eigenproblems is outsourced to *PETSc* [48] and *SLEPc* [49]. Finite element meshes and

quadrature rules are set up using *libmesh* [50]. Such a holistic approach has the benefit of rapid application development but also lags behind the more specialized numerical capabilities of OMEN when it comes to transport simulations.

NEMO 5 software implementation. NEMO 5 is a pure C++ code that organizes physical problems into C++ modules that interact with each other. For example, in a Schrödinger-Poisson simulation the Poisson solver is indifferent to what other physical model was used to compute the density. Consequently the same Poisson module is used for closed-boundary eigenvalue problems, open-boundary transport problems, and semiclassical simulations. All physical modules are able to employ the same suite of input/output modules and numerical solvers, which standardizes the functionality in this respect.

Parallelism is solely based on MPI. NEMO 5 implements a home-grown parallelization class that encapsulates the task of generating multiple levels of parallelism, creating appropriate communicators, and distributing the problems to the MPI processes at hand so that the parallelization aspect of the simulator is decoupled from the physics. The class is used in different types of simulations, providing parallelization hierarchies ranging from one to four levels. A more detailed description of the strategies employed in order to achieve load-balanced parallelization is going to be presented in [51].

For a quantum transport simulation the parallelization hierarchy is the same as in Fig. 22, with the exception of inverted k- and E-levels. For strain calculations the sole parallelization level consists of a spatial decomposition of the structure. This decomposition can be three-dimensional. For quantum eigenstate computations the spatial parallelization is augmented with k-parallelization where applicable.

0.1.4 Science Driver for Metric Problem

Double-Gate UTB Transistors

Almost every two years the transistor area is divided by two, leading to an exponential increase of their number per integrated circuit (IC). As a consequence, the functionality and the performances of the electronic devices have kept improving since the first radio transistor in the 1950's.

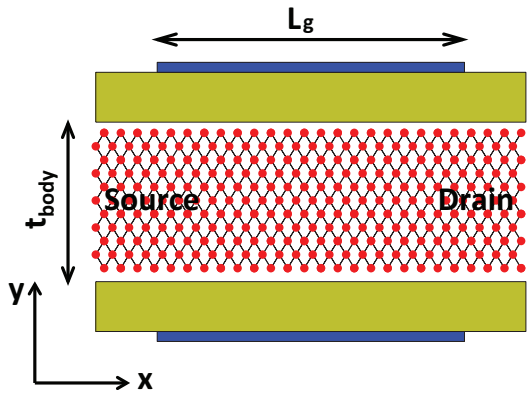

Figure 5: Schematic view of a double-gate ultra-thin-body transistor. Transport occurs along the x-axis while the electrons are confined along the y-direction. The transistor is characterized by its gate contact length L_g and its body thickness t_{body}. The total number of atoms (red dots) composing the simulation domain is usually smaller than N_A=100,000 for this type of structures.

The dimensions as well as the electrical properties of the transistors are determined by the International Technology Roadmap for Semiconductors (ITRS) [52]. For many years now, the structure of the transistors has remained the same, a planar, single-gate (SG), silicon (Si) metal-oxide-semiconductor field-effect transistor (MOSFET), but their size has been continuously reduced from one technology node to the other. Currently the 32nm technology node is in production and Intel, the largest transistor manufacturer worldwide, is massively producing devices with a gate length that does not exceed 30nm and a a gate pitch length of 112.5nm [53].

The solutions for the next technology node, 22nm, are under development and will still be based on the planar SG Si MOSFET. So will probably be the 15nm technology node in about 4 years. However, at one point in the future, it will be extremely difficult if not impossible to further scale the planar SG Si MOSFET and maintain good operation conditions due to material, structural, and functional problems.

A potential candidate to replace the planar SG Si MOSFET in the future is the double-gate (DG) ultra-thin-body (UTB) field-effect transistor (FET) as shown in Fig. 5. It can be made of strained Si or of any more exotic material with better transport properties than Si. Modeling this class of next

generation devices, predicting their transport properties, and comparing the performances of different designs before their fabrication can be very helpful to the semiconductor industry and substantially decrease the development costs. Very often, the simulation capabilities are ahead of experiments and can guide them towards the right direction.

The electron flow through DG UTB FETs is completely dominated by quantum mechanical effects and disorder on an atomistic scale. To address these issues, state-of-the-art computer aided design tools like OMEN and NEMO 5 are required.

OMEN : The scientific motivation behind this project is the simulation of DG UTB FETs for the 12nm technology node, i. .e 3 nodes ahead of time. These future transistors will be characterized by a gate length L_g=12nm, a semiconductor channel with a body thickness of around t_{body}=5nm, and high-κ oxide layers of thickness t_{ox}=3nm and relative dielectric constant ϵ_R=20. Two different devices will be considered here, one made of $In_{0.53}Ga_{0.47}As$ with transport along the <100> crystal axis, and one made of strained Si with transport along the <110> crystal axis.

Even at very short gate lengths, no device truly operates at its ballistic limit. $In_{0.53}Ga_{0.47}As$-based transistors come close, but Si-based transistors exhibit a ballistic ratio of sometimes less than 60%. It is therefore necessary to include the interactions of the electrons and holes with their environment into the modeling effort, especially electron-phonon scattering which is the dominant limiting factor at the nanometer scale. Electron-phonon scattering is a dissipative mechanism coupling different momentum and energies together and requiring a lot of inter-processor communication [13, 45]. However, this effect is essential and cannot be neglected in order to give accurate results. A multi-level parallel version of the electron-phonon scattering model has been implemented in OMEN. It will be shown in Section 0.1.6 that for a given structure, the simulation time scales relatively well up to 170k cores for the transistors considered in this project. The goal is to decrease the simulation time by a factor of 2 at the end of Q4 so that in the future, UTB devices can be investigated more efficiently in the presence of electron-phonon scattering.

NEMO 5 : Solving a quantum transport problem consists of solving a set of equations for many different energies, wavevectors, and voltages. In the

ballistic limit these problems are decoupled and therefore well parallelizable, as has been demonstrated by near-perfect scaling of OMEN up to 222,720 cores on Jaguar. Being able to simulate devices at their ballistic limit is an important step in order to understand nanoelectronic devices and a prerequisite to more sophisticated simulations with scattering, which usually is treated as a correction to the ballistic result. NEMO 5, being a much younger code than OMEN, does not exhibit the expected near-ideal scaling when it comes to massive parallelization of ballistic problems. Optimizing this strong scaling behavior is essential in order to investigate state-of-the-art devices while harnessing large computational resources in an efficient way.

The considered structure is a UTB designed for the 22nm technology node, with gate length L_g=22nm, oxide thickness t_{ox}=1.5nm both on the upper and lower side, and an extension of the source and drain regions of 10nm each, bringing the total length to 42nm.

InAs-GaAs Quantum Dot Strain and Eigenstates

Quantum dots, sometimes termed artificial atoms [54], are nanostructures where a 3D confinement of carriers is achieved. The energy spectrum of these carriers is discrete, like in atoms, as opposed to 0D, 1D or 2D confined structures where the energy spectrum is continuous. The arguably most popular type of quantum dot is indium arsenide (InAs) embedded in gallium arsenide (GaAs). Manufacturing of these structures can be achieved by self-assembly in the Stranski-Krastanow growth mode. Due to the large lattice mismatch of 7% between InAs and GaAs, it is energetically preferable for InAs to form little chunks rather than a flat plane when it epitaxially grown on GaAs beyond a critical thickness of about 1.6 monolayers [58]. The resulting dot has a nontrivial three-dimensional strain profile which substantially influences the carrier energies.

InAs/GaAs quantum dots have many applications mainly in the field of optoelectronics. At room temperature quantum dots can serve as an active material for high-efficiency, low-threshold lasers [57]. When being operated at low temperatures, potential applications include quantum computation, quantum cryptography and single-photon sources [56].

The central quantity in the theoretical modeling of all these devices is an accurate description of the electronic states in the strained quantum dot. The dots themselves typically contain a few hundred thousand to a few million atoms, giving rise to a countable number of continued electron and hole

states. It is however also important to model the vicinity of the dot up to a certain distance [55], which brings the simulation domain size up to tens of millions of atoms. Moreover, it was shown that an atomistic full-band description is detrimental in order to obtain the correct quantum state symmetries and the effect of disorder.

The considered structure is a $(5\text{nm})^3$ cube of InAs embedded in a cube of GaAs of varying size. The harmonic Keating variant of the valence force field model [17] is employed for the strain relaxation. The 20-band $\text{sp}^3\text{d}^5\text{s}^*$ tight-binding model is chosen to model the electronic structure. In contrast to purely electronic devices, the valence band also needs to be modeled accurately, which necessitates the inclusion of spin-orbit coupling and therefore doubling the size of the Hamiltonian.

0.1.5 The Model and Algorithm

Quantum Transport (OMEN and NEMO 5)

For quantum transport problems, the core of OMEN and NEMO 5 is to solve the Schrödinger equation with open boundary conditions (OBCs) for each electron/hole energy E and momentum k. The OBCs couple the simulation domain to its surrounding environment and allows for electrons/holes to enter and exit a finite domain. In the device modeling community, the most popular way of treating the Schrödinger equation with OBCs is based on the Non-equilibrium Green's Function formalism, which consists in solving the following equations

$$(1) \quad \big(\mathbf{E} - \mathbf{H}(k) - \mathbf{V} - \mathbf{\Sigma}^{RB}(E,k) - \mathbf{\Sigma}^{RS}(E,k)\big) \cdot \mathbf{G}^R(E,k) \;=\; \mathbf{I}$$

$$(2) \quad \mathbf{G}^<(E,k) \;=\; \mathbf{G}^R(E,k) \cdot \big(\mathbf{\Sigma}^{<B}(E,k) + \mathbf{\Sigma}^{<S}(E,k)\big) \cdot \mathbf{G}^A(E,k).$$

In Eq. 1 and 2, the unknowns are the retarded $\mathbf{G}^R(E,k)$ and lesser $\mathbf{G}^<(E,k)$ Green's functions at the energy E and momentum k. They are matrices of size $N_A \times N_{orb}$ where N_A is the number of atoms composing the simulation domain and N_{orb} the number of orbitals describing the properties of each atom. For example, N_{orb}=10 in the sp^3d^5s* tight-binding model without spin orbit coupling, 1 s orbital, 3 p orbitals, 1 excited s orbital called $s*$, and 5 d orbitals. The single elements of the retarded and lesser Green's Functions, $G_{ij}^{R\sigma_1\sigma_2}(E,k)$ and $G_{ij}^{<\sigma_1\sigma_2}(E,k)$, respectively, represent the coupling between two orbitals σ_1 and σ_2 situated on two atoms i and

j. The diagonal matrix $\mathbf{E}$ contains the electron/hole energy E, $\mathbf{H}(k)$ is the device Hamiltonian which includes the on-site energy and coupling elements between the different orbitals and atoms $H_{ij}^{\sigma_1\sigma_2}$, while the entries of the diagonal matrix $\mathbf{V}$ are the electrostatic potential at each atomic site. The open boundary conditions are cast into the matrices $\mathbf{\Sigma}^{RB}(E,k)$ and $\mathbf{\Sigma}^{<B}(E,k)$. Finally, the scattering mechanisms such as electron-phonon are described by the scattering self-energies $\mathbf{\Sigma}^{RS}(E,k)$ and $\mathbf{\Sigma}^{<S}(E,k)$. Note that $G_{ij}^{A\sigma_1\sigma_2}(E,k)$=conj$(G_{ji}^{A\sigma_2\sigma_1}(E,k))$ and that a greater Green's Function $\mathbf{G}^{>}(E,k)$ exists whose definition is identical to Eq. (2), except that $<$ is replaced by $>$.

Equations 1 and 2 must be solved for each possible E and k. This can be done using a recursive algorithm instead of directly inverting or multiplying large matrices [47]. If the scattering self-energies $\mathbf{\Sigma}^{RS}(E,k)$ and $\mathbf{\Sigma}^{<S}(E,k)$ are turned off, as in ballistic simulations, each Schrödinger equation can be solved independently from the others since nothing couples the different energies and momentum. However, when scattering is present, everything becomes more complicated. For example, $\mathbf{\Sigma}^{\gtrless S}(E,k)$ and $\mathbf{\Sigma}_{nn}^{RS}(E,k)$ for electron-phonon scattering are defined as [45]

$$\begin{aligned}
\mathbf{\Sigma}_{nn}^{\gtrless S}(E,k) &= \sum_{l,i,j}\int_{-\pi}^{\pi}\frac{\mathrm{d}q}{2\pi}\sum_{\omega_{ph}}\mathcal{V}_{nlln}^{ij}(\omega_{ph},q)\cdot\nabla_i\mathbf{H}_{nl}\cdot\Big(n^{ph}(\omega_{ph})\cdot\mathbf{G}_{ll}^{\gtrless}(E\pm\hbar\omega_{ph},k-q)+ \\
&\quad (3)\quad (n^{ph}(\omega_{ph})+1)\cdot\mathbf{G}_{ll}^{\gtrless}(E\mp\hbar\omega_{ph},k-q)\Big)\cdot\nabla_j\mathbf{H}_{ln}, \\
\mathbf{\Sigma}_{nn}^{RS}(E,k) &\approx \frac{1}{2}(\mathbf{\Sigma}_{nn}^{>S}(E,k)-\mathbf{\Sigma}_{nn}^{<S}(E,k)).
\end{aligned}$$

All $\mathbf{\Sigma}$'s are block diagonal matrices, where each block $\mathbf{\Sigma}_{nn}^{S}(E,k)$ is of size $N_{orb}\times N_{orb}$. In Eq. (3), ω_{ph} is the confined phonon frequency, $n^{ph}(\omega_{ph})$ the Bose distribution of the phonons with frequency ω_{ph}, and $\nabla_i\mathbf{H}_{nm}$ the derivative of the nearest-neighbor coupling matrix $\mathbf{H}_{nm}$ with respect to the coordinate i=x, y, or z along the bond $\mathbf{R}_m-\mathbf{R}_n$ joining the atoms n and m. To simplify the calculation, all the self-energies are assumed local in position, i. e. only the diagonal $\mathbf{\Sigma}_{nn}^{S}(E,k)$ elements are considered [13]. The form factor $\mathcal{V}_{nmmn}^{ij}(\omega_{ph},q)$ is defined in Ref. [45] and contains information about the phonon modes.

It becomes clear from Eq. (3) that one energy-momentum (E,k) pair is connected to many other $(E\pm\hbar\omega_{ph},k-q)$ pairs through the electron-phonon

scattering self-energies. The number of connections depends on the number of points used to discretize the phonon momentum q between $-\pi$ and π and on the number of frequencies ω_{ph} necessary to span the entire phonon spectrum. The difficulty is to efficiently exchange information between CPUs in order to compute Eq. (3).

Since the expression for the scattering self-energy $\mathbf{\Sigma}^{<S}(E,k)$ in Eq. (3) depends on the Green's Function $\mathbf{G}^{<}(E \pm \hbar\omega, k-q)$ in Eq. (2), these two equations must be iteratively solved till convergence is achieved. The number of required iterations is labeled $N_{SC,iter}$. The same procedure must be applied to the retarded Green's Function $\mathbf{G}^{R}(E,k)$ and its corresponding retarded scattering self-energy $\mathbf{\Sigma}^{RS}(E,k)$.

Once that Eq. (1) and (2) have been solved for each energy E and momentum k, the carrier and current densities can be calculated in the NEGF formalism as

$$n(\mathbf{r}) = -i \sum_{j} \int_{-\pi}^{\pi} \frac{\mathrm{d}k}{2\pi} \int \frac{\mathrm{d}E}{2\pi} \mathrm{tr}\left(\mathbf{G}^{<}_{jj}(E,k)\right) \delta(\mathbf{r}-\mathbf{R_j}) \tag{4}$$

and

$$\begin{aligned} \mathbf{J}(\mathbf{r}) &= \frac{e}{2\hbar} \sum_{i_1 i_2} \int_{-\pi}^{\pi} \frac{\mathrm{d}k}{2\pi} \int \frac{\mathrm{d}E}{2\pi} \mathrm{tr}(\mathbf{H}_{i_1 i_2} \cdot \mathbf{G}^{<}_{i_2 i_1}(E,k) - \mathbf{G}^{<}_{i_1 i_2}(E,k) \cdot \mathbf{H}_{i_2 i_1}) \\ &\quad (\mathbf{R}_{i_2} - \mathbf{R}_{i_1}) \delta(\mathbf{r} - \mathbf{R_{i_1}}), \end{aligned} \tag{5}$$

respectively. The trace operator tr runs over the atomic orbitals σ.

Finally, the solution of Eq. (1) and (2) directly depends on the electrostatic potential $V(\mathbf{r})$ which forms the diagonal entries of the matrix $\mathbf{V}$ in Eq. (1). In turn, $V(\mathbf{r})$ is related to the charge density $n(\mathbf{r})$ through Poisson equation. This yields another self-consistent loop between Eq. (1) and (2) and Poisson equation, as illustrated by the solid orange line in Fig. 2. The number of required iterations to reach convergence between $n(\mathbf{r})$ and $V(\mathbf{r})$ is labeled $N_{Poiss,iter}$.

Ballistic simulations represent a special case of Eq. (1) and (2) since the scattering self-energies $\mathbf{\Sigma}^{RS}(E,k)$ and $\mathbf{\Sigma}^{<S}(E,k)$ are equal to 0. Therefore, no self-consistent iteration between the Green's Functions and the scattering self-energies are required, which can be interpreted as $N_{SC,iter}$=1.

Furthermore, in case of ballistic transport, the NEGF formalism can be replaced by a Wave Function approach, that is computationally much for efficient in 2-D and 3-D structures. There is only one equation to solve

$$\left(\mathbf{E} - \mathbf{H}(k) - \mathbf{\Sigma}^{RB}(k,E)\right) \cdot \mathbf{C}(k,E) = \mathbf{Inj}(k,E), \tag{6}$$

where the multiple right-hand-side vector $\mathbf{Inj}(k, E)$ contains the electronic states injected from the device contacts. Equation 6 has the form of a sparse linear system of equations 'Ax=b" where the unknown is the coefficient vector $\mathbf{C}(k, E)$. More details about the Wave Function approach and how the carrier and current densities are calculated can be found in Ref. [9].

InAs/GaAs Quantum Dot Strain and Eigenstates (NEMO 5)

Strain relaxation:
Strain relaxation is done by minimizing an energy functional U with respect to all degrees of freedom. The degrees of freedom are the displacements $\mathbf{d}_i$ for every atom i. Minimization is done by a Newton iteration for finding the solutions of $\nabla U = 0$:

$$\mathbf{F}(\mathbf{R}) \equiv \mathbf{F}(\mathbf{d}_1, \mathbf{d}_2, ...) \equiv \nabla U(\mathbf{d}_1, \mathbf{d}_2, ...) = 0. \tag{7}$$

$$\mathbf{R}^{(n+1)} = \mathbf{R}^{(n)} + \Delta\mathbf{R}^{(n)} = \mathbf{R}^{(n)} - \mathbf{J}_{\mathbf{F}}^{-1}(\mathbf{R}^{(n)})\mathbf{F}(\mathbf{R}^{(n)}). \tag{8}$$

Here $\mathbf{J}_{\mathbf{F}}$ denotes the Jacobian of the vectorial function $\mathbf{F}$. In the present example $n_{Newton} = 4$ iterations are required until convergence is reached. During every Newton iteration a linear system $\mathbf{Ax} = \mathbf{b}$ must be solved with $\mathbf{A} \equiv \mathbf{J}_{\mathbf{F}}, \mathbf{x} \equiv -\Delta\mathbf{R}^{(n)}, \mathbf{b} \equiv \mathbf{F}(\mathbf{R})$. Due to the immense size for the $3N \times 3N$ matrix $\mathbf{A}$, where N is the number of atoms, an iterative algorithm is used that relies on matrix-vector multiplications (generalized minimum residual with block-Jacobi preconditioner).

Electronic Structure:
Interior eigenstates of the tight-binding Hamiltonian are found by performing an explicitly restarted Lanczos algorithm [49] using *SLEPc*. The Krylov subspace dimension (40), the target energy (1.0eV), the accuracy of the obtained eigenvalues (10^{-8}eV) and the number of requested eigenvalues (6) are held constant.

0.1.6 Q2 Baseline Problem Results

OMEN: Strong Scaling of the Electron-Phonon Scattering Model

OMEN Q2 benchmark is about the simulation of the InGaAs and Si ultra-thin-body (UTB) field-effect transistors (FETs) described in Section 0.1.4 in the presence of electron-phonon scattering. Some technical details about each structure is summarized below:

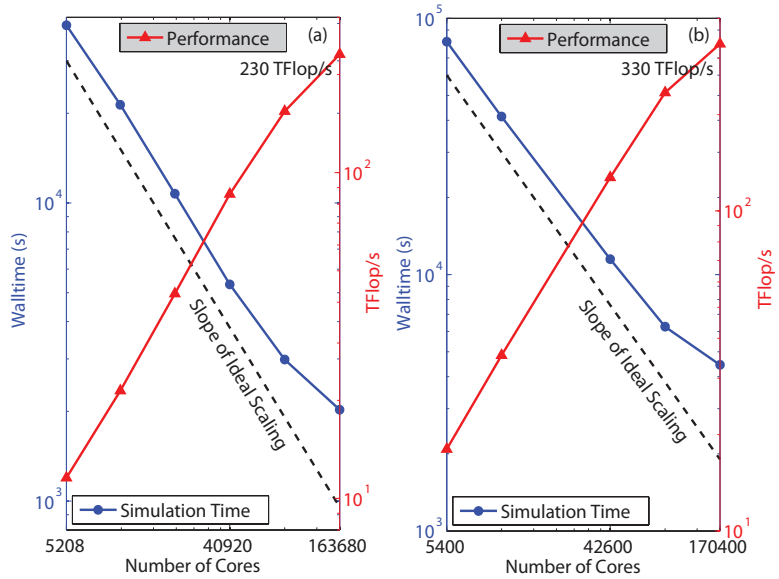

Figure 6: OMEN strong scaling results for the simulation of a <100> InGaAs UTB FET (a) and a <110> Si UTB FET (b) in the presence of electron-phonon scattering. The blue curves with circles refer to the walltime, the red curves with triangles to the sustained performance. The time and flop/s to self-consistently solve Eq. (1), (2), and (3) is reported. A total for $N_{SC,iter}$=10 and 14 iterations are required to reach convergence for the InGaAs and Si devices, respectively.

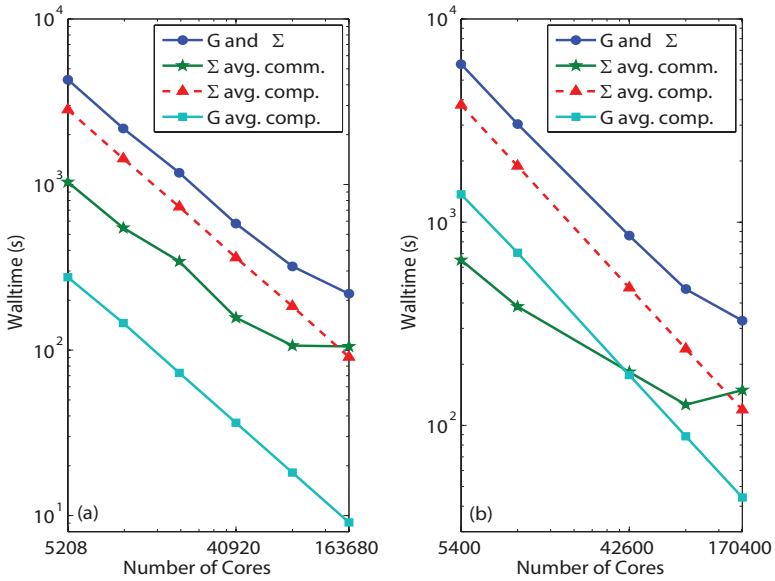

Figure 7: Decomposition of the simulation time for 1 single self-consistent iteration between Eq. (1), (2), and (3) for the same InGaAs (a) and Si (b) devices as in Fig. 6. The total time to solve Eq. (1), (2), and (3) (blue lines with circles), the time to solve Eq. (3) only, without the interprocessor communication time (red lines with triangles), the communication time associated with Eq. (3) (green lines with stars), and the time to solve Eq. (1) and (2) (cyan lines with squares) are reported as function of the number of CPUs.

- InGaAs UTB FET: it is a n-type device (donor doping only) with an In concentration of 53%, electrons flowing along the <100> crystal axis, and confinement along (100). The structure has a body thickness t_{body}=5nm, a total length L=42nm, and contains N_A=4970 atoms, excluding the insulator layer surrounding the channel, which are only accounted for in Poisson equation. The $sp^3d^5s^*$ tight-binding model without spin-orbit coupling is used to describe each atoms so that N_{orb}=10 orbitals are retained. The size of the Hamiltonian matrix describing the simulation domain is therefore N=$N_A \times N_{orb}$=49700. The number of momentum points N_k is an input parameter and is fixed to 31 ranging from -π/a_0 to π/a_0, where a_0=0.58686nm is the lattice constant of $In_{0.53}Ga_{0.47}As$. The number of energy points $N_E(k)$ is different for each k-point and ranges from 721 to 2212. It is determined by the minimum of the contact bandstructure and the position of the source and drain Fermi levels. Initially, to determine $N_E(k)$, the number of CPUs attributed to each momentum point is identical, but then load balance is applied so that each CPU treats the same number of energy points. The phonon spectrum is discretized using $N_{\omega_{ph}}$=9 phonon energies in Eq. 3.

- Si UTB FET: it is a n-type device again, but with electron transport along the <110> crystal axis and confinement along (100). A 1% uniaxial stress is applied along the transport direction to obtain better performance. The structure has the same dimensions as the InGaAs one, but contains N_A=6068 atoms. Using the $sp^3d^5s^*$ tight-binding model without spin-orbit coupling to characterize each atom, the size of the Hamiltonian matrix $\mathbf{H}$ in Eq. 1 is N=60680. For the Si device, N_k can be reduced to 25 without loosing any accuracy, $N_E(k)$ ranges from 1241 to 1914 and the same load balance scheme as introduced before is used. Finally, the number of phonon energies is set to $N_{\omega_{ph}}$=10.

Figure 6 shows the time required to simulate the InGaAs (left) and Si (right) UTB FETs as function of the number of CPUs. It also reports the sustained performance in TFlop/s. This is a strong scaling experiment since the structure dimensions, N_k, $N_E(k)$, and $N_{\omega_{ph}}$ are fixed, only the number of CPUs increases. What is measured is the time and Flop/s to self-consistently solve Eq. (1), (2), and (3) at a given external voltage configuration (source-to-drain voltage V_{ds}=0.7 V, gate-to-source voltage V_{gs}=0.5 V). The iterative solution of Eq. (1), (2), and (3) stops when the value of the charge and

current density in Eq. (4) and (5), respectively, does not vary more than a pre-defined criterion, usually less than 0.1% variation. Convergence is achieved after $N_{SC,iter}$=10 iterations for the InGaAs device and 14 for the Si device. Figure 6 reports the time to compute these 10 and 14 iterations, as well as the time to initialize the simulation environment, solve Poisson equation once, and close the simulation at the end. The numerical results (current and charge densities) do not depend on the number of CPUs, they are always the same.

The scaling of the simulation time is good till 81840 (85200) cores for the InGaAs (Si) device, but then starts to saturate on 163680 (170400) cores. To understand this phenomenon, the time to compute one single self-consistent iteration between Eq. (1), (2), and (3) out of 10 (14) is shown in Fig. 7. The total time, the calculation time for the Green's Functions in Eq. (1) and (2), the calculation time for the scattering self-energies in Eq. (3), excluding the inter-processor time, and the interprocessor communication time in Eq. (3) are reported. The deterioration of the scaling behavior of OMEN comes from an increase of the communication time between 81840 (85200) and 163680 (170400) cores. The goal of this project is therefore to improve the communication scheme of OMEN and to obtain better computational performance. This will help simulate larger device structures within a shorter time and provide new guidelines to design the next generation nanoelectronic transistors.

Note that for all the experiments considered here, each CPU treats only one single momentum point. The number of energy points assigned to each CPU decreases as the total number of CPU N_{tot} increases. For example, in the Si UTB FET case, each CPU treats 31 energy points when N_{tot}=5400, 2 energy points when N_{tot}=85200, and finally only 1 energy point when N_{tot}=170400.

As a metric to measure the improvement of OMEN between Q2 and Q4, the simulation of the Si UTB FET on 85200 cores has been chosen. The total number of instructions, the floating point instructions, the L2 cache misses, the real cycles, and the user cycles were measured at the end of Q2 using PAPI counters and are reported in Tables 2 and 3. In the first measurement (Table 2), the PAPI counters are initialized right after the function MPI_Init and read right before MPI_Finalize. In the second experiment (Table 3), the PAPI counters are initialized before each self-consistent iteration and read after each of them. Furthermore, the PAPI counters are also used to extract data from the initialization phase before the first self-consistent iteration

PAPI Event Name	Counter Value
Total Instructions	2.67e18
FP Instructions	1.44e18
L2 Cache misses	9.52e14
Real cycles	1.66e13
Real (μs)	6.40e9
User cycles	1.66e13
User (μs)	6.39e9

Table 2: PAPI hardware counter data for the Q2 Si UTB benchmark on 85200 cores. Measurement from the beginning to the end of the simulation.

and after the last iteration. All iterations require almost the same number of instructions, floating point instructions, L2 cache misses, and have the same duration, except the first one. In this case, Eq. (3) is not solved and the scattering self-energies are set to 0. Starting from the second iteration, the Green's Functions from the previous iteration are used to compute Eq. (3).

NEMO 5

Strong scaling of ballistic quantum transport. This benchmark tests the four-level parallelization at a constant numerical complexity. The four levels consist of: voltage (12 points), energy (574 points), wavevector (15 points) and contacts (2). In the benchmark voltage parallelization is always complete (any MPI process computes only one voltage). Contact parallelization is at the bottom of the hierarchy and would only enter the picture above 12x574x15=103680 processes.

The simulated structure consists of two domains: one for the Poisson equation which consists of 9000 atoms, and a subset of 5695 atoms for the quantum domain. The simulation time is dominated by the quantum problem being solved on the latter domain. A 10-band $sp^3d^5s^*$ model without spin-orbit coupling is employed for the electronic structure, resulting in a (sparse) matrix size of 56950 for an individual linear equation system.

Different voltage settings necessitate varying amounts of outer-loop Poisson iterations, which are by nature sequential, to reach self-consistency. The simulation is leveled to do only 2 such iterations per voltage to compensate for this effect, and hence the benchmark is not end-to-end.

The Q2 baseline is shown in Fig. 8. Two areas of possible improvements

PAPI Event Name	Init	It. 1	It. 2	It. 3	It. 4	It. 5	It. 6	It. 7	$\cdots$
Total Instructions	3.86e16	4.63e16	2.0e17	1.99e17	1.98e17	2.0e17	2.0e17	2.0e17	$\cdots$
FP Instructions	1.49e14	5.25e16	1.07e17	1.07e17	1.07e17	1.07e17	1.07e17	1.07e17	$\cdots$
L2 Cache misses	1.76e12	2.97e13	7.3e13	7.88e13	7.89e13	7.33e12	7.32e12	7.3e12	$\cdots$
Real cycles	3.67e11	2.42e11	1.23e12	1.22e12	1.22e12	1.23e12	1.23e12	1.23e12	$\cdots$
Real (μs)	1.41e8	9.34e7	4.72e8	4.71e8	4.70e8	4.72e8	4.73e8	4.73e8	$\cdots$
User cycles	3.63e11	2.42e11	1.23e12	1.22e12	1.22e12	1.23e12	1.23e12	1.23e12	$\cdots$
User (μs)	1.40e8	9.34e7	4.72e12	4.71e8	4.7e8	4.72e8	4.73e8	4.72e8	$\cdots$

PAPI Event Name	$\cdots$	It. 8	It. 9	It. 10	It. 11	It. 12	It. 13	It. 14	Close
Total Instructions	$\cdots$	1.99e17	2.0e17	1.99e17	1.99e17	1.99e17	1.99e17	1.99e17	2.64e15
FP Instructions	$\cdots$	1.07e17	1.07e17	1.07e17	1.07e17	1.07e17	1.07e17	1.07e17	1.26e11
L2 Cache misses	$\cdots$	7.3e12	7.31e12	7.34e12	7.31e12	7.31e12	7.31e12	7.31e12	1.45e11
Real cycles	$\cdots$	1.22e12	1.23e12	1.23e12	1.22e12	1.23e12	1.23e12	1.22e12	2.23e10
Real (μs)	$\cdots$	4.7e8	4.73e8	4.72e8	4.71e8	4.72e8	4.72e8	4.71e8	8.6e6
User cycles	$\cdots$	1.22e12	1.23e12	1.23e12	1.22e12	1.23e12	1.23e12	1.22e12	2.18e10
User (μs)	$\cdots$	4.7e8	4.73e8	4.71e8	4.7e8	4.71e8	4.72e8	4.71e8	8.4e6

Table 3: PAPI hardware counter data for the Q2 Si UTB benchmark on 85200 cores. Measurement for each single G-Σ self-consistent iteration as well as the initialization and closing phases.

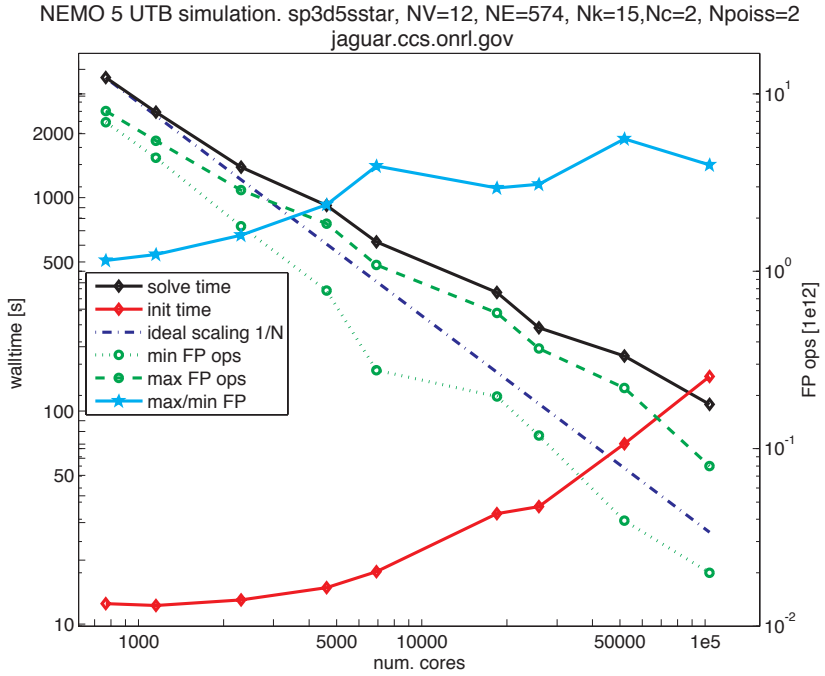

Figure 8: NEMO 5 strong scaling of ballistic quantum transport through an ultrathin-body (UTB) transistor: baseline results.

can be identified:

1. The discrepancy between minimum and maximum number of floating point iterations per process points at load imbalances. These are likely to be caused by the custom multilievel parallelization and task distribution class.

2. At high process counts the time for initialization becomes appreciable and eventually overtakes the solution time. This might be connected to collective file I/O operations.

End-to-end weak scaling of million-atom computations. These benchmarks test strain and quantum state calculations when the physical size of the structure is scaled up. A cube of varying size is partitioned into $n_x \times n_y \times n_z$ smaller cubes with $N_{proc} = n_x n_y n_z$. This cube contains N

Event	Total	Root	Min()	Max()
INS	30254138455409076	280034983914	248553383012	361645376258
FP OPs	93372674237289	3433964657	900291959	3434021087
L2 DCM	75911120950827	736548391	708094766	907337520

Table 4: PAPI hardware counter data for the NEMO 5 Q2 ballistic quantum transport benchmark on 103680 PEs. The initialization required $\sim$ 145 s. INS are the retired instruction counts. FP OPs is a count of the number of executed floating point operations. L2 DCM is the number of L2 data cache misses measured. One sees an imbalance in the floating point work.

Event	Total	Root	Min()	Max()
INS	43004292849913314	329172805948	244553056057	473939755860
FP OPs	5952648720531652	21399404118	20025115794	79696565618
L2 DCM	22818535026750	45613753	41629438	904760521

Table 5: PAPI hardware counter data for the NEMO 5 Q2 ballistic quantum transport benchmark on 103680 PEs. The solve and shutdown required $\sim$ 107.52 s. INS are the retired instruction counts. FP OPs is a count of the number of executed floating point operations. L2 DCM is the number of L2 data cache misses measured. As in the initialization phase, an imbalance in the floating point work is reported.

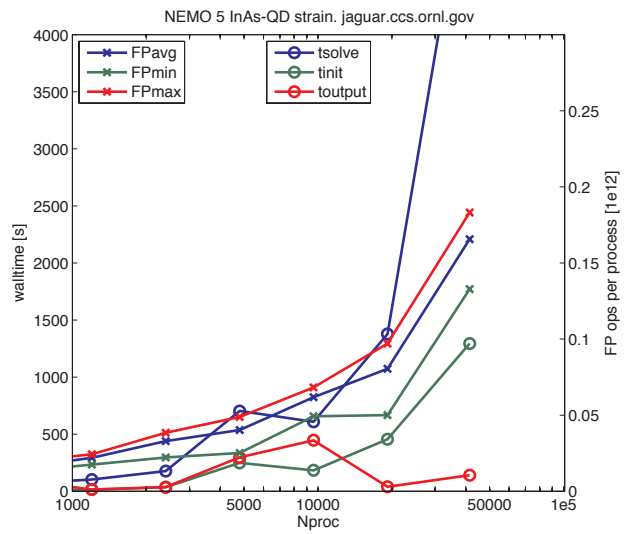

Figure 9: Weak scaling of end-to-end NEMO 5 strain calculation

atoms with $N_{DOF} = 3N$ (strain) or $N_{DOF} = 20N$ (electronic structure) degrees of freedom. In the experiment, the size of the $(5\text{nm})^3$ quantum dot in the center of the cube is held constant such that the strain profile and the obtained eigenstates converge to a well-defined values as the size of the surroundings is scaled up.
The measured simulation time is divided into three phases: an initialization phase where the structure and the necessary equations and parameters are set up (t_{init}), a solve phase (t_{solve}), and a phase where results are saved to file and the simulator is shut down (t_{output}).

Strain relaxation:
A total of n_{Mv} matrix-vector multiplications of FP_{solve} floating-point operations is needed until convergence is reached. The baseline data for this experiment is shown in Fig. 9.
It can be seen that the total number of floating point operations scales superlinearly. This might be attributed to a deteriorating quality of the preconditioner and thus a rising n_{Mv}.

Electronic Structure:
We will report benchmark results for electronic structure computations we

are developing in NEMO 5 in FY11 Q4. The primary algorithm employed in this problem is a Lanczos iteration to resolve a specified number of eigenvalues to a fixed accuracy.

0.1.7 Developments, Enhancements, and Q4 Benchmark Results

Developments and Enhancements -OMEN

The goal here is to reduce the simulation time of a Si UTB transistor including electron-phonon scattering on 85200 cores by a factor of at least 2. As first approach, we tried to improve the communication scheme since inter-processor communication accounts for about 25 to 30% of the total time on 85200 cores and is larger than the computation time on 170400 cores, as shown in Fig. 7 right. The idea was to replace the "MPI_Isend-MPI_Receive" scheme by one-sided MPI communication operations involving MPI_Win_create, MPI_Get, and MPI_Put. However, this attempt was unsuccessful, the communication time did not really decrease and even if it had, this would not have been enough to reach the project objectives. In effect, the communication time is less than 50% of the total time so that even making it disappear does not reduce the total time by a factor of 2.

As an alternative, it was decided to improve the solution of Eq. (1) and (2). To do that, it was first noticed that the size of all the matrices in Eq. (1) and (2) could be divided by a factor of up to 2 if OMEN would work with the primitive unit cell of UTB transistors instead of a more convenient, but larger unit cell. Before going further, it is probably necessary to explain why OMEN did not work with the smallest possible unit cell (the primitive unit cell) in the first place. There is an historical reason for that: OMEN was first developed to treat 3-D nanowires, for which the lateral dimensions are confined and not periodic as in 2-D ultra-thin-body structures. Hence, at the beginning of any simulation, during the initialization phase, an Hamiltonian matrix $\mathbf{H}$ was created and only the diagonal entries of this matrix had to be updated later to account for the electron energy and the electrostatic potential.

In a 2-D UTB, as targeted here, one of the lateral dimensions, for example z, is periodic and induces a k-dependence for all the physical quantities (density-of-states and transmission) so that the Hamiltonian matrix takes

the following form

$$(9) \quad \mathbf{H} = \mathbf{H_{00}} + \mathbf{H_{01}} \cdot \exp(ik\Delta_z) + \mathbf{H_{10}} \cdot \exp(-ik\Delta_z),$$

where $\mathbf{H}_{00}$ describes the central unit cell of the device, $\mathbf{H}_{01}$ ($\mathbf{H}_{10}$) the connection to the unit cell situated above (below) at z=+Δ_z (z=-Δ_z), and $\exp(\pm ik\Delta_z)$ is a phase factor. Here, Δ_z represents the width of a unit cell along the periodic direction. When OMEN was extended to 2-D structures, it was easier and more convenient to keep the simulation flow unchanged. The Hamiltonian matrix should be created once at the beginning and not be too much altered. Hence, during the initialization phase, a matrix $\hat{\mathbf{H}}$ defined as

$$(10) \quad \hat{\mathbf{H}} = \mathbf{H_{00}} + \mathbf{H_{01}} + \mathbf{H_{10}},$$

was generated, as well as two index vectors "up_index" and "down_index" listing all the elements in $\hat{\mathbf{H}}$ that should be multiplied by a phase factor, either $\exp(+ik\Delta_z)$ or $\exp(-ik\Delta_z)$.

The advantage of this approach was that it did not require much change to the code to go from 3-D nanowires to 2-D UTBs: the construction of the Hamiltonian had to be slightly modified and a function "update_phase" had to be introduced so that OMEN could reconstruct the Hamiltonian from Eq. (9) based on Eq. (10) and the two vectors "up_index" and "down_index". However, there is also a disadvantage with this approach. Each entry of the matrix $\hat{\mathbf{H}}$ can only depend on a single phase factor (1 for $\mathbf{H}_{00}$, $\exp(ik\Delta_z)$ for $\mathbf{H}_{01}$, and $\exp(-ik\Delta_z)$ for $\mathbf{H}_{10}$). To fulfill this requirement, the thickness Δ_z of the periodic unit cell along the z direction must be chosen large enough to avoid that any matrix entry h_{ij} is defined as h_{ij}=a_{ij}+$b_{ij} * \exp(\pm ik_z\Delta_z)$. For certain applications, as for example when z is aligned with the (110) crystal axis, Δ_z had to be extended to $\sqrt{2} * a_0$, where a_0 is the lattice constant of the UTB material, while it could have been reduced to $a_0/\sqrt{2}$ if OMEN would have worked with primitive unit cells, and therefore smaller Hamiltonian $\mathbf{H}$ matrices.

As part of the FY11 OMB PMM OASCR Software Effectiveness Metric program, the structure and the simulation flow of OMEN have been modified so that each sub-matrix $\mathbf{H}_{00}$, $\mathbf{H}_{01}$, and $\mathbf{H}_{10}$ could be independently created at the beginning of each device simulation. In this way, there is no more any constraint on the thickness Δ_z of the UTB structure and OMEN can utilize primitive unit cells when needed. The major difficulty has been to modify

the function "update_phase" so that the matrices $\mathbf{H_{00}}$, $\mathbf{H_{01}} \cdot \exp(ik\Delta_z)$, and $\mathbf{H_{10}} \cdot \exp(-ik\Delta_z)$ can be efficiently added. In effect, they all have a different sparse pattern leading to some complications if they are very often summed up. Hence, to accelerate their addition, the sparse pattern of the sum of the three matrices has been stored in the matrix $\mathbf{H_{00}}$ during the initialization phase so that it can be used each time the phase factors in $\mathbf{H}$ are updated.

To summarize, in order to accelerate the simulation time of OMEN when electron-phonon scattering is considered, the device size has been reduced so that the size of the Hamiltonian matrix $\mathbf{H}$ and scattering self-energies $\mathbf{\Sigma}$ in Eq. (1) and (2) becomes smaller. To do that, about 70% of the class formerly called "Hamilton.H" and now "ElHamilton.H" had to be modified. This represents more than 1000 lines of code that have been rewritten to speed up OMEN by a factor of 2.17 while simulating a Si UTB transistor with electron-phonon scattering on 85200 cores.

Developments and Enhancements -NEMO5

NEMO5

Three different simulation types were looked at during Q2: a strong scaling example (*UTB*), which was the primary target, and two weak scaling examples (*Strain* and *Eigenstates*). The following developments and enhancements related to these examples took place within the scope of the ASCR GPRA / PMM project:

- **Better compiler options:** It was recognized that a significant speedup can be achieved in PETSc when specifying the flag `--with-debugging=0` during compilation of PETSc and SLEPc. This yielded significant performance improvements for all three simulation examples. The mentioned flag was found to be more significant than the `-O3` compiler flag.

- **Better matrix preconditioning:** For the weak scaling example *Strain* the preconditioning in the Q2 baseline was done using a simple block Jacobi method (PETSc option `bjacobi`). This method was chosen due to the lack of memory-scalability of the adaptive Schwarz method implementation (PETSc option `asm`) in PETSc which provides superior preconditioning. It was discovered during this exercise that the `asm` implementation featured some arrays which scaled with the global

number of variables. As a result from the interaction of the NEMO5 team with the PETSc team, this shortcoming could be removed. The changes will be part of the next release version of PETSc.

- **Better load balancing:** The Q2 baseline results of the strong scaling example *UTB* did not have an optimal distribution of jobs onto the processes (i.e., some processes needed to solve more jobs than others). The reason for this behavior was analyzed, and it was found that the chosen multilevel parallelization strategy could not yield optimal results. An additional strategy was implemented in the code to provide balancing at very high process counts. In the Q4 results the number of tasks per process is now optimally balanced for all numbers of cores. However, some load imbalance remains because of differences in the computational load of individual tasks. As there is only one task per process left at 103680 cores, this imbalance cannot be further improved unless the problem is sized up.

- **Faster initialization of the simulation:** From the Q2 baseline scaling graph it was observed that the initialization time, i.e. the time required to reach the point where the actual solution is carried out by PETSc or SLEPc, increased and eventually dominated the entire simulation time as the process count approached 1e5 cores. Two reasons were identified:
  Domain decomposition of large physical domains: In the weak scaling examples, the large physical structure is partitioned into many small pieces which are stored individually. The partitioning involves the setup of couplings to neighboring partitions, and neighbors of neighbors. Although local sub-communicators were already used in Q2 to compute the coupling tables between neighboring processes, residual MPI Barriers made the code behave sequentially in the number of partitions. Removal of these barriers resulted in an asynchronous operation mode, thus speeding up the computation of the coupling tables while retaining the correctness of the results.
  Read-in of material parameters from file: It was observed that the read-in of the material parameters from file took as long as 1 minute at very large process counts for the simple reason that every process needed to access the same file on disk. This flaw was corrected by letting the master process read the file and broadcast the contents to all other pro-

PAPI Event Name	Counter Value
Total Instructions	1.17e18
FP Instructions	4.13e17
L2 Cache misses	3.56e14
Real cycles	7.66e12
Real (μs)	2.95e9
User cycles	7.66e12
User (μs)	2.94e9

Table 6: PAPI hardware counter data for the Q4 Si UTB benchmark on 85200 cores. Measurement from the beginning to the end of the simulation.

cesses, a strategy that had already been adapted for the input deck in Q2. This strategy reduced the time for setting up material parameters to less than 5 seconds on 103680 cores.

0.1.8 Q4 Benchmark Results -OMEN

After the code was successfully modified and debugged, the same numerical experiment as during Q2 (see Tables 2 and 3) was performed. A Si UTB transistor was simulated on 85200 cores including electron-phonon scattering until convergence between the Green's Functions G and self-energies Σ is achieved. The results are shown in Tables 6 and 7. The key achievement is a reduction of the simulation time from 6400 sec down to 2950 sec corresponding to a speed up factor of 2.17, more than the required factor 2. The time reduction can mainly be attributed to a decrease of the total instruction counts, as it can be seen in Tables 2 and 6, which is a direct consequence of the reduction of the device unit cell and the corresponding Hamiltonian matrix and scattering self-energies.

As complementary verification of the optimization process, the same timing experiment as above has been conducted on 42600 cores instead of 85200. The times and operation counts are reported in Table 8. As it can be observed, the exact same simulation takes 5900 sec on 42600 cores after Q4 while it lasted 6400 sec on 85200 cores before the code modifications. Here again, the improvement factor larger than 2 is confirmed.

PAPI Event Name	Init	It. 1	It. 2	It. 3	It. 4	It. 5	It. 6	It. 7	$\cdots$
Total Instructions	3.00e16	6.14e15	8.05e16	8.07e16	8.09e16	8.12e16	8.08e16	8.08e16	$\cdots$
FP Instructions	1.38e14	6.36e15	3.13e16	3.13e16	3.13e16	3.13e16	3.13e16	3.13e16	$\cdots$
L2 Cache misses	3.68e12	5.02e12	2.6e13	2.6e13	2.61e13	2.59e13	2.59e13	2.59e13	$\cdots$
Real cycles	2.89e11	3.51e10	5.15e11	5.17e11	5.19e11	5.21e11	5.18e11	5.19e11	$\cdots$
Real (μs)	1.11e8	1.35e7	1.98e8	1.99e8	1.99e8	2.0e8	1.99e8	2.0e8	$\cdots$
User cycles	2.86e11	3.51e10	5.15e11	5.17e11	5.18e11	5.21e11	5.18e11	5.18e11	$\cdots$
User (μs)	1.10e8	1.35e7	1.98e8	1.99e8	1.99e8	2.0e8	1.99e8	1.99e8	$\cdots$

PAPI Event Name	$\cdots$	It. 8	It. 9	It. 10	It. 11	It. 12	It. 13	It. 14	Close
Total Instructions	$\cdots$	8.03e16	8.05e16	8.1e16	8.12e16	8.0e16	8.18e16	8.13e16	6.35e14
FP Instructions	$\cdots$	3.13e16	3.13e16	3.13e16	3.13e16	3.13e16	3.13e16	3.13e16	1.44e11
L2 Cache misses	$\cdots$	2.59e13	2.59e13	2.87e13	2.59e13	2.58e13	2.6e13	2.59e13	5.43e10
Real cycles	$\cdots$	5.15e11	5.17e11	5.22e11	5.22e11	5.13e11	5.27e11	5.23e11	4.93e9
Real (μs)	$\cdots$	1.98e8	1.99e8	2.01e8	2.01e8	1.97e8	2.03e8	2.01e8	1.9e6
User cycles	$\cdots$	5.15e11	5.17e11	5.22e11	5.22e11	5.12e11	5.27e11	5.23e11	4.94e9
User (μs)	$\cdots$	1.99e8	1.99e8	2.01e8	2.01e8	1.97e8	2.03e8	2.01e8	1.85e6

Table 7: PAPI hardware counter data for the Q4 Si UTB benchmark on 85200 cores. Measurement for each single G-Σ self-consistent iteration as well as the initialization and closing phases.

PAPI Event Name	Counter Value
Total Instructions	1.16e18
FP Instructions	4.13e17
L2 Cache misses	3.47e14
Real cycles	1.54e13
Real (μs)	5.90e9
User cycles	1.53e13
User (μs)	5.82e9

Table 8: PAPI hardware counter data for the Q4 Si UTB benchmark on 42600 cores. Measurement from the beginning to the end of the simulation.

0.1.9 Q4 Benchmark Results -NEMO5

UTB (strong scaling)

The performance metrics of the Q4 run at 103680 cores are shown in table 9 and compared to the Q2 run. It is observed that the overall instruction count is about 30% lower in the Q4 run, which is mainly attributed to the removal of debugging symbols. The large time difference in the initialization time is however mainly due to the improved read-in of the material database. The total number of floating-point operations is about the same in both runs, providing experimental evidence that the problem statement and its associated computational burden did not change. The load imbalance measured as the ratio between the maximum and minimum number of floating-point operations on a process, decreased from a factor of 4 to a factor of 2.

Figure 10 displays the strong scaling of the Q4 run. Compared to the Q2 run, the graph now exhibits a more uniform behavior and the simulation time is dominated by the actual solve time at all process counts. The remaining deviation from ideal scaling is likely to be due to inherent differences of the computational load between individual tasks. Choosing a bigger example, i.e. one where the maximum parallelization is not reached and one process is assigned multiple tasks even at the maximum process count, might alleviate this imbalance.

The bottom-line speed-up factor achieved between the Q2 and Q4 runs

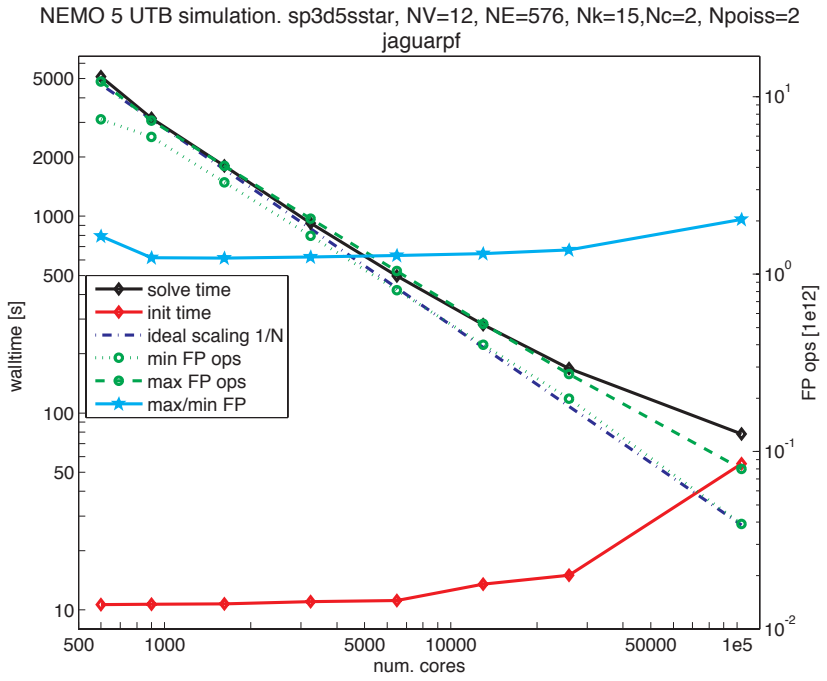

Figure 10: NEMO 5 strong scaling of ballistic quantum transport through an ultrathin-body (UTB) transistor: Q4 results.

Metric	Q2	Q4
INSinit	30.25e15	22.49e15
FPinit	93.37e12	3.54e12
tinit	145.00s	55.23s
INSsolve	43.00e15	31.30e15
FPsolve	5.95e15	6.00e15
FPmax	79.70e9	79.75e9
FPmin	20.02e9	39.13e9
tsolve	107.52s	78.42s
ttotal	252.52s	133.65s

Table 9: Improvements of the NEMO5 strong scaling example *UTB*, as measured by PAPI, at 103680 cores. INSinit, INSsolve: total number of instructions during initialization and solve phase. FPinit, FPsolve: total number of floating point instructions during initializaion and solve phase. FPmin, FPmax: minimum and maximum number of floating-point operations for an individual process during solve phase. tinit, tsolve, ttotal: wallclock time duration of initialization and solve phases, and total simulation time.

amounts to 1.89 at 103680 cores. The problem formulation and the simulation results stayed are the same in both runs.

Strain (weak scaling)

The performance metrics of the Q4 run at 41472 cores are shown in table 10 and compared to the Q2 run. The constant matrix size and similar total number of floating-point operations provide empirical evidence that the problem formulation as well as its computational burden did not change between the runs. However, the initialization and in particular the solve time both exhibit significant reductions. The reduction in initialization time can

be attributed to a faster set-up of the atomistic domain, and the reduction in solve time is a combination of better compiler options and improved preconditioning.

Metric	**Q2**	**Q4**
N	994.121.664	994.121.664
FPsolve	6.87e15	6.75e15
FPmax	183.17e9	179.67e9
FPmin	132.81e9	130.66e9
tinit	1294.57s	990.83s
tsolve	5610.34s	862.17s
toutput	141.03s	149.08s
ttotal	7045.94s	2002.08s

Table 10: Improvements of the NEMO5 weak scaling example *Strain*, as measured by PAPI, at 41472 cores. N: matrix size. FPsolve, FPmin, FPmax: number of floating-point instructions during solve phase (total, minimum and maximum for an individual process) tinit, tsolve, ttotal: wallclock time duration of initialization, solve and output phases, and total simulation time.

The weak scaling is shown in Fig. 11. The scale is the same as in the Q2 run, and the reduction of initialization and solve time is apparent. The time required for writing the results stayed essentially the same, as no improvements were made. This time is substantially lower than the other two and can be controlled easily by increasing the number of created files.

The bottom-line speed-up factor achieved between the Q2 and Q4 runs amounts to 3.52 at 41472 cores. The problem formulation and the simulation results stayed are the same in both runs.

Eigenstates (weak scaling)

# MPI procs	DOFs	Atoms	# mat-vec	tinit [s]	tsolve [s]	toutput [s]
600	14.224.896	4.741.632	2.098	6.94	60.08	104.36
1200	28.584.384	9.528.128	2.690	13.96	97.08	12.67
2400	57.746.496	19.248.832	4.252	28.61	131.50	38.85
4800	115.843.416	38.614.472	5.365	59.92	180.80	126.34
9600	231.926.328	77.308.776	7.651	143.15	258.56	438.15
19200	461.971.968	153.990.656	9.265	340.30	351.36	42.12
41472	994.121.664	331.373.888	21.379	990.83	862.17	149.08

Table 11: NEMO5 weak scaling example *Strain* during the Q4 run.

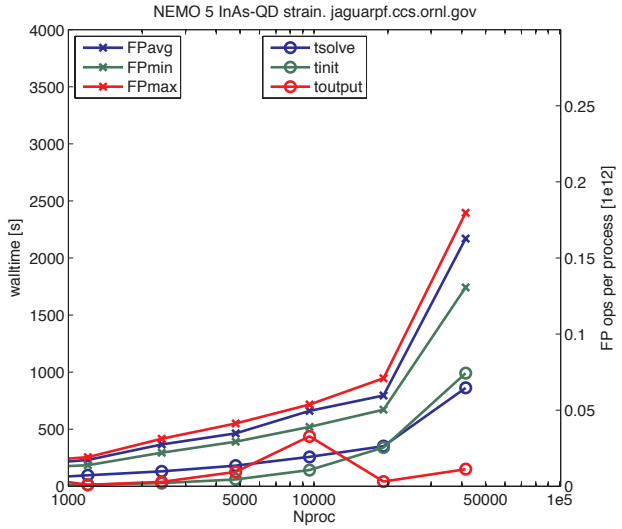

Figure 11: Weak scaling of end-to-end NEMO 5 strain calculation: Q4 results.

The performance metrics of the Q4 run at 41472 cores are shown in table 12 and compared to the Q2 run.

The weak scaling is shown in Fig. 12.

The bottom-line speed-up factor achieved between the Q2 and Q4 runs amounts to 2.73 at 41472 cores. The problem formulation and the simulation results stayed are the same in both runs.

Application Summary of Results and Analysis - OMEN

The goal of the FY11 OMB PMM OASCR Software Effectiveness Metric program has been reached for OMEN since the simulation time on 85200 cores of a Si UTB with electron-phonon scattering could be reduced by a factor 2.17. Furthermore, after Q4, the simulation time on 42600 cores is smaller than it was after Q2 on 85200 cores.

What should be noted is that the algorithms of OMEN were not really

Metric	Q2	Q4
Nlancz	349	349
FPsolve	11.55e15	11.48e15
FPmax	351.0e9	349.3e9
FPmin	156.1e9	154.4e9
tinit	1001.3s	618.4s
tsolve	2256.1s	488.4s
toutput	147.5s	141.0s
ttotal	3404.9s	1247.8s

Table 12: Improvements of the NEMO5 weak scaling example *Eigenstates*, as measured by PAPI, at 41472 cores. Nlancz: number of Lanczos iterations to achieve six converged eigenstates. FPsolve, FPmin, FPmax: number of floating-point instructions during solve phase (total, minimum and maximum for an individual process) tinit, tsolve, ttotal: wallclock time duration of initialization, solve and output phases, and total simulation time.

changed, but the translation of the physical problem into a numerical problem was greatly enhanced. Consequently, the increase of the communication time as function of the number of cores, as illustrated in Fig. 7, persists and still needs some improvement. This will be done outside of the FY11 OMB PMM OASCR Software Effectiveness Metric program, but the experience collected during this project will be really useful for that.

Application Summary of Results and Analysis - NEMO5

The ASCR PMM Software exercise yielded the following core results for the NEMO5 code:

1. **Bottom-line speed-up factor of 1.89, 3.52 and 2.73 were achieved in the scope of the ASCR PMM Software project for the three simulation examples (at 103680, 41472 and 41472 cores, respectively).** The performance metric goal has therefore been met by

# MPI procs	DOFs	Atoms	# Lanczos its.	tinit [s]	tsolve [s]	toutput [s]
12	116.640	5.832	140	0.24	138.95	0.67
24	276.480	13.824	161	0.18	162.46	0.75
48	540.000	27.000	161	0.26	178.58	2.03
96	933.120	46.656	186	0.34	197.85	4.81
192	1.946.720	97.336	296	0.58	303.85	9.24
1200	12.721.120	636.056	294	3.57	399.68	11.00
2400	25.194.240	1.259.712	345	7.91	475.58	26.86
4800	48.122.080	2.406.104	352	20.63	526.81	111.94
9600	98.260.000	4.913.000	314	52.37	494.67	365.34
19200	196.006.880	9.800.344	327	155.44	511.45	40.34
41472	420.491.520	21.024.576	349	618.36	488.44	140.98
82944	842.883.840	42.144.192	395	2405.81	623.49	364.42

Table 13: NEMO5 weak scaling example *Eigenstates* during the Q4 run.

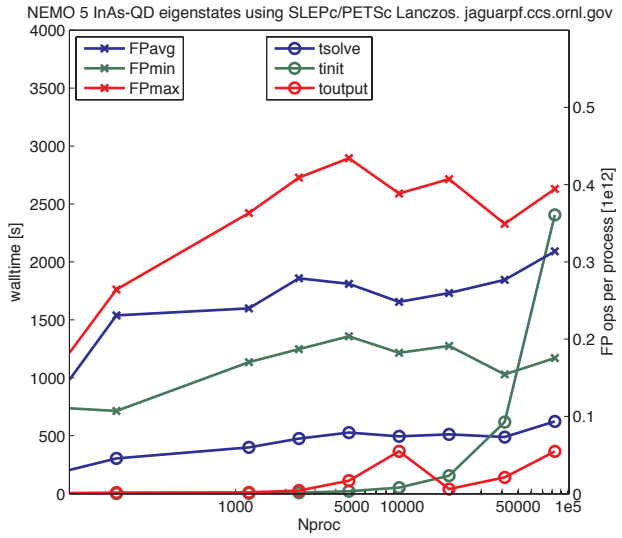

Figure 12: Weak scaling of end-to-end NEMO eigenstate calculation: Q4 results.

NEMO5.

2. **Memory and performance bottlenecks in PETSc were discovered and eliminated.** Memory bottlenecks were corrected in the ASM precoditioner and the AO mapping routines. A performance bottleneck was corrected in the PETSc terminal output routines. These improvements will be part of the next release of PETSc. PETSc is a high-impact code used by thousands of codes, and the present improvements can be termed significant.

3. **Initialization time was significantly reduced.** These improvements will benefit all future large-scale simulations with the NEMO5 code.

Outlook:

- For large structures, the domain decomposition into tens of thousands of domains is still not scalable. Further optimization of this is nontrivial.

- Compiler-related enhancements have not yet been exhausted. Use different compiler, more optimization options.

- Load balancing is not perfect, as shown by the differing min/max values for the floting-point instructions. Some of the imbalance is inherent to the problem; however, by making the domain decomposition more balanced one might achieve further improvements in the weak scaling examples.

0.2 LAMMPS

0.2.1 Introduction

LAMMPS [109] (Large-scale Atomic/Molecular Massively Parallel Simulator,
lammps.sandia.gov) is a classical molecular dynamics and particle simulation code. It has potentials for soft materials (biomolecules, polymers) and solid-state materials (metals, semiconductors) and coarse-grained or mesoscopic systems. It can be used to model atoms, or more generically, as a parallel particle simulator at the atomic, meso, or continuum scale. LAMMPS runs on single processors or in parallel using message-passing techniques and a spatial-decomposition of the simulation domain. The code is designed to be easy to modify or extend with new functionality. It is distributed from the website (lammps.sandia.gov/download.html) as an open source code under the terms of the GPL by Sandia National Laboratories, a US Department of Energy laboratory. The main authors of LAMMPS are Steve Plimpton, Aidan Thompson, and Paul Crozier. A long list of additional contributing authors is available on the LAMMPS webpage. Funding for LAMMPS development has come primarily from DOE (OASCR, OBER, ASCI, LDRD, Genomes-to-Life).

0.2.2 Background and Motivation

During the past few years, there has been a great deal of interest among the molecular dynamics (MD) developers and users communities to harness the impressive compute power of Graphics Processing Units (GPUs). We have repeatedly heard from many LAMMPS users that they would like to be able to run LAMMPS on GPUs to accelerate their simulations. Some users have even developed CUDA[13] kernels to be able to run LAMMPS simulations using NVIDIA[14] GPUs.

The future success of LAMMPS will require that it run effectively on next-generation supercomputers. A broad range of critical science problems can be addressed using LAMMPS particle simulations, including a wide variety of problems in materials science, biology, and nanoscience. Particle simulation codes tend to scale and perform well on the high performance computing

[13] CUDA C Programming Guide, GPU Computing at NVIDIA's Developer website
[14] developer.nvidia.com

(HPC) platforms of today, and are anticipated to be among the applications that most effectively harness the compute power of next-generation HPC platforms.

It is clear that the future of HPC will include many-core architectures, and that many of those platforms will include GPU accelerators to get to high flop rates with reasonable energy consumption. NVIDIAs GPUs and the CUDA programming model appear to be leading the way towards such next-generation supercomputing. Other hardware manufacturers and other programming models may catch up, but application developers do not have the luxury of waiting to see which will become the most predominant and useful. Codes that perform well on next-generation HPC hardware will be preferred over those that cannot take full advantage of the accelerators.

Developers of other MD codes (HOOMD, NAMD, etc) have already shown impressive speed-ups by using GPUs, but few codes have the rich feature set and wide-ranging force field capabilities that LAMMPS includes. We know of no other code that comes close to LAMMPS in terms of available force fields and pair styles, which include: ReaxFF, COMB, EAM, MEAM, Stillinger-Weber, Tersoff, CHARMM, Amber, OPLS, Gromacs, AI-REBO, eFF, DPD, Lennard-Jones, peridynamics, granular, GayBerne, SRD.

0.2.3 Capability Overview

We aim to retain the rich feature set and simulation capabilities of LAMMPS while harnessing the compute power of GPUs. Ultimately, we want to achieve shorter time-to-solution for the particle simulations that accelerate LAMMPS users science. It is hoped that we will be able to achieve 5x to 100x speed-ups on typical LAMMPS simulations by effectively utilizing the available GPUs on hybrid HPC platforms. It is important to note that (unlike other codes such as HOOMD) we are not aiming to optimize for running on a single GPU. While GPU-enabled LAMMPS will run well on a single GPU, the target platform is a CPU/GPU hybrid HPC platform.

GPU-enabled LAMMPS aims for dual parallelism: spatial decomposition with MPI between CPU cores, and force decomposition with CUDA on individual GPUs. During the FY11 enhancement work, we will leverage LAMMPSs existing spatial decomposition and MPI infrastructure, while adding a second layer of parallelism in the underlying force calculations.

The community of researchers interested in developing LAMMPSs GPU capabilities is large and growing, but it is also diverse and geographically

distributed. We aim to leverage the work of all of our GPU-LAMMPS developers, integrating the best efforts into useful GPU modules for LAMMPS. Currently, LAMMPS has some limited capabilities for harnessing GPUs for particle simulation. Specifically, the following pair potentials are available as part of the GPU package: the CMM coarse grained potential (lammps.sandia.gov/doc/pair_cmm.html), the LJ, LJ 9-6, and LJ + coulomb potentials, and the Gay-Berne potential (lammps.sandia.gov/doc/pair_gayberne.html). We intend to expand this list to include many more pair potentials, including the EAM pair potential that will be necessary to run our chosen test problem on a GPU-based cluster.

0.2.4 Science Driver for Metric Problem

A small number of standard LAMMPS benchmark problems are bundled with the LAMMPS distribution, and typical computational performance results reported for these problems on the LAMMPS website (http://lammps.sandia.gov/bench.html). Rather than devise new benchmark problems, we simply leverage our existing benchmarks and results. Specifically, we use LAMMPS's standard EAM metallic solid benchmark problem for purposes of this study. This test problem involves simulation of a nanoscopic sample of solid copper, with the base case consisting of 32,000 copper atoms. We employ in Q2 the embedded atom method (EAM) potential to represent the interactions between the copper atoms. In such interaction models, forces are completely short-ranged and consequently truncated at 4.95 angstroms. Each copper atom interacts with approximately 45 neighboring atoms that reside within that cutoff. The standard Velocity Verlet algorithm is used to integrate the equations of motion forward in time with a 5 femtosecond time-step in the micro-canonical (NVE) ensemble. We plan to extend the study to include Lennard-Jones type interactions in Q3. For the L-J interaction, we will begin with the rapid melt problem.

Note that the Cu_u3.eam file contains the atomic interaction parameters required for simulation of copper using the EAM potential. This file is distributed with LAMMPS and can be found in the bench sub-folder.

```
# bulk Cu lattice

variable x index 1
variable y index 1
variable z index 1
```

```
variable xx equal 20*$x
variable yy equal 20*$y
variable zz equal 20*$z

units metal
atom_style atomic

lattice fcc 3.615
region box block 0 ${xx} 0 ${yy} 0 ${zz}
create_box 1 box
create_atoms 1 box

pair_style eam
pair_coeff 1 1 Cu_u3.eam

velocity all create 1600.0 376847 loop geom

neighbor 1.0 bin
neigh_modify    every 1 delay 5 check yes

fix 1 all nve

timestep 0.005
thermo 50

run 100
```

This simple test problem can be seen as representative of a wide array of particle simulations that can be performed with LAMMPS. Speed improvements that are realized for this test problem will likely carry over to many other simulations of interest to LAMMPS users. Even though this test problem consists of little more than thermal vibrations of atoms within a crystal lattice, its computational cost requirements (CPU time per atom per time-step) are expected to be very similar to other EAM simulations of solid metallic materials. Such atomistic simulations can be used to investigate atomic-level materials characteristics (thermophysical properties, diffusivity, melting properties, defect properties, shear and tensile strength, etc.) and other atomistic-level phenomena.

0.2.5 The Model and Algorithm

The EAM potential that is used to model the copper-copper inter-atomic interactions is from [111, 112]. The EAM is based on the quasiatom scheme where all atoms are embedded in a field composed of all the other atoms composing the host, and impurities are considered locally uniform. The embedding energy is a function of the electron density. If one considers each atom as an individual impurity in the system, the cohesive energy can be

evaluated because the impurity is a functional of the electron density of the remaining unperturbed host system. This is the basic premise: the impurity potential is determined by the electron density of the host prior to adding the impurity. The total potential is the sum of both the unperturbed host electron density potential and the position and charge potential of the impurity. Note that energy in such systems has been shown to be well approximated by a pair interaction.

The function form for the total energy E_{tot} and the contribution, E_i, of the ith atom to the total are:

$$(11) \quad E_i = F(\rho_{h,i}) + \frac{1}{2}\sum_{j\neq i}\phi(R_{ij}) \ ,$$

$$(12) \quad E_{tot} = \sum_i F_i(\rho_{h,i}(R_{ij})) + \frac{1}{2}\sum_{i,j,i\neq j}\phi_{i,j}(R_{ij}) \ .$$

where F_i is the embedding energy, $\rho_{h,i}$ is the density of the host at position R_i *without* atom i, $\phi_{i,j}$ is the short-range pair potential, and $R_{i,j}$ is the distance between the position of atoms i and j. $F(\rho)$ and $\phi(r)$ are determined (typically empirically) from the properties of the corresponding pure metal. It is empirically found that for many repulsive pair potentials one can assign a particular form (as a function of charge) $st\ \phi_{ij}(r) \sim Z_i(r)Z_j(r)/r$ to represent the interaction of two neutral screened atoms.

Additional information about the EAM potential as implemented in LAMMPS can be found on the LAMMPS website: lammps.sandia.gov/doc/pair_eam.html.

0.2.6 Q2 Baseline Problem Results

Existing benchmarking data for the test problem can be found on the LAMMPS website (http://lammps.sandia.gov/bench.html#eam) for strong scaling (Figure 13) and weak scaling (Figure 14) parallel computational efficiency results. Figure 13 shows fixed-size parallel efficiency for the same 32,000 atom problem run on different numbers of processors, and Figure 14 shows scaled-size efficiency for runs with 32,000 atoms per processor. Thus a scaled-size 64-processor run is done on a system that has 2,048,000 atoms, and a 32,000 processor run includes roughly a billion atoms. Each curve is normalized to be 100% efficient on a single processor for the respective machine. One-processor times-to-solution are shown in parentheses.

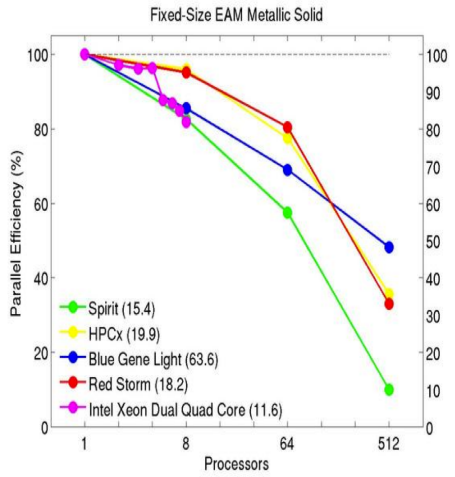

Figure 13: Strong scaling parallel efficiencies for LAMMPS simulations of the EAM metallic solid problem on up to 512 processors. Simulations were performed on five different machines.

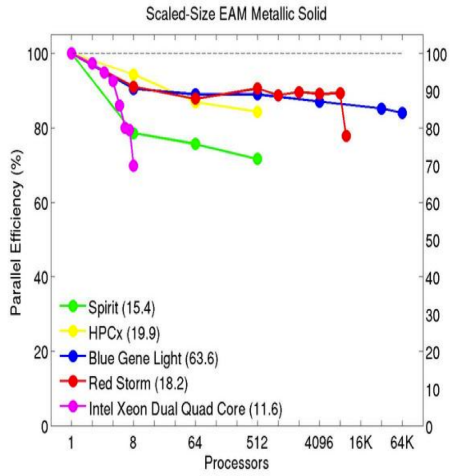

Figure 14: Weak scaling parallel efficiencies for LAMMPS simulations of the EAM metallic solid problem on up to 64,000 processors. Simulations were performed on five different machines.

The Q2 problem is to calculate the molecular dynamics trajectory of a 32000 atom system of copper atoms for one hundred time steps. The thermodynamic properties (temperature, pressure, potential energy, and total energy) of the system are computed three times: once at initialization, after 50 time steps, and at the end of the run at the 100th time step. We computed the problem with both the Portland Group Inc. (PGI) and Intel compilers. We will add GNU Compiler Class (GCC) builds of the binary in Q3 and strive to accelerate the force computation using GPU enhancements in Q4. The small walltime of the Q2 baseline runs should not be a problem unless it gets so small that it is below the accurate measuring ability of our tools. In this case, the plan will be to increase the numbers of atoms in the system or simply run more timesteps (i.e. change the "run 100" command to "run 1000"), since the simulation time scales almost perfectly with the number of timesteps.

Direct use of the GPUs for this problem will focus on accelerating the pair and neighbor calculation times. We expect that comparing a single node instance of the problem with and without the GPU will reveal a significant speed-up with the GPU. The enhancement will be reported by first comparing the single node performance of the Q2 and Q4 algorithms. In reference to Table 15, one notes a turning point of diminishing returns for strong scaling the Q2 problem. This is the point where the parallel computation of coarse grained parallel tasks does not compensate for the increased communication overheads. The distributed version, we predict, may be complicated by the communication features of the algorithm that will likely be impacted by the deeper memory structures in the Q4 code. It is not clear yet how to balance the bulk synchronous behavior of the flow of the force computation kernel. Even if communication is asynchronous, a result generated on the GPU (or some form of it) needs to be copied off the GPU to a communication buffer the in the host's memory prior to sending a message. On receive, the message has to be parsed by the CPU and useful data for next iteration copied from memory to the GPU. This traversal is costly may diminish the overall impact of the single node speed-ups. This is an important problem facing the entire HPC community today. We will aim to identify fine grained tasks well suited for the GPU and balance the global execution in an MPI environment. It is noted that as of the end of Q2, CUDA 4.0 is available for review. This version includes many advances delivering a more realistic and effective use of a hybrid programming model between a host CPU and GPU to the user, as well as multi-GPU computing. One can now test and explore features such

as peer-to-peer communication enabling the direct communication between GPUs without the system memory copy overhead and remote operations such as load / store from one GPU to another. Another important feature is the idea of Uniform Virtual Addressing (UVA) whereby a single address space for both the CPU and GPU memory is now supported. There are new pointer features that allow developers to determine a physical memory location from pointer value. Last, until now, analyzing the execution behavior on GPUs has not been supported. While we care mostly about time to execute a problem, the process of optimizing the performance behavior or designing new algorithms is greatly enhanced by the ability to interrogate the manner in which a process is executed. The new version of CUDA, 4.0, supports automated performance analysis and improved debugging capabilities enabling measurement of instruction and memory throughput features of user applications and kernel occupancy anaysis.

Table 15 reports performance numbers for a strong scaling study of the 32K Cu atom EAM benchmark on the target architecture. Note that the time reaches an apparent minimum between 16 and 352 PEs as the wall time increases slightly between these runs. In each execution event, the numerical values of the observables were consistent to the last reported digit. Table 16 reports the output of the thermodynamic quantities and the total energy. Expert user intuition suggests that the thermodynamic values are correct, and that there is no error in the simulations.

0.2.7 Developments, Enhancements, and Q4 Benchmark Results

We added the Lennard-Jones [110] interaction in Q3 to extend the host + GPU studies we began in Q2. The interaction has the form:

$$E = 4\epsilon[(\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^{6}] \quad (13)$$

where ϵ is the depth of the potential well, σ is the distance where the interparticle potential is zero, and r is the distance between the particles. Note that the inputs required for the Lennard-Jones melt simulation are contained in `in.lj`. This file is distributed with LAMMPS and can be found in the bench sub-folder as is the case for the Q2 inputs, `in.eam`. The Lennard-Jones input decks are reproduced for the reader in the appendix of this document.

PEs=352	INS	FP_OPs	CYCLEs	Time(s)
Intel	962014997534	8209100529	2127997216	.886665
PGI	1007371906713	8578087428	2182934238	.910071

Table 14: Performance Benchmark Results for PGI and Intel builds of the 32K Cu atom EAM potential benchmark problem.

PEs	1 no MPI	1	4	8	16	352
INS	20329365103	20329369096	24915217839	21579319975	25904255039	1007371906713
FP_OPs	8362966560	8363672762	8370073483	8377980036	8382717868	8578087428
T[s]	10.751913	11.728978	3.005760	1.401148	.831466	.910071

Table 15: The benchmark problem is computed repeatedly on an increasing number of processes on the target architecture. The numbers of from the PGI-built executable.

Time Step	$Temp$[K]	E_{pair}[eV]	E_{tot}[eV]	$Pressure$[bars]
0	1600	-113280	-106662.09	18703.573
50	781.69049	-109873.35	-106640.13	52273.088
100	801.832	-109957.3	-106640.77	51322.821

Table 16: The observables computed in each run reported for this benchmark are identical in each execution event up to the number of digits presented here.

The revised Q3 plan was to compare the scaling modes of the `user-cuda` and `gpu` packages being rapidly developed in the LAMMPS library for the EAM and L-J $1/r$ potential computations. In what follows we describe briefly the differences in these efforts and then present measured results for the runs.

Differences between LAMMPS's CPU-based and CPU+GPU-based compute kernels

LAMMPS was originally written to run on a single CPU processor or on multiple CPU processors in parallel using MPI and a spatial-decomposition of the simulation domain. Over the past couple of years, new capabilities have been added to LAMMPS to allow it to harness the compute power of GPUs to accelerate simulations. Faster simulation can be achieved by performing a force decomposition on the GPU threads as well as the usual spatial decomposition using MPI between compute nodes. This dual parallelism allows LAMMPS to harness even more of the available flops on systems that include GPUs.

LAMMPS now includes two optional packages that allow users to harness GPUs. The two packages are called the "GPU" package and the "USER-CUDA" package. The "GPU" package was written by Mike Brown (Oak Ridge National Laboratory), and the "USER-CUDA" package was written primarily by Christian Trott at the University of Technology Ilmenau, Germany.

Both packages can be run on NVIDIA GPUs that are associated with the CPUs of a compute node. Both use CUDA kernels to accelerate the most compute-intensive parts of the LAMMPS simulation by performing those calculations efficiently on the GPUs. Typically, the most compute-intensive parts of the LAMMPS run that are performed on the GPU are the interatomic force calculations and the associated neighbor listing. Both packages provide GPU versions of many pair styles and for long-range Coulombic interactions via the PPPM command. Both accelerate a LAMMPS calculation using GPUs, but they do it in different ways. As a consequence, for a particular simulation on specific hardware, one package may be faster than the other.

The "GPU" package is specifically designed to exploit common GPU hardware configurations where one or more GPUs are coupled with many cores of multi-core CPUs, e.g. within a node of a parallel machine. In the "GPU" package, atom-based data (e.g. coordinates, forces) moves back-and-forth between the CPU(s) and GPU every timestep. Neighbor lists can be constructed on the CPU or on the GPU. The charge assignment and force

interpolation portions of PPPM can be run on the GPU. The FFT portion, which requires MPI communication between processors, runs on the CPU. Asynchronous force computations can be performed simultaneously on the CPU(s) and GPU. The "GPU" package provides NVIDIA support as well as more general OpenCL support, so that the same functionality can eventually be supported on a variety of GPU hardware. Generally speaking, for best performance, users of LAMMPS's "GPU" package should use multiple CPUs per GPU, as provided by most multi-core CPU/GPU configurations. Because of the large number of cores within each GPU device, it may be more efficient to run on fewer processes per GPU when the number of particles per MPI process is small (hundreds of particles); this can be necessary to keep the GPU cores busy. The "GPU" package can be built for single, mixed, or double precision. The latter requires that the available GPU cards support double precision.

The "USER-CUDA" package designed to allow an entire LAMMPS calculation, for many timesteps, to run entirely on the GPU (except for inter-processor MPI communication), so that atom-based data (e.g. coordinates, forces) do not have to move back-and-forth between the CPU and GPU. The speed-up advantage of this approach is typically better when the number of atoms per GPU is large. Data will stay on the GPU until a timestep where a non-GPU-ized fix or compute is invoked. Whenever a non-GPU operation occurs (fix, compute, output), data automatically moves back to the CPU as needed. This may incur a performance penalty, but should otherwise work transparently. Neighbor lists for GPU-ized pair styles are constructed on the GPU. The USER-CUDA package supports the use of only a single CPU (core) with each GPU. This package offers more speed-up relative to CPU performance when the number of atoms per GPU is large, e.g. on the order of tens or hundreds of thousands. As noted above, this package will continue to run a simulation entirely on the GPU(s) (except for inter-processor MPI communication), for multiple timesteps, until a CPU calculation is required, either by a fix or compute that is non-GPU-ized, or until output is performed (thermo or dump snapshot or restart file). The less often this occurs, the faster the simulation will run.

The "GPU" package is often faster than the "USER-CUDA" package in the regime where the number of atoms per GPU is relatively small. The crossover point, in terms of atoms/GPU at which the "USER-CUDA" package becomes faster depends strongly on the pair style. For example, for a simple Lennard-Jones system the crossover (in single precision -and as de-

scribed in the LAMMPS manual) is often about 50,000 to 100,000 atoms per GPU. Both packages compute bonded interactions (bonds, angles, etc) on the CPU. This means a model with bonds will force the "USER-CUDA" package to transfer per-atom data back-and-forth between the CPU and GPU every timestep. If the "GPU" package is running with several MPI processes assigned to one GPU, the cost of computing the bonded interactions is spread across more CPUs and hence the "GPU" package can run faster. When using the "GPU" package with multiple CPUs assigned to one GPU, its performance depends to some extent on high bandwidth between the CPUs and the GPU. Hence its performance is affected if full 16 PCIe lanes are not available for each GPU. In HPC environments this can be the case if S2050/70 servers are used, where two devices generally share one PCIe 2.0 16x slot. Also many multi-GPU mainboards do not provide full 16 lanes to each of the PCIe 2.0 16x slots. The "GPU" package accelerates only pair force, neighbor list, and PPPM calculations. The "USER-CUDA" package currently supports a wider range of pair styles and can also accelerate many fix styles and some compute styles, as well as neighbor list and PPPM calculations. The "GPU" package uses more GPU memory than the "USER-CUDA" package. This is generally not a problem since typical runs are computation-limited rather than memory-limited.

0.2.8 Q4 Benchmark Results

In Q4, we have built the latest `gpu` and `user-cuda` packages in support of two different $\frac{1}{r}$ interactions. The details of these builds are provided in the appendix and the reader is advised to read these sections carefully as they contain relevant corrections to the suggested build process in the documentation available at the time of testing.

Before proceeding, the following information describes the node structure of the 8 processor core + 1 GPU target in Q4:

```
HOST :

processor : 0-7
vendor_id : GenuineIntel
cpu family : 6
model : 26
model name : Intel(R) Xeon(R) CPU           X5550  @ 2.67GHz
```

```
cpu MHz : 2666.852

GPU :

NVIDIA Corporation NVIDIA CUDA
Device 0: "Tesla C2050"
CUDA Driver Version:                           3.20
CUDA Runtime Version:                          3.20

  Type of device:                                GPU
  Compute capability:                            2
  Double precision support:                      Yes
  Total amount of global memory:                 2.62445 GB
  Number of compute units/multiprocessors:       14
  Number of cores:                               448
  Total amount of constant memory:               65536 bytes
  Total amount of local/shared memory per block: 49152 bytes
  Total number of registers available per block: 32768
  Warp size:                                     32
  Maximum number of threads per block:           1024
  Maximum group size (# of threads per block)    1024 x 1024 x 64
  Maximum item sizes (# threads for each dim)    65535 x 65535 x 1
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Clock rate:                                    1.147 GHz
  Concurrent copy and execution:                 Yes
  Run time limit on kernels:                     No
  Integrated:                                    No
  Support host page-locked memory mapping:       Yes
  Compute mode:                                  Default
  Concurrent kernel execution:                   Yes
  Device has ECC support enabled:                Yes
```

USER_CUDA package

To summarize, the `user-cuda` based runs cannot make use of multiple host processes coupled to the GPU through CUDA at this time. As a result, the gains within a node do not alone satisfy the metric. The force evaluation appears to have some scaling bugs in memory that were not fully determined.

It is likely that the Uniform Virtual Addressing of newer (¿4.0) CUDA installations will help eliminate this problem. CUDA 4.0 did exist but only on a small (single node) partition of the target architecture. For instance, a single MPI process computing the 32K Cu atom EAM problem from Q2 achieved 10.1489 s in Q4, and a single MPI process plus the CUDA extension for computing force terms over neighboring pairs of atoms was executed at 1.93222 s. Superficially, the gains look incredible. However, the MPI version of the force calculator can still exploit 7 additional processes on the node. Indeed, the 8 MPI process execution without the GPU achieves 1.35875 s -faster than the GPU enhanced CUDA version.

Perhaps it is the case that there is not sufficient work to do for such a small problem. To test this theory, we increase the number of atoms for which the force was to be computed and started again from a single node. For a single MPI process it takes 2872.59 s to complete the force computation for 1000 time steps of 1 million atoms. For the single MPI process + CUDA version, the same problem executes the main loop in 378.595 s. The performance data is printed below for this run for the curious reader. It is no surprise, but it should be noted that memory copies and thread synchronization are expensive operations.

```
LAMMPS (17 Sep 2011)
# Using LAMMPS_CUDA
USER-CUDA mode is enabled
# CUDA: Activate GPU
# Using device 0: Tesla C2050
Simulating 1000188 atoms for 1000 time steps.
Lattice spacing in x,y,z = 3.615 3.615 3.615
Created orthogonal box = (0 0 0) to (227.745 227.745 227.745)
  1 by 1 by 1 processor grid
Created 1000188 atoms
# CUDA: VerletCuda::setup: Allocate memory on device for maximum of 1010000 atoms.
# CUDA: Using precision: Global: 4 X: 4 V: 4 F: 4 PPPM: 4
Setting up run ...
# CUDA: VerletCuda::setup: Upload data...
# CUDA: Total Device Memory useage post setup: 748.937500 MB
Memory usage per processor = 210.466 Mbytes
Step Temp E_pair E_mol TotEng Press
       0         1600   -3540665.2            0   -3333810.3    18687.277
```

```
WARNING: # CUDA: You asked for a Verlet integration using Cuda, but several fixes
This can cause a severe speed penalty due to frequent data synchronization betwee
      50    780.33209   -3433977.5            0   -3333092.8    52251.235
     100    797.86016   -3436258.5            0   -3333107.7    51421.774
     150    797.24653     -3436181            0   -3333109.5    51478.082
     200    797.43673   -3436204.2            0   -3333108.2    51477.913
     250    797.44708     -3436207            0   -3333109.6    51480.283
     300    798.47405   -3436339.8            0   -3333109.6    51437.185
     350    797.94332   -3436268.8            0   -3333107.2    51465.627
     400    797.88071   -3436260.5            0     -3333107    51478.299
     450    797.30176   -3436184.2            0   -3333105.6    51496.071
     500    798.19117     -3436301            0   -3333107.4    51468.809
     550    797.74382   -3436240.2            0   -3333104.5    51477.474
     600     797.0209   -3436146.2            0   -3333103.9    51509.191
     650    797.24622   -3436174.2            0   -3333102.8    51505.501
     700    797.18333   -3436168.2            0   -3333104.9    51503.579
     750    797.76229   -3436241.5            0   -3333103.3    51481.847
     800     798.0322     -3436277            0     -3333104    51465.718
     850    796.56305   -3436084.8            0   -3333101.6    51525.746
     900    797.58513   -3436218.5            0   -3333103.3    51487.654
     950     797.3577     -3436187            0   -3333101.2    51493.542
    1000    798.29428     -3436309            0   -3333102.1    51454.288
Loop time of 375.115 on 1 procs for 1000 steps with 1000188 atoms

Pair  time (%) = 54.3035 (14.4765)
Neigh time (%) = 32.5784 (8.68492)
Comm  time (%) = 25.6912 (6.8489)
Outpt time (%) = 0.10574 (0.0281886)
Other time (%) = 262.436 (69.9615)

Nlocal:    1.00019e+06 ave 1.00019e+06 max 1.00019e+06 min
Histogram: 1 0 0 0 0 0 0 0 0 0
Nghost:    176122 ave 176122 max 176122 min
Histogram: 1 0 0 0 0 0 0 0 0 0
Neighs:    0 ave 0 max 0 min
Histogram: 1 0 0 0 0 0 0 0 0 0
FullNghs:  7.55114e+07 ave 7.55114e+07 max 7.55114e+07 min
Histogram: 1 0 0 0 0 0 0 0 0 0
```

```
Total # of neighbors = 75511371
Ave neighs/atom = 75.4972
Neighbor list builds = 163
Dangerous builds = 1
# CUDA: Free memory...
##IPM2v0.xx#######################################################
#
# command   : ./lmp_gcc_cu32_sm20 -cuda on
# start     : Tue Sep 27 12:56:15 2011   host      : dirac01
# stop      : Tue Sep 27 13:02:39 2011   wallclock : 383.92
# mpi_tasks : 1 on 1 nodes               %comm     : 0.00
# omp_thrds : 1                          %omp      : 0.00
# files     :                            %i/o      : 0.00
# mem [GB]  : 0.82                       gflop/sec : 0.00
#
#           :       [total]        <avg>          min          max
# wallclock :        383.92       383.92       383.92       383.92
# MPI       :          0.00         0.00         0.00         0.00
# %wall     :
#   MPI     :                       0.00         0.00         0.00
# #calls    :
#   MPI     :           607          607          607          607
# mem [GB]  :          0.82         0.82         0.82         0.82
#
#                              [time]        [count]        <%wall>
# cudaThreadSynchroni          87.79          59672          22.87
# cudaMemcpy                   63.99          44616          16.67
# cudaGetDeviceCount            1.44              1           0.38
# cudaMemcpyToSymbol            0.87          41563           0.23
# cudaLaunch                    0.17          20569           0.04
# cudaSetupArgument             0.06         111206           0.02
# cudaHostAlloc                 0.06              8           0.02
# cudaFreeHost                  0.05              8           0.01
# cudaConfigureCall             0.03          20569           0.01
# cudaFree                      0.02             81           0.01
# cudaMalloc                    0.01            272           0.00
# cudaMemset                    0.01           1183           0.00
```

```
# cudaGetLastError              0.01          21251           0.00
# MPI_Allreduce                 0.00            488           0.00
# cudaStreamCreate              0.00              3           0.00
# cudaSetDevice                 0.00              2           0.00
# cudaGetDeviceProper           0.00              1           0.00
# MPI_Bcast                     0.00            101           0.00
# MPI_Barrier                   0.00              2           0.00
# MPI_Scan                      0.00              1           0.00
# cudaGetDevice                 0.00              1           0.00
# cudaSetDeviceFlags            0.00              1           0.00
# MPI_Comm_size                 0.00              4           0.00
# MPI_Comm_rank                 0.00              9           0.00
# MPI_Init                      0.00              1           0.00
# MPI_Finalize                  0.00              1           0.00
#
```

This again appears to be a considerable win for the hybrid algorithm. However, as in the 32K atom force computation, executing the 1 million atom problem with 8 MPI processes on the node yields 412.402 s for completion and the win for the host + GPU case plus CUDA enhancements is only nominal. This trend persists regardless of how the problem is weakly scaled within the node. In fact, at 8 million atoms, the host + CUDA version failed to execute for lack of memory, whereas the problem can be executed with the MPI only code on the node in 3344.07 s -slightly less than an hour.

The next scaling mode we wanted to test was the strong scaling mode for a fixed problem. Consider the following screen shot of such a study over 1 million atoms on the target architecture:

```
1M ATOMS:
MPI-only version of the force computation:
Loop time of 2872.59 on 1 procs for 1000 steps with 1000188 atoms (1p1n)
Loop time of 764.202 on 4 procs for 1000 steps with 1000188 atoms (4p1n)
Loop time of 755.475 on 4 procs for 1000 steps with 1000188 atoms (4p4n)
Loop time of 412.402 on 8 procs for 1000 steps with 1000188 atoms (8p1n)
Loop time of 389.187 on 8 procs for 1000 steps with 1000188 atoms (8p8n)
Loop time of 161.931 on 20 procs for 1000 steps with 1000188 atoms (20p,20n)
Loop time of 110.603 on 30 procs for 1000 steps with 1000188 atoms (30p,30n)
Loop time of 102.942 on 32 procs for 1000 steps with 1000188 atoms (32p4n)
```

```
Loop time of 50.1122 on 64 procs for 1000 steps with 1000188 atoms (64p8n)
Loop time of 20.6911 on 160 procs for 1000 steps with 1000188 atoms (160p,20n)
Loop time of 14.1628 on 240 procs for 1000 steps with 1000188 atoms (240p30n)

MPI + CUDA 3.2 version of the force computation:
Loop time of 378.595 on 1 procs for 1000 steps with 1000188 atoms
Loop time of 125.999 on 4 procs for 1000 steps with 1000188 atoms
Loop time of 72.6575 on 8 procs for 1000 steps with 1000188 atoms
Loop time of 34.5002 on 20 procs for 1000 steps with 1000188 atoms
Loop time of 26.6687 on 30 procs for 1000 steps with 1000188 atoms
```

We notice that after 4 nodes, the MPI only implementation can always win in strong scaling mode by utilizing the full set of processor cores for segments of the force computation over pairs of neighboring atoms in the system.

GPU package

For the `gpu` package, atom-specific data that describes coordinates and force terms are moved between the host processors and the GPU every timestep. The neighbor lists can be constructed on the host or the GPU. In distinction to the `user-cuda` package, this package was written to explicitly utilize NVIDIA instructions on the GPU and was designed to allow very clever scaling modes so that entire force computations may be passed to the GPU or a fractional split may be defined such that the GPU computes only part of the force computation over neighboring pairs of atoms. Further, the ability to allow multiple MPI processes to share a GPU allows for important run modes to be decided by the informed user. Here, we present results from 3 studies that reveal the strong and weak scaling modes, as well as the throttle ability to fractionally assign work to the GPU.

Figure 15 depicts the strong scaling. The Q4 algorithm out performs the best Q2 algorithm cleanly. Both algorithms were still minimizing the wall-time when the test completed because the amount of available hardware was exhausted. Thus, 240 PEs versus 240 PEs + 30 GPUs the gain was 5.80736 s / 2.86258 s = 2.028X speed-up.

Figure 16 depicts the weak scaling comparison of the two algorithms. The Q4 algorithm out performs the best Q2 algorithm cleanly. Both algorithms exhibit outstanding weak scaling at the processor counts tested, 8, 16, 64, 128 PEs. However, the hybrid algorithm is always faster. Table 17 depicts the raw measured data. For instance, the 1 million (1,000,188 atoms to be exact) test case executes over 4X faster than the corresponding MPI algorithm on

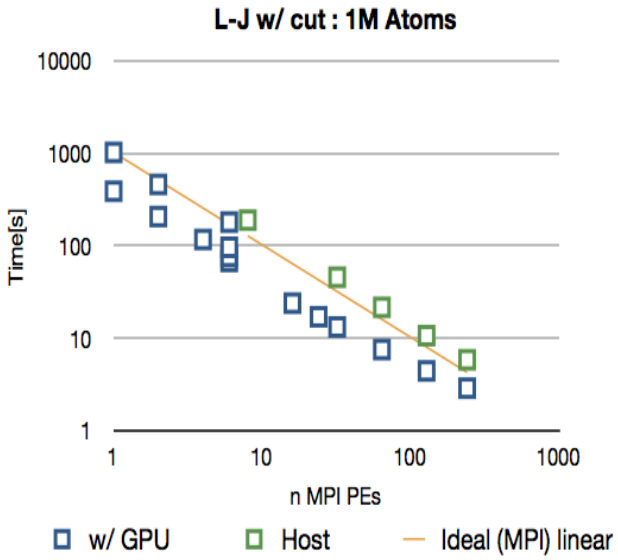

Figure 15: The best MPI runtimes to compute the Lennard-Jones potential for 1 million atoms and increasing process count are compared to the best hybrid MPI + GPU version of the force computation. The ideal strong scaling line is based on the best theoretical MPI strong scaling case. It is provided as a basis for comparison.

Atoms	Time[s] Hybrid	Time[s] MPI only	PEs
500000	23.6967	94.3719	8
1000188	23.8014	96.8414	16
4000000	24.3787	100.255	64
8001504	23.6476	101.092	128

Table 17: Raw Data for weak scaling Q4 force computation study.

the same PE count.

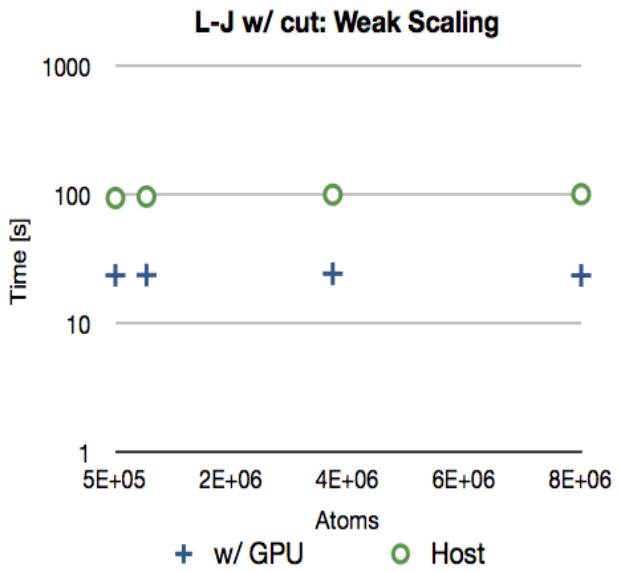

Figure 16: The best MPI runtimes to compute the Lennard-Jones potential for increasing numbers of atoms on a correspondingly increasing process count are compared to the best hybrid MPI + GPU version of the force computation.

Last, Figure 17 demonstrates the times affiliated with different fractions of the problem being assigned to the GPU for a fixed problem on a single node. Some additional raw data affiliated with the runs is printed at the bottom of this section in Table 18.

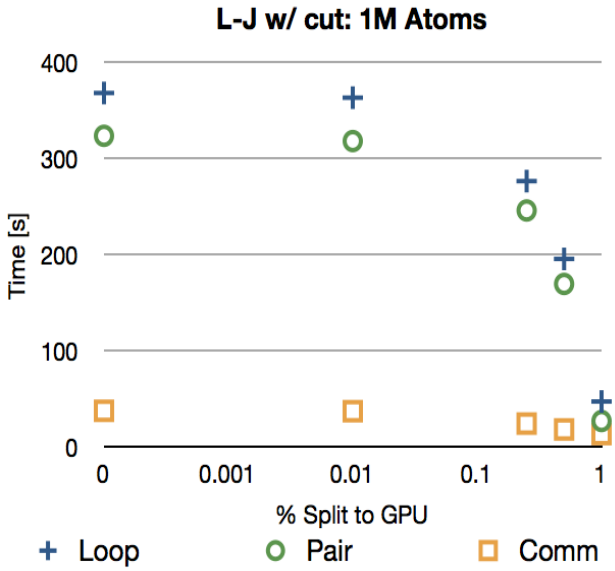

Figure 17: The raw data for this plot is found in Table 18. It is noted that the performance for this single node, fixed 1 M atom problem improves with increasing the amount of work executed by the GPU.

Frac. Split	Loop Time[s]	Pair Time[s]	Comm Time[s]
0.0001	368.205	323.681	37.7386
0.01	363.327	318.209	37.3178
0.25	276.59	246.052	24.4422
0.5	195.679	169.656	18.3587
1	47.4061	27.1221	13.5273

Table 18: Raw Data for a Q4 fixed 1M atom force computation study on a single node of the target architecture. The fraction of the pair computation executed on the GPU is varied demonstrating improved performance as a larger fraction of the problem is assigned to the GPU.

0.3 OSIRIS: Plasma based acceleration

0.3.1 Introduction

Particle accelerators are critical to scientific discovery in the United States Department of Energy (DOE) program and indeed the world. Of the 28 facilities listed in the 2003 DOE report Facilities for the Future of Science: A Twenty-Year Outlook, 14 involve accelerators. The development and optimization of accelerators are essential for advancing our understanding of the fundamental properties of matter, energy, space, and time, and for enabling research in aspects of materials science, chemistry, geosciences, and bioscience. Modeling of accelerator components and simulation of beam dynamics are necessary for understanding and optimizing the performance of existing accelerators and for optimizing the design and cost effectiveness of future accelerators.

In the next decade, the High Energy Physics (HEP) community will explore the energy frontier by operating and upgrading the Large Hadron Collider (LHC). It will also explore novel concepts and technologies necessary for the design of the next lepton collider with capabilities beyond the LHC. This includes the construction and operation of new facilities such as BELLA and FACET at which R&D for these new acceleration technologies will be undertaken. It will also be exploring the intensity frontier by designing high intensity proton sources for neutrino physics and the rare process program uses accelerators to study the properties of nuclear matter and the structure of the nucleus, understand the mechanism of quark confinement, and create and study the quark-gluon plasma state. The flagship DOE Nuclear Physics (NP) accelerators are CEBAF at Jefferson Lab and RHIC at Brookhaven National Laboratory (BNL). The worldwide nuclear physics community has identified the construction of a polarized electron-ion collider as a long-term objective in the ongoing effort to understand and answer questions in these areas. DOE/NP is considering two approaches for such a facility: e-RHIC and ELIC. DOE/NP is also constructing a facility for rare isotope beams (FRIB) that will permit studies of nuclei far from stability that promise to radically improve our understanding of atomic nuclei.

0.3.2 Background and Motivation

Today's accelerators are complex, large, and expensive. For example, the LHC is 30 kilometers in circumference and cost over 10 billion dollars to build. To reduce this cost, new accelerator technology is needed. Computer simulations have the potential to greatly reduce the time and cost it takes from the conceptual design of an accelerator to operation. One technology that is receiving much attention is called plasma-based acceleration . In plasma based acceleration a short and intense laser or relativistic particle beam (the driver) propagates through a plasma near the speed of light. The light pressure of the laser or the space charge forces from the particle beam displaces plasma electrons. The ions then pull the electrons back towards where they started thereby creating a plasma wave wake with a phase velocity near the speed of light. Schematics of both are shown in Figure 1. The accelerating (electric) field in these wakes are more than 1000 times higher than those in existing accelerators. Properly shaped and phased electrons or positron beams (witness beams) are loaded onto the wake and they surf to ultra-high energies in very short distances. Recently, experiments using a laser driver have demonstrated the feasibility of generating GeV class quasi-monoenergetic beams. Experiments using a electron beam driver have demonstrated gradients in 50GeV/m and wake excitation over pump depletion distances. To accurately simulate this process one needs to model how a short and intense driver evolves over large distances, how the wake is excited and how it evolves, and how the properties of the witness beams evolve as they are accelerated. In many cases the excitation of the wake is nonlinear. The fields within the wakestructure are electromagnetic (not electrostatic) in nature so a full electromagnetic treatment is needed. The trajectories of individual electrons can cross each other and the oscillation energy of an electron in the wake can reach and exceed its rest mass. This makes purely theoretical descriptions difficult and it necessitates the use of simulation tools that track the motion of individual particles (a fully kinetic description) and that solve for the relativistically correct equations of motion. The leading kinetic description is the particle-in-cell (PIC) method.

0.3.3 Capability Overview

There is now a well established hierarchy of full kinetic methods for modeling plasma based acceleration (many developed within SciDAC). The most

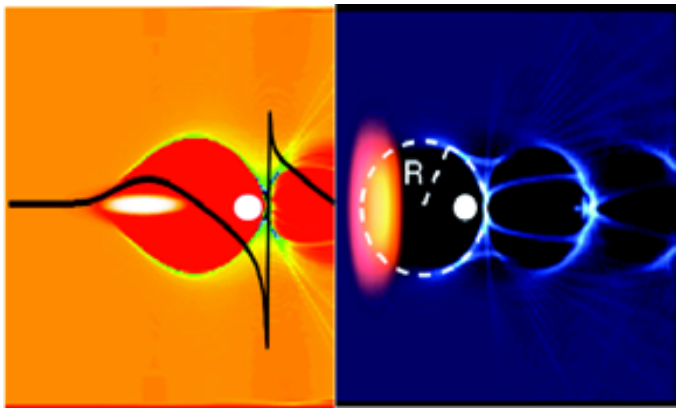

Figure 18: On the left is an electron beam (white) moving from right to left. It forms a wakefield (density of plasma is shown. A lineout of the accelerating field is shown in black. A trailing bunch is shown in white in the back of the wakefield. On the right a laser (orange) is moving from right to left. It also creates a wakefield. The wakefield in both cases is a moving bubble of a radius R. A trailing beam is shown in white as well.

'complete' method is the fully explicit, full Maxwell solver, fully relativistic, PIC method. In this method, a large number of macro-particles are initialized with positions within a grid and momenta. Each timestep, the currents (and in some cases the charge density) are deposited onto the appropriate corners or segments of the grid. Maxwells equations are then advanced forward to calculate new electric and magnetic fields. This is most commonly done using the finite difference time domain (FDTD) method for which the fields need to be appropriately centered with respect to the cell (some are located at corners, some at the midplane of the cell boundaries, while others are defined in the middle of the cell) to achieve second order accuracy. This is commonly done using the Yee mesh. The fields can then be used to calculate the force on each particle so that their momentum and position can be advanced forward to the next time step after which the current can be determined and the whole sequence can be repeated. This loop can then be repeated the desired number of timesteps. The cell size is chosen to resolve the smallest scale length of physical relevance and the time step is then determined from the Courant condition. The shortest length of relevance is usually the laser wavelength for a laser driver and the wakes wavelength for a particle beam driver. In some cases the charged particles can radiate x-rays

as they are accelerated. The wavelength of the x-rays is far too short to be resolved, and can be handled by secondary algorithms. While this algorithm is straightforward to explain, production codes are now very complex due to parallelization, single core optimization, various choices for how to deposit the current, higher order particle shapes, complex boundary conditions, diagnostics, and physics packages required to realistically model phenomena of interest. The details of how these are done in OSIRIS will be given later.

The next method within the hierarchy takes advantage of the fact that a short pulse laser does not backscatter. In this limit, the laser pulse can be separated out into a central frequency (corresponding to a forward moving wave) and an envelope which is a complex number describing the pulse shape and length. Only the evolution of the envelope (both amplitude and phase) for the laser is calculated and it is explicitly assumed that the time scale that the laser evolves over is much longer than a laser period (in fact longer even than a plasma wake period). Therefore, the time steps and grids used to advance the laser envelope can now be larger than the laser period and wavelength. The particles still oscillate at the laser frequency and wavenumber. However, one can further separate out the fields and forces into plasma wakefields and laser fields and then average over the laser frequency and wavenumber (assuming there is no backscatter). The result is that the force on the particles has the usual Lorentz force from the plasma wakefields and a ponderomotive or light pressure force from the laser. This is idea is called the ponderomotive guiding center (PGC) approximation . The potential savings is very large and they scale as the output energy of the accelerator section. One potential issue is that as the laser propagates deeply into the plasma it pump depletes. However, the PGC averaging procedure breaks down for very intense pump strengths and the envelope approximation can breakdown for long propagation distances.

Another method that takes advantage of the fact that there is no backscatter was recently proposed by Vay. In this method, one does the simulation is a Lorentz boosted frame. In a Lorentz frame moving at a speed near the speed of light with the laser, the laser will appear Lorentz expanded (by a factor $(1 + v_f/c)\gamma_f$ where $\gamma_f = (1 - v_f^2/c^2)^{-1/2}$ and v_f is the velocity of the frame and c is the speed of light. The plasma (now moving towards you) is now Lorentz contracted (by a factor γ_f). In this frame the shortest wavelength is still the laser wavelength (so long as there is not light reflected as this light will have a very short wavelength in this frame) which has also been Lorentz expanded, and the number of wavelengths within the laser is a

Lorentz invariant, i.e., it is the same in every reference frame. So although the laser is longer the number of cells required to resolve it does not change and the number of cells used for the simulation does not change. In physical units the time step is now larger (by $(1 + v_f/c)\gamma_f$). The required number of time steps is therefore smaller by $(1 + v_f/c)^2\gamma_f^2$ (a factor $(1 + v/c)$ arises because the pulse and plasma are moving towards each other). The optimal frame to transform into is one moving near the group velocity of light within the plasma, $v_g = (1 - \omega_p^2/\omega_0^2)^{1/2}$. In this frame $\gamma_f = \gamma_g = \omega_0/\omega_p$ such that the speed up once again scales as $(\omega_0/\omega_p)^2$. And just like in the PGC method, the speed up scales at the output energy of the accelerator section. This technique is attractive because in principal it should work for full pump depletion distances and for arbitrary pump strengths. However, one must use smaller ?f when investigating self-trapping which occurs when the wake gets so large that background plasma electrons become self-injected into the wake and are accelerated. This is because the plasma is represented by ?f fewer particles which can lead to a statistical issue.

The last method in the hierarchy is the quasi-static approximation (which is used together with the PGC when using a laser driver). The quasi-static approximation exploits the fact that during the time it takes the laser or particle beam driver to pass by a plasma electron the laser and particle beam appear static, i.e., neither their shape nor, their energy distribution (for a laser, this the frequency content) change. Under this approximation, one can calculate the wake for a given driver shape and then use the potentials and fields within this wake to get source terms to advance the laser or particle beam forward with time steps that resolve the time scale that the driver evolves over. These time scales are much larger than a plasma period. A laser driver typically evolves over a Rayleigh length, $k_0 w_0^2/2$ while for a particle beam evolves over the betatron period $k_p/\sqrt{2\gamma}$ (the period that particles oscillate transversely in the wakefield). For a laser driver the speed up in theory could therefore be the product of a factor of $(\omega_0/\omega_p)^2$ from the use of PGC and an additional factor of $((\omega_0/\omega_p)k_p^2 w_0^2/2$, i.e., $(\omega_0/\omega_p)^3 k_p^2 w_0^2/2$ from the quasi-static approximation. However, accuracy and numerical stability issues keep the time step for the laser solver to $(\omega_0/\omega_p)k_p\Delta$ where Δ is the transverse cell size. For a particle beam driver the speed up scales as $(2\gamma)^{1/2}$. Furthermore, the algorithm requires a predictor corrector loop so the field solve and particle push has additional overhead. This technique has already been used to verify some scaling laws out to 100 GeV. An advantage of the quasi-static model is that it is not susceptible to numerical

Cerenkov radiation so smaller transverse cell sizes can be used for narrow trailing bunches. In addition, mesh refinement should be easier to implement in principle. Calculating radiation reaction is also more straightforward. At present no prescription for handling self-trapping is available when making the quasi-static approximation.

As the above indicates there is not a single code or algorithm that fits all for modeling plasma-based acceleration. Currently, there are a suite of production codes that have been supported under the advanced accelerator part of SciDAC, OSIRIS, VORPAL , and QuickPIC. In addition, another code Warp has been supported for electron cloud work. It has recently been used to develop the boosted frame technique and is starting to be used by some in SciDAC for LWFA simulations. OSIRIS is a full PIC code that also can run using the boosted frame technique primarily to study self-injection. VORPAL is a full PIC code that also can operate with the PGC, with cut cells for complicated boundaries, and is currently being updated to run using the boosted frame. Warp has been used to run laser wakefield simulations using the boosted frame technique (for very high output energies in the external injection and weak beam loading regime). QuickPIC is a 3d quasi-static code which is used for both laser and particle beam drivers.

The above codes have been compared against each other for short test problems. However, they are not identical and achieve different speeds and different parallel scalability. For this project the OSIRIS codes will be used which has achieved the highest parallel scalability. The most recent features added to OSIRIS include dynamic load balancing, higher order particle splines which when combined with current smoothing and compensation for improved energy conservation, a diagnostic which accumulates and tracks the trajectories for a pre-selected group of particles, and perfectly matched layers for transmitting light out of the simulation domain. In addition, new physics packages beyond that in standard PIC have been added. These include tunnel and impact ionization as well as a relativistically correct binary collision operator. Very recently, the ability to carry out simulations in a Lorentz boosted frame was implemented.

OSIRIS is highly optimized on a single processor while maintaining high parallel efficiency, i.e., $> 80\%$ for strong scaling (fixed problem size) on nearly 300k processors of the Jülich BlueGene Jugene Computer. Specifically, for the strong scaling study at the processor count there was 1820 cells and 58000 particles on each processor. OSIRIS has also demonstrated very good scaling on a variety of other platforms and supercomputers, including the Cray XT4

and XT5 at Oak Ridge. Specifically, on Jaguar it has 61% efficiency on a strong scaling study to 128000 cores with 4000 cells and 64000 particles on each core at the end. The key figure of merit for a PIC code is how many particles per cell can be advanced per time step. Using quadratic interpolation, OSIRIS achieves 570 ns particle for an entire time step (including the field solve, particle push, and current deposition) using 16 particles per cell on Jaguar.

0.3.4 Science Driver for Metric Problem

Understanding and controlling the injection and acceleration of self-trapped electrons is a grand challenge in plasma-based acceleration research. Research in plasma-based acceleration has made incredible progress. Experiments and simulations demonstrated in 2004 the formation of the first quasi-monoenergetic energy and narrow divergence beams from laser plasma accelerators and simulations together with theory have shown how intense lasers can lead to self-injection and mono-energetic beams in nonlinear wake (bubble) regimes. In addition, experiments and simulations of beam driven wakefields have shown that wakes with gradients over 50 GeV/m can be excited over distances where high efficiencies can be achieved.

However, the requirements of the 'quality' of the output beam for high-energy physics and for next generation light sources are very difficult to achieve. Next generation experiments and current simulation efforts are aimed at developing schemes for improving both the energy spread and the angular divergence (emittance) of beams produced by laser or particle beam drivers and for accelerating positrons. A further challenge is determining simple methods for injecting beams of particles into the wakefields. This is particularly challenging when using a laser since one needs to synchronize the laser and the injected particles. Simulations play an invaluable tool for this problem as they can unequivocally demonstrate where injected electrons come from and they can use plasma and laser parameters that are not yet feasible to produce in the laboratory. For example by tracking particles in the simulations it was shown that the injected electrons originate roughly one laser spot size off axis and that the very strong fields of the nonlinear wake lead to a large transverse emittance, i.e., unacceptable angular divergence. In addition, new ideas such as using density ramps or ionization in separate injection sections are being explored.

0.3.5 The Model Algorithm

Particle-in-Cell (PIC) codes model plasmas as particles which interact self-consistently via the electromagnetic fields they themselves produce. These models work at the most fundamental, microscopic level. As a result, they are the most compute intensive model in plasma physics. PIC codes are used in almost all areas of plasma physics, such as fusion energy research, plasma accelerators, space physics, ion propulsion, and plasma processing, and many other areas. PIC algorithms are also used in cosmology, astrophysics, accelerator physics, and semi-classical quantum simulations. Unlike molecular dynamics codes widely used in chemistry, where particles interact as binary pairs, particles in PIC codes interact via fields calculated on a grid. PIC codes are possible whenever there is some differential equation which describes the fields in terms of particle sources. There are a variety of PIC codes in common use, differentiated by the kinds of forces retained in the model. The simplest is the electrostatic force, described by a Poisson equation. More complex are the Darwin (non-radiative electromagnetics) and the fully electromagnetic models. Because PIC codes contain the richest physics, they are increasingly being used to validate reduced plasma descriptions, such as fluid models. PIC algorithms are sometimes used in conjunction with molecular dynamic codes, to model the electrostatic part of a more complex binary interaction. PIC codes generally have three important procedures in the main iteration loop. The first is the deposit, where some particle quantity, such as a charge, is accumulated on a grid via interpolation to produce a source density. Various other quantities can also be deposited, such as current densities, depending on the model. The second important procedure is the field solver, which solves Maxwells equations or a subset to obtain the electric and/or magnetic fields from the source densities. Finally, once the fields are obtained, the particle forces are found by interpolation from the grid, and the particle coordinates are updated, using Newtons second law and the Lorentz force. The particle processing parts dominate over the field solving parts in a typical PIC application.

The OSIRIS framework is a state-of-the-art, fully explicit, massively parallel, electromagnetic PIC code. The code contains algorithms for simulations in 1D, 2D and 3D cartesian, and 2D azimuthally symmetric cylindrical coordinates. PIC models work at the most fundamental, microscopic level and are therefore the most compute intensive model in plasma physics, so the code was designed from scratch to be massively parallel. The code has been de-

ployed in many HPC systems, with many diverse architectures/interconnects/IO systems worldwide, from small clusters to very large systems.

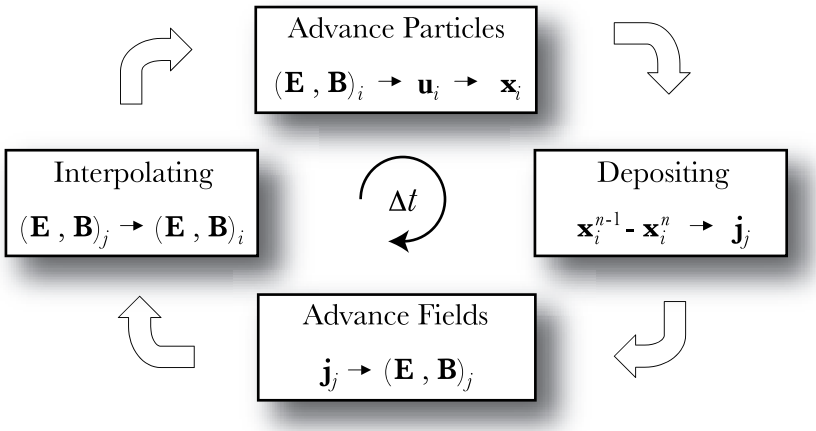

Figure 19: OSIRIS: EM-PIC loop.

After setting the initial conditions for the problem being simulated, the OSIRIS main simulation loop follows the standard electromagnetic particle-in-cell paradigm, as described in figure 19. In this loop particles are advanced using the Lorentz force resulting from the self-consistent electromagnetic field in the simulation. These fields are defined on a grid, and need to be interpolated at the particle positions using some form of weighing function, effectively connecting grid and particles quantities. The simplest method to achieve this is the so called nearest grid interpolation method, or zero order weighing, where the field values at the grid point nearest to the particle position are used to determine the force acting on the particle. However, as the particle passes through cell boundaries, the jumps in field quantities will result in a relatively noisy both in space and in time. To overcome this higher order interpolation schemes must used that combine the contributions of multiple grid points using a smooth weighing function. Traditionally PIC codes use linear interpolation, that combines the values of the two nearest grid points in each dimension, which is acceptable for many simulation problems. Higher order splines can be used for lowering the noise, with the tradeoff of requiring additional calculations for each particle. OSIRIS implements 1st to 4th order interpolation schemes (linear, quadratic, cubic and quartic splines) which can be selected at runtime, offering the user the option to choose the

best trade-off between noise level and computation cost for the problem at hand.

Regardless of the interpolation level in use, field interpolation calculations require knowledge of the grid point index closest to the particle position, and the distance between the particle and the grid point, normalized to the cell size. The standard approach is to store the particle positions a floating point quantity relative to some chosen reference point, and to determine the required quantities when required. However this poses two problems: the calculation of the cell index may represent an important computational cost, as it involves a real to integer conversion and more importantly results in a lower numerical precision being used for particle interpolation calculations when these are further away from the chosen reference point, which in turn leads to additional noise in the simulation. To overcome these issues, improving global numerical accuracy and insuring uniform noise properties across the whole simulation box, OSIRIS stores particle positions as a set of two values, the first being an integer indicating the cell the particle is in, and the second the position of the particle inside the cell.

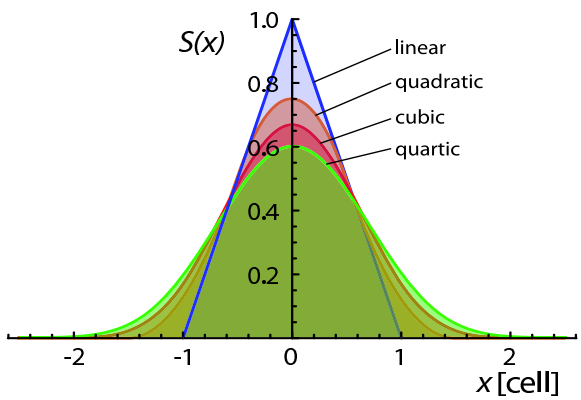

Figure 20: Particle interpolation shapes.

Particle momenta and positions are then advanced in time through a leap-frog scheme, which provides 2nd order accuracy in time. This requires that particles positions and momenta are not time-centered, with particle positions being known at time t, and particle momenta being known at $t-\frac{\Delta t}{2}$. As a result, field quantities will also have to be known at time t, and the values interpolated at particle positions will be used to calculate the Lorentz force and advance particle momenta from $t-\frac{\Delta t}{2}$ to $t+\frac{\Delta t}{2}$. OSIRIS implements

	linear	quadratic	cubic	quartic
S_{-2}				$\frac{1}{384}(1-2x)^4$
S_{-1}		$\frac{1}{8}(1-2x)^2$	$\frac{1}{6}(1-x)^3$	$\frac{1}{96}\left(-16x^4+16x^3+24x^2-44x+19\right)$
S_0	$1-x$	$\frac{3}{4}-x^2$	$\frac{1}{6}\left(3x^3-6x^2+4\right)$	$\frac{1}{4}x^4-\frac{5}{8}x^2+\frac{115}{192}$
S_1	x	$\frac{1}{8}(1+2x)^2$	$\frac{1}{6}\left(-3x^3+3x^2+3x+1\right)$	$\frac{1}{96}\left(-16x^4-16x^3+24x^2+44x+19\right)$
S_2			$\frac{1}{6}x^3$	$\frac{1}{384}(1+2x)^4$

Table 19: Particle interpolation functions. S_i is the interpolation weight for cell i with respect to the particle cell and position x is normalized to cell size.

this advance using a relativistic Boris pusher. This method, considers the effects of the electric and magnetic forces separately, splitting the particle advance into electric field impulses and magnetic rotations. This allows for easy calculation of the time centered relativistic gamma factor, that is not affected by the magnetic rotation but is required to calculate the magnetic force, and results in a second order accurate momentum integration. After completing this integration, the momentum is known at time $t+\frac{\Delta t}{2}$, which is used to advance particle positions in time from t to $t+\Delta t$.

After advancing particle quantities, the resulting current density is calculated on the grid at time $t+\frac{\Delta t}{2}$ to self-consistently evolve the electromagnetic field, as described below. This requires the current to be deposited on the grid using the desired interpolation order which should be consistent with the interpolation scheme used to calculate the fields for the forces, otherwise non-physical self-forces can arise. This step will connect particle and grid quantities, closing the loop on particle/grid interaction. The current density deposition scheme must also be chosen to match the electromagnetic field solver, and specifically to obey the charge continuity equation on the grid, $\vec{\nabla}\cdot\vec{j}=-d\rho/dt$, where $\vec{\nabla}\cdot\vec{j}$ is calculated using the same finite difference grid differencing operator used for the EM field-solver, and ρ is computed by depositing the charge using the interpolation scheme mentioned above. To achieve exact charge conservation, OSIRIS follows the current deposition scheme of Villasenor and Buneman. In these schemes the deposited current density is calculated from the continuity equation by analyzing the variation of the deposited particle charge on the grid due to particle motion in this time step, achieving charge conservation to numerical precision. Esirkepov's work extends the previous Villasenor technique by deriving a general method that can be applied to any particle weighing function. OSIRIS implements

a combination of both methods, for interpolation weights from 1st up to 4th order, that splits particle trajectories in segments lying inside the same cell, and uses analytical solutions for accurate current deposition of these segments.

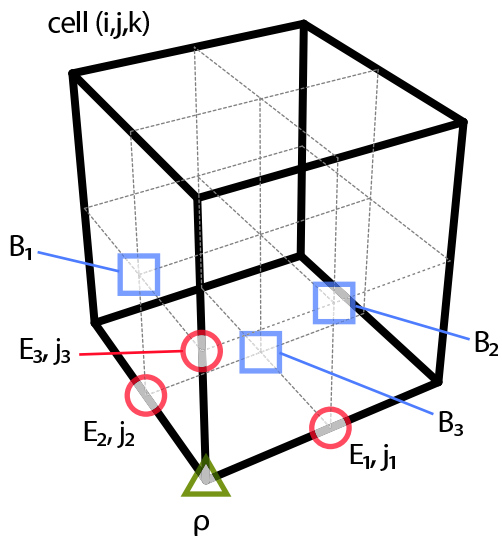

Figure 21: Position of grid quantities inside simulation cell.

Finally, closing the simulation loop, the field quantities are then advanced from time t to time $t + \Delta t$ using a local Maxwell solver, that is equivalent to the Finite Difference Time Domain (FDTD) method using a Yee mesh, adding the contribution of the plasma currents to the electrical field advance equation. As stated above, this method approximates the spatial derivatives using finite differences. Electric and magnetic quantities are defined in different positions inside the simulation cell (see figure 21) allowing for second order accuracy in space for this operator. The standard FDTD method requires that the electric and magnetic field to be also non-time centered. However, as we saw above, this is a requirement for the particle push. To overcome this issue, we follow the method proposed by Boris that splits the magnetic field integration into 2 steps. OSIRIS first advances the magnetic field from time t to time $t + \frac{\Delta t}{2}$ using the electric field values at time t, then performs a standard electric field advance from t to $t + \Delta t$ using the magnetic field and current density defined at time $t + \frac{\Delta t}{2}$, and finally advances the magnetic field from $t + \frac{\Delta t}{2}$ to $t + \Delta t$. This method has been demonstrated to have the same accuracy as the FDTD method, while allowing the electric

and magnetic fields to be time centered, and not requiring any additional storage for the magnetic field.

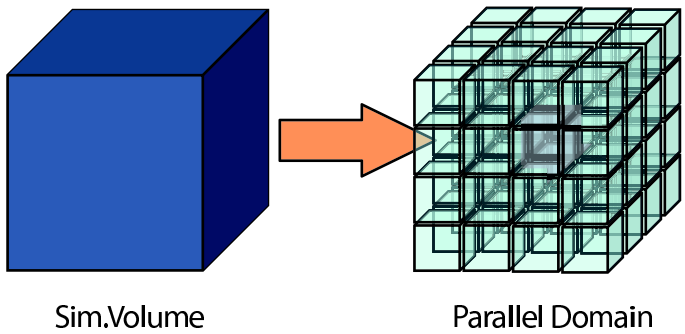

Figure 22: OSIRIS: Parallel domain decomposition.

Given the detail required for realistic simulations, particle models are very demanding in terms of computational resources. With this in mind OSIRIS was designed from scratch to be used in massively parallel computer systems. Particle-in-cell codes are good candidates for parallelization since the particle calculations are inherently local, requiring only information close to the particle position on the grid. The field solver chosen for OSIRIS is also a local field solver, which requires less parallel communication than a global field solver such as a spectral solver. With this in mind OSIRIS splits the problem across multiple cpus using a spatial domain decomposition that is implemented using MPI, as shown in figure 22.

In this parallel partition each node will handle a fraction of the global simulation grid and the particles that occupy that region of simulation space. The simulation grids use guard cells that hold values belonging to neighboring nodes, that are updated only once per iteration. These guard cells must be enough to accommodate both the needs of the field solver and the particle interpolation scheme. Whenever possible multiple grid quantities are packed into a single message, to minimize the number of messages sent. After advancing particles, these are checked to see if they have crossed the node boundary, and if so, they are sent to the appropriate neighboring node. This poses an additional difficulty when compared with the parallelization of the grid quantities, given that the number of particles being received by any given node is not known ahead of time. This process generally involves a set of 2 messages: the first informs the receiving node of how many particles it needs to expect, and the second holds all particle data, if any. In OSIRIS

communication is processed one dimension at a time, first along x, then y, then z. This allows communication with only immediate neighbors (six in 3D), that greatly simplifies code structure and minimizes communication. The total amount of data being communicated at every time step is relatively small when compared to the simulation volume inside a given node, and this data will only be communicated once at every time step. This means that communication itself presents a very low overhead in the parallelization efficiency, and the biggest challenge is to maintain an even load (number of particles) across multiple nodes. This parallelization strategy has proven to be very effective, and OSIRIS simulations are now routinely performed in the Jugene system, currently the largest in the world in number of cores, and strong scaling tests at the recent Extreme Scaling Workshop in Julich (2009) have demonstrated very good scaling up to ~ 300 thousand cores.

0.3.6 Q2 Baseline Problem

Modeling self-injection and acceleration to GeV or higher energies in nonlinear wakefields require full particle-in-cell simulations as well as large computational resources. In addition, controlled self-injection including ionization can also make the use of the boosted frame technique more challenging due to a limited number of simulation particles, and it can lead to significant load imbalance. While there are various self-injection scenarios including using intense lasers in a self-guided regime, using field-ionization, by the laser, using density gradients, and using multiple lasers, from a high performance computing perspective their is much overlap in the simulation requirements to study these scenarios. Therefore, the high performance computing lessons learned from studying one injection method will apply to other injection methods.

Simulations of self-injection LWFA are extremely demanding and push the limits of running particle-in-cell codes on the most advanced HPC hardware. They are usually performed at the lower limits of resolution and number of particles per cell, without the inclusion of ion effects. However, the dependence of the simulation observables such as the charge, slice energy spectra and emittance of the self-injected bunches that exist in multiple "buckets" of the plasma wave wake on numerical parameters, such as higher order particle shape, box size, and cell size, and also physical parameters such as laser pulse length, laser spot size, spot size asymmetries, plasma density and ion motion, is not fully known. Another critical issue in performing these simulations on

10000 or more PEs is that of parallel load balance. The self-injection mechanism itself places particles into a local region of real space. This leads to a severe accumulation of particles in a very narrow region at the back of each plasma wavelength. If a large number of PEs is used and the region where particles accumulate resides on a very small number of the total number of PEs then a severe load balance will arise. Furthermore, the position of these particles in the simulation moving window will not remain constant so that any redistribution of particles across the PEs will have to be done regularly during the simulation.

To study these effects, we chose a set of 4 problems as the Q2 baseline. The first problem models a uniform warm plasma with a temperature distribution parameter of $u_{\mathrm{th}} = 0.01$c with the goal of measuring code performance under a perfectly load balanced simulation. In this scenario, particle diffusion across parallel nodes happens uniformly so the total number of particles per node remains approximately constant. This is also a good overall performance test, as these plasma conditions will resemble those on most of the simulation box for the laser wakefield runs.

For the laser wakefield scenarios we chose 3 problems. Two of them correspond to the interaction of a 200 TW (6 Joule) laser interacting with uniform plasma with a density of $1.5 \times 10^{18}\mathrm{cm}^{-3}$ plasma with an intensity sufficient to trigger self-injection, under different numerical and physical conditions. We chose different grid resolutions, different number of particles per cell, and mobile/immobile ions. The third LWFA run corresponds to a PW (30J) laser propagating in a $.5 \times 10^{18}\mathrm{cm}^{-3}$, where ion motion is expected to play an important role. These 3 Q2 baseline simulations will give us detailed information on the relevant numerical and physical effects that will be useful for future studies of other scenarios. Importantly, they will provide information regarding the degree of load balance that arises.

We use quadratic shaped particles for the current deposition and field interpolation for all the simulations, as this gives the best tradeoff between reducing aliasing at the expense of higher floating point counts for these problems. Finally the simulations were stopped before the complete details of the observables could be measured in order to fit in a 1 million cpu hour limit. However, each simulation ran long enough for key physical processes (such as self-injection) to occur and to capture the relevant flow of the computation. These simulations will be completed later to obtain all the physics results. Table 20 shows a summary of the most relevant simulation parameters for the 4 simulations.

The warm plasma test performed quite well yielding a global performance of 7.62×10^{10} particle pushes per second, which gives an average of 1.38 million particles pushes per second per PE (for quadratic interpolation). The performance of the code did not fluctuate over the ~ 10 wall clock hours that the simulation ran and there was no measurable performance degradation. The average load imbalance, measured as the ratio between the maximum number of particles in a PE over the average number of particles in a PE was 1.00011, effectively maintaining a perfect load balance over the full course of the simulation, as expected. The performance measurement therefore sets a baseline for comparison with other scenarios where the load imbalance will hinder code performance. Table 20 shows a summary of the performance measurements for all the simulations. The measured machine events for the warm test and run 1 of the LWFA simulations are reported in Table 22

Figure 23 shows the results for one the LWFA simulations (Run 2), illustrating the main physical aspects of this scenario: the driver laser pulse, shown in the right of the plot in blue and red isosurfaces drives a high amplitude plasma wave, shown in red-yellow-blue isosurfaces, that will then trap and accelerate background plasma particles. A small subset of the total simulation particles, with the color scale representing particle kinetic energies, is also plotted, showing a significant accumulation of particles at the back of the accelerating bucket. A hole was cut through the plasma wave isosurfaces to reveal the particles inside.

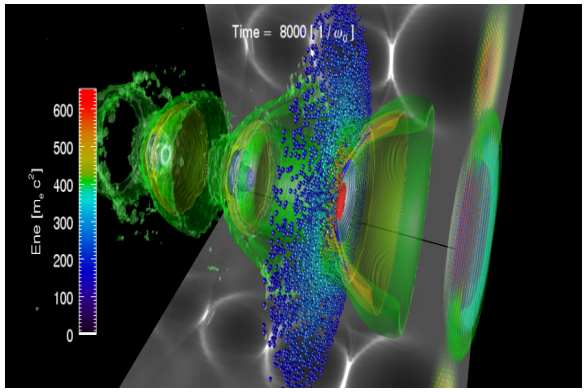

Figure 23: OSIRIS: Laser Wakefield Accelerator Baseline Run 2.

Different laser parameters result in different injection behavior as expected. For the parameters of run 3, significant injection also occurs in the

Run	Grid	Simulation Box $[c/\omega_0]$	Particles	Iterations	Laser a_0	Ions
Warm test	$6144 \times 6144 \times 1536$	$614.4 \times 614.4 \times 153.6$	4.46×10^{11}	5600	n/a	n/a
Run 1	$8064 \times 480 \times 480$	$806.4 \times 1171.88 \times 1171.88$	3.72×10^{9}	41000	4.0	fixed
Run 2	$8832 \times 432 \times 432$	$1766.4 \times 2041.31 \times 2041.31$	6.59×10^{9}	47000	4.58	fixed
Run 3	$4032 \times 312 \times 312$	$806.4 \times 1171.88 \times 1171.88$	1.26×10^{10}	52000	4.0	moving

Table 20: OSIRIS: Simulation parameter summary for runs on 55295 PEs.

Run	Warm	Run 1	Run 2	Run 3	Run 1.b
Parallel partition	$48 \times 48 \times 24$	$96 \times 24 \times 24$	$96 \times 24 \times 24$	$96 \times 24 \times 24$	$384 \times 12 \times 12$
Total PE	55296	55296	55296	55296	55296
Total Particles	4.64×10^{11}	3.17×10^{9}	1.60×10^{9}	3.04×10^{9}	3.37×10^{9}
Min. Particles/node	8.39×10^{6}	1.20×10^{3}	1.84×10^{1}	6.01×10^{4}	4.73×10^{3}
Max. Particles/node	8.39×10^{6}	1.11×10^{6}	2.02×10^{6}	1.89×10^{6}	5.27×10^{5}
Avg. Particles/node	8.39×10^{6}	5.73×10^{4}	2.89×10^{4}	5.49×10^{4}	6.10×10^{4}
Avg. Imbalance	1.00	18.85	78.62	51.29	8.68
Performance [part./s]	7.62×10^{10}	4.22×10^{9}	1.39×10^{9}	2.39×10^{9}	8.22×10^{9}
Performance / PE [10^6 part./s]	1.377	0.0764	0.0251	0.0432	0.1487
Push Time [μs / part]	0.726	13.09	39.85	23.12	6.73
Normalized Push Time [μs / part]	0.726	0.695	0.507	0.451	0.774

Table 21: Simulation performance. Average imbalance is calculated as the ratio between the maximum number of particles in a PE over the average number of particles in a PE, and the normalized push time as the push time divided by the average imbalance.

Machine Event	Warm test (5600 iter)	LWFA run 1 (41000 iter)
INS	7.50×10^{18}	7.62×10^{18}
FP OPs	2.47×10^{18}	1.37×10^{17}
L2 DCM	3.46×10^{15}	6.31×10^{14}
Time[s]	3.41×10^{4}	3.36×10^{4}

Table 22: Summary of Measured Machine Events for the *warm* and *lwfa run 1* Q2 55295 PE baselines. The reported numbers are not normalized to the iteration count. INS is the total number of instructions retired, FP OPs is the total number of floating point operations executed, and L2 DCM is a count of the number of times a L2 data cache miss happened during execution.

second accelerating bucket as shown in figure 24 where particles have already moved away from the back of this bucket and into the accelerating structure (again, only a small subset of the total simulation particles is plotted here). As an example of typical diagnostics for these runs we also show a plot of the pseudo-potential, Ψ_x, for the same run / iteration in figure 25. Tracking the pseudo-potential that individual particles feel as a function of time (and their spatial coordinates) will shed light on the injection process. Figure 26 plots the particle phasespace density at the end of runs 2 and 3, showing evidence of particle trapping and acceleration, highlighting the multiple accelerating buckets present in these scenarios and the self-injection in these buckets.

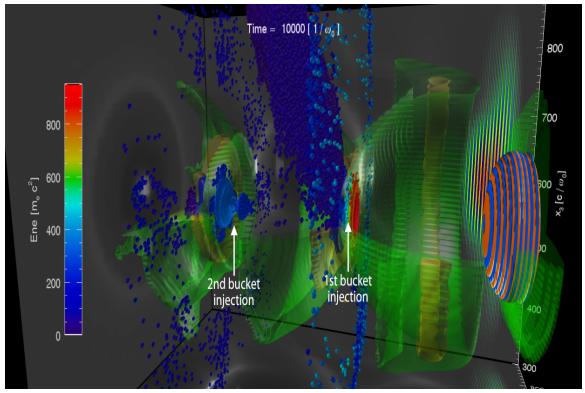

Figure 24: OSIRIS: Multiple self-injection in run 3.

The performance of the laser wakefield simulations was, as expected, lower than the reference warm plasma test as shown in the summary table 21 This

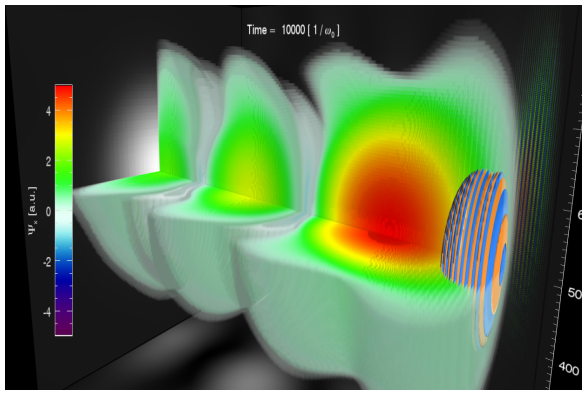

Figure 25: OSIRIS: Volume rendering of the pseudo-potential diagnostic, Ψ_x, for run 3.

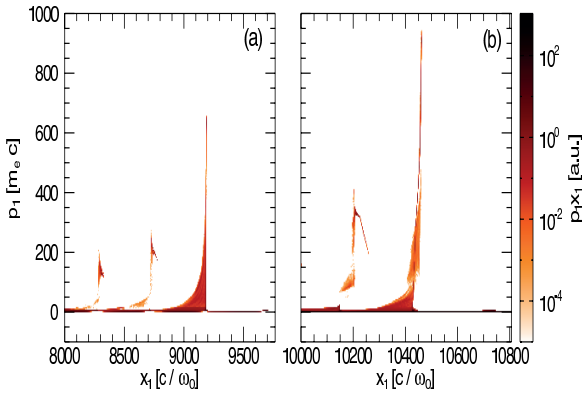

Figure 26: Particle phasespace density (longitudinal momentum vs. longitudinal position) for (a) run 2 at time $t = 8000\,\omega_0^{-1}$ and (b) run 3 at time $t = 10000\,\omega_0^{-1}$ showing evidence of particle trapping and acceleration.

was due mostly to significant load imbalance that occurs as a result of the physics of the underlying problem. The average load imbalance over the 3 runs ranged from 18.8 to 78.6 and the overall code performance was reduced by the same factors, when compared to the warm test. We can examine this in detail in figures 27 and 28. Figure 27 shows the number of particles per node for a slice of the parallel partition close to the center of the simulation box. There is a significant accumulation of particles in a small number of nodes that correspond to the simulation region at the back of the accelerating buckets, with a maximum of 1.37 million particles in this node, compared with an average of only 55.6 thousand particles in all nodes. (Note that if 50 times few PEs were used then the load imbalance would be less than 2). This situation persists throughout most of the simulation, as shown in figure 28, that shows maximum, minimum, and average number of particles per PE as a function of iteration. As soon as the plasma enters the simulation box it starts to accumulate behind the laser and a significant imbalance is maintained over the whole simulation length. The load imbalance does depend on iteration number since these particles do move in time (although slowly) from one PE to another.

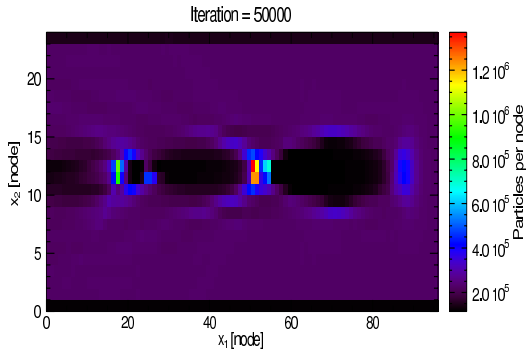

Figure 27: Number of particles per node for a slice of the parallel partition close to the center of the simulation box, for run 3.

Due to the large load imbalance there is significant room for improvement. The main challenge for laser wakefield simulations is therefore to improve the load balance in the simulation. We also performed a run identical to run 1 (run 1.b in table 21), except the problem was decomposed differently (rather using partitions of $96 \times 24 \times 24$ we used $384 \times 12 \times 12$). This led to a

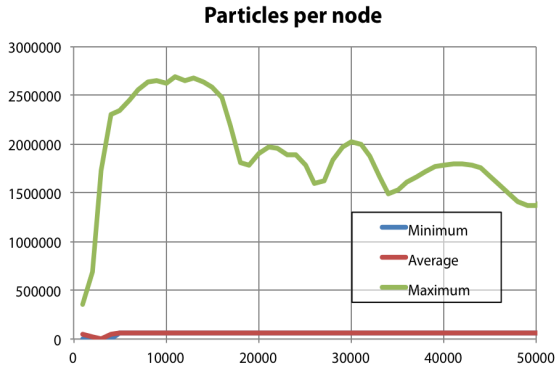

Figure 28: Time evolution of the maximum, minimum and average number of particles per PE, for run 3.

factor of 2 improvement in the load imbalance indicating that repartitioning infrequently can have dramatic improvements. Combined with algorithm improvements on a single PE such as the use of SSE vector code using single precision operations, we are targeting 1-2 orders of magnitude improvement in performance. Such improvements will enable the completion (400,000, 3,000,000, and 6,000,000 iterations for Runs 1, 3 and 2 respectively) of the Q2 baseline simulations in .5 to 4 million Jaguar hours. Currently they have only run 50,000 iterations each. These improvements will enable physics explorations using high fidelity simulations of 1-10 GeV LWFA stages.

0.3.7 Developments, Enhancements, and Q4 Benchmark Results

During Q2 we identified two main opportunities for performance enhancement, specifically optimizing key kernels on a single PE and improved algorithms for load imbalance. W implemented a single precision, single instruction multiple data (SIMD) version for the particle push and deposit algorithm using the SSE (Streaming SIMD Extensions) available on most x86 hardware, and in Jaguar computing nodes in particular. The SSE vector units available on these computing cores allow for the same instruction to be applied on 4 distinct single precision values at once, enabling much greater floating point performance. To use these units, the data to be processed must be loaded into vector registers (4×32 bit int/float) and the operations

must be coded using Assembly or C intrinsics. Although some compilers can also generate SSE code by automatically identifying vector parallelism opportunities in the source code, optimal performance can only be achieved by rethinking the algorithm and implementing it by hand for 4-way vector parallelism. Unlike other technologies, such as GPGPUs, the overall system architecture does not suffer a significant change, and we still have a cache based memory system. This means memory access strategies can remain the same but, since we will be processing more data per clock cycle, the impact of memory access overhead will be greater.

SIMD version of the PIC algorithm

For a typical OSIRIS simulation (e.g. the Warm.3d test) the code spends over 90% of execution time in advancing the particles and depositing the current which corresponds to only $\sim 10\%$ of the source code. We therefore decided to focus our SIMD development efforts only on these kernels, and keep the remaining code / memory structure intact. These kernels are also excellent candidates for vector optimization, since the operations on each particle are mostly independent from each other, with the exception of current deposition, where memory conflicts may occur. We also performed detailed testing with a single precision version of the algorithm implemented in Fortran 90, and found that for most cases the algorithm performs well with the reduced precision. The exception are problems where we are looking for faint effects on top of much stronger ones, where the reduction in precision may eliminate the required detail. For these scenarios, the double precision Fortran version should be used. However, for the laser wakefield simulations in this project, single precision calculations appears to reproduce all the necessary physics.

The parallelization strategy is straightforward: i) we load 4 particles into the vector unit, ii) we interpolate the EM fields for these 4 particles, iii) we push the 4 particles, iv) we create up to 4×4 virtual particles for current deposition and v) we store the 4 particles back to main memory. We then turn our attention to the virtual particles created and i) load 4 virtual particles into the vector unit, ii) calculate the current contribution for the 4 virtual particles and iii) accumulate this current in the global electric current grid. This last step is the only step done serially: we loop through each of the 4 virtual particles and deposit the previously calculated current for each one. This is required as more than one of the virtual particles may be in the same simulation cell, and there would be a memory conflict when accumulating

the results on the global grid. However, this is a very small part of the total algorithm, and has negligible impact overall.

As mentioned earlier, we are now processing much more data per clock cycle, and global memory access is now much more expensive than calculations, so the algorithm also had to be optimized to reduce the use of temporary memory and optimize cache (L2) use. Special attention was also given to loading / storing particles from main memory, especially because we wanted to keep the rest of the code intact. This meant that particles are stored with different x, y and z position components for the same particle sequentially in memory, which is optimal for the normal serial algorithm. However, for the vector version, the optimal memory organization would be to store individual components in separate sequential arrays i.e. one for x, one for y and one for z. The solution found was to make use of the vector shuffle operations that efficiently exchange parts of the vector registers: i) we read 3 vectors (12 positions) sequentially and ii) shuffle them to get one vector of 4 x positions, one vector of 4 y positions and one vector of 4 z positions. This 4×3 transpose is done in the registers and is very efficient, with a minimal overhead of about 10 cycles, and allows us to use efficient vector memory read operations while maintaining the rest of the code structure. For storing the particles back to memory, the opposite operation is performed.

Finally, the SSE algorithm was implemented using C, since SSE Fortran Intrinsics are not available. However, interfacing Fortran and C code has become straightforward and no special steps had to be taken.

SIMD performance

Table 24 below shows the SIMD peak code performance on a single core of an Intel i7 3.20 GHz, compiled with Intel Compiler 11.0 with -Ofast. In 3D, the code is able to push almost 14 million particles per second, using linear interpolation, and 6.4 million particles per second using quadratic interpolation.

Comparing the SSE code performance with Fortran code performance on the same system, we find that we obtain excellent speedups on all geometries / interpolation levels, as shown in figure 29. It should also be noted that for lower interpolation levels the compiler is able to generate SSE code from the original Fortran code, which also accounts for smaller speedups. We find that 3rd order interpolation levels have an optimal memory amount over calculation ratio.

	Push time [ns]		Performance [M Part/s]	
Level	2D	3D	2D	3D
1	44	73	22.5	13.7
2	70	156	14.3	6.4
3	101	324	9.9	3.1
4	149	625	6.7	1.6

Table 23: Single core OSIRIS SSE performance for multiple interpolation levels in 2D and 3D.

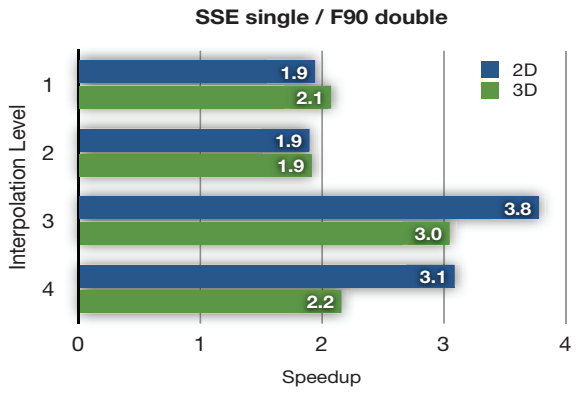

Figure 29: OSIRIS: Single core OSIRIS SSE speedup for multiple interpolation levels in 2D and 3D.

Load Imbalance

Having optimized single core performance we turned our attention to the imbalance problem. As explained before, load imbalance can be very challenging for LWFA problems. As the wake is excited plasma electrons form a large density spike in the rear or the bubble and as the simulation progresses some of these electrons are trapped and move forward within the bubble. These leads to particles may accumulating on a small subset of cores, leading to performance degradation, as the node with the most computational load will dominate overall simulation time. To address this issue, we attempted two different techniques, that can be used simultaneously: shared memory parallelism and dynamic load balance. For shared memory parallelism we took advantage of the fact that cores inside a computing node (up to 12 for Jaguar) share memory access, which allows them to better distribute the load among them and improve on global imbalance. For dynamic load balance we attempt to readjust node (or core in a pure MPI version) boundaries dynamically in an attempt to keep an even load across all computational elements.

Shared Memory Parallelism

The default parallelization strategy for OSIRIS was designed for a distributed memory system, where each core or MPI process handles a separate region of simulation space. This has proven extremely efficient because at high core counts the simulation space handled by each core is quite small, and the probability that significant load imbalance will occur is quite high. The idea behind the use of shared memory parallelism is to take advantage of the fact that many cores actually share memory access. We can therefore have them share a given simulation region, whose volume will be much larger, and distribute the particles inside this larger volume evenly across these cores. The workload for these cores will always be perfectly balanced; globally some imbalance may still occur, but since the volume handled by each group of cores is much larger, the probability for significant load imbalance will be lower.

As with the SSE code we started by parallelizing the particle pusher: each MPI process will be responsible for a larger region of simulation space to be shared by a given number of cores. This process will spawn n_T threads (that should match the number of cores available) to process the particles,

dividing particles evenly across threads. As in the SSE code this may result in memory conflicts, as particles being processed by different threads may try to deposit current to the same memory location. To overcome this, we use n_T copies of the electric current grid, and each thread will only deposit to 1 of these copies. After this has finished, the algorithm will accumulate all current grids in a single current grid, also using dividing the work over n_T threads.

After parallelizing the particle pusher, the effects from Amdahl's law start to be important, therefore, we proceeded to parallelize several other sections of the algorithm, including the field solver, current smoother, boundary processing of particles / fields and particle sorting. This parallelization was simpler than the particle pusher as these routines generally consist of an external loop that can be easily split among threads. This parallelization also has the advantage of reducing the total node communication volume; however since the messages in OSIRIS are generally small and hence latency dominated, this was not noticeable. In principle, choosing the largest possible value for n_T should provide the optimal code performance, as it would significantly lower load imbalance. However, when compared to the standard algorithm, the shared memory parallelization has the overhead of zeroing multiple current grid copies and the reduction of the current grids, so a balance needs to be found.

The actual implementation was done using OpenMP, and the figure 30 below shows a comparison of the performance between the reference 1 thread / MPI process implementation, and up to 12 threads per MPI process. As we can see there is only a 10% slowdown up to 6 threads per MPI process when compared to the reference parallelization. However, when we move to 12 threads per MPI process we see a significant drop in performance.

A detailed analysis of the code performance is shown in figure 30. As we increase the number of threads, the algorithm overhead (reducing the current grid, etc.) increases causing a performance degradation. However, this does not account for all of the significant drop from 6 to 12 threads. We also find a significant drop in the performance of the particle push. This is mainly an effect of the Non-Uniform Memory Architecture (NUMA) used in a Jaguar XT5 node: up to 6 cores (1 cpu), all cores / threads will access memory that is local to its cpu; when we move to 12 cores (2 cpus) then some cores / threads will access memory that belongs to the neighboring cpu, causing a performance degradation. We have implemented an enhancement to overcome this: the temporary electric current grids are allocated in memory

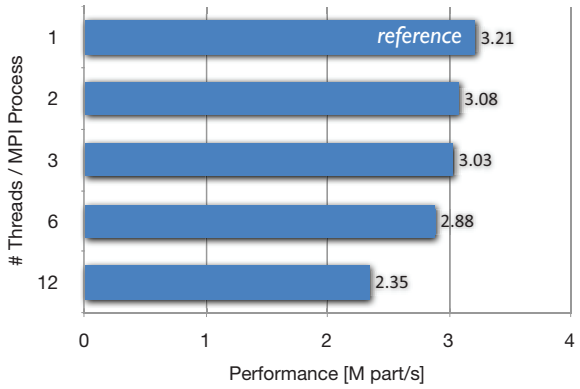

Figure 30: The total thread number (thread count times MPI Processes) remains constant.

that is local to each thread (i.e. it belongs to the same cpu). Without this, performance would be $\sim 15\%$ worse at 12 threads.

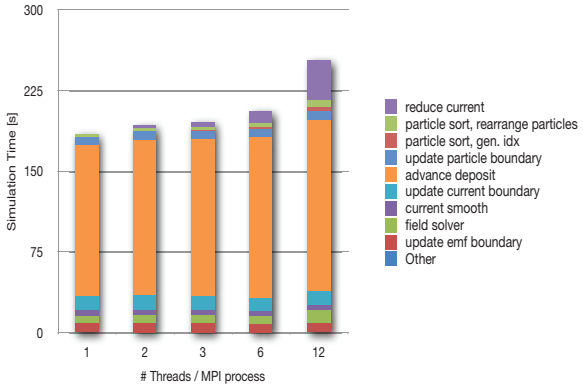

Figure 31: OSIRIS: Detailed algorithm performance for shared memory parallelism.

This effect therefore limits the scalability for SMP parallelism to a number of threads per MPI process that matches the number of cores per cpu, which for the Jaguar XT5 partition means 6 threads per MPI process. Currently, this leads to a factor of ~ 2 speedup for the cases run. It should be noted that other systems may exhibit better improved performance depending on the details of the memory architecture: for example, tests done on an IBM BlueGene/P system showed negligible performance degradation from 1 to 4

threads per MPI process (maximum value for a BlueGene/P node).

Dynamic Load Balance

Implementing SMP parallelism reduced load imbalance and improved performance, but for the laser wakefield simulations the performance was still below the optimal code performance. Although each MPI process now handles a simulation region up to 6 times the size of the Q2 baseline, significant imbalance can still occur. The concept behind dynamic load balance in PIC codes (figure 32) is simple: when this occurs change the simulation partition by changing the position of node boundaries, in order to keep the simulation load as balanced as possible. The algorithm has 2 steps: i) determining the best possible partition from current computational load and ii) redistribute the domain boundaries. However, finding an optimal partition can be quite difficult, and redistributing the domains can have significant overhead.

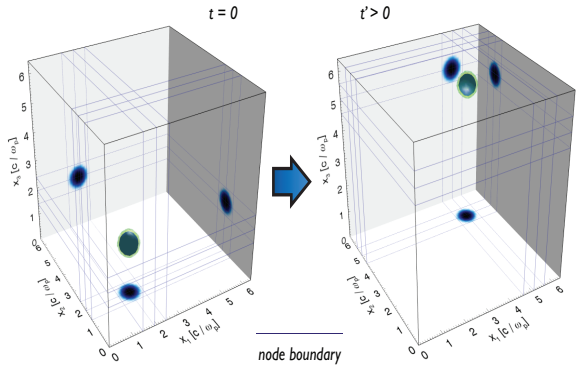

Figure 32: OSIRIS: Dynamic load balancing of parallel particle-in-cell simulations.

The only scenario for which a single solution exists for the best parallel partition is for a one-dimensional parallel partition. However, at high core counts, one must resort to multi-dimensional partitions, and attempting to improve the load imbalance in one direction may result in worse results in the other directions. One possible solution is to assume that the computational load is a separable function (i.e. load is a product of $f_x(x) \times f_y(y) \times fz(z)$): by doing this dynamic load balance in each direction becomes independent of each other. However, just by considering a spherical plasma, we see that

this is generally not true, and while in some cases there is a performance improvement, sometimes this results in a worse partition. Furthermore, at very high core counts, there isn't significant room to move your boundaries, given that we will be close to the minimum limit of cells per node, and therefore our chances for optimization are quite low. Regarding the redistribution of the node boundaries, the algorithm finds the overlap between the previous partition and the new partition, and builds a message pattern moving the data to and from the required nodes. For small partition changes this is very efficient, with very little data transfer. However, for large partition changes a given node may be sending / receiving data from hundreds of nodes, and this results in a significant overhead.

Despite our best efforts, we were unable to obtain a performance boost through dynamic load balance. Figure 33 below shows the best result obtained in our tests. This is a 2D slice of particle density, and the gray lines show the node boundaries. There is a similar node partition perpendicularly to this slice. The code attempted to dynamic load balance along x_2 and x_3 simultaneously. Overall there was a 30% improvement in imbalance, but a 5% drop in overall performance. Looking at the plot we see that the code concentrated more nodes close to the propagation axis, but we also see why the LWFA problem has a very challenging load imbalance issue.

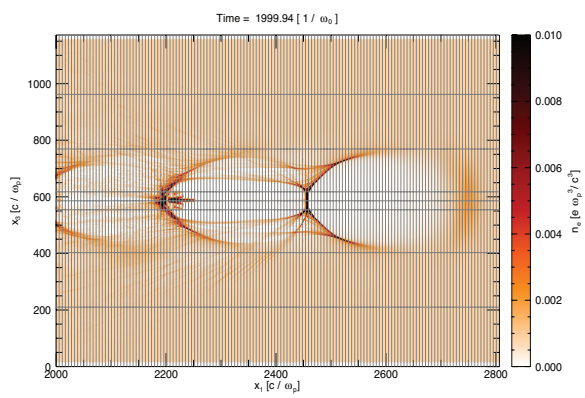

Figure 33: Particle density $x_2 - x_1$ slice for LWFA-01 run. The gray lines show the parallel node boundaries. The perpendicular direction features a similar node partition.

We also attempted choosing partitions with less nodes along propagation direction, to allow for dynamic load balance along propagation direction,

but in these scenarios the imbalance degrades significantly. Dynamic load balancing in the transverse directions also did not yield big improvements, since we have very different distributions from slice to slice, and dynamic load balancing in just x_2 or x_3 does not improve performance. While we were not able to find improvements with the dynamic load imbalance with the framework for a hybrid OpenMP/MPI together with dynamic load balancing across the nodes, all the pieces are in place for further experimentation. As noted the LWFA problem is a worst case scenario and we effectively carried out a strong scaling study.

0.3.8 Q4 Benchmark Results

Based on the results of the developments presented we chose to use for the Q4 benchmarks the single precision SIMD code. For the peak code performance tests we used the pure MPI implementation, and for the laser wakefield runs we used the hybrid MPI/OpenMP algorithm with 6 threads / MPI process / CPU. Table X shows a summary of the benchmarks run. We chose to run 2 sets of problems: a `small` set, where we repeat the Q2 tests with the same hardware (55k cores partition), and a large set where we test the code performance on the full Jaguar system (220k cores partition) and strong scale the LWFA-01 run.

55k partition runs

The goal of the 55k partition runs was to assess the performance improvements of the code using the same hardware as in the Q2 benchmarks.The test parameters are exactly the same as in Q2, and only the binary has changed. As before, the Warm.3d test establishes a baseline for expected code performance for realistic plasmas, and the 3 laser wakefield runs focus on analyzing different aspects of LWFA. Figure 34 below shows the performance results for these runs (Performance is measured in global particle pushes per second). For comparison the results from Q2 are also included.

As shown above, the peak performance measured was 1.8×10^{11} part / s for the Warm.3d test, a 2.4 speedup from Q2, that results from a combination of the SSE code and general code / compiler improvements. For the laser wakefield runs the speedup ranged from 6.9 to 7.4, also benefiting from shared memory parallelism and improved parallel partition, reducing the simulation imbalance by a factor that ranged from 3.6 to 4.4. Overall, the average

Run	Partition [cores]	Performance [G part/s]	Push Time [μs]	Average Imbalance	TFLOPS	INS/FP	Q2 Speedup
Warm.3d	55296	179.95	0.307	1.00	169.92	1.28	2.36
LWFA - 01		29.66	1.864	3.64	31.18	6.39	7.03
LWFA - 02		27.43	2.016	4.75	28.02	7.69	7.37
LWFA - 03		61.20	0.903	2.31	58.25	3.84	6.92
Frozen.3d linear	221184	1463.52	0.151	1.00	516.92	1.34	n / a
Frozen.3d quadratic		784.04	0.282	1.00	736.12	1.20	n / a
Warm.3d weak scale		741.20	0.298	1.00	700.09	1.21	9.73
Warm.3d strong scale		719.80	0.307	1.00	679.68	1.28	9.45
LWFA - 01 - strong scale		70.91	3.119	4.66	76.55	9.48	16.80

Table 24: Single core OSIRIS SSE performance for multiple interpolation levels in 2D and 3D.

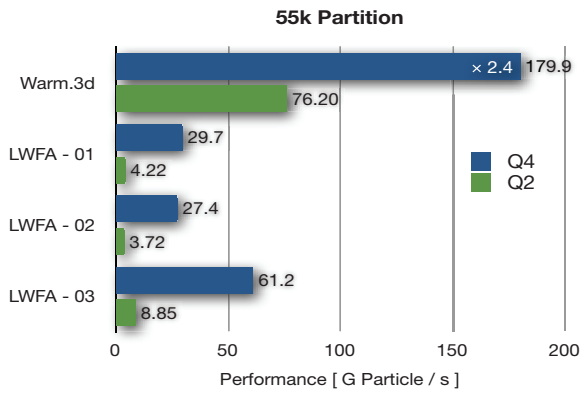

Figure 34: OSIRIS: Code performance for the 55k partition tests, showing both the Q4 (blue) and Q2 (green) results.

speedup for the laser wakefield runs was a factor of 7, meaning that the LWFA - 01 run can now be completed in the same hardware (55k cores) in 10.3 hours, rather than the original 2 days and 17.8 hours.

The breakdown of the speedup can be seen in the figure 35 below for the LWFA-01 run. The speedup comes from a combination of 3 factors: i) a new parallel partition yields a 1.91 times improvement, ii) the SMP parallelism yields an additional speedup of 1.82 and iii) the SSE code yields an additional speedup of 2.02 leading to a total speedup of 7.03. The behavior was similar for the other laser wakefield runs.

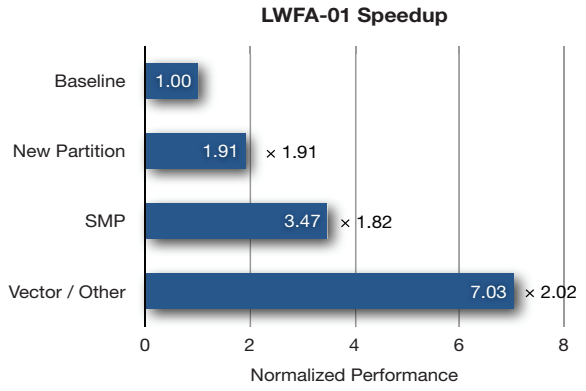

Figure 35: OSIRIS: Breakdown of performance speedup for the LWFA-01 run.

220k partition runs

The goal of the 220k partition runs was to analyze code performance in the full system, and to establish our capability in scaling the test problems in this hardware partition. A series of tests were chosen to establish peak code performance (frozen linear and frozen quadratic), strong and weak scaling of well balanced problems (warm.3d strong and warm.3d weak), and strong scaling of one of the laser wakefield problems. The hardware partition used was exactly 4 times the size of the partition used in Q2, and corresponds to 98.7% of the full system. Figure 36 below summarizes the performance results for these runs.

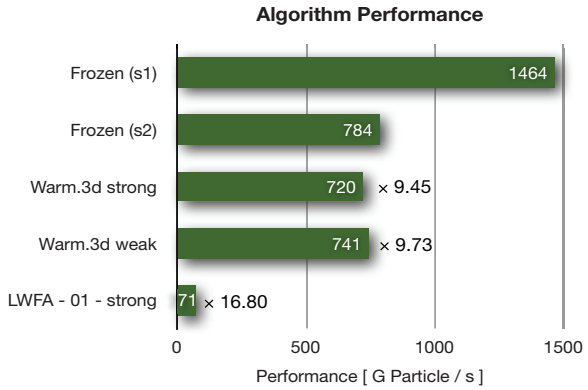

Figure 36: OSIRIS: Code performance for the 220k partition tests.

The frozen tests measure the asymptotic limit for code performance by modeling a zero temperature plasma with 1.86×10^{12} particles. This means that there is no particle motion, which leads to optimal performance in current deposition and minimizes the communications for particle boundaries. For linear interpolation the code performance was 1.46×10^{12} particles / s and for quadratic interpolation 0.78×10^{12} particles / s. The Warm.3d tests focus on a more realistic scenario that uses a warm plasma distribution that more accurately mimics typical simulation parameters, while maintaining a balanced load. Two scaling tests were made, for strong and weak scaling of this simulation. The total number of particles in these simulations were 4.64×10^{11} and 1.86×10^{12} particles respectively. The code performance was very similar in both cases, on the order of 0.73×10^{12} particles / s, slightly below the frozen plasma limit. In both scenarios the code was found to to

hyper-scale from Q2, with a speedup of ~ 9.5 with only 4 times the number cores.

Regarding the floating point performance of the code, figure X below summarizes the results obtained for the large scale tests. For comparison the HPL benchmark for Jaguar was included. The code reaches very high floating point / parallel efficiency for the full system, reaching 0.74 PFlops peak performance or 32% of Rpeak (42% of Rmax). Quadratic Interpolation was found to improve memory / operation ratio and yield a higher FLOP count, while linear interpolation has higher particle performance but lower FLOP count.

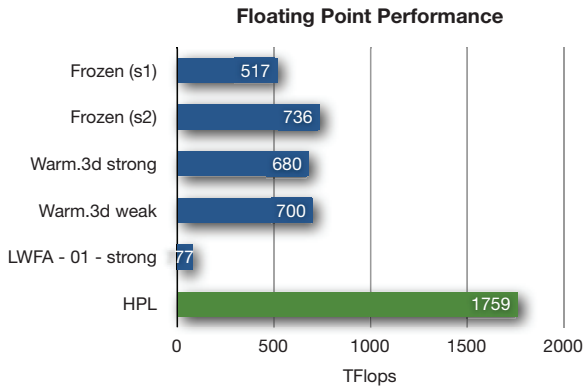

Figure 37: Floating point performance for the 220k partition tests. For comparison the HPL benchmark for Jaguar was included (green).

Finally, figure 38 below shows a visualization of the results from the strong scaling of the LWFA-01 run. The global performance of this simulation was 7.09×10^{10} particles / s, or 3.12μs core push time, which results in a $16.8\times$ speedup from Q2. The load imbalance was 4.66, larger than for the 55k partition tests, which is expected since the node volume decreased by a factor of 4. The floating point performance of the code was 77 TFlops (3.3% of Rpeak). When compared with the Warm.3d runs we see that the code ran $9.8\times$ slower. This is mainly due to the imbalance shown above and the fact that we have a very small final problem size, on average $\sim$ 15k particles per node, which results in a parallel overhead of over 50%. These numbers indicate that the LWFA-01 simulation, that would take 2 days and 17.8 hours in Q2, can now be performed in under 4 hours using the full Jaguar XT5 partition.

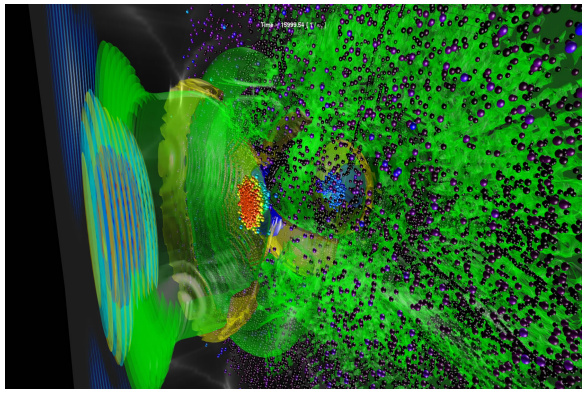

Figure 38: OSIRIS: Visualization from the strong scaling of the LWFA-01 run

Baseline Q4 Problem Results

For Q4, we concentrated on running run 1 out to completion by taking advantage of the algorithm improvements and doing a strong scaling to the full machine. This simulation models a 200TW (6 Joule) laser propagating through .5cm of a $1.5x10^{18}cm_{-3}$ plasma. We previously simulated this using first order (linear) particle shapes and a resolution of $k_0\Delta_z = .2$ and $k_p\Delta_x = k_p\Delta_y = .116$ using 2 electrons per cell and immobile ions. It was found that two groups of electrons are self-trapped. One is in the first bucket and another in the second bucket. After a laser propagation distance of .75cm the amount of charge in the first bucket was .3 nC with an energy of 1.5 GeV, a FWHM energy spread of 3.8% , and a normalized emittance of 35 and 29 mm-mr in the two transverse planes respectively. These results were published in 2007. The electrons in the second bucket arises from electrons at the surface getting injected. These electrons are not present for realistic density profiles so we do not discuss them here. Subsequently, we have rerun this case using second order particle shapes with 2 and 8 electrons per cell but with the same resolution. We presented preliminary results on these comparisons earlier this year at the 2011 Particle Accelerator Conference and they will be published in the conference proceedings.

In run 1, we use quadratic particle shapes and two particles per cell, however the resolution is improved by a factor of 2 (and the time step as well) in the laser propagation direction and by a factor of 1.61 in each transverse direction, so $k_0\Delta_z = .1$ and $k_p\Delta_x = k_p\Delta_y = .072$. In figure 39we show a

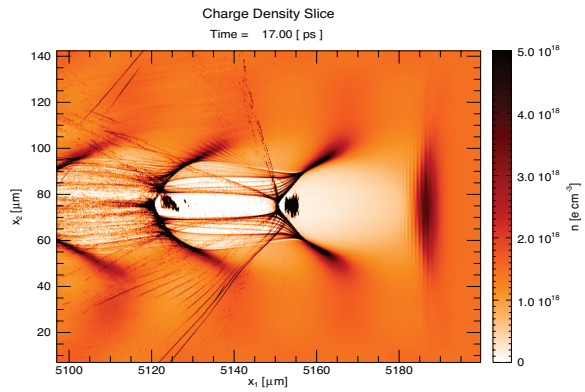

Figure 39: A 2D slice of the electron density showing the electrons injected into the first two buckets

slice of the electron density at a propagation distance of .51cm. Just as in the previously published results, one can see the two buckets with the self-trapped bunches in each bucket. In Table 25 we give the charge, average energy, energy of the peak, energy spread (defined two ways), and emittance for the in both transverse directions for the electrons in the first bucket at .46cm (not the end of the simulation but a time common to all runs). For the Q4 run there is .3643 nC in the first bucket with an average energy of .9625 GeV, a FWHM energy spread of 44.9 MeV (4.6), and emittances of 19.42 and 18.71 mm-mr in the two transverse planes respectively. Most quantities are within 10% The two exceptions are the charge (the linear particle shape run has 25% less charge) and the emittance is significantly reduced in the higher resolution (Q4) run. The high resolution run has 50% lower RMS value for the two transverse planes. This improvement in emittance is very important for both collider and light source applications. These results indicate quadratic particle shapes should be used and that more convergence studies may be necessary for the emittance.

Figure 40: Comparison of the energy spectra of the beam in the first bucket for the runs listed in table 25. Black is 2ppc linear, green is 2ppc quadratic, blue is 8ppc quadratic and red is Q4 high resolution.

In figure 40 we plot the energy spectra for each of the runs in Table 25. These spectra together with Table 25 help to show that the details of the spectra are very similar. In this plot one can see that the energy distributions

	2ppc Linear	2ppc Quad	8ppc Quad	Q4 HR
Charge [pC]	284	339	347	366
Avg. Ene [MeV]	1074.7	1052.6	1054.7	1048.1
StdDev Ene [MeV]	53.4	76.3	75.6	85.5
Peak Ene [MeV	1031.8	979.0	984.5	962.6
Ene FWHM [MeV]	34.5	50.1	19.8	44.5
ϵ_{Ny} mm mr	29.6	26.7	28.8	19.2
ϵ_{Nz} mm mr	33.9	30.7	25.6	18.6

Table 25: Comparison of the electron beam in the first bucket from runs with the same physical parameters but different numerical parameters

all show a dominate peak near 1 GeV and a second peak near 1.2 GeV. The largest peak is at .9625 GeV in the high resolution and at 1031.8 GeV in the low resolution linear particle shape run. The energy of the peaks are very similar for all of the simulations using quadratic particle shapes. It is also worth looking at the details of the 6D phase space for this beam. In figures 26, 42, and 43 we plot $p1$ vs. $x1$, $p2$ vs.$p1$ and $P_\perp$ vs. $x_\perp$ for the particles in the first bucket. These plots show that the electrons in the spike near 1.2 GeV are also very localized in space (in x1) and they have a very small angular divergence (emittance). In addition, the angular divergence of the electrons in the lower energy peak is also relatively small. Therefore, more analysis of these beams are needed to understand the energy dependent emittance and the slice emittance and energy spread.

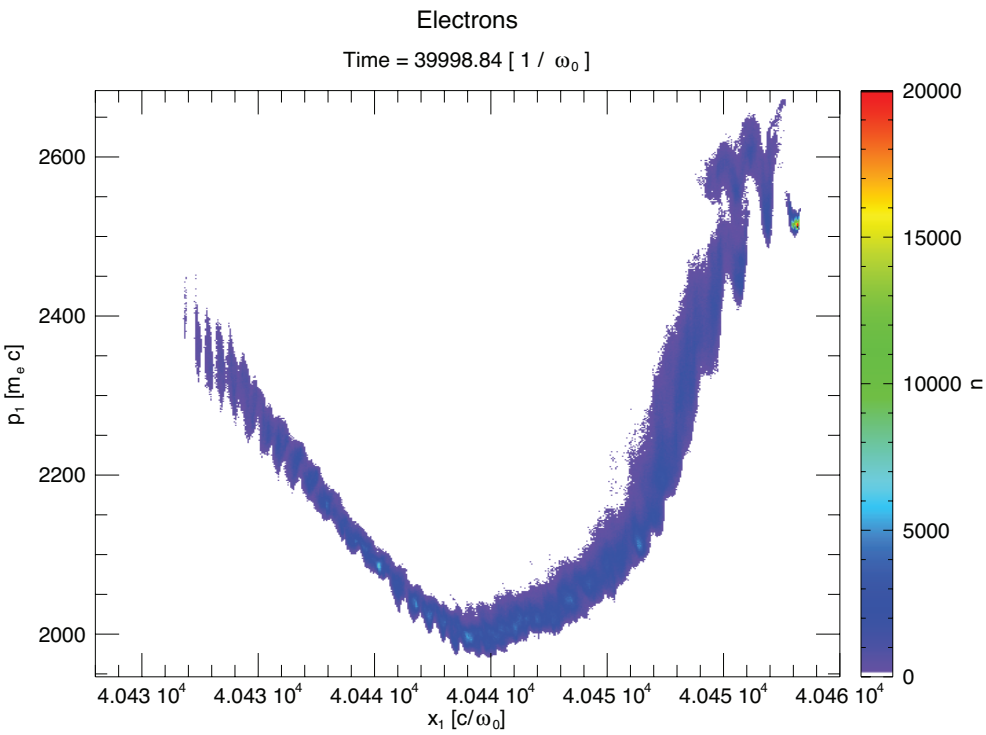

Figure 41: The p1 vs. x1 phase space of the injected beam (laser propagates in x1) at .46cm

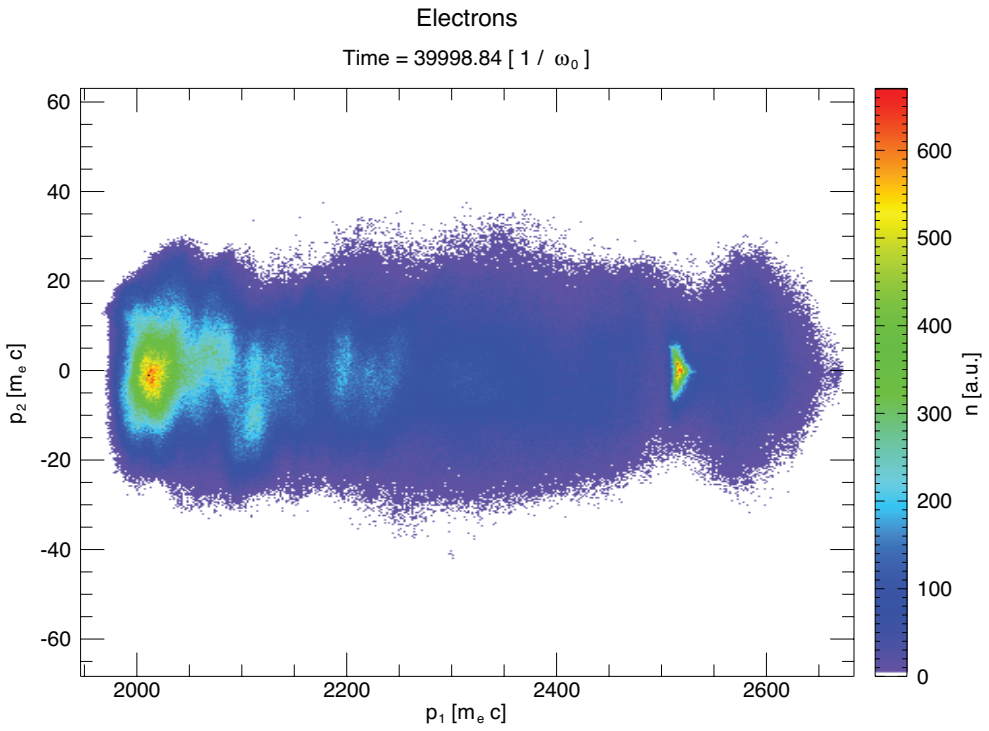

Figure 42: The p2 vs. p1 phase space of the injected beam (laser propagates in x1) at .46cm

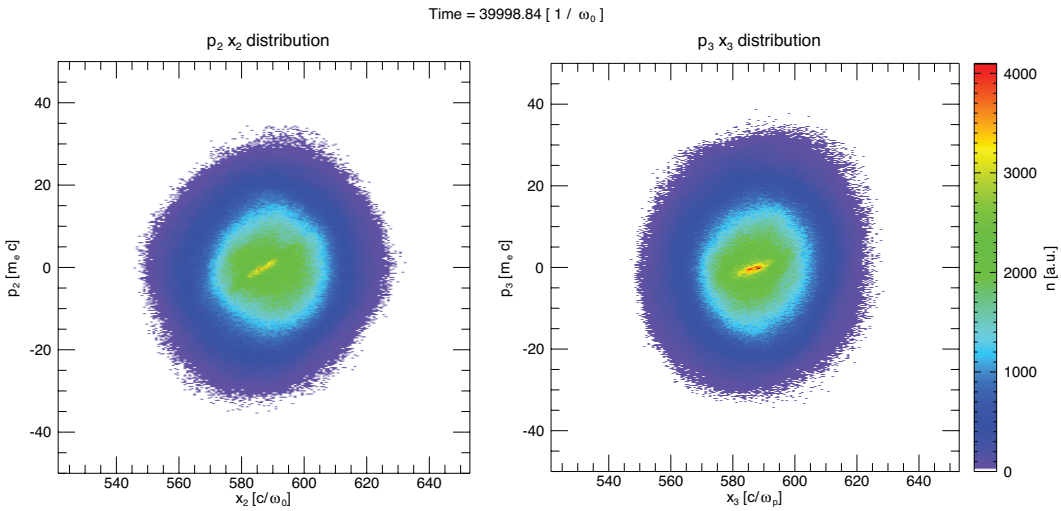

Figure 43: The p2 vs. x2 and p3 vs. x3 phase space of the injected beam (laser propagates in x1) at .46cm

0.4 eSTOMP

0.4.1 Introduction

The goal of our OMB PMM DOE SC ASCR metric is to increase the process and property detail of a uranium bioremediation simulation. In this case, we will simulate a field experiment at a former uranium mill tailings site where the enrichment of indigenous metal-reducing bacteria via acetate biostimulation catalyzes the conversion of uranium in a groundwater plume from the mobile +6 oxidation state to immobile solid-associated uranium in the +4 oxidation state. This effectively removes uranium from solution resulting in groundwater concentrations below actionable standards. Three-dimensional, coupled variably-saturated flow and biogeochemical reactive transport modeling is used to better understand the interplay of transport rates and biogeochemical reaction rates that determine the location and magnitude of key reaction products. A comprehensive reaction network is used to simulate the impacts on uranium behavior of pulsed acetate amendment, seasonal water table variation, spatially-variable physical (hydraulic conductivity, porosity) and geochemical (reactive surface area) material properties. A principal challenge is the mechanistic representation of biologically-mediated terminal electron acceptor process (TEAP) reactions whose products significantly alter geochemical controls on uranium mobility through increases in pH, alkalinity, exchangeable cations, and highly reactive reduction products. The objectives are to double the degrees of freedom and achieve scalability of this eSTOMP application.

0.4.2 Background and Motivation

For comprehensively detailed depictions of complex subsurface systems, numerical simulation is the principal vehicle for assembling the fundamental process models and addressing the spatially varying material properties required for predictions. Advanced computing can provide the high performance and large memory to simultaneously address long simulation periods (e.g., 10,000 year period for spent nuclear fuel regulatory compliance), comprehensive treatment of coupled processes (e.g., geology, hydrology, biology, chemistry), resolution of multiscale variability in material properties, and uncertainties in conceptual process models and parameters. Effective use of state-of-the-art computational capability and performance can enable

new lines of research where larger, higher resolution datasets can be used as testbeds to develop new theories and process models that will ultimately lead to more reliable predictions of field scale behavior. In fact, the development and incorporation of new multiscale process models of fundamental behaviors is ongoing: 1) molecular-scale models of mineral surface reactions, 2) in silico models of metal reducing bacteria, 3) pore-scale models of multiphase flow and multicomponent reactive transport, and 4) simulations of multiphase flow and multicomponent reactive transport at the scale of the fundamental Darcy representative elementary volume. These developments reflect advancements in measurement and characterization technology (e.g., NMR, X-ray synchrotron, neutron scattering, electromagnetic and seismic geophysical methods) that are providing unprecedented detail on structures and processes in the subsurface environment. The expectation is for comprehensive simulations with high spatial and process detail to begin to link and integrate research at different conceptual length scale domains (e.g., molecular to microscopic, pore to meso, etc.) to resolve long-standing subsurface upscaling issues.

While there has been progress, accurate and detailed characterization of spatial property distributions in the subsurface remains a first-order challenge facing reactive transport modeling efforts. Subsurface properties such as permeability and reactivity are known to vary over orders of magnitude across a wide range of spatial scales. This heterogeneity gives rise to preferential flow paths or barriers, incomplete mixing of reactants, limited effectiveness of engineered solutions, and slow release of reactive components through diffusion-controlled mass transfer processes. Conversely, we often struggle to identify which small scale (i.e., sub-grid resolution) features, processes, and or conditions impact/control the field-scale behaviors of interest. This is due, in part, to the severe computational burden of resolving a wide range of length and time scales to obtain accurate predictions of the subsurface processes. Consequently, the current practice is to develop models based on the continuum scale where Darcys law applies and the geometrical complexity of the pore space is averaged and replaced by empirically determined effective parameters or constitutive relations. For complex, coupled, nonlinear processes (e.g., multiphase flow, biogeochemical reactions, and geomechanics), the lack of mechanistic upscaling undermines the reliability of predictive simulations used to inform decisions affecting the environment and human health.

0.4.3 Capability Overview

The computationally intensive research performed in this project will use eSTOMP, the extreme-scale computing version of the Subsurface Transport Over Multiple Phases (STOMP) subsurface simulator (White and Oostrom 2006). The STOMP simulator is Pacific Northwest National Laboratory's (PNNL) premier subsurface multifluid flow and reactive transport simulation capability for addressing DOE missions in environmental restoration, carbon sequestration, and alternative energy resources. The process models in STOMP are used to address a large range of subsurface environments, including nonisothermal conditions, fractured media, multiple-phase systems, nonwetting fluid entrapment, soil freezing conditions, nonaqueous phase liquids, biogeochemical reactions, radioactive decay, solute transport, dense brines, nonequilibrium dissolution, and surfactant-enhanced dissolution and mobilization of organics.

Written in Fortran-90, eSTOMP solves the conservation equations for compressible variably saturated flow and multicomponent reactive transport based on finite-volume discretization. A Newton-Krylov solution procedure is employed for the nonlinear global implicit formulation. The bi-conjugate gradient solution methodology with incomplete LU preconditioning from the PETSc solver library is employed. eSTOMP was engineered for multicore-based supercomputers using the Global Arrays Toolkit (GA) (Nieplocha et al. 2006). GA presents a global shared memory view of the data to the programmer while providing the underlying mapping and communication of distributed data that exploits data locality. The superiority of GA for small messages and fine-grain parallelism is particularly significant for multicore architectures. GA is compatible with the message passing interface (MPI) that the PETSc solver in eSTOMP is based on.

0.4.4 Science Driver for Metric Problem

The goal of our OMB PMM DOE SC ASCR metric is to increase the process and property detail of a uranium bioremediation simulation. In this case, we will simulate a field experiment at a former uranium mill tailings site where the enrichment of indigenous metal-reducing bacteria via acetate biostimulation catalyzes the conversion of uranium in a groundwater plume from the mobile +6 oxidation state to immobile solid-associated uranium in the +4 oxidation state. This effectively removes uranium from solution resulting in

groundwater concentrations below actionable standards. Three-dimensional, coupled variably-saturated flow and biogeochemical reactive transport modeling is used to better understand the interplay of transport rates and biogeochemical reaction rates that determine the location and magnitude of key reaction products. A comprehensive reaction network is used to simulate the impacts on uranium behavior of pulsed acetate amendment, seasonal water table variation, spatially-variable physical (hydraulic conductivity, porosity) and geochemical (reactive surface area) material properties. A principal challenge is the mechanistic representation of biologically-mediated terminal electron acceptor process (TEAP) reactions whose products significantly alter geochemical controls on uranium mobility through increases in pH, alkalinity, exchangeable cations, and highly reactive reduction products.

The objectives are to reduce the wall clock time to simulation completion and improve the scalability/efficiency of this eSTOMP application.

0.4.5 The Model and Algorithm

The governing flow and transport equations are the partial differential equations for the conservation of water and solute mass. Kinetic and equilibrium reactions are represented as a system of ordinary differential and algebraic equations (DAEs). Variably saturated flow is represented through constitutive functions that relate capillary pressure to saturation and relative permeability. The flow and transport equations are solved following an integral volume finite difference approach with the nonlinearities in the discretized equations resolved through Newton-Raphson iteration. Linear systems which result from the Newton-Raphson linearization or the solute transport solution are solved with the biconjugate gradient stabilized method with ILU preconditioning. Solute mass conservation and reactions are sequentially solved following the solution of the flow equations.

The STOMP simulator solves transient flow and transport problems in the subsurface environment in one, two, or three dimensions. Coordinate systems must be orthogonal and currently are limited to Cartesian, tilted Cartesian, and cylindrical, where the vertical coordinate of the cylindrical system must be aligned with the gravitational vector. All boundary conditions, sources, and sinks can be time variant. Hydrogeologic properties, which can be anisotropic, can be spatially varied throughout the computational domain within the resolution of a node volume. A variety of boundary conditions are available, which may be applied selectively over the boundary

surfaces. Unspecified boundary surfaces are assumed to be zero flux surfaces. Inactive nodes can be specified by the user and boundary conditions can be applied to surfaces separating active (computational) and inactive (noncomputational) nodes.

Checkpoint/restart capabilities have also been included in the simulator, which can resume a simulation with or without alterations to input parameters. Output from the simulator can be written both to files and to the standard input/output device (e.g., screen). Output forms include time histories of selected variables, time snapshots of selected variables across the computational domain, and variable integrals for sources and fluxes across boundary and internode surfaces.

The water mass conservation equation, shown below, equates the time rate of change of water mass within a control volume with the flux of water mass crossing the control volume surface.

$$\partial_t[\sum_{\gamma=j,g,i}(n_D\omega_\gamma^w\rho_\gamma s_\gamma)] = -\sum_{\gamma=l,g}(\nabla\mathbf{F}_\gamma^w + \nabla\mathbf{J}_\gamma^w) - \nabla\mathbf{F}_l^S + \dot{m}^w \tag{14}$$

where

$$\mathbf{F}_\gamma^w = -\frac{\omega_\gamma^w\rho_\gamma k_{r\gamma}\mathbf{k}}{\mu_\gamma}(\nabla P_\gamma + \rho_\gamma g\mathbf{z}_g) \;\; for\; \gamma = 1, g \tag{15}$$

$$\mathbf{J}_\gamma^w = -\tau_\gamma n_D\rho_\gamma s_\gamma\frac{M^w}{M_\gamma}D_\gamma^w\nabla\chi_\gamma^w \;\; for\; \gamma = l, g \tag{16}$$

$$\mathbf{F}_l^S = D_l^S\nabla S_l \tag{17}$$

The solute conservation equation, shown below, equates the time rate of change of solute within a control volume with the flux of solute crossing the control volume surface. In the STOMP simulator, solute is partitioned among the fluid and solid phases assuming thermodynamic and geochemical equilibrium conditions. Solute transport occurs by advection and diffusion-dispersion through the aqueous phase.

$$\begin{aligned}\partial_t C &= -\sum_{\gamma=l,g,n}(\nabla[C_\gamma\mathbf{V}_\gamma]) + \dot{m}^C - \dot{R}^C C \\ &+ \sum_{\gamma=j,g,i}\nabla[(\tau_\gamma s_\gamma n_D D_\gamma^C + s_\gamma n_D\mathbf{D}_{h_\gamma})\nabla C_\gamma]\end{aligned} \tag{18}$$

where

$$(19)\quad \mathbf{V}_\gamma = -\frac{k_{r\gamma}\mathbf{k}}{\mu_\gamma}(\nabla P_\gamma + \rho_\gamma g \mathbf{z}_g)\ \ for\ \gamma = l, g, n\ .$$

Equilibrium reactions (i.e., mass action equations relating species activities through an equilibrium constant) are represented as:

$$(20)\quad C_j = K_{eq\,j} \prod_{N_{eq\,j}} (C_i)^{e_i}\ \forall i \neq j\ ;\ j = 1, N_{eq}^{eq}\ .$$

Kinetic equations define kinetic components where a stochiometrically weighted sum of species concentrations vary in time according to a weighted sum of kinetic rates:

$$(21)\quad \frac{d\sum_{N_{tk,j}^S} b_i C_i}{dt} = \sum_{N_{tk,j}^r} c_k R_k\ \ j = 1, N_{kn}^{eq}\ .$$

A variety of popular reaction rate models are available with the STOMP simulator including those based on simple first order reactions, transition state theory, and Monod kinetics.

The nonlinear system of mixed differential and algebraic equations (DAEs) is formulated with the backward Euler method and Gauss-Jordan matrix decomposition. Newton-Raphson iteration is used to solve the reaction system.

0.4.6 Q2 Baseline Problem Results

The baseline simulation targets the processes, properties, and conditions controlling the behavior of uranium in a groundwater plume during engineered bioremediation. In this case, acetate is injected into a uranium-contaminated groundwater plume to stimulate metal reducing bacteria to catalyze the conversion of aqueous U(VI) to immobile U(IV). Uranium behavior is simulated under pulsed acetate amendment, seasonal water table variation, and spatially-variable physical (hydraulic conductivity, porosity) and geochemical (reactive surface area) material properties. The goal of the simulation is to investigate the interplay of transport rates and biogeochemical reaction rates that determine the location and magnitude of key reaction products.

The 18 m long, 20 m wide, and 6.3 m deep model domain is resolved at 0.1 m resolution with 2,268,000 grid cells. The spatially variable geology is characterized with 5 lithofacies: clay fill, sandy gravel, muddy gravel, fine,

and the underlying Wasatch formation, which acts as a bottom confining layer. Variably-saturated flow (i.e., Richards Equation) through the hydrogeologic system is driven by upgradient and downgradient Dirichlet pressure boundary conditions that reflect the seasonal decline in the water table elevation during the field experiment. This effectively reduces the saturated thickness from 3.3 to 2.5 m. All other hydrologic boundaries are treated as no-flow Neumann conditions. The transient simulation is initiated from a steady state flow condition. Initial and upgradient solute concentrations are based on field observations with influent and effluent solute mass fluxes treated as Neumann boundary conditions. 102 biogeochemical species and 7 minerals are simulated in the reaction network. A genome-scale model for one metal-reducing bacteria, *Geobacter sulfurreducens*, is being tested in this simulation.

The goal is to simulate the 110 day period of acetate injection and the following 50 days of subsurface behavior. A nominal 1 hour time step is used but is automatically reduced if the convergence rate drops below criteria. 52,800 processor cores are used for the baseline problem, mapped to a 60 x 44 x 20 processor core grid that corresponds to the 180 x 200 x 63-cell grid.

In addition to the main input file, there are three $\sim$ 25 MB auxiliary files that are read in for the material type distribution, initial uranium concentrations, and the reactive site density. Nominal output of 10 variables at a single location is generated every time step. Complete spatial distribution snapshots of 18 key components are generated every 6 simulated hours, resulting in $\sim$ 400 GB of output. Each checkpoint file generates $\sim$ 6 GB of output. Checkpointing at 10 day intervals results in a total output of $\sim$ 500 GB.

The Q2 baseline simulation targeted 24 simulated hours of the uranium bioremediation experiment. Every 6 simulated hours, a checkpoint/restart file and spatial output for the 18 selected components were generated. 106272 Krylov iterations (PETSc) and 722 STOMP steps were executed to complete the baseline simulation over 300 physical time steps. Machine event counts measured during execution of the baseline problem are presented in Table 26.

Our objective is to reduce the wall clock time for the baseline simulation by at least a factor of 2. An efficiency result for this system will be demonstrated in Q4.

INS	FP_OPs	Cycles	Time[s]
2113081740471208727	378976827416438	31924984661256	12278.84

Table 26: PAPI hardware counter data for the Q2 Uranium bioremediation experiment executed on 158,400 PEs of which only 52,800 PEs were utilized for computation. This overprescription of hardware is to account for a large memory demand in the current implementation.

0.4.7 Developments, Enhancements, and Q4 Benchmark Results

The Q2 application was improved on 2 fronts: per core memory allocation and time spent in I/O.

Memory allocation In the Q2 simulation, the large per core memory requirements limited the number of computational cores per node to 4. This meant that the 52,800 process job had to allocate, but not use, an additional 105,600 processor cores. In Q3 and Q4, eSTOMP was modified to use only distributed arrays for the chemical species, eliminating the temporary allocation of 102 field arrays in local memory. Each of these arrays is comprised of one double-precision word for each grid cell in the 2,268,000 grid cell problem domain. This modification resulted in 1.72 GB less memory usage per core. This not only improved the on chip-processor utilization by a factor of three (i.e., all 12 processor cores on Jaguar nodes could now be used), it permitted the problem to be solved with smaller processor counts.

I/O In the Q2 application, the periodic output of field and checkpoint/restart files accounted for 79.5% of the wall clock time for the baseline simulation. There were two major factors that contributed to the large amount of time spent on I/O. The first is the I/O bottleneck caused by the serialization of I/O through the head node. Also, during this process, there was added organizational complexity due to the NUMA specific memory affinity issues arising during memory copies and data modifications caused by the overprescription of process cores to meet the memory demands. The second is the use of ASCII file formats, which in reading and writing large files (e.g., checkpoint/restart), requires considerably more time than binary formatted files due to formatting, spaces, new line characters, and comments or annnotations. There are extra BYTEs of data written each event to add comments and space tabbing required to make the files human readable, plain text.

eSTOMP has several output options ranging from high frequency output

ASCII	Binary
3253456688	2739744000

Table 27: Comparison of aggregated file sizes for the Q2 problem with the unformatted ASCII versus binary writes. The aggregated size should be 151 (fields) x 2,268,000 (double precision numbers per field) x 8 BYTEs per double precision number = 2,739,744,000 BYTEs.

of specific variables at specific locations, to the periodic output of entire field arrays of a specific variable, to the less frequent output of checkpoint/restart information. In the Q2 baseline, the output of 10 variables at a reference node location every time step was determined not to be a significant contributor to the large amount of wall clock time spent in I/O. Thus, the focus of the Q3 and Q4 effort was the reduction of wall clock time spent on the periodic output of the complete spatial distributions for 18 field variables and the periodic generation of checkpoint/restart files.

The original eSTOMP code used a sequence of GA toolkit commands (see `http://www.emsl.pnl.gov/docs/global/c_nga_ops.html` for more details) to move data from the local processes to a global array that allowed it to be easily manipulated for output. This included 1) `nga_put` commands to move local data from the local process to a global array, 2) `nga_sync` commands to ensure that all local processes had completed their 'put' of data into the global array, and 3) `nga_get` commands for the head node to get the data in preparation for writing the data to disk. To keep the memory requirements manageable, the writing of the ASCII files was done in buffered ($\sim 4N_xN_y$) slices.

It was determined that the use of a reduction operation using the GA toolkit command `ga_dgop` was more efficient than the sequence of the GA `get`, `sync`, and `put` operations. In this case, a memory buffer sized for the entire targeted field array output was created for each process. Each process writes their local fraction of the targeted field array into the buffer in the location it would occupy in the complete natural array. The `ga_dgop` command is essentially a wrapper for the `MPI_reduce` command, which is used to send each local piece of the write buffer to the write process. This approach was used for output of the field variables and the checkpoint/restart files.

Once the contents of the write buffer are completely updated and accessible from the head node, it can be written to disk as a binary file. A buffered

write was executed $N_x N_y N_z/4$ slices. One binary file was generated for each selected spatially distributed field variable and for each checkpoint/restart file. In Table[27] the differences in the aggregated magnitudes of the unformatted ASCII writes versus the packed binary writes are compared for the Q2 problem.

0.4.8 Q4 Benchmark Results

The raw performance measurements for the Q4 enhanced eSTOMP runs versus the Q2 baseline results are presented in Table [28].

eSTOMP Summary of Performance for FY11 Benchmarks

	Q2	Q4	Q2 : Q4
PEs	158,400	26,400	6
INS	2.11e+18	2.619e+17	8.0665
FP OPs	378976827416438	416965910025780	0.9089
L2 DCM	19491864496712	13739811264029	1.4186
Time[s]	12278.840 (11620.592)	4798.922 (4554.582)	2.5587
Cost [CPU Hrs]	540,269	35,192	15.352

Table 28: Summary and Comparison of Measured Machine Events for the Uranium Bioremediation problem.

The combination of the following optimizations resulted in over 2.5X reduction in total wall clock time and over 15X savings in the measure of CPU cost of the Q2 problem with the Q4 enhanced code: reduced per core memory requirements leads to fewer processors to execute the same problem, replacing the sequence of GA put, sync, and get commands with the reduce operation, packed binary instead of ASCII output files. It is noted that in Q4 111,871 PETSc iterations over 755 STOMP iterations were completed versus 106,272 PETSc and 722 STOMP iterations in Q2. Thus, the gains per iteration are even greater than reported in the table, but quantifying the differences is an unnecessary detail here. However, it is noted that the added FP OPs in Q4 are partly accounted for by this fact, and the new reduction-based I/O write routine.

While this was sufficient to satisfy the performance metric guidance, we are currently optimizing parallel I/O routines based on Lustre file system calls. It is necessary to briefly describe the unused features of the tar-

get Lustre [113] file system before describing an algorithm designed explicitly to deliver better performance and scalability for STOMP I/O phases. (Wikipedia provides a decent description of the components of a Lustre configuration [114].) The meta-data servers (MDS) of the Lustre file system maintain the file names and directories in the filesystem, and state management of these (ie file open, close, etc.). The object storage targets (OSTs) store chunks of files as data objects which may be stripped across one or more OSTs according to a number of prescibed patterns. The object storage servers (OSSs) provide file service, and network request handling for a set of OSTs. The Spider Lustre installation at ORNL has 672 OSTs at 7 TB storage capacity per OST. By default, user jobs are given a 4 OST target with a 1 MB default stripe access size.

The details of the Q2 problem make for an interesting I/O write scenario because the magnitude of data to be moved to disk is small and the process of assembling the data requires the coordination of a distributed set of processes with an imposed index mapping. To wit, the problem is on a $180 \times 200 \times 63$ spatial lattice and distributed over some virtual process volume (ie, $52800 \sim 60 \times 44 \times 20$ in Q2). Each observable is a double precision element defined at each lattice point in space. For the stated problem, the major I/O write events occur when all 136 double precision fields and 2 integer fields are written per check point plus 15 double precision fields for plotting. Thus, there are 151 double precision fields that will need to be written -the dominant part of these I/O events. This equates to 2,739,744,000 BYTEs, or 2.55 GB per big event as previously noted. For the Q2 problem, 5 such events take place at 0, 6, 12, 18, 24 Hrs in the simulation time.

Several stand alone codes were designed to enhance the performance of the I/O read and write phases for the described Q2 problem. In eSTOMP, the GA-based I/O write routine uses pre-set memory buffers, a *global array*, that allow for remote put and get operations. When it is time to write the observables or a check point event, then processes execute a loop over the fields copying a buffer of $4N_xN_y$ elements at a time from the work array to the global array for subsequent fields, the root process gets the data in an ordered manner and writes it to disk. This process is inherently sequential since it demands that the writing process assemble the data and write one buffer of a single field observable at a time, and has clear opportunities for enhancements. The results presented are for the write operation (read enhancements are similar and essentially the reverse of the writes) and demonstrate the gains from executing a synchronous MPI algorithm that allows two different

levels of parallelism to be introduced in the collection of distributed data / formation of the natural indexing of the physical fields, and the process of writing the data to Lustre with multiple writers simultaneously.

The new routine defines $niopes = kio \cdot nfld$ I/O processes per major event where there are kio I/O PEs per field, and $nfld$ fields in general. Process ranks and classes modulo $nfld$ are used to define the process IDs to which the field data is copied. Within these classes, kio sub-communicators are formed to separate the data magnitude temporarily stored per I/O process, and further increase the scalability and performance of network assembly of the data by simulatneous communications through different communicators. The routine is synchronous in the sense that each process is involved in the transformation within one of the kio groups per field variable since the simulation data is domain decomposed and one must visit each field variable to complete this phase.

Once all the data is reorganized to write buffers, a parallel write with offset is made into a single Lustre file per event. The performance of this algorithm depends on the manner in which the Lustre partition is configured. In order to determine the correct set of parameters to use for the eSTOMP problem, the new algorithm, and the Lustre file system, an oracle code was written to execute the Q2 problem parameters over different initializations of the algorithm and file system, i.e. an exhaustive search was performed for parameter optimization. In this example, 3300 PEs from the target architecture were used. This resource allocation was decided because after the corrections to the memory problems were implemented, the Q2 problem could execute to completion cleanly with this size request and in less than a third of the time as the Q2 allocation required. The parameters that the oracle code sets in the file system that are related to but independent from the problem parameters are the number of OSTs, the stripe size in BYTEs, and the access pattern. In our runs, a round robin stripe pattern was always used, the range of stripe sizes tested was $1MB, 2MB, 4MB, 8MB, 16MB$, and the range of OSTs was $1, 2, 4, 8, 16, 32$. The algorithm parameters tested values of kio in $1, 2, 3, 4, 6, 8$, and, given $nfld = 151$ for the Q2 problem, the $niopes$ tested were $151, 302, 453, 604, 906, 1208$. The results of the runs are plotted in Figure [45], and the best reproducible runs with the new algorithm are printed in Table [29]. It is noted that the best version of the new algorithm is 54.5X faster than the ASCII version of the I/O write phase for 5 checkpoints, and 10.7X faster than the best binary version of the I/O write phase for 5 checkpoints. These codes are now being incorporated into

production code but were not ready for production testing given the tight deadlines of this exercise.

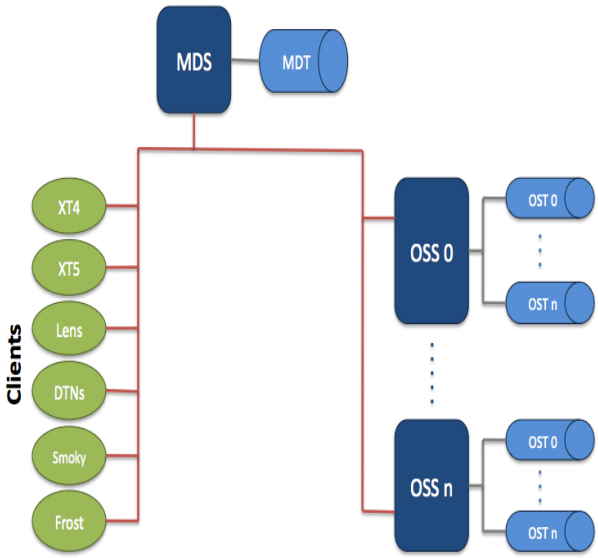

Figure 44: Spider file system configuration at ORNL's NCCS.

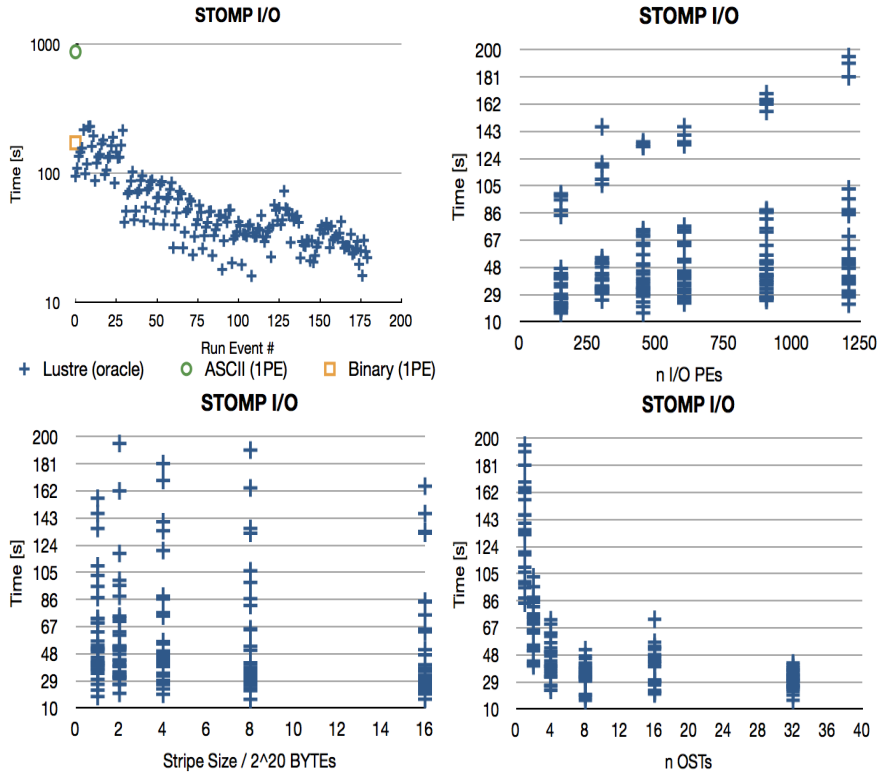

Figure 45: Results of new STOMP I/O Lustre-based I/O algorithm parameter study.

Lustre-based STOMP I/O			
Time[s]	Stripe Size [B]	n OSTs	n I/O PEs
16.064482	8388608	8	151
16.156173	16777216	32	453
18.029697	1048576	8	151
19.627054	4194304	8	151
19.776406	16777216	32	151
STOMP ASCII Unformatted I/O			
875.422139			1
STOMP Binary Formatted I/O			
172.573337			1

Table 29: Consistently fast parameter sets employing the Lustre explicit STOMP I/O algorithm. Only paramemeter sets that completed *all* the Q2 problem I/O in less than $20s$ are printed. The stand-alone MPI test codes that exactly mimick STOMP's GA-based I/O are compared.

Bibliography

[1] G. E. Moore, "Cramming more components onto integrated circuits", Electronics **38**, 114-117 (1965).

[2] W. Fichtner, D. J. Rose, and R. E. Blank, "Semiconductor device simulation", IEEE Trans. on Elec. Dev. **30**, pp. 1018-1030 (1983).

[3] C. S. Rafferty, M. R. Pinto, and R. W. Dutton, "Iterative methods in semiconductor device simulation", IEEE Trans. on Elec. Devices **32**, pp. 2018-2027 (1985).

[4] S. Selberherr, A. Schutz, and H. W. Potzl, "MINIMOS-A two-dimensional MOS transistor analyzer", IEEE Trans. on Elec. Devices **27**, pp. 1540-1550 (1980).

[5] C. Jacoboni and L. Reggiani "The Monte Carlo method for the solution of charge transport in semiconductors with applications to covalent materials", Reviews of Modern Physics **55**, 645, (1983).

[6] M. V. Fischetti and S. E. Laux, "Monte carlo analysis of electron transport in small semiconductor devices including band-structure and space-charge effects", Phys. Rev. B **38**, 9721 (1988).

[7] U. Ravaioli and D. K. Ferry, "MODFET Ensemble Monte Carlo Model Including the Quasi-Two-Dimensional Electron Gas", IEEE Trans. on Elec. Devices **33**, pp. 677-681 (1986).

[8] T. Wang and K. Hess, "Calculation of the Electron Velocity Distribution in High Electron Mobility Transistors Using an Ensemble Monte Carlo Method", J. of App. Phys. **57**, pp. 5336-5344 (1985).

[9] M. Luisier, G. Klimeck, A. Schenk, and W. Fichtner, "Atomistic Simulation of Nanowires in the $sp^3d^5s^*$ Tight-Binding Formalism: from Boundary Conditions to Strain Calculations, Phys. Rev. B, **74**, 205323 (2006).

[10] M. Luisier, A. Schenk, and W. Fichtner, "Three-Dimensional Full-Band Simulations of Si Nanowire Transistors", Proc. of International Electron Devices Meeting (IEDM) 2006, 811 (2006).

[11] M. Luisier and G. Klimeck, "Full-band and atomistic simulation of n- and p-doped double-gate MOSFETs for the 22nm technology node", Int. Conf. on Simulation of Semiconductor Processes and Devices (SISPAD) Hakone, Japan (2008).

[12] M. Luisier and G. Klimeck, "Atomistic, Full-Band Design Study of InAs Band-to-Band Tunneling Field-Effect Transistors", IEEE Elec. Dev. Lett. **30**, 602-604 (2009).

[13] M. Luisier and G. Klimeck, "Atomistic full-band simulations of silicon nanowire transistors: Effects of electron-phonon scattering", Phys. Rev. B **80**, 155430 (2009).

[14] M. Fuechsle, S. Mahapatra, FA Zwanenburg, M. Friesen, MA Eriksson, and M.Y. Simmons. Spectroscopy of few-electron single-crystal silicon quantum dots. *Nature Nanotechnology*, 5(7):502–505, 2010.

[15] G. Klimeck, S.S. Ahmed, N. Kharche, M. Korkusinski, M. Usman, M. Prada, and T.B. Boykin. Atomistic Simulation of Realistically Sized Nanodevices Using NEMO 3-DPart II: Applications. *IEEE Transactions on Electron Devices*, 54(9):2090–2099, 2007.

[16] J. C. Slater and G. F. Koster, "Simplified LCAO Method for the Periodic Potential Problem", Phys. Rev. **94**, 1498-1524 (1954).

[17] Z. Sui and I. P. Herman, "Effect of strain on phonons in Si, Ge, and Si/Ge heterostructures", Phys. Rev. B **48**, 17938 (1993).

[18] M. Luisier, A. Schenk, and W. Fichtner, "Three-Dimensional Full-Band Simulations of Si Nanowire Transistors", IEDM Tech. Digest **2006**, 811 (2006).

[19] M. Luisier, A. Schenk, and W. Fichtner, "Atomistic treatment of interface roughness in Si nanowire transistors with different channel orientations", Appl. Phys. Lett. **90**, 102103 (2007).

[20] M. Luisier, A. Schenk, and W. Fichtner, "Full-band atomistic study of source-to-drain tunneling in Si nanowire transistors", Int. Conf. on Simulation of Semiconductor Processes and Devices (SISPAD) Vienna, Austria (2007).

[21] M. Luisier and G. Klimeck, "Full-band and atomistic simulation of n- and p-doped double-gate MOSFETs for the 22nm technology node", Int. Conf. on Simulation of Semiconductor Processes and Devices (SISPAD) Hakone, Japan (2008).

[22] M. Luisier and G. Klimeck, "Full-Band and Atomistic Simulation of Realistic 40 nm InAs HEMT", IEDM Tech. Digest **2008**, 887-890 (2008).

[23] N. Kharche, G. Klimeck, D.-H. Kim, J. A. del Alamo, and M. Luisier, "Performance Analysis of Ultra-Scaled InAs HEMTs", IEDM 2009, Baltimore MD, USA (2009).

[24] M. Luisier and G. Klimeck, "Performance limitations of graphene nanoribbon tunneling FETS due to line edge roughness" 66th Device Research Conference, Penn State University, PA (2009).

[25] M. Luisier and G. Klimeck, "Performance analysis of statistical samples of graphene nanoribbon tunneling transistors with line edge roughness", App. Phys. Lett. **94**, 223505 (2009).

[26] M. Luisier and G. Klimeck, "Atomistic, Full-Band Design Study of InAs Band-to-Band Tunneling Field-Effect Transistors", IEEE Elec. Dev. Lett. **30**, 602-604 (2009).

[27] M. Luisier and G. Klimeck, "Investigation of $In_xGa_{1-x}As$ Ultra-Thin-Body Tunneling FETs using a Full-Band and Atomistic Approach", International Conference on Simulation of Semiconductor Processes and Devices, SISPAD 2009, San Diego CA, USA (2009).

[28] M. Luisier and G. Klimeck, "Performance Comparisons of Tunneling Field-Effect Transistors made of InSb, Carbon, and GaSb-InAs Broken Gap Heterostructures", IEDM 2009, Baltimore MD, USA, (2009).

[29] S. Agarwal, G. Klimeck, and M. Luisier, "Leakage Reduction Design Concepts for Low Power Vertical Tunneling Field-Effect Transistors, accepted for publication in IEEE Elec. Dev. Lett. (2010).

[30] M. Luisier and G. Klimeck, "Simulations of Nanowire Tunneling Transistors: from the WKB Approximation to Full-Band Phonon-Assisted Tunneling accepted for publication in J. of App. Phys. (2010).

[31] T. A. Davis, "A column pre-ordering strategy for the unsymmetric-pattern multifrontal method", ACM Trans. on Math. Software **30**, 165 (2004).

[32] O. Schenk and K. Gärtner, "Solving Unsymmetric Sparse Systems of Linear Equations with PARDISO, Journal of Future Generation Computer Systems", J. of Future Generation Computer Systems **20**, 475 (2004).

[33] X. S. Li and J. W. Demmel "SuperLU_DIST: A Scalable Distributed Memory Sparse Direct Solver for Unsymmetric Linear Systems", ACM Trans. on Math. Software **29**, 110 (2003).

[34] P. R. Amestoy, I. S. Duff, and J.-Y. L'Excellent, "Multifrontal parallel distributed symmetric and unsymmetric solvers" Comput. Methods in Appl. Mech. Eng. **184**, 501 (2000).

[35] T. B. Boykin, M. Luisier, and G. Klimeck, "Multi-band transmission calculations for nanowires using an optimized renormalization method", Phys. Rev. B **77**, 165318 (2008).

[36] M. Luisier, A. Schenk, W. Fichtner, T. B. Boykin, and G. Klimeck, "A parallel sparse linear solver for nearest-neighbor tight-binding problems", Proc. of the 14th international Euro-Par conference on Parallel Processing, 790-800 (2008).

[37] J. Dongarra, "Basic Linear Algebra Subprograms Technical Forum Standard", International Journal of High Performance Applications and Supercomputing, **16**, 1–111 (2002).

[38] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and

D. Sorensen, "LAPACK User's Guide", Third Edition, SIAM, Philadelphia (1999).

[39] S. Datta, "Electronic Transport in Mesoscopic Systems", Cambridge University Press, Cambridge (1995).

[40] P. M. Gresho and R. L. Sani, "Incompressible Flow and the Finite Element Method: Isothermal Laminar Flow", John Wiley and Sons, New York (2000).

[41] R. E. Bank, D. J. Rose, and W. Fichtner, "Numerical Methods for Semiconductor Device Simulation", IEEE Trans. Electron Dev. **30**, 1031 (1983).

[42] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls", ACM Trans. Math. Softw. **22**, 469 (1996).

[43] R. S. Tuminaro, M. Heroux, S. A. Hutchinson, and J. N. Shadid, "Official Aztec User's Guide: Version 2.1" (1999).

[44] D. H. Kim and J. A. del Alamo, "30-nm InAs Pseudomorphic HEMTs on an InP Substrate With a Current-Gain Cutoff Frequency of 628 GHz", IEEE Elec. Dev.Lett. **29**, 830-833 (2008).

[45] M. Luisier, "A parallel implementation of electron-phonon scattering in nanoelectronic devices up to 95k cores",Proceedings of the 2010 ACM/IEEE Conference on Supercomputing, (2010).

[46] S. Jin, Y. J. Park, and H. S. Min. A three-dimensional simulation of quantum transport in silicon nanowire transistor in the presence of electron-phonon interactions. *J. Appl. Phys.*, 99:123719, 2006.

[47] G. Klimeck. Quantum and semi-classical transport in RTDs in NEMO 1-D. *Journal of Computational Electronics*, 2:177, 2003.

[48] S. Balay, K. Buschelman, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, B.F. Smith, and H. Zhang. PETSc users manual. 2003.

[49] Vicente Hernandez, Jose E. Roman, and Vicente Vidal. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Transactions on Mathematical Software*, 31(3):351–362, September 2005.

[50] B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey. `libMesh`: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations. *Engineering with Computers*, 22(3–4):237–254, 2006. `http://dx.doi.org/10.1007/s00366-006-0049-3`.

[51] S. Steiger, M. Povolotskyi, T. Kubis, H.-H. Park, B. Haley, and G. Klimeck. Nemo 5: A parallel multiscale nanoelectronics modeling tool. *Manuscript in preparation*, 2011.

[52] http://www.itrs.net/

[53] P. Packan et al., 'High Performance 32nm Logic Technology Featuring 2nd Generation High-k + Metal Gate Transistors", Proc. of the International Electron Devices Meeting (IEDM) 2009, paper 28.4 (2009).

[54] M.A. Kastner, O. Klein, T.M. Lyszczarz, P. Mankiewich, D.C. Shaver, S. Wind, D. Abusch-Magder, D.J. Goldhaber-Gordon, and N.Y. Morgan. Artificial Atoms. *Phys. Rev. Lett*, 63:1893, 1990.

[55] M. Usman, H. Ryu, I. Woo, D.S. Ebert, and G. Klimeck. Moving Toward Nano-TCAD Through Multimillion-Atom Quantum-Dot Simulations Matching Experimental Data. *Nanotechnology, IEEE Transactions on*, 8(3):330–344, 2009.

[56] P. Michler, A. Kiraz, C. Becher, WV Schoenfeld, PM Petroff, L. Zhang, E. Hu, and A. Imamoglu. A quantum dot single-photon turnstile device. *Science*, 290(5500):2282, 2000.

[57] MV Maximov, Y.M. Shernyakov, AF Tsatsulnikov, AV Lunev, AV Sakharov, VM Ustinov, A.Y. Egorov, AE Zhukov, AR Kovsh, PS Kopev, et al. High-power continuous-wave operation of a InGaAs/AlGaAs quantum dot laser. *Journal of Applied Physics*, 83:5561, 1998.

[58] S. Fafard, Z. R. Wasilewski, C. Ni. Allen, D. Picard, M. Spanner, J. P. McCaffrey, and P. G. Piva. Manipulating the energy levels of semiconductor quantum dots. *Phys. Rev. B*, 59(23):15368–15373, Jun 1999.

[59] M. Luisier and G. Klimeck, "Numerical strategies towards peta-scale simulations of nanoelectronics devices", Parallel Computing **36**, 117-128 (2010).

[60] S. Abraham, Spencer Abraham announces department of energy 20-year science facility plan, text available at $http://www.er.doe.gov/Sub/Newsroom/News_Releases/DOE-SC/2003/Energy20-YearScienceFacilityPlan.htm$, 2003.

[61] E.W. Bethel, et al. "Occam's Razor and Petascale Visual Data Analysis," Journal of Physics: Conference Series 180, 012084 (2009).

[62] C. K. Birdsall, A. B. Langdon, V. Vehedi, and J. P. Verboncoeur, Plasma Physics via Computer Simulations, Adam Hilger, Bristol, 1991.

[63] I. Blumenfeld et al., Nature 445, 741 (2007).

[64] C.E. Clayton et al., Self-guided laser wakefield acceleration beyond 1 GeV using ionization-induced injection, to appear in Phys. Rev. Lett.

[65] E. Cormier-Michel, et al. , "Unphysical kinetic effects in laser wakefield accelerators modeled with particle-in-cell codes," Phys. Rev. E. 78, 016404 (2008).

[66] E. Cormier-Michel, et al. "Control of focusing fields in laser-plasma accelerators using higher-order modes," PRSTAB submitted.

[67] B.M. Cowan, et al, "Characteristics of an envelope model for laser plasma accelerator simulation", J. Comput. Phys., submitted Computer Physics Communications.

[68] V. Decyk, T. Singh, and S. Friedman, Graphical Processing Unit-Based Simulations, in Proc. Intl. Computational Accelerator Physics Conf. (ICAP2009), 2009,

[69] Viktor K. Decyk and Tajendra V. Singh, submitted to

[70] E. Esarey et al., IEEE Trans. Plasma Sci. 24, 252 (1996).

[71] E. Esarey et al., Phys. Rev. Lett. 79, 2682 (1997).

[72] J. Faure et al., Nature 431, 541 (2004).

[73] J. Faure et al., Nature 444, 737 (2006).

[74] R.A.Fonscea et al., Lect. Notes Comp. Science 2329, III-342, Springer-Verlag, (2002).

[75] C. G. R. Geddes et al., Nature 431, 538 (2004).

[76] C.G.R. Geddes et al. "Production of high-quality electron bunches by dephasing and beam loading in channeled and unchanneled laser plasma accelerators," Phys. Plasmas 12, 056709 (2005).

[77] C. G. R. Geddes et al., J. Phys. Conf. Ser. 78, 12021 (2007).

[78] C.G.R. Geddes et al, Phys. Rev. Lett. 100, 215004 (2008).

[79] Geddes et al. "Laser Plasma Particle Accelerators: Large Fields for Smaller Facility Sources," SciDAC Review 13 (Summer, 2009);http://www.scidacreview.org/0903/index.html

[80] A.J. Gonsalves et al., in Proc. Particle Accel. Conf., IEEE, Piscataway, NJ (2007).

[81] Daniel F. Gordon et al., IEEE Trans. Plasma Sci, Vol. 28, 1135 (2000).

[82] C.K. Huang et al. Journal of Computational Physics 217 658-679 (2006).

[83] C. Huang, et al., "Recent results and future challenges for large scale Particle-In-Cell simulations of plasma-based accelerator concepts", J. of Physics: Conference Series 180, 012005 (2009).

[84] C. Joshi, "Plasma accelerators," Scientific American, Vol. 294, 41-47 (Feb. 2006).

[85] N. Kirby et al. "Transverse emittance and current of multi-GeV trapped electrons in a plasma wakefield accelerator," Phys. Rev. STAB 12, 051302 (2009).

[86] X. Kong, et al., "Particle-in-cell Simulaions with Charge- Conserving Current Deposition on Graphic Processing Units," submitted to J. Computational Phys.,2010.

[87] S. P. D. Mangles et al., Nature 431, 535 (2004).

[88] W. P. Leemanset al., 2005 IEEE Trans. Plasma Sci. 33, 8 (2005).

[89] W. P. Leemans et al., Nature Physics 2, 696 (2006).

[90] W.P. Leemans and E.H. Esarey, "Laser-driven plasma-wave electron accelerators," Physics Today 2009.

[91] W. Lu et al in Phys. Rev. ST Accel. Beams 10, 061301 (2007)

[92] Samuel F. Martins et al., "Exploring laser-wakefield-accelerator regimes for near-term lasers using particle-in-cell simulation in Lorentz-boosted frames," Nature Physics, 6, 311 - 316 (2010)

[93] Samuel F. Martins et al., Computer Physics Communications, 181, 869-875 (2010).

[94] E. Cormier-Michel et al., accepted to PRE (2008).

[95] P. Mora, T.M. Antonsen Jr., Kinetic modeling of intense, short laser pulses propagating in tenuous plasmas, Phys. Plasmas 4 (1997) 217

[96] C. Nieter and J. Cary, J. Comp. Phys. 196, 448 (2004).

[97] E. Oz, et al., "Ionization-induced electron trapping in ultra-relativistic plasma wakes," Physical Review Letters. 98, 084801(2007).

[98] K. Paul, et al., "Benchmarking the codes VORPAL, OSIRIS and Quick-PIC with Laser Wakefield Acceleration Simulations," Proc. Advanced. Accel. Concepts Workshop (2008).

[99] Pak A., et al., "Injection and trapping of tunnel ionized electrons into laser-produced wakes," Physical Review Letters, Vol. 104, 025003 (January 2010).

[100] A. Pukhov and J. Meyer-ter-Vehn , 2002 Appl. Phys. B 74, 355 .

[101] O. Rubel, et al., "Coupling Visualization and Data Analysis for Knowledge Discovery from Multi-dimensional Scientific Data," Procedia Computer Science 1, 1751 (2010).

[102] Tajima T. and Dawson J. M. , Phys. Rev. Lett. 43, 267 (1979)

[103] F. S. Tsung et al., Phys. Rev. Lett. 93, 185002 (2004) .

[104] F. S. Tsung et al., Physics of Plamas 13, 056708 (2006) .

[105] M. Tzoufras et al., Phys. Rev. Lett. vol. (2008).

[106] M. Tzoufras, PhD Dissertation UCLA 2008.

[107] Vay J.-L., Phys. Rev. Lett. 98, 130405 (2007).

[108] J.-L. Vay, et al., "Modeling laser wakefield accelerators in a Lorentz boosted frame", submitted to J. Comput. Phys.

[109] S. Plimpton, Fast Parallel Algorithms for Short-Range Molecular Dynamics, J Comp Phys, 117, 1-19 (1995).

[110] Lennard-Jones, J. E. , "On the Determination of Molecular Fields," pp463-477, Proc. R. Soc. Lond. A 106 (738) (1924).

[111] Daw, Baskes, Phys Rev Lett, 50, 1285 (1983).

[112] Daw, Baskes, Phys Rev B, 29, 6443 (1984).

[113] `http://wiki.lustre.org/index.php/Main_Page` .

[114] `http://en.wikipedia.org/wiki/Lustre_(file_system)` .

Appendices

0.5 A. Software Libraries

0.5.1 Modules Available on the Target Architectures

Q2 Modules: *jaguarpf.ccs.ornl.gov*

```
------------------------------------ /opt/cray/xt-asyncpe/3.7/modulefiles ------------------------------------
xtpe-barcelona     xtpe-mc12          xtpe-shanghai
xtpe-istanbul      xtpe-mc8           xtpe-target-native

---------------------------------------------- /opt/modulefiles ----------------------------------------------
Base-opts/2.1.27HD                                    pgi/10.1.0
Base-opts/2.1.27HD.lusrelsave                         pgi/10.2.0
Base-opts/2.1.29HD                                    pgi/10.3.0(default)
Base-opts/2.1.29HD.lusrelsave                         pgi/10.4.0
Base-opts/2.1.41HD                                    pgi/10.5.0
Base-opts/2.1.41HD.lusrelsave                         pgi/10.8.0
Base-opts/2.1.50HD                                    pgi/11.0.0
Base-opts/2.1.50HD.lusrelsave                         pgi/6.2.5
Base-opts/2.1.50HD_PS08                               pgi/7.0.7
Base-opts/2.1.50HD_PS08.lusrelsave                    pgi/7.1.6
Base-opts/2.2.27                                      pgi/7.2.3
Base-opts/2.2.27.lusrelsave                           pgi/7.2.4
Base-opts/2.2.31                                      pgi/7.2.5
Base-opts/2.2.31.lusrelsave                           pgi/8.0.1
Base-opts/2.2.31A                                     pgi/8.0.2
Base-opts/2.2.31A.lusrelsave                          pgi/8.0.3
Base-opts/2.2.41                                      pgi/8.0.4
Base-opts/2.2.41.lusrelsave                           pgi/8.0.5
Base-opts/2.2.41A                                     pgi/8.0.6
Base-opts/2.2.41A.lusrelsave                          pgi/9.0.1
Base-opts/2.2.73(default)                             pgi/9.0.2
Base-opts/2.2.73.lusrelsave                           pgi/9.0.3
PrgEnv-cray/1.0.0                                     pgi/9.0.4
PrgEnv-cray/1.0.1(default)                            pkgconfig/0.15.0(default)
PrgEnv-cray/1.0.2                                     torque/2.3.2-snap.200807092141
PrgEnv-gnu/2.1.27HD                                   torque/2.4.1b1-snap.200905191614(default)
PrgEnv-gnu/2.1.29HD                                   xt-asyncpe/1.0c
PrgEnv-gnu/2.1.41HD                                   xt-asyncpe/1.1
PrgEnv-gnu/2.1.50HD                                   xt-asyncpe/1.2
PrgEnv-gnu/2.1.50HD_PS08                              xt-asyncpe/2.0
PrgEnv-gnu/2.2.27                                     xt-asyncpe/2.0.34
PrgEnv-gnu/2.2.31                                     xt-asyncpe/2.1
PrgEnv-gnu/2.2.31A                                    xt-asyncpe/2.3
PrgEnv-gnu/2.2.41                                     xt-asyncpe/2.4
PrgEnv-gnu/2.2.41A                                    xt-asyncpe/2.5
PrgEnv-gnu/2.2.73(default)                            xt-asyncpe/3.0
PrgEnv-intel/1.0.0(default)                           xt-asyncpe/3.1
PrgEnv-pathscale/2.1.27HD                             xt-asyncpe/3.1.20
PrgEnv-pathscale/2.1.29HD                             xt-asyncpe/3.2
PrgEnv-pathscale/2.1.41HD                             xt-asyncpe/3.3
PrgEnv-pathscale/2.1.50HD                             xt-asyncpe/3.4
PrgEnv-pathscale/2.1.50HD_PS08                        xt-asyncpe/3.5
PrgEnv-pathscale/2.2.27                               xt-asyncpe/3.7(default)
PrgEnv-pathscale/2.2.31                               xt-asyncpe/3.8
```

```
PrgEnv-pathscale/2.2.31A
PrgEnv-pathscale/2.2.41
PrgEnv-pathscale/2.2.41A
PrgEnv-pathscale/2.2.73(default)
PrgEnv-pgi/2.1.27HD
PrgEnv-pgi/2.1.29HD
PrgEnv-pgi/2.1.41HD
PrgEnv-pgi/2.1.50HD
PrgEnv-pgi/2.1.50HD_PS08
PrgEnv-pgi/2.2.27
PrgEnv-pgi/2.2.31
PrgEnv-pgi/2.2.31A
PrgEnv-pgi/2.2.41
PrgEnv-pgi/2.2.41A
PrgEnv-pgi/2.2.73(default)
acml/4.0.1a
acml/4.1.0
acml/4.2.0
acml/4.3.0(default)
acml/4.4.0
apprentice2/5.0.0
apprentice2/5.0.1
apprentice2/5.0.2(default)
apprentice2/5.1.0
blcr/0.7.3
cce/7.0.0
cce/7.0.1
cce/7.0.2
cce/7.0.3
cce/7.0.4
cce/7.1.0
cce/7.1.1
cce/7.1.2
cce/7.1.3
cce/7.1.4.111
cce/7.1.5(default)
cce/7.1.6
cce/7.2.4
cce/7.2.7
cce/7.3.1
cray/MySQL/5.0.64-1.0000.2342.16.1
cray/account/1.0.0-2.0202.19482.49.17
cray/audit/1.0.0-1.0202.22161.0
cray/csa/3.0.0-1_2.0202.21426.77.6
cray/job/1.5.5-0.1_2.0202.21413.56.6
cray/projdb/1.0.0-1.0202.19483.52.1
dwarf/8.2.0
dwarf/8.4.0
dwarf/8.6.0
dwarf/8.8.0
dwarf/9.5.0
dwarf/9.9.0(default)
elf/0.8.10
elf/0.8.12(default)
fftw/2.1.5
fftw/2.1.5.1
fftw/3.1.1
xt-asyncpe/3.9
xt-asyncpe/4.0
xt-asyncpe/4.5
xt-atp/1.0(default)
xt-boot/2.1.27HD
xt-boot/2.1.29HD
xt-boot/2.1.41HD
xt-boot/2.1.50HD
xt-boot/2.1.50HD_PS08
xt-boot/2.2.27
xt-boot/2.2.31
xt-boot/2.2.31A
xt-boot/2.2.41
xt-boot/2.2.41A
xt-boot/2.2.73
xt-catamount/2.1.27HD
xt-catamount/2.1.29HD
xt-catamount/2.1.41HD
xt-catamount/2.1.50HD
xt-catamount/2.1.50HD_PS08
xt-craypat/5.0.0
xt-craypat/5.0.1
xt-craypat/5.0.2(default)
xt-craypat/5.1.0
xt-lgdb/1.2(default)
xt-libc/2.1.27HD
xt-libc/2.1.29HD
xt-libc/2.1.41HD
xt-libc/2.1.50HD
xt-libc/2.1.50HD_PS08
xt-libsci/10.2.1
xt-libsci/10.3.1
xt-libsci/10.3.5
xt-libsci/10.3.6
xt-libsci/10.3.8
xt-libsci/10.3.8.1
xt-libsci/10.3.9
xt-libsci/10.4.0
xt-libsci/10.4.1
xt-libsci/10.4.2
xt-libsci/10.4.3
xt-libsci/10.4.4(default)
xt-libsci/10.4.5
xt-libsci/10.5.0
xt-lustre-ss/2.1.27HD_1.6.5
xt-lustre-ss/2.1.29.HD_ORNL.nic1_1.6.5
xt-lustre-ss/2.1.29HD_1.6.5
xt-lustre-ss/2.1.29HD_ORNL.nic10_1.6.5
xt-lustre-ss/2.1.29HD_ORNL.nic11_1.6.5
xt-lustre-ss/2.1.29HD_ORNL.nic12_1.6.5
xt-lustre-ss/2.1.29HD_ORNL.nic2_1.6.5
xt-lustre-ss/2.1.29HD_ORNL.nic5_1.6.5
xt-lustre-ss/2.1.29HD_ORNL.nic6_1.6.5
xt-lustre-ss/2.1.41HD_1.6.5
xt-lustre-ss/2.1.50HD.PS04.lus.1.6.5.steve.8062_1.6.5
xt-lustre-ss/2.1.50HD.PS04.lus.1.6.5.steve.8072_1.6.5
xt-lustre-ss/2.1.50HD.PS08.lus.1.6.5.steve.8099_1.6.5
```

```
fftw/3.2.0
fftw/3.2.1
fftw/3.2.2
fftw/3.2.2.1(default)
fftw/3.2.2.1.bak
gcc/3.3.3
gcc/4.1.2
gcc/4.2.0.quadcore
gcc/4.2.3
gcc/4.2.4
gcc/4.3.2
gcc/4.4.1
gcc/4.4.2(default)
gcc/4.4.3
gcc/4.4.4
gcc-catamount/3.3
gnet/2.0.5
hdf5/1.8.2.2
hdf5/1.8.2.3
hdf5/1.8.3.0
hdf5/1.8.3.1(default)
hdf5/1.8.4.0
hdf5/1.8.4.1
hdf5/1.8.5.0
hdf5-parallel/1.8.2.2
hdf5-parallel/1.8.2.3
hdf5-parallel/1.8.3.0
hdf5-parallel/1.8.3.1
hdf5-parallel/1.8.4.0
hdf5-parallel/1.8.4.1(default)
hdf5-parallel/1.8.5.0
intel/11.1.046(default)
intel/11.1.064
iobuf/1.0.6(default)
java/jdk1.6.0_17
java/jdk1.6.0_20(default)
libfast/1.0
libfast/1.0.2
libfast/1.0.3
libfast/1.0.4
libfast/1.0.5
libfast/1.0.6
libfast/1.0.7(default)
libfast/orig.1.0.3
libscifft-pgi/1.0.0(default)
mazama/5.1.0(default)
moab/5.2.3
moab/5.2.4
moab/5.3.0
moab/5.3.3
moab/5.3.6(default)
modules/3.1.6
modules/3.1.6.5(default)
mrnet/2.0.1.1(default)
mrnet/2.2.0.1
mrnet/3.0.0
netcdf/3.6.2
xt-lustre-ss/2.1.50HD.PS08.lus.1.6.5.steve.8119_1.6.5
xt-lustre-ss/2.1.50HD_1.6.5
xt-lustre-ss/2.1.50HD_PS04_1.6.5
xt-lustre-ss/2.1.50HD_PS08_1.6.5
xt-lustre-ss/2.1.UP00_ORNL.nic12_1.6.5
xt-lustre-ss/2.1.UP00_ORNL.nic2_1.6.5
xt-lustre-ss/2.1.UP00_ORNL.nic30_1.6.5
xt-lustre-ss/2.1.UP00_ORNL.nic3_1.6.5
xt-lustre-ss/2.1.UP00_ORNL.nic40_1.6.5
xt-lustre-ss/2.1.UP00_ORNL.nic51_1.6.5
xt-lustre-ss/2.1.UP00_ORNL.nic52_1.6.5
xt-lustre-ss/2.2.27.lus.1.6.5.steve.8154_1.6.5
xt-lustre-ss/2.2.27.lus.1.6.5.steve.8182_1.6.5
xt-lustre-ss/2.2.27.lus.1.6.5.steve.8189_1.6.5
xt-lustre-ss/2.2.27_1.6.5
xt-lustre-ss/2.2.31.lus.1.6.5.steve.8211_1.6.5
xt-lustre-ss/2.2.31A.lus.1.6.5.steve.8259_1.6.5
xt-lustre-ss/2.2.31A_1.6.5
xt-lustre-ss/2.2.31_1.6.5
xt-lustre-ss/2.2.41A_1.6.5
xt-lustre-ss/2.2.41_1.6.5
xt-lustre-ss/2.2.73_1.6.5
xt-lustre-ss/2.2.UP01.lus.1.6.5.steve.8265_1.6.5
xt-lustre-ss/2.2_1.6.5
xt-mpt/2.1.27HD
xt-mpt/2.1.29HD
xt-mpt/2.1.41HD
xt-mpt/2.1.50HD
xt-mpt/2.1.50HD_PS08
xt-mpt/3.0.1
xt-mpt/3.0.2
xt-mpt/3.0.4
xt-mpt/3.1.0
xt-mpt/3.1.0.4
xt-mpt/3.1.0.6
xt-mpt/3.1.0.7
xt-mpt/3.1.1
xt-mpt/3.1.2
xt-mpt/3.2.0
xt-mpt/3.3.0
xt-mpt/3.4.0
xt-mpt/3.4.1
xt-mpt/3.4.2
xt-mpt/3.5.0
xt-mpt/3.5.1
xt-mpt/4.0.0(default)
xt-mpt/4.0.2
xt-mpt/4.0.3
xt-mpt/4.1.0.1
xt-mpt/4.1.1
xt-mpt/5.0.0
xt-os/2.1.27HD
xt-os/2.1.29HD
xt-os/2.1.41HD
xt-os/2.1.50HD
xt-os/2.1.50HD_PS08
xt-os/2.2.27
```

```
netcdf/4.0.0.2                                      xt-os/2.2.31
netcdf/4.0.0.3                                      xt-os/2.2.31A
netcdf/4.0.1.0                                      xt-os/2.2.41
netcdf/4.0.1.1                                      xt-os/2.2.41A
netcdf/4.0.1.2                                      xt-os/2.2.73
netcdf/4.0.1.3(default)                             xt-papi/3.6
netcdf/4.1.1.0                                      xt-papi/3.6.1a
netcdf-hdf5parallel/4.0.0.2                         xt-papi/3.6.2
netcdf-hdf5parallel/4.0.0.3                         xt-papi/3.6.2.2
netcdf-hdf5parallel/4.0.1.0                         xt-papi/3.7.2(default)
netcdf-hdf5parallel/4.0.1.1                         xt-papi/3.7.2.0.5
netcdf-hdf5parallel/4.0.1.2                         xt-pe/2.1.27HD
netcdf-hdf5parallel/4.0.1.3(default)                xt-pe/2.1.29HD
netcdf-hdf5parallel/4.1.1.0                         xt-pe/2.1.41HD
pathscale/3.2                                       xt-pe/2.1.50HD
pathscale/3.2.99(default)                           xt-pe/2.1.50HD_PS08
petsc/2.3.3a                                        xt-pe/2.2.27
petsc/3.0.0                                         xt-pe/2.2.31
petsc/3.0.0.1                                       xt-pe/2.2.31A
petsc/3.0.0.10(default)                             xt-pe/2.2.41
petsc/3.0.0.2                                       xt-pe/2.2.41A
petsc/3.0.0.3                                       xt-pe/2.2.73
petsc/3.0.0.4                                       xt-service/2.1.27HD
petsc/3.0.0.6                                       xt-service/2.1.29HD
petsc/3.0.0.8                                       xt-service/2.1.41HD
petsc/3.1.00                                        xt-service/2.1.50HD
petsc-complex/2.3.3a                                xt-service/2.1.50HD_PS08
petsc-complex/3.0.0                                 xt-service/2.2.27
petsc-complex/3.0.0.1                               xt-service/2.2.31
petsc-complex/3.0.0.10(default)                     xt-service/2.2.31A
petsc-complex/3.0.0.2                               xt-service/2.2.41
petsc-complex/3.0.0.3                               xt-service/2.2.41A
petsc-complex/3.0.0.4                               xt-service/2.2.73
petsc-complex/3.0.0.6                               xtgdb/1.0.0(default)
petsc-complex/3.0.0.8                               xtpe-target-catamount
petsc-complex/3.1.00                                xtpe-target-cnl
pgi/10.0.0

------------------------------------------- /opt/cray/modulefiles -------------------------------------------
atp/1.0.1(default)          xt-mpich2/5.0.0(default)    xt-mpt/3.1.0.4              xt-mpt/3.5.0
atp/1.0.2                   xt-mpt/2.1.27HD             xt-mpt/3.1.0.6              xt-mpt/3.5.1
atp/1.0.3                   xt-mpt/2.1.29HD             xt-mpt/3.1.0.7              xt-mpt/4.0.0(default)
perftools/5.1.0(default)    xt-mpt/2.1.41HD             xt-mpt/3.1.1                xt-mpt/4.0.2
pmi/1.0-1.0000.7628.10.2.ss xt-mpt/2.1.50HD             xt-mpt/3.1.2                xt-mpt/4.0.3
pmi/1.0-1.0000.7901.22.1.ss xt-mpt/2.1.50HD_PS08        xt-mpt/3.2.0                xt-mpt/4.1.0.1
stat/1.0.0(default)         xt-mpt/3.0.1                xt-mpt/3.3.0                xt-mpt/4.1.1
stat/1.1.0                  xt-mpt/3.0.2                xt-mpt/3.4.0                xt-mpt/5.0.0
trilinos/10.0.1(default)    xt-mpt/3.0.4                xt-mpt/3.4.1                xt-shmem/5.0.0(default)
trilinos/10.2.0             xt-mpt/3.1.0                xt-mpt/3.4.2

------------------------------------------- /opt/modules/3.1.6 ----------------------------------------------
modulefiles/modules/dot         modulefiles/modules/module-info modulefiles/modules/null
modulefiles/modules/module-cvs  modulefiles/modules/modules     modulefiles/modules/use.own

------------------------------------------- /sw/xt5/modulefiles ---------------------------------------------
DefApps                             gptl/3.6(default)                   ompi/1.7a1r22760
MiscApps                            gptl/3.6.3                          ompi/DT/routed-pgi-uos
```

```
adios/0.9.10
adios/0.9.10_phdf5
adios/0.9.12
adios/0.9.13
adios/1.0
adios/1.1.0
adios/1.1.amr1
adios/1.2
adios/1.2.1(default)
adios/1.2_jc
adios/trunk
arpack/2008.03.11(default)
atlas/3.8.2
atlas/3.8.2-fPIC-dualcore
atlas/3.8.3
atlas/3.8.3-fPIC-dualcore(default)
autoconf/2.63(default)
automake/1.10.1(default)
automake/1.11.1
aztec/2.1(default)
banner/1.3.2(default)
blas/ref(default)
blas/ref-dualcore
blas-goto/1.0
blas-goto/2.0(default)
boost/1.44.0
bugget/2.0(default)
cdo/1.3.1
cdo/1.3.2(default)
cmake/2.6.1(default)
cmake/2.6.2
cmake/2.6.4
cmake/2.8.0
cpmd/3.13.1
cpmd/3.13.2(default)
ddt/2.4.1
ddt/2.4.1-11140
ddt/2.5.1-12323
ddt/2.6.1(default)
ddt/3.0-rc1
ddt/3.0-rc3
doxygen/1.5.6
doxygen/1.5.8
doxygen/1.5.9
doxygen/1.6.1(default)
dragon/1.0a(default)
emacs/23.1(default)
esmf/4.0.0r_O
esmf/4.0.0r_g
esmf/4.0.0rp1_O(default)
esmf/4.0.0rp1_g
esmf/4.0.0rp2_O
esmf/4.0.0rp2_g
ferret/6.1(default)
ferret/6.3
fftpack/5-r4i4
fftpack/5-r8i4
gptl_pmpi/3.6(default)
gptl_pmpi/3.6.3
grace/5.1.21
grace/5.1.22(default)
gromacs/3.3.3
gromacs/4.0.5(default)
gsl/1.11
gsl/1.11-dualcore
gsl/1.12
gsl/1.12-dualcore(default)
gv/3.6.7(default)
hdf5/1.6.7
hdf5/1.6.7_par
hdf5/1.6.8
hdf5/1.6.8_par
hdf5/1.8.1
hdf5/1.8.1_par
hdf5/1.8.2
hdf5/1.8.2_par
hpctoolkit/4.9.2(default)
hypre/2.0.0(default)
hypre/2.0.0-setdir
hypre/2.4.0b
hypre/2.4.0b-craypat
hypre/2.4.0b-debug
idl/6.4(default)
imagemagick/6.4.2(default)
imagemagick/6.5.4
iota/0.2.1
iota/0.2.2(default)
iota/0.2.3
iota/0.2.4
iota/1.0.0
java/1.5.0.06
java/1.6.0.06
java-jdk/1.5.0.06
java-jdk/1.6.0.06(default)
java-jre/1.5.0.06(default)
lammps/22Jul09
lammps/4Mar08
lammps/9Sep10
lammps/May08(default)
lapack/3.1.1(default)
lapack/3.1.1-dualcore
lapack/3.1.1-fPIC
libgd/2.0.35(default)
liblut/0.9.6
liblut/0.9.7
liblut/0.9.8
liblut/0.9.9(default)
liblut/1.0.0
libxml2/2.7.6(default)
libxslt/1.1.26(default)
lsq/0.1.0
lsq/1.0.0
m4/1.4.11(default)
makedepf90/2.8.8(default)
ompi/DT/routing-pgi
ompi/DT/routing-pgi-uos
ompi/experimental(default)
ompi/standard
p-netcdf/1.0.2
p-netcdf/1.0.3(default)
p-netcdf/1.1.1
p3dfft/2.4(default)
p3dfft/2.4-fftw
p3dfft/2.4.sav
parmetis/3.1
parmetis/3.1.1(default)
perl-gd/2.44(default)
petsc/2.3.3-debug
petsc/3.0.0-custom
petsc/3.0.0-debug
petsc-complex/2.3.3-debug
petsc-complex/3.0.0-debug
pgplot/5.2(default)
pltar/0.8.9(default)
pltar/0.9.0
pltar/1.0.0
pspline/1.0(default)
python/2.5.2
python/2.5.2-netcdf(default)
qt/4.3.4(default)
ruby/1.8.7
ruby/1.9.1(default)
silo/4.7(default)
silo/test
spdcp/1.0.0(default)
sprng/2.0b(default)
stagesub/1.0.2
stagesub/1.0.3(default)
starccm/5.06.010
subversion/1.4.6
subversion/1.5.0(default)
subversion/1.6.12
sundials/2.3.0(default)
superlu/3.0(default)
superlu_dist/2.2(default)
swig/1.3.36(default)
szip/2.1(default)
szip/2.1.tpb
tau/2.17.2
tau/2.17.3
tau/2.18.1(default)
tau/2.19
tcl_tk/8.5.8(default)
tkdiff/4.1.4(default)
totalview/8.6.0-1(default)
totalview/8.7.0-1
totalview/8.9.0-0
trilinos/10.0.4
trilinos/10.2.2
trilinos/10.4.0
trilinos/8.0.3
```

```
fftpack/5-r8i8(default)          matlab/7.5                       trilinos/9.0.2
fftw/3.1.2                       matlab/7.7                       udunits/1.12.4
fftw/3.1.2-dualcore              matlab/7.8(default)              udunits/1.12.9(default)
fftw/3.2                         mercurial/1.0.2                  umfpack/5.1.1(default)
fftw/3.2-dualcore                mercurial/1.3(default)           valgrind/3.3.1
fftw/3.2.1                       metis/4.0(default)               valgrind/3.4.1(default)
fftw/3.2.2                       mpe2/1.0.6(default)              vampir/2.2.0
fftw/3.2.2-dualcore              mpip/3.1.2(default)              vampir/2.3.0
fpmpi/1.0                        mumps/4.7.3_par(default)         vampir/2.3.0-OLCF3-101217
fpmpi/1.1(default)               mumps/4.9_par                    vampir/2.4.0-OLCF3-110215(default)
fpmpi_papi/1.0                   mxml/2.6(default)                vampirtrace/5.10(default)
fpmpi_papi/1.1(default)          namd/2.6                         vampirtrace/5.10b-libwrap
gamess/2008Mar04                 namd/2.7b1(default)              vampirtrace/5.10b100802
gamess/2009Jan12(default)        namd/2.7b4/PrgEnv-gnu/2.2.41A    vampirtrace/5.10ornl
ghostscript/8.64(default)        nano/2.2.6(default)              vampirtrace/5.10rc1
git/1.6.0                        ncl/5.0.0(default)               vampirtrace/5.11ornl
git/1.6.0.4                      nco/3.9.4(default)               vampirtrace/5.11ornl-beta
git/1.6.2.4                      nco/3.9.8                        vampirtrace/5.8
git/1.6.4(default)               ncview/1.93c(default)            vampirtrace/5.8.2
globalarrays/4.0.8(default)      ncview/1.93g                     vampirtrace/5.8_noatime
globalarrays/4.1.1               ncview/1.93g-netcdf4             vampirtrace/5.9
globalarrays/4.2                 nedit/5.5(default)               vampirtrace/5.9-beta
globus/4.2.1(default)            netcdf/3.6.2                     vampirtrace/5.9b100716
globus/5.0.2                     netcdf/4.0.0                     vim/7.1
gmake/3.81(default)              netcdf/4.0.0_par                 vim/7.2(default)
gnuplot/4.2.3                    netcdf/4.1                       visit/1.11.1(default)
gnuplot/4.2.4(default)           netcdf/4.1.1                     visit/2.0.2
gnuplot/4.2.5                    netcdf/4.1.1_par                 visit/2.1.2
gptl/3.4.1                       netcdf/4.1_par                   visit/2.2.0
gptl/3.4.3                       nose/0.10.4(default)             vmd/1.8.7(default)
gptl/3.4.7                       ompi/1.4a1-dtr103                wgrib/1.8.0.12o(default)
gptl/3.5                         ompi/1.4a1r21772                 wgrib/1.8.0.13b
gptl/3.5.1                       ompi/1.7a1r22177                 workflow/xgc-monitor-3.1(default)
gptl/3.5.2                       ompi/1.7a1r22229                 xt-find/0.2.0(default)
```

Q4 Modules: *jaguarpf.ccs.ornl.gov*

```
---------------------------------- /opt/cray/xt-asyncpe/4.9/modulefiles ----------------------------------
craype-hugepages2m   xtpe-barcelona       xtpe-mc8             xtpe-shanghai
craype-hugepages8m   xtpe-istanbul        xtpe-network-gemini  xtpe-target-native
xtpe-accel-nvidia20  xtpe-mc12            xtpe-network-seastar

------------------------------------------- /opt/modulefiles -------------------------------------------
Base-opts/2.1.27HD                                  petsc-complex/3.1.04(default)
Base-opts/2.1.27HD.lusrelsave                       pgi/10.0.0
Base-opts/2.1.29HD                                  pgi/10.1.0
Base-opts/2.1.29HD.lusrelsave                       pgi/10.2.0
Base-opts/2.1.41HD                                  pgi/10.3.0
Base-opts/2.1.41HD.lusrelsave                       pgi/10.4.0
Base-opts/2.1.50HD                                  pgi/10.5.0
Base-opts/2.1.50HD.lusrelsave                       pgi/10.8.0
Base-opts/2.1.50HD_PS08                             pgi/10.9.0(default)
Base-opts/2.1.50HD_PS08.lusrelsave                  pgi/11.0.0
Base-opts/2.2.27                                    pgi/11.3.0
Base-opts/2.2.27.lusrelsave                         pgi/11.5.0
Base-opts/2.2.31                                    pgi/11.7.0
Base-opts/2.2.31.lusrelsave                         pgi/11.8.0
```

```
Base-opts/2.2.31A
Base-opts/2.2.31A.lusrelsave
Base-opts/2.2.41
Base-opts/2.2.41.lusrelsave
Base-opts/2.2.41A
Base-opts/2.2.41A.lusrelsave
Base-opts/2.2.73(default)
Base-opts/2.2.73.lusrelsave
PrgEnv-cray/1.0.0
PrgEnv-cray/1.0.1
PrgEnv-cray/1.0.2(default)
PrgEnv-cray/1.0.2.orig
PrgEnv-cray/1.0.2.patch
PrgEnv-cray/1.0.2.test
PrgEnv-cray/latest
PrgEnv-cray/next
PrgEnv-cray/previous
PrgEnv-cray/test
PrgEnv-gnu/2.1.27HD
PrgEnv-gnu/2.1.29HD
PrgEnv-gnu/2.1.41HD
PrgEnv-gnu/2.1.50HD
PrgEnv-gnu/2.1.50HD_PS08
PrgEnv-gnu/2.2.27
PrgEnv-gnu/2.2.31
PrgEnv-gnu/2.2.31A
PrgEnv-gnu/2.2.41
PrgEnv-gnu/2.2.41A
PrgEnv-gnu/2.2.73(default)
PrgEnv-gnu/next
PrgEnv-gnu/previous
PrgEnv-intel/1.0.0(default)
PrgEnv-intel/next
PrgEnv-intel/previous
PrgEnv-pathscale/2.1.27HD
PrgEnv-pathscale/2.1.29HD
PrgEnv-pathscale/2.1.41HD
PrgEnv-pathscale/2.1.50HD
PrgEnv-pathscale/2.1.50HD_PS08
PrgEnv-pathscale/2.2.27
PrgEnv-pathscale/2.2.31
PrgEnv-pathscale/2.2.31A
PrgEnv-pathscale/2.2.41
PrgEnv-pathscale/2.2.41A
PrgEnv-pathscale/2.2.73(default)
PrgEnv-pathscale/next
PrgEnv-pathscale/previous
PrgEnv-pgi/2.1.27HD
PrgEnv-pgi/2.1.29HD
PrgEnv-pgi/2.1.41HD
PrgEnv-pgi/2.1.50HD
PrgEnv-pgi/2.1.50HD_PS08
PrgEnv-pgi/2.2.27
PrgEnv-pgi/2.2.31
PrgEnv-pgi/2.2.31A
PrgEnv-pgi/2.2.41
PrgEnv-pgi/2.2.41A
pgi/6.2.5
pgi/7.0.7
pgi/7.1.6
pgi/7.2.3
pgi/7.2.4
pgi/7.2.5
pgi/8.0.1
pgi/8.0.2
pgi/8.0.3
pgi/8.0.4
pgi/8.0.5
pgi/8.0.6
pgi/9.0.1
pgi/9.0.2
pgi/9.0.3
pgi/9.0.4
pkgconfig/0.15.0(default)
torque/2.3.2-snap.200807092141
torque/2.4.1b1-snap.200905191614(default)
xt-asyncpe/1.0c
xt-asyncpe/1.1
xt-asyncpe/1.2
xt-asyncpe/2.0
xt-asyncpe/2.0.34
xt-asyncpe/2.1
xt-asyncpe/2.3
xt-asyncpe/2.4
xt-asyncpe/2.5
xt-asyncpe/3.0
xt-asyncpe/3.1
xt-asyncpe/3.1.20
xt-asyncpe/3.2
xt-asyncpe/3.3
xt-asyncpe/3.4
xt-asyncpe/3.5
xt-asyncpe/3.7
xt-asyncpe/3.8
xt-asyncpe/3.9
xt-asyncpe/4.0
xt-asyncpe/4.5
xt-asyncpe/4.6
xt-asyncpe/4.7
xt-asyncpe/4.9(default)
xt-asyncpe/5.01
xt-atp/1.0(default)
xt-boot/2.1.27HD
xt-boot/2.1.29HD
xt-boot/2.1.41HD
xt-boot/2.1.50HD
xt-boot/2.1.50HD_PS08
xt-boot/2.2.27
xt-boot/2.2.31
xt-boot/2.2.31A
xt-boot/2.2.41
xt-boot/2.2.41A
xt-boot/2.2.73
xt-catamount/2.1.27HD
```

```
PrgEnv-pgi/2.2.73(default)
PrgEnv-pgi/next
PrgEnv-pgi/previous
acml/4.0.1a
acml/4.1.0
acml/4.2.0
acml/4.3.0
acml/4.4.0(default)
apprentice2/5.0.0
apprentice2/5.0.1
apprentice2/5.0.2(default)
apprentice2/5.1.0
apprentice2/5.1.3
blcr/0.7.3
cce/7.2.7
cce/7.3.1
cce/7.3.2
cce/7.3.3
cce/7.3.4(default)
cce/7.4.0
cce/7.4.1
cce/7.4.2
chapel/1.3.0(default)
cray/MySQL/5.0.64-1.0000.2342.16.1
cray/account/1.0.0-2.0202.19482.49.17
cray/audit/1.0.0-1.0202.22161.0
cray/csa/3.0.0-1_2.0202.21426.77.6
cray/job/1.5.5-0.1_2.0202.21413.56.6
cray/projdb/1.0.0-1.0202.19483.52.1
dwarf/8.2.0
dwarf/8.4.0
dwarf/8.6.0
dwarf/8.8.0
dwarf/9.5.0
dwarf/9.9.0(default)
elf/0.8.10
elf/0.8.12(default)
fftw/2.1.5
fftw/2.1.5.1
fftw/2.1.5.2
fftw/3.1.1
fftw/3.2.0
fftw/3.2.1
fftw/3.2.2
fftw/3.2.2.1(default)
fftw/3.2.2.1.bak
gcc/3.3.3
gcc/4.1.2
gcc/4.2.0.quadcore
gcc/4.2.3
gcc/4.2.4
gcc/4.3.2
gcc/4.4.1
gcc/4.4.2
gcc/4.4.3
gcc/4.4.4
gcc/4.5.1(default)
xt-catamount/2.1.29HD
xt-catamount/2.1.41HD
xt-catamount/2.1.50HD
xt-catamount/2.1.50HD_PS08
xt-craypat/5.0.0
xt-craypat/5.0.1
xt-craypat/5.0.2(default)
xt-craypat/5.1.0
xt-craypat/5.1.3
xt-lgdb/1.2(default)
xt-libc/2.1.27HD
xt-libc/2.1.29HD
xt-libc/2.1.41HD
xt-libc/2.1.50HD
xt-libc/2.1.50HD_PS08
xt-libsci/10.2.1
xt-libsci/10.3.1
xt-libsci/10.3.5
xt-libsci/10.3.6
xt-libsci/10.3.8
xt-libsci/10.3.8.1
xt-libsci/10.3.9
xt-libsci/10.4.0
xt-libsci/10.4.1
xt-libsci/10.4.2
xt-libsci/10.4.3
xt-libsci/10.4.4
xt-libsci/10.4.5
xt-libsci/10.5.0(default)
xt-lustre-ss/2.1.27HD_1.6.5
xt-lustre-ss/2.1.29.HD_ORNL.nic1_1.6.5
xt-lustre-ss/2.1.29HD_1.6.5
xt-lustre-ss/2.1.29HD_ORNL.nic10_1.6.5
xt-lustre-ss/2.1.29HD_ORNL.nic11_1.6.5
xt-lustre-ss/2.1.29HD_ORNL.nic12_1.6.5
xt-lustre-ss/2.1.29HD_ORNL.nic2_1.6.5
xt-lustre-ss/2.1.29HD_ORNL.nic5_1.6.5
xt-lustre-ss/2.1.29HD_ORNL.nic6_1.6.5
xt-lustre-ss/2.1.41HD_1.6.5
xt-lustre-ss/2.1.50HD.PS04.lus.1.6.5.steve.8062_1.6.5
xt-lustre-ss/2.1.50HD.PS04.lus.1.6.5.steve.8072_1.6.5
xt-lustre-ss/2.1.50HD.PS08.lus.1.6.5.steve.8099_1.6.5
xt-lustre-ss/2.1.50HD.PS08.lus.1.6.5.steve.8119_1.6.5
xt-lustre-ss/2.1.50HD_1.6.5
xt-lustre-ss/2.1.50HD_PS04_1.6.5
xt-lustre-ss/2.1.50HD_PS08_1.6.5
xt-lustre-ss/2.1.UP00_ORNL.nic12_1.6.5
xt-lustre-ss/2.1.UP00_ORNL.nic2_1.6.5
xt-lustre-ss/2.1.UP00_ORNL.nic30_1.6.5
xt-lustre-ss/2.1.UP00_ORNL.nic3_1.6.5
xt-lustre-ss/2.1.UP00_ORNL.nic40_1.6.5
xt-lustre-ss/2.1.UP00_ORNL.nic51_1.6.5
xt-lustre-ss/2.1.UP00_ORNL.nic52_1.6.5
xt-lustre-ss/2.2.27.lus.1.6.5.steve.8154_1.6.5
xt-lustre-ss/2.2.27.lus.1.6.5.steve.8182_1.6.5
xt-lustre-ss/2.2.27.lus.1.6.5.steve.8189_1.6.5
xt-lustre-ss/2.2.27_1.6.5
```

```
gcc/4.5.2
gcc-catamount/3.3
gnet/2.0.5
hdf5/1.8.2.2
hdf5/1.8.2.3
hdf5/1.8.3.0
hdf5/1.8.3.1
hdf5/1.8.4.0
hdf5/1.8.4.1
hdf5/1.8.5.0(default)
hdf5-parallel/1.8.2.2
hdf5-parallel/1.8.2.3
hdf5-parallel/1.8.3.0
hdf5-parallel/1.8.3.1
hdf5-parallel/1.8.4.0
hdf5-parallel/1.8.4.1
hdf5-parallel/1.8.5.0(default)
intel/11.1.046(default)
intel/11.1.064
iobuf/1.0.6(default)
java/jdk1.6.0_17
java/jdk1.6.0_20
java/jdk1.6.0_24(default)
libfast/1.0
libfast/1.0.2
libfast/1.0.3
libfast/1.0.4
libfast/1.0.5
libfast/1.0.6
libfast/1.0.7
libfast/1.0.8(default)
libfast/orig.1.0.3
libscifft-pgi/1.0.0(default)
mazama/5.1.0(default)
moab/5.2.3
moab/5.2.4
moab/5.3.0
moab/5.3.3
moab/5.3.6(default)
modules/3.1.6
modules/3.1.6.5(default)
mrnet/2.0.1.1
mrnet/2.2.0.1
mrnet/3.0.0(default)
netcdf/3.6.2
netcdf/4.0.0.2
netcdf/4.0.0.3
netcdf/4.0.1.0
netcdf/4.0.1.1
netcdf/4.0.1.2
netcdf/4.0.1.3
netcdf/4.1.1.0(default)
netcdf-hdf5parallel/4.0.0.2
netcdf-hdf5parallel/4.0.0.3
netcdf-hdf5parallel/4.0.1.0
netcdf-hdf5parallel/4.0.1.1
netcdf-hdf5parallel/4.0.1.2
xt-lustre-ss/2.2.31.lus.1.6.5.steve.8211_1.6.5
xt-lustre-ss/2.2.31A.lus.1.6.5.steve.8259_1.6.5
xt-lustre-ss/2.2.31A_1.6.5
xt-lustre-ss/2.2.31_1.6.5
xt-lustre-ss/2.2.41A_1.6.5
xt-lustre-ss/2.2.41_1.6.5
xt-lustre-ss/2.2.73_1.6.5
xt-lustre-ss/2.2.UP01.lus.1.6.5.steve.8265_1.6.5
xt-lustre-ss/2.2_1.6.5
xt-mpt/2.1.27HD
xt-mpt/2.1.29HD
xt-mpt/2.1.41HD
xt-mpt/2.1.50HD
xt-mpt/2.1.50HD_PS08
xt-mpt/3.0.1
xt-mpt/3.0.2
xt-mpt/3.0.4
xt-mpt/3.1.0
xt-mpt/3.1.0.4
xt-mpt/3.1.0.6
xt-mpt/3.1.0.7
xt-mpt/3.1.1
xt-mpt/3.1.2
xt-mpt/3.2.0
xt-mpt/3.3.0
xt-mpt/3.4.0
xt-mpt/3.4.1
xt-mpt/3.4.2
xt-mpt/3.5.0
xt-mpt/3.5.1
xt-mpt/4.0.0
xt-mpt/4.0.2
xt-mpt/4.0.3
xt-mpt/4.1.0.1
xt-mpt/4.1.1
xt-mpt/5.0.0
xt-mpt/5.1.3(default)
xt-os/2.1.27HD
xt-os/2.1.29HD
xt-os/2.1.41HD
xt-os/2.1.50HD
xt-os/2.1.50HD_PS08
xt-os/2.2.27
xt-os/2.2.31
xt-os/2.2.31A
xt-os/2.2.41
xt-os/2.2.41A
xt-os/2.2.73
xt-papi/3.6
xt-papi/3.6.1a
xt-papi/3.6.2
xt-papi/3.6.2.2
xt-papi/3.7.2(default)
xt-papi/3.7.2.0.5
xt-papi/4.1.0.0.2
xt-papi/4.1.2
xt-pe/2.1.27HD
```

```
netcdf-hdf5parallel/4.0.1.3                          xt-pe/2.1.29HD
netcdf-hdf5parallel/4.1.1.0(default)                 xt-pe/2.1.41HD
pathscale/3.2                                        xt-pe/2.1.50HD
pathscale/3.2.99(default)                            xt-pe/2.1.50HD_PS08
petsc/2.3.3a                                         xt-pe/2.2.27
petsc/3.0.0                                          xt-pe/2.2.31
petsc/3.0.0.1                                        xt-pe/2.2.31A
petsc/3.0.0.10                                       xt-pe/2.2.41
petsc/3.0.0.2                                        xt-pe/2.2.41A
petsc/3.0.0.3                                        xt-pe/2.2.73
petsc/3.0.0.4                                        xt-service/2.1.27HD
petsc/3.0.0.6                                        xt-service/2.1.29HD
petsc/3.0.0.8                                        xt-service/2.1.41HD
petsc/3.1.00                                         xt-service/2.1.50HD
petsc/3.1.04(default)                                xt-service/2.1.50HD_PS08
petsc-complex/2.3.3a                                 xt-service/2.2.27
petsc-complex/3.0.0                                  xt-service/2.2.31
petsc-complex/3.0.0.1                                xt-service/2.2.31A
petsc-complex/3.0.0.10                               xt-service/2.2.41
petsc-complex/3.0.0.2                                xt-service/2.2.41A
petsc-complex/3.0.0.3                                xt-service/2.2.73
petsc-complex/3.0.0.4                                xtgdb/1.0.0(default)
petsc-complex/3.0.0.6                                xtpe-target-catamount
petsc-complex/3.0.0.8                                xtpe-target-cnl
petsc-complex/3.1.00

------------------------------------------ /opt/cray/modulefiles ------------------------------------------
atp/1.0.1                            xt-mpt/2.1.50HD_PS08
atp/1.0.2                            xt-mpt/3.0.1
atp/1.0.3                            xt-mpt/3.0.2
atp/1.1.0(default)                   xt-mpt/3.0.4
gdb/7.2(default)                     xt-mpt/3.1.0
papi/4.1.2(default)                  xt-mpt/3.1.0.4
perftools/5.1.0                      xt-mpt/3.1.0.6
perftools/5.1.3(default)             xt-mpt/3.1.0.7
perftools/5.2.0                      xt-mpt/3.1.1
pmi/1.0-1.0000.7628.10.2.ss          xt-mpt/3.1.2
pmi/1.0-1.0000.7901.22.1.ss          xt-mpt/3.2.0
pmi/1.0-1.0000.8160.39.1.ss(default) xt-mpt/3.3.0
stat/1.0.0                           xt-mpt/3.4.0
stat/1.1.0                           xt-mpt/3.4.1
stat/1.1.2(default)                  xt-mpt/3.4.2
tpsl/1.0.0(default)                  xt-mpt/3.5.0
trilinos/10.0.1                      xt-mpt/3.5.1
trilinos/10.2.0                      xt-mpt/4.0.0
trilinos/10.6.0(default)             xt-mpt/4.0.2
trilinos/10.6.2.0                    xt-mpt/4.0.3
xt-mpich2/5.0.0                      xt-mpt/4.1.0.1
xt-mpich2/5.1.3(default)             xt-mpt/4.1.1
xt-mpt/2.1.27HD                      xt-mpt/5.0.0
xt-mpt/2.1.29HD                      xt-mpt/5.1.3(default)
xt-mpt/2.1.41HD                      xt-shmem/5.0.0
xt-mpt/2.1.50HD                      xt-shmem/5.1.3(default)

------------------------------------------ /opt/modules/3.1.6 ------------------------------------------
modulefiles/modules/dot         modulefiles/modules/module-info modulefiles/modules/null
modulefiles/modules/module-cvs  modulefiles/modules/modules     modulefiles/modules/use.own
```

```
------------------------------------------- /sw/xt5/modulefiles -------------------------------------------
DefApps                                gptl/3.5.1                             ompi/1.7ft_a4
adios/0.9.10                           gptl/3.5.2                             ompi/DT/routed-pgi-uos
adios/0.9.10_phdf5                     gptl/3.6(default)                      ompi/DT/routing-pgi
adios/0.9.12                           gptl/3.6.3                             ompi/DT/routing-pgi-uos
adios/0.9.13                           gptl_pmpi/3.6(default)                 ompi/experimental(default)
adios/1.0                              gptl_pmpi/3.6.3                        ompi/standard
adios/1.1.0                            grace/5.1.21                           p-netcdf/1.0.2
adios/1.1.amr1                         grace/5.1.22(default)                  p-netcdf/1.0.3(default)
adios/1.2                              gromacs/3.3.3                          p-netcdf/1.1.1
adios/1.2.1(default)                   gromacs/4.0.5(default)                 p-netcdf/1.2.0
adios/1.2_jc                           gsl/1.11                               p3dfft/2.4(default)
adios/1.3                              gsl/1.11-dualcore                      p3dfft/2.4-fftw
adios/trunk                            gsl/1.12                               p3dfft/2.4.sav
arpack/2008.03.11(default)             gsl/1.12-dualcore(default)             parmetis/3.1
atlas/3.8.2                            gv/3.6.7(default)                      parmetis/3.1.1(default)
atlas/3.8.2-fPIC-dualcore              hdf5/1.6.7                             perl-gd/2.44(default)
atlas/3.8.3                            hdf5/1.6.7_par                         petsc/2.3.3-debug
atlas/3.8.3-fPIC-dualcore(default)     hdf5/1.6.8                             petsc/3.0.0-custom
autoconf/2.63(default)                 hdf5/1.6.8_par                         petsc/3.0.0-debug
autoconf/2.68                          hdf5/1.8.1                             petsc-complex/2.3.3-debug
automake/1.10.1(default)               hdf5/1.8.1_par                         petsc-complex/3.0.0-debug
automake/1.11.1                        hdf5/1.8.2                             pgplot/5.2(default)
automake/1.11.1-a                      hdf5/1.8.2_par                         pltar/0.8.9(default)
aztec/2.1(default)                     hdf5/1.8.5-patch1                      pltar/0.9.0
banner/1.3.2(default)                  hdf5/1.8.7                             pltar/1.0.0
blas/ref(default)                      hpctoolkit/4.9.2(default)              pspline/1.0(default)
blas/ref-dualcore                      hypre/2.0.0(default)                   python/2.5.2
blas-goto/1.0                          hypre/2.0.0-setdir                     python/2.5.2-netcdf(default)
blas-goto/2.0(default)                 hypre/2.4.0b                           python/2.7.1
boost/1.44.0                           hypre/2.4.0b-craypat                   python/2.7.2
bugget/2.0(default)                    hypre/2.4.0b-debug                     qt/4.3.4(default)
cdo/1.3.1                              hypre/2.7.0b                           ruby/1.8.7
cdo/1.3.2(default)                     idl/6.4(default)                       ruby/1.9.1(default)
cmake/2.6.1(default)                   imagemagick/6.4.2(default)             ruby/1.9.2
cmake/2.6.2                            imagemagick/6.5.4                      silo/4.7(default)
cmake/2.6.4                            iota/0.2.1                             silo/test
cmake/2.8.0                            iota/0.2.2(default)                    spdcp/1.0.0(default)
cpmd/3.13.1                            iota/0.2.3                             sprng/2.0b(default)
cpmd/3.13.2(default)                   iota/0.2.4                             stagesub/1.0.2
ddt/2.4.1                              iota/1.0.0                             stagesub/1.0.3(default)
ddt/2.4.1-11140                        java/1.5.0.06                          starccm/5.06.010(default)
ddt/2.5.1-12323                        java/1.6.0.06                          starccm/6.05.061
ddt/2.6.1                              java-jdk/1.5.0.06                      subversion/1.4.6
ddt/3.0-16706(default)                 java-jdk/1.6.0.06(default)             subversion/1.5.0(default)
ddt/3.0-17491                          java-jre/1.5.0.06(default)             subversion/1.6.12
ddt/3.0-rc1                            lammps/1Jul11                          subversion/1.6.15
ddt/3.0-rc3                            lammps/22Jul09                         subversion/1.6.17
doxygen/1.5.6                          lammps/4Mar08                          sundials/2.3.0(default)
doxygen/1.5.8                          lammps/9Sep10(default)                 superlu/3.0(default)
doxygen/1.5.9                          lammps/May08                           superlu_dist/2.2(default)
doxygen/1.6.1(default)                 lapack/3.1.1(default)                  swig/1.3.36(default)
dragon/1.0a(default)                   lapack/3.1.1-dualcore                  szip/2.1(default)
emacs/23.1(default)                    lapack/3.1.1-fPIC                      szip/2.1.tpb
esmf/4.0.0r_O                          libgd/2.0.35(default)                  tau/2.17.2
esmf/4.0.0r_g                          liblut/0.9.6                           tau/2.17.3
```

```
esmf/4.0.0rp1_O                    liblut/0.9.7                       tau/2.18.1(default)
esmf/4.0.0rp1_g                    liblut/0.9.8                       tau/2.19
esmf/4.0.0rp2_O(default)           liblut/0.9.9(default)              tcl_tk/8.5.8(default)
esmf/4.0.0rp2_g                    liblut/1.0.0                       tkdiff/4.1.4(default)
esmf/5.2.0-p1_O                    libtool/2.4                        totalview/8.6.0-1(default)
esmf/5.2.0-p1_g                    libxml2/2.7.6(default)             totalview/8.7.0-1
esmf/5.2.0-p1_with-lapack+netcdf_O libxslt/1.1.26(default)            totalview/8.9.0-0
esmf/5.2.0-p1_with-lapack+netcdf_g lsq/0.1.0                          trilinos/10.0.4
esmf/5.2.0-p1_with-netcdf_O        lsq/1.0.0                          trilinos/10.2.2
esmf/5.2.0-p1_with-netcdf_g        lustredu/1.0(default)              trilinos/10.4.0
esmf/5.2.0_O                       m4/1.4.11(default)                 trilinos/8.0.3
esmf/5.2.0_g                       m4/1.4.14                          trilinos/9.0.2
ferret/6.1(default)                makedepf90/2.8.8(default)          udunits/1.12.4
ferret/6.3                         matlab/7.5                         udunits/1.12.9(default)
fftpack/5-r4i4                     matlab/7.7                         umfpack/5.1.1(default)
fftpack/5-r8i4                     matlab/7.8(default)                valgrind/3.3.1
fftpack/5-r8i8(default)            mercurial/1.0.2                    valgrind/3.4.1(default)
fftw/3.1.2                         mercurial/1.3(default)             vampir/2.3.0
fftw/3.1.2-dualcore                mercurial/1.7.5                    vampir/2.4.0-OLCF3-110323
fftw/3.2                           metis/4.0(default)                 vampir/2.5.0-OLCF3-110621
fftw/3.2-dualcore                  mpe2/1.0.6(default)                vampir/2.5.0-OLCF3-110715-2M
fftw/3.2.1                         mpip/3.1.2(default)                vampir/2.5.0-OLCF3-110815(default)
fftw/3.2.2                         mumps/4.7.3_par(default)           vampirtrace/5.10
fftw/3.2.2-dualcore                mumps/4.9_par                      vampirtrace/5.10b-libwrap
fpmpi/1.0                          mxml/2.6(default)                  vampirtrace/5.10b100802
fpmpi/1.1(default)                 namd/2.6                           vampirtrace/5.10ornl
fpmpi_papi/1.0                     namd/2.7b1(default)                vampirtrace/5.10rc1
fpmpi_papi/1.1(default)            namd/2.7b4/PrgEnv-gnu/2.2.41A      vampirtrace/5.11.1
gamess/2008Mar04                   nano/2.2.6(default)                vampirtrace/5.11.1-etimesync
gamess/2009Jan12(default)          ncl/5.0.0(default)                 vampirtrace/5.11iofsl-beta
gamess/2010Oct01                   nco/3.9.4(default)                 vampirtrace/5.11ornl(default)
ghostscript/8.64(default)          nco/3.9.8                          vampirtrace/5.11ornl-beta
git/1.6.0                          ncview/1.93c(default)              vampirtrace/5.12-etimesync
git/1.6.0.4                        ncview/1.93g                       vampirtrace/5.12ornl
git/1.6.2.4                        ncview/1.93g-netcdf4               vampirtrace/5.8
git/1.6.4(default)                 nedit/5.5(default)                 vampirtrace/5.8.2
git/1.7.6                          netcdf/3.6.2                       vampirtrace/5.8_noatime
globalarrays/4.0.8(default)        netcdf/4.0.0                       vampirtrace/5.9
globalarrays/4.1.1                 netcdf/4.0.0_par                   vampirtrace/5.9-beta
globalarrays/4.2                   netcdf/4.1                         vampirtrace/5.9b100716
globus/4.2.1(default)              netcdf/4.1.1                       vim/7.1
globus/5.0.2                       netcdf/4.1.1_par                   vim/7.2(default)
globus/5.0.4                       netcdf/4.1.3                       visit/1.11.1(default)
gmake/3.81(default)                netcdf/4.1_par                     visit/2.0.2
gnuplot/4.2.3                      nose/0.10.4(default)               visit/2.1.2
gnuplot/4.2.4(default)             ompi/1.4a1-dtr103                  visit/2.2.0
gnuplot/4.2.5                      ompi/1.4a1r21772                   vmd/1.8.7(default)
gptl/3.4.1                         ompi/1.7a1r22177                   wgrib/1.8.0.12o(default)
gptl/3.4.3                         ompi/1.7a1r22229                   wgrib/1.8.0.13b
gptl/3.4.7                         ompi/1.7a1r22760                   workflow/xgc-monitor-3.1(default)
gptl/3.5                           ompi/1.7ft_a3                      xt-find/0.2.0(default)
```

Q2 Modules: *carver.nersc.gov (dirac)*

```
------------------------------ /usr/common/nsg/opt/Modules/3.2.7/config/versions ------------------------------
3.2.7
```

```
--------------------------------- /usr/common/nsg/opt/Modules/3.2.7/modulefiles ---------------------------------
dot         module-cvs  module-info modules     null        use.own

------------------------------------- /usr/common/usg/Modules/modulefiles -------------------------------------
CTSS                            idl/8.0(default)                openmpi-gnu/1.4.3
R/2.12.0                        intel/11.1(default)             openmpi-intel/1.4.2(default)
abinit/6.0.3                    ipm/0.983                       openmpi-intel/1.4.3
abinit/6.2.2                    ipm/2.00(default)               osg/1.2-ce(default)
abinit/6.4.1(default)           jasper/1.900.1(default)         papi/4.0.0(default)
amber/10.0                      jpeg/6b(default)                paraview/3.8.1(default)
amber/11(default)               lammps/2010.07.04(default)      parpack/2.1(default)
autoconf/2.61(default)          lapack/3.2.1(default)           perl/5.8.8(default)
bison/2.4                       lapack-gnu/3.2.1                pgi/10.0
boost/1.45(default)             mathematica/7.0(default)        pgi/10.3
cdo/1.4.1(default)              matlab/R2010a(default)          pgi/10.4
cmake/2.8.0                     matlab/R2010b                   pgi/10.5
cmake/2.8.2(default)            matlab/dptfoo                   pgi/10.8(default)
cp2k/2010.06.22(default)        matlab-nofonts/R2010a           pgi/10.9
cuda/3.2(default)               mkl/10.2(default)               pgi/11.1
cuda/4.0b                       molpro/2008.1                   pgi/9.0
ddt/2.5.1                       molpro/2008.1.serial            pgi-gpu/10.5(default)
ddt/2.6(default)                molpro/2009.1                   pgi-gpu/10.8
einspline/0.9.2(default)        molpro/2010.1(default)          pgplot/5.2
espresso/4.1.2                  mvapich-gcc/1.2rc1              ploticus/2.41
espresso/4.2.1(default)         mvapich2-gcc/1.5.1p1            pnetcdf/1.1.0(default)
fca/1.2(default)                mysql/5.0.51(default)           postg/9.0.1(default)
fftw/2.1.5(default)             nag/21                          postgres/8.4.3(default)
fftw/2.1.5-gnu                  nag/21_gnu                      postgres/9.0.1
fftw/3.2.2                      nag/22(default)                 python/2.6.4(default)
fftw/3.2.2-gnu                  nag/22_gnu                      python/2.6.5
fftw-Intel/2.1.5                nag-gnu/21                      python/2.7
fftw-Intel/3.2.2                nag-gnu/22(default)             python/2.7.1
fftw-gnu/2.1.5                  namd/2.7(default)               qchem/3.2.0.2
fftw-gnu/3.2.2                  namd/2.7b2                      qchem/3.2.0.2-intel(default)
g09/a2                          namd/2.7b2_cuda                 siesta/3.0-b(default)
g09/a2.2                        ncar/5.1.1                      silo/4.7.2(default)
g09/b1(default)                 ncar/5.1.1-test                 sprng/2.0
ga/4.2(default)                 ncar/5.2.0                      sqlite/3.6.23(default)
ga/4.2-intel                    ncar/5.2.1(default)             subversion/1.6.4(default)
ga/4.2.i8                       nco/4.0.2                       szip/2.1(default)
gamess/24Mar2007R6(default)     nco/4.0.5(default)              szip/2.1-gnu
gamess/25Mar2010R2              ncview/1.93g(default)           tig
gcc/4.4.2(default)              netcdf/4.0.1                    totalview/8.7.0
gcc/4.5.2                       netcdf/4.1.1(default)           training/1.0
gnuplot/4.4.0(default)          nvidia-driver-util/3.2(default) udunits/1.12.9(default)
gpu-util/1.0                    nvidia-driver-util/4.0b         usg/1.1(default)
grace/5.1.22(default)           nwchem/5.1.1                    vasp/4.6.35(default)
gromacs/4.0.7(default)          nwchem/5.1.1-gnu                vasp/5.2
gromacs/4.0.7_serial            nwchem/6.0(default)             vasp/5.2.11
gsl/1.13(default)               openmpi/1.4.1                   vasp/5.2.11.tbdyn
gsl/1.14                        openmpi/1.4.1-gnu               vasp/5.2.8
h5part/1.6.1(default)           openmpi/1.4.2(default)          vasp/5.2.orig
hdf/4.2.5                       openmpi/1.4.3                   webdev/1.0
hdf/4.2.5_gnu                   openmpi/1.4.3beta               wien2k/10.1(default)
hdf5/1.8.3(default)             openmpi/tvtest                  wien2k/2009.2
hdf5/1.8.6                      openmpi-gcc/1.4.1               wien2k/2009.2.test
hdf5-parallel/1.8.3(default)    openmpi-gcc/1.4.2(default)      xcrysden/1.5.21(default)
```

```
hdf5-parallel/1.8.3-gnu         openmpi-gcc/1.4.2.test          xml2/0.4(default)
hdf5-parallel/1.8.3-intel       openmpi-gcc/1.4.3               zlib/1.2.3
hdfview/2.6.1                   openmpi-gnu/1.4.1               zlib/1.2.3-gnu
hmpp/2.4                        openmpi-gnu/1.4.2(default)      zlib/1.2.5(default)
idl/7.1                         openmpi-gnu/1.4.2.test

------------------------------------ /usr/common/acts/Modules/modulefiles ------------------------------------
ga/4.3.2i8            petsc/3.0.0_O         petsc/3.1_O_c++       slepc/3.0.0_O         superlu_dist/2.3_O
metis/4.0             petsc/3.0.0_O_c++     petsc/3.1_O_complex   slepc/3.0.0_O_c++     superlu_dist/2.3_g
metis/4.0_O           petsc/3.0.0_O_complex petsc/3.1_g           slepc/3.0.0_O_complex tao/1.10.1_O
metis/4.0_g           petsc/3.0.0_g         petsc/3.1_g_c++       slepc/3.0.0_g         tao/1.10.1_g
parmetis/3.1.1        petsc/3.0.0_g_c++     petsc/3.1_g_complex   slepc/3.0.0_g_c++     trilinos/10.4.0
parmetis/3.1.1_O      petsc/3.0.0_g_complex scalapack/1.8.0       slepc/3.0.0_g_complex
parmetis/3.1.1_g      petsc/3.1(default)    scalapack/1.8.0_O     superlu/4.0_O
petsc/3.0.0           petsc/3.1_O           scalapack/1.8.0_g     superlu/4.0_g

------------------------------------ /usr/common/ftg/Modules/modulefiles ------------------------------------
bupc/2.10.2-gcc-4.4.2         bupc/2.12.0-gcc-4.4.2         bupc/2.12.1-pgi-10.8(default)
bupc/2.10.2-pgi-10.3          bupc/2.12.0-icc-11.1          bupc/2.12.1-pgi-10.9
bupc/2.10.2-pgi-10.4          bupc/2.12.0-pgi-10.8          bupc-mpicc/2.10.2-gcc-4.4.2
bupc/2.10.2-pgi-10.5          bupc/2.12.0-pgi-10.9          bupc-mpicc/2.10.2-pgi-10.5
bupc/2.10.2-pgi-10.8          bupc/2.12.1-gcc-4.4.2
bupc/2.10.2-pgi-10.9          bupc/2.12.1-icc-11.1

----------------------------------- /usr/common/graphics/Modules/modulefiles -----------------------------------
ParaView/3.8.1                ferret/6.4(default)          vmd/1.8.7(default)
avs-express/7.2.1(default)    tecplot/360_2009_R2(default)
```

Q4 Modules: *carver.nersc.gov (dirac)*

```
------------------------------- /usr/common/nsg/opt/Modules/3.2.7/config/versions -------------------------------
3.2.7

---------------------------------- /usr/common/nsg/opt/Modules/3.2.7/modulefiles ----------------------------------
dot         module-cvs  module-info modules     null        use.own

-------------------------------------- /usr/common/usg/Modules/modulefiles --------------------------------------
CTSS                            idl/8.0(default)                openmpi-intel/1.4.3
R/2.10.1                        intel/11.1(default)             openmpi-intel/1.4.4beta
R/2.12.0                        ipm/0.983                       osg/1.2-ce(default)
R/2.12.1(default)               ipm/2.00(default)               papi/4.0.0(default)
R/2.13.1                        jasper/1.900.1(default)         paraview/3.8.1(default)
abinit/6.0.3                    jpeg/6b(default)                parpack/2.1(default)
abinit/6.2.2                    lammps/2010.07.04(default)      perl/5.8.8(default)
abinit/6.4.1(default)           lammps/2011.08                  pgi/10.0
amber/10.0                      lapack/3.2.1(default)           pgi/10.3
amber/11(default)               lapack/3.3.0                    pgi/10.4
autoconf/2.61(default)          lapack-gnu/3.2.1                pgi/10.5
bison/2.4                       lapack-gnu/3.3.0                pgi/10.8(default)
boost/1.45(default)             mathematica/7.0(default)        pgi/10.9
cdo/1.4.1(default)              mathematica-nofonts/7.0(default) pgi/11.1
cmake/2.8.0                     matlab/R2010a(default)          pgi/9.0
cmake/2.8.2(default)            matlab/R2010b                   pgi-gpu/10.5(default)
cmake/2.8.5                     matlab-nofonts/R2010a(default)  pgi-gpu/10.8
cp2k/2010.06.22(default)        matlab-nofonts/R2010b           pgplot/5.2
cp2k/2011.10.05                 mercurial/1.7.3(default)        pgplot-pgi/5.2(default)
```

```
cuda/3.2                          mercurial/1.8.1                   ploticus/2.41
cuda/4.0(default)                 mkl/10.2(default)                 pnetcdf/1.1.0(default)
ddt/2.5.1                         molpro/2008.1                     postg/9.0.1(default)
ddt/2.6                           molpro/2008.1.serial              postgres/8.4.3(default)
ddt/3.0(default)                  molpro/2009.1                     postgres/9.0.1
einspline/0.9.2(default)          molpro/2010.1(default)            python/2.6.4
espresso/4.1.2                    molpro/2010.1.mppx                python/2.6.5
espresso/4.2.1(default)           mongodb/1.8.1(default)            python/2.7
espresso/4.3.2                    mvapich-gcc/1.2rc1                python/2.7.1(default)
fca/1.2(default)                  mvapich2-gcc/1.5.1p1              qchem/3.2.0.2
fftw/2.1.5(default)               mysql/5.0.51(default)             qchem/3.2.0.2-intel(default)
fftw/2.1.5-gnu                    nag/21                            siesta/3.0-b(default)
fftw/3.2.2                        nag/21_gnu                        silo/4.7.2(default)
fftw/3.2.2-gnu                    nag/22(default)                   sprng/2.0
fftw-Intel/2.1.5                  nag/22_gnu                        sqlite/3.6.23(default)
fftw-Intel/3.2.2                  nag-gnu/21                        subversion/1.6.17
fftw-gnu/2.1.5                    nag-gnu/22(default)               subversion/1.6.4(default)
fftw-gnu/3.2.2                    nag-intel/23(default)             szip/2.1(default)
g09/a2                            namd/2.7                          szip-gnu/2.1(default)
g09/a2.2                          namd/2.8(default)                 szip-intel/2.1(default)
g09/b1(default)                   ncar/5.1.1                        texinfo/4.12
ga/4.2(default)                   ncar/5.1.1-test                   texlive/2008(default)
ga/4.2-intel                      ncar/5.2.0                        tig
ga/4.2.i8                         ncar/5.2.1(default)               totalview/8.7.0
gamess/24Mar2007R6(default)       ncar/6.0.0                        totalview/8.9.1
gamess/25Mar2010R2                nco/4.0.2                         totalview/8.9.1-1(default)
gcc/4.4.2(default)                nco/4.0.5(default)                training/1.0
gcc/4.5.2                         ncview/1.93g(default)             udunits/1.12.9(default)
gnuplot/4.4.0(default)            netcdf/4.0.1                      usg/1.1(default)
gpaw/0.8.0.8092(default)          netcdf/4.1.1(default)             vasp/4.6.35(default)
gpi/1.0                           netcdf-gnu/4.1.1(default)         vasp/5.2
gpu-util/1.0                      netcdf-intel/4.1.1(default)       vasp/5.2.11
grace/5.1.22(default)             nvidia-driver-util/3.2            vasp/5.2.11.IKPTD
gromacs/4.0.7(default)            nvidia-driver-util/4.0(default)   vasp/5.2.11.tbdyn
gromacs/4.0.7_serial              nwchem/5.1.1                      vasp/5.2.12
gsl/1.13(default)                 nwchem/5.1.1-gnu                  vasp/5.2.8
gsl/1.14                          nwchem/6.0(default)               vasp/5.2.orig
h5part/1.6.1(default)             openmpi/1.4.1                     webdev/1.0
hdf/4.2.5                         openmpi/1.4.1-gnu                 wien2k/10.1(default)
hdf/4.2.5_gnu                     openmpi/1.4.2(default)            wien2k/10.1.debug
hdf5/1.8.3(default)               openmpi/1.4.3                     wien2k/10.1.test
hdf5/1.8.6                        openmpi/1.4.3beta                 wien2k/11.1
hdf5-gnu/1.8.6(default)           openmpi/tvtest                    wien2k/2009.2
hdf5-intel/1.8.6(default)         openmpi-gcc/1.4.1                 wien2k/2009.2.test
hdf5-intel/1.8.7                  openmpi-gcc/1.4.2(default)        xcrysden/1.5.21(default)
hdf5-parallel/1.8.3(default)      openmpi-gcc/1.4.2.test            xemacs/21.4.22
hdf5-parallel/1.8.3-gnu           openmpi-gcc/1.4.3                 xml2/0.4(default)
hdf5-parallel/1.8.3-intel         openmpi-gnu/1.4.1                 zlib/1.2.3
hdfview/2.6.1                     openmpi-gnu/1.4.2(default)        zlib/1.2.5(default)
hmpp/2.4(default)                 openmpi-gnu/1.4.2.test            zlib-gnu/1.2.3
hmpp/2.5.1                        openmpi-gnu/1.4.3                 zlib-gnu/1.2.5(default)
idl/7.1                           openmpi-intel/1.4.2(default)      zlib-intel/1.2.5(default)

------------------------------------- /usr/common/acts/Modules/modulefiles -------------------------------------
ga/5.0.2              parmetis/3.1.1_g      petsc/3.1_g           slepc/3.1_0           superlu_dist/2.5_g
ga/5.0.2_0            petsc/3.0.0           petsc/3.1_g_c++       slepc/3.1_0_complex   tao/1.10.1_0
ga/5.0.2_g            petsc/3.0.0_0         petsc/3.1_g_complex   slepc/3.1_g           tao/1.10.1_g
```

```
hypre/2.7.0b          petsc/3.0.0_O_c++     scalapack/1.8.0       slepc/3.1_g_c++       tau/2.20.2
hypre/2.7.0b_O        petsc/3.0.0_O_complex scalapack/1.8.0_O     slepc/3.1_g_complex   tau/2.20.3(default)
hypre/2.7.0b_g        petsc/3.0.0_g         scalapack/1.8.0_g     superlu/4.0_O         trilinos/10.4.0
metis/4.0             petsc/3.0.0_g_c++     slepc/3.0.0_O         superlu/4.0_g         trilinos/10.6.4
metis/4.0_O           petsc/3.0.0_g_complex slepc/3.0.0_O_c++     superlu/4.1_O         trilinos/10.6.4_O
metis/4.0_g           petsc/3.1(default)    slepc/3.0.0_O_complex superlu/4.1_g         trilinos/10.6.4_g
overture/24a.d        petsc/3.1_O           slepc/3.0.0_g         superlu_dist/2.3_O
parmetis/3.1.1        petsc/3.1_O_c++       slepc/3.0.0_g_c++     superlu_dist/2.3_g
parmetis/3.1.1_O      petsc/3.1_O_complex   slepc/3.0.0_g_complex superlu_dist/2.5_O

------------------------------------- /usr/common/ftg/Modules/modulefiles -------------------------------------
bupc/2.10.2-gcc-4.4.2         bupc/2.12.0-icc-11.1          bupc/2.12.2-gcc-4.4.2
bupc/2.10.2-pgi-10.3          bupc/2.12.0-pgi-10.8          bupc/2.12.2-icc-11.1
bupc/2.10.2-pgi-10.4          bupc/2.12.0-pgi-10.9          bupc/2.12.2-pgi-10.8(default)
bupc/2.10.2-pgi-10.5          bupc/2.12.1-gcc-4.4.2         bupc-mpicc/2.10.2-gcc-4.4.2
bupc/2.10.2-pgi-10.8          bupc/2.12.1-icc-11.1          bupc-mpicc/2.10.2-pgi-10.5
bupc/2.10.2-pgi-10.9          bupc/2.12.1-pgi-10.8          hpctoolkit
bupc/2.12.0-gcc-4.4.2         bupc/2.12.1-pgi-10.9

----------------------------------- /usr/common/graphics/Modules/modulefiles ------------------------------------
ParaView/3.8.1               ferret/6.4(default)          tecplot/360_R2011
avs-express/7.2.1(default)   tecplot/360_2009_R2(default) vmd/1.8.7(default)
```

0.6 Application Inputs, Scripts, Environments

0.6.1 STOMP

Q2 and Q4 Input Settings

The inputs required to execute the benchmark studied are too large to print here. The interested reader may refer to STOMP's simulation scenarios and in particular time card 24:

```
#  File: /projects/rifle/case24/input_main
#  Last Changed: Mon Mar 28 13:08:32 2011 by Scott Waichler <scott.waichler@pnl.gov>
~Simulation Title Card
1,
3-D model for Rifle Big Rusty experiment,
Scott Waichler,
Pacific Northwest National Laboratory,
March 21, 2011,
00:00:00 PM,
5,
Case 24,
3-D flow field representing Rifle Big Rusty experiment,
Demonstration case for high-capacity computing with full chemistry and frequent output,
New denser grid at 0.1 m in all dimensions,
Heterogeneous material types from Yi-Ju Bott and Chris Murray,
```

The files required are:

```
   80 -rw-r--r--    1 xxxxx  staff     40477 Aug 26 15:53 grid.input
   24 -rwxr--r--    1 xxxxx  staff     11041 Aug 26 15:53 rxn.dat
51904 -rw-r--r--    1 xxxxx  staff  26574480 Aug 26 15:53 material2.inp
48728 -rw-r--r--    1 xxxxx  staff  24948000 Aug 26 15:53 site.dat
48728 -rw-r--r--    1 xxxxx  staff  24948000 Aug 26 15:53 u_init.dat
```

The files are used to initialize the material types and the reaction processes as a spatial distribution on the target lattice. Much of this data is taken directly from field survey. The cards that were included in these studies: Solution Control Card, Grid Card, Rock/Soil Zonation Card, Mechanical Properties Card, Hydraulic Properties Card, Saturation Function Card, Aqueous Relative Permeability Card, Solute/Porous Media Interaction Card, Initial Conditions Card (ie, gas pressure, aqueous pressure,etc.), Boundary Conditions Card (ie, left upgradient boundary, defined head and oxygen, DOC concentrations WEST, hydraulic gradient, aqueous concentrations, right downgradient boundary, defined head EAST, hydraulic gradient, outflow, etc.), injection sites (ie, injection wells are repeated at x=8.05 m, and from y=5.05 to 14.95 m domain intervals), etc.

Other information such as the units (pressure (Pa), species aqueous concentrations (mol/L), etc.), simulation duration, check point and observable data I/O frequency, the mapping details of the discretized spatial lattice to the virtual process lattice are also annotated.

The only notable change between the Q2 and Q4 input decks is that the number of processes required to execute the Q2 problem was dramatically reduced in Q4 and thus the virtual process grid dimensions changed from $3 \times (60 \times 44 \times 20 :=)52,800 = 158,400$ to $30 \times 22 \times 5 = 3300$.

Q2 Compilation Script

```
ALL: STOMP1

include MakeIncl.user
PETSC_DIR = /autofs/na1_home/fang/xt5/software/petsc-3.1-p4
PETSC_ARCH = cray-xt5
include ${PETSC_DIR}/${PETSC_ARCH}/conf/petscvariables
include ${PETSC_DIR}/conf/variables
include ${PETSC_DIR}/conf/rules

ALL_INCLUDES = $(STOMP_INCLUDES)
ALL_LIBS = $(GA_LIB)
#FLINKERS = ${FLINKER} ${INCLUDES}

EXECF = stomp1_timer.x
#FC_INTEL = ifort -w
FC =
FLINKER =
#FLINKER = mpif90 -r8 -pg -save -I.
FLINKER = ftn -I.
# normal compile
FC = $(FLINKER) $(INCLUDES) $(ALL_INCLUDES)
DEBUG_FLAG = -g -C
OPTIMIZE_FLAG = -O

SOURCEF = grid_mod.F90 rd_mod.F90 banner.F90 bcf1.F90 bcj1.F90 bcp1.F90\
          cap1.F90 chk1.F90 cisc1.F90 difmn.F90 dpor1.F90 drcvl.F90 drcvlb.F90\
          eckechem.F90 elc1.F90 factor.F90 fnhgbl.F90 fntblx.F90 fntbly.F90\
          fsplnx.F90 fsplny.F90 GA_grid.F90 incrm1.F90 intlz.F90 jcbp.F90\
          jcbwl.F90 ksp1.F90 lcase.F90 ldo1.F90 petscapp.F90 petsccalls.F90\
          porsty.F90 qsat1.F90 rd_chem.F90 rdbc1.F90 rdchr.F90 rddpr.F90\
          rdgrid.F90 rdhydr.F90 rdic1.F90 rdijk.F90 rdinac.F90 rdini.F90\
          rdinpl.F90 rdint.F90 rdlrp.F90 rdmech.F90 rdou1.F90 rdouun.F90\
          rdrock.F90 rdrst.F90 rdsclf1.F90 rdsf1.F90 rdsolu.F90 rdsp1.F90\
          rdsr1.F90 rdtf1.F90 rdtp1.F90 rdunit.F90 refnod.F90 rkl1.F90\
          rklt1.F90 rsdl1.F90 sbnd1.F90 sjcbl.F90 smc1.F90 sorc1.F90\ splinx.F90\
          spliny.F90 step.F90 stomp1.F90 tmstep.F90 tortu.F90 trpgl1.F90\
          update_nodes.F90 updt1.F90 watlad.F90 watlqv.F90 watsp.F90 wrmsg.F90\
          wrplot.F90 wrrst.F90 stomp1_driver.F90 watgsd.F90 airgsd.F90\
```

```
          average_v.F90 tport1.F90 crntnb.F90 sjcbl.F90 updtc.F90 sort1.F90

FOBJS =   bgc.o chem.o bgc_param.o grid_mod.o rd_mod.o petscapp.o\
          banner.o bcf1.o bcj1.o bcp1.o cap1.o chk1.o cisc1.o difmn.o\
          dpor1.o drcvl.o drcvlb.o eckechem.o elc1.o factor.o fntblx.o\
          fntbly.o fsplnx.o fsplny.o GA_grid.o incrm1.o intlz.o jcbp.o\
          jcbwl.o ksp1.o lcase.o ldo1.o petsccalls.o porsty.o qsat1.o\
          rd_chem.o rdbc1.o rdchr.o rddpr.o rdgrid.o rdhydr.o rdic1.o\
          rdijk.o rdinac.o rdini.o rdinpl.o rdint.o rdlrp.o rdmech.o\
          rdou1.o rdouun.o rdrock.o rdsclf1.o rdsf1.o rdsolu.o rdsp1.o\
          rdsr1.o rdtf1.o rdtp1.o rdunit.o refnod.o rkl1.o rklt1.o\
          rsdl1.o smc1.o sorc1.o splinx.o spliny.o step.o stomp1.o\
          tmpr1.o tmstep.o tortu.o update_nodes.o updt1.o watlad.o\
          watlqv.o watsp.o wrmsg.o wrplot.o wrrst.o stomp1_driver.o\
          watgsd.o airgsd.o average_v.o tport1.o crntnb.o sjcbl.o\ sbnd1.o\
          updtc.o sprp1.o sort1.o rdrst.o stomp_terminate.o idata.o\
          s_part.o cbgc.o user_t.o Biogeochem.o Biogeochem_log.o mrate.o

OBJS = $(FOBJS)

STOMP1: ${OBJS}
        -${FC} -o stomp1_timer.x $(OBJS) $(ALL_INCLUDES) $(ALL_LIBS) ${PETSC_LIB}

#clean :
#       /bin/rm -f *.o *.mod $(ECECF)
```

MakeIncl.user

```
# This file contains user-specified component-specific variables.
# Other settings can be specified through environment variables,
# e.g., CXX, CXX_FLAGS, or command line arguments to make.
# Also, other makefile segments can be included here.
# ---------------------------------------------------------------
# Include path directives, including paths to Fortran modules
STOMP_INCLUDES =  -I/autofs/na1_home/fang/xt5/software/ga_lib/include \
                  -I/autofs/na1_home/fang/xt5/software/petsc-3.1-p4

# Library paths and names
GA_LIB  = \
        -L/autofs/na1_home/fang/xt5/software/petsc-3.1-p4/linux-gnu-c-opt/lib -lpetsc \
        -L/autofs/na1_home/fang/xt5/software/ga_lib/lib -lga -larmci \
        -L/ccs/home/fang/profiling/ -lpapic \
        -L/opt/xt-tools/papi/3.6.2.2/lib -lpapi -lpfm -lsci -lm
```

Q2 Run Script

```
#!/bin/bash
#PBS -V
#PBS -A csc074stomp
#PBS -N test
#PBS -j oe
```

```
# Number of compute cores (i.e. size=8 means 4 nodes with 2 cores each)
#PBS -l walltime=12:00:00,size=158400

cd $PBS_O_WORKDIR
date

rm -f core*
limit coredumpsize 128megabytes

## -N: number of procs per node, -n: number of procs
aprun -n 52800 -N 4 /tmp/work/fang/rifle_src/stomp1_output.x
```

Q2 Runtime Environment

```
Currently Loaded Modulefiles:
  1) modules/3.1.6
  2) DefApps
  3) torque/2.4.1b1-snap.200905191614
  4) moab/5.3.6
  5) /opt/cray/xt-asyncpe/default/modulefiles/xtpe-istanbul
  6) cray/MySQL/5.0.64-1.0000.2342.16.1
  7) xtpe-target-cnl
  8) xt-service/2.2.73
  9) xt-os/2.2.73
 10) xt-boot/2.2.73
 11) xt-lustre-ss/2.2_1.6.5
 12) cray/job/1.5.5-0.1_2.0202.21413.56.6
 13) cray/csa/3.0.0-1_2.0202.21426.77.6
 14) cray/account/1.0.0-2.0202.19482.49.17
 15) cray/projdb/1.0.0-1.0202.19483.52.1
 16) Base-opts/2.2.73
 17) pgi/10.3.0
 18) xt-libsci/10.4.4
 19) pmi/1.0-1.0000.7628.10.2.ss
 20) xt-mpt/4.0.0
 21) xt-pe/2.2.73
 22) xt-asyncpe/3.7
 23) PrgEnv-pgi/2.2.73
 24) xt-papi/3.6.2.2

roche@jaguarpf-login7:~> qstat -f -ln -u fang > qs.txt
roche@jaguarpf-login7:~> cat qs.txt

nid00004:
                                                                    Req'd  Req'd   Elap
Job ID               Username Queue    Jobname          SessID NDS   TSK Memory Time  S Time
-------------------- -------- -------- ---------------- ------ ----- --- ------ ----- - -----
399403.nid00004      fang     batch    test              17743     1 158400    --  12:00 C 04:03
   batch1/0
399643.nid00004      fang     batch    test                --      1 158400    --  10:00 Q   --
    --
399658.nid00004      fang     batch    test                --      1 158400    --  10:00 Q   --
    --
399732.nid00004      fang     batch    test                --      1    360    --  01:00 Q   --
    --
roche@jaguarpf-login7:~> date
Fri Apr  1 17:04:24 EDT 2011
```

Q4 Compilation Script

The Global Arrays library, `ga5.0.2`, was built on the Cray XT5 at ORNL. One simply requests an interactive session so that the the compute nodes are targeted during the build. The library was configured thusly: `./configure --with-mpi --with-portals --enable-i4 --prefix=/tmp/work/roche`.

```
clear ;
echo ' ';
date ;
echo ' ';
echo '... starting C builds';
echo ' ';
          echo 'cc -c krp-init.c';
  cc -c ${PAPI_INCLUDE_OPTS} krp-init.c;
  echo 'cc -c krp-rpt.c ';
          cc -c ${PAPI_INCLUDE_OPTS} krp-rpt.c;
          echo 'cc -c krp-rpt-init.c';
          cc -c ${PAPI_INCLUDE_OPTS} krp-rpt-init.c;
          echo 'cc -c krp-rpt-init-sum.c';
          cc -c ${PAPI_INCLUDE_OPTS} krp-rpt-init-sum.c;
          echo 'cc -c krp-rpt-init-sum2.c';
          cc -c ${PAPI_INCLUDE_OPTS} krp-rpt-init-sum2.c;
echo ' ';
echo '... starting Fortran builds';
echo ' ';
          echo 'ftn -c petscapp.F90';
          ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} petscapp.F90;
          echo 'ftn -c chem.F90';
          ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} chem.F90;
          echo 'ftn -c grid_mod.F90';
          ftn -c grid_mod.F90;
          echo 'ftn -c rd_mod.F90';
          ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rd_mod.F90;
          echo 'ftn -c banner.F90';
          ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} banner.F90;
          echo 'ftn -c bcf1.F90';
          ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} bcf1.F90;
          echo 'ftn -c bgc.F90';
          ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} bgc.F90 ;
          echo 'ftn -c bgc_param.F90';
          ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} bgc_param.F90;
          echo 'ftn -c bcj1.F90';
          ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} bcj1.F90;
          echo 'ftn -c bcp1.F90';
          ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} bcp1.F90;
          echo 'ftn -c cap1.F90';
          ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} cap1.F90;
          echo 'ftn -c chk1.F90';
          ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} chk1.F90;
          echo 'ftn -c cisc1.F90';
          ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} cisc1.F90;
          echo 'ftn -c difmn.F90';
          ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} difmn.F90;
          echo 'ftn -c dpor1.F90';
          ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} dpor1.F90;
          echo 'ftn -c drcvl.F90';
```

```
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} drcvl.F90;
echo 'ftn -c drcvlb.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} drcvlb.F90;
echo 'ftn -c eckechem.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} eckechem.F90;
echo 'ftn -c elc1.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} elc1.F90;
echo 'ftn -c factor.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} factor.F90;
echo 'ftn -c fnhgbl.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} fnhgbl.F90;
echo 'ftn -c fntblx.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} fntblx.F90;
echo 'ftn -c fntbly.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} fntbly.F90;
echo 'ftn -c fsplnx.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} fsplnx.F90;
echo 'ftn -c fsplny.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} fsplny.F90;
echo 'ftn -c GA_grid.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} GA_grid.F90;
echo 'ftn -c incrm1.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} incrm1.F90;
echo 'ftn -c intlz.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} intlz.F90;
echo 'ftn -c jcbp.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} jcbp.F90;
echo 'ftn -c jcbwl.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} jcbwl.F90;
echo 'ftn -c ksp1.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} ksp1.F90;
echo 'ftn -c lcase.F90';
ftn -c -i4 ${GA_INCL_OPT}_INCL_OPT} lcase.F90;
echo 'ftn -c ldo1.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} ldo1.F90;
echo 'ftn -c petsccalls.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} petsccalls.F90;
echo 'ftn -c porsty.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} porsty.F90;
echo 'ftn -c qsat1.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} qsat1.F90;
echo 'ftn -c rd_chem.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rd_chem.F90;
echo 'ftn -c rdbc1.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdbc1.F90;
echo 'ftn -c rdchr.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdchr.F90;
echo 'ftn -c rddpr.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rddpr.F90;
echo 'ftn -c rdgrid.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdgrid.F90;
echo 'ftn -c rdhydr.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdhydr.F90;
echo 'ftn -c rdic1.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdic1.F90;
echo 'ftn -c rdijk.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdijk.F90;
```

```
echo 'ftn -c rdinac.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdinac.F90;
echo 'ftn -c rdini.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdini.F90;
echo 'ftn -c rdinpl.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdinpl.F90;
echo 'ftn -c rdint.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdint.F90;
echo 'ftn -c rdlrp.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdlrp.F90;
echo 'ftn -c rdmech.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdmech.F90;
echo 'ftn -c rdou1.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdou1.F90;
echo 'ftn -c rdouun.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdouun.F90;
echo 'ftn -c rdrock.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdrock.F90;
echo 'ftn -c rdrst.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdrst.F90;
echo 'ftn -c rdsclf1.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdsclf1.F90;
echo 'ftn -c rdsf1.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdsf1.F90;
echo 'ftn -c rdsolu.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdsolu.F90;
echo 'ftn -c rdsp1.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdsp1.F90;
echo 'ftn -c rdsr1.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdsr1.F90;
echo 'ftn -c rdtf1.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdtf1.F90;
echo 'ftn -c rdtp1.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdtp1.F90;
echo 'ftn -c rdunit.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rdunit.F90;
echo 'ftn -c refnod.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} refnod.F90;
echo 'ftn -c rkl1.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rkl1.F90;
echo 'ftn -c rklt1.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rklt1.F90;
echo 'ftn -c rsdl1.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} rsdl1.F90;
echo 'ftn -c sbnd1.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} sbnd1.F90;
echo 'ftn -c sjcbl.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} sjcbl.F90;
echo 'ftn -c smc1.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} smc1.F90;
echo 'ftn -c sorc1.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} sorc1.F90;
echo 'ftn -c splinx.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} splinx.F90;
echo 'ftn -c spliny.F90';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} spliny.F90;
echo 'ftn -c sprp1.F90' ;
```

```
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} sprp1.F90;
        echo 'ftn -c step.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} step.F90;
        echo 'ftn -c stomp1.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} stomp1.F90;
        echo 'ftn -c tmpr1.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} tmpr1.F90;
        echo 'ftn -c tmstep.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} tmstep.F90;
        echo 'ftn -c tortu.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} tortu.F90;
        echo 'ftn -c trpgl1.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} trpgl1.F90;
        echo 'ftn -c update_nodes.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} update_nodes.F90;
        echo 'ftn -c updt1.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} updt1.F90;
        echo 'ftn -c watlad.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} watlad.F90;
        echo 'ftn -c watlqv.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} watlqv.F90;
        echo 'ftn -c watsp.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} watsp.F90;
        echo 'ftn -c wrmsg.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} wrmsg.F90;
        echo 'ftn -c wrplot.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} wrplot.F90;
        echo 'ftn -c wrrst.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} wrrst.F90;
        echo 'ftn -c stomp1_driver.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} stomp1_driver.F90;
        echo 'ftn -c watgsd.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} watgsd.F90;
        echo 'ftn -c airgsd.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} airgsd.F90;
        echo 'ftn -c average_v.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} average_v.F90;
        echo 'ftn -c tport1.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} tport1.F90;
        echo 'ftn -c crntnb.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} crntnb.F90;
        echo 'ftn -c sjcbl.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} sjcbl.F90;
        echo 'ftn -c updtc.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} updtc.F90;
        echo 'ftn -c sort1.F90';
        ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} sort1.F90;
echo 'ftn -c stomp_terminate.o ';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} stomp_terminate.F90;
echo 'ftn -c idata.o ';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} idata.F90 ;
echo 'ftn -c s_part.o ';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} s_part.F90 ;
echo 'ftn -c cbgc.o ';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} cbgc.F90;
echo 'ftn -c user_t.o ';
ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} user_t.F90;
```

```
   echo 'ftn -c Biogeochem.o ';
   ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} Biogeochem.F90;
   echo 'ftn -c Biogeochem_log.o ';
   ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} Biogeochem_log.F90;
   echo 'ftn -c mrate.o';
   ftn -c -i4 ${GA_INCL_OPT} ${PETSC_INCL_OPT} mrate.F90;

echo '... past intermediate builds';
echo '... starting binary build';
echo 'ftn -o xstmp-q4 bgc.o chem.o bgc_param.o grid_mod.o rd_mod.o petscapp.o banner.o bcf1.o bcj1.o
bcp1.o cap1.o chk1.o cisc1.o difmn.o dpor1.o drcvl.o drcvlb.o eckechem.o elc1.o factor.o fntblx.o
fntbly.o fsplnx.o fsplny.o GA_grid.o incrm1.o intlz.o jcbp.o jcbwl.o ksp1.o lcase.o ldo1.o
petsccalls.o porsty.o qsat1.o rd_chem.o rdbc1.o rdchr.o rddpr.o rdgrid.o rdhydr.o rdic1.o rdijk.o
rdinac.o rdini.o rdinpl.o rdint.o rdlrp.o rdmech.o rdou1.o rdouun.o rdrock.o rdsclf1.o rdsf1.o
rdsolu.o rdsp1.o rdsr1.o rdtf1.o rdtp1.o rdunit.o refnod.o rkl1.o rklt1.o rsdl1.o smc1.o sorc1.o
splinx.o spliny.o step.o stomp1.o tmpr1.o tmstep.o tortu.o update_nodes.o updt1.o watlad.o watlqv.o
watsp.o wrmsg.o wrplot.o wrrst.o stomp1_driver.o watgsd.o airgsd.o average_v.o tport1.o crntnb.o
sjcbl.o sbnd1.o updtc.o sprp1.o sort1.o rdrst.o stomp_terminate.o idata.o s_part.o cbgc.o user_t.o
Biogeochem.o Biogeochem_log.o mrate.o krp-init.o  krp-rpt-init.o  krp-rpt.o ${PAPI_POST_LINK_OPTS}
${GA_POST_LINK_OPTS} -L${PETSC_DIR}/lib -lcraypetsc ${CRAY_PETSC_FORTRAN_INCPATH_CNL_OPTS} -lm ';
ftn -o xstmp-q4 bgc.o chem.o bgc_param.o grid_mod.o rd_mod.o petscapp.o banner.o bcf1.o bcj1.o
bcp1.o cap1.o chk1.o cisc1.o difmn.o dpor1.o drcvl.o drcvlb.o eckechem.o elc1.o factor.o fntblx.o
fntbly.o fsplnx.o fsplny.o GA_grid.o incrm1.o intlz.o jcbp.o jcbwl.o ksp1.o lcase.o ldo1.o
petsccalls.o porsty.o qsat1.o rd_chem.o rdbc1.o rdchr.o rddpr.o rdgrid.o rdhydr.o rdic1.o rdijk.o
rdinac.o rdini.o rdinpl.o rdint.o rdlrp.o rdmech.o rdou1.o rdouun.o rdrock.o rdsclf1.o rdsf1.o
rdsolu.o rdsp1.o rdsr1.o rdtf1.o rdtp1.o rdunit.o refnod.o rkl1.o rklt1.o rsdl1.o smc1.o sorc1.o
splinx.o spliny.o step.o stomp1.o tmpr1.o tmstep.o tortu.o update_nodes.o updt1.o watlad.o watlqv.o
watsp.o wrmsg.o wrplot.o wrrst.o stomp1_driver.o watgsd.o airgsd.o average_v.o tport1.o crntnb.o
sjcbl.o sbnd1.o updtc.o sprp1.o sort1.o rdrst.o stomp_terminate.o idata.o s_part.o cbgc.o user_t.o
Biogeochem.o Biogeochem_log.o mrate.o krp-init.o  krp-rpt-init.o  krp-rpt.o ${PAPI_POST_LINK_OPTS}
${GA_POST_LINK_OPTS} -L${PETSC_DIR}/lib -lcraypetsc ${CRAY_PETSC_FORTRAN_INCPATH_CNL_OPTS} -lm ;
echo '... past binary build: xstmp-q4';
```

Q4 Run Script

```
#PBS -V
#PBS -l walltime=04:00:00,size=3300
#PBS -A csc074stomp
#PBS -N stmp-q4
#PBS -j oe

cd ${PBS_O_WORKDIR}
time aprun -n 3300 ./xstmp-q4
```

Q4 Runtime Environment

There is minimal work required to define some environment variables and relevant paths not covered by the modules on the target: Note that the PETSC_INCL_OPT is weird. It is due to how the PETSc headers are included in the codes.

```
module load xt-papi/3.6.2 petsc
export PETSC_INCL_OPT=-I/opt/petsc/3.1.04/real/PGI/linux;
export GA_DIR=/tmp/work/roche;
export GA_INCL_OPT=-I${GA_DIR}/include;
export GA_POST_LINK_OPTS="-L${GA_DIR}/lib -lga"
```

So the module list reads:

```
Currently Loaded Modulefiles:
  1) modules/3.1.6
  2) DefApps
  3) torque/2.4.1b1-snap.200905191614
  4) moab/5.3.6
    5) /opt/cray/xt-asyncpe/default/modulefiles/xtpe-istanbul
  6) cray/MySQL/5.0.64-1.0000.2342.16.1
  7) xtpe-target-cnl
  8) xt-service/2.2.73
  9) xt-os/2.2.73
 10) xt-boot/2.2.73
 11) xt-lustre-ss/2.2_1.6.5
 12) cray/job/1.5.5-0.1_2.0202.21413.56.6
 13) cray/csa/3.0.0-1_2.0202.21426.77.6
 14) cray/account/1.0.0-2.0202.19482.49.17
15) cray/projdb/1.0.0-1.0202.19483.52.1
16) Base-opts/2.2.73
17) pgi/10.9.0
18) xt-libsci/10.5.0
19) pmi/1.0-1.0000.8160.39.1.ss
20) xt-mpt/5.1.3
21) xt-pe/2.2.73
22) xt-asyncpe/4.9
23) PrgEnv-pgi/2.2.73
24) xt-papi/3.6.2
25) petsc/3.1.04
```

Q4 Compilation Script - STOMP Lustre Study

```
cmpl.f-usr :
ftn -c f-usr.f90 ;
cc -c ${PAPI_INCLUDE_OPTS} krp-init.c ;
cc -c ${PAPI_INCLUDE_OPTS} krp-rpt-init.c ;
cc -c ${PAPI_INCLUDE_OPTS} krp-rpt-init-sum.c ;
cc -c ${PAPI_INCLUDE_OPTS} krp-rpt.c ;
cc -c get-my-lattice.c ;
cc -c get-lattice-data.c ;
cc -c get-lattice-indices.c ;
cc -c ${LIBLUT_INCLUDE_OPTS} wr-lstr-orcl-b-sync.c ;
cc -c stmp-uf-rd.c ;
cc -c stmp-f-rd.c ;
cc -c wr.c ;
cc -c ascii-wr.c ;
ftn -o xf-usr f-usr.o krp-init.o krp-rpt-init.o krp-rpt-init-sum.o krp-rpt.o get-lattice-indices.o
get-my-lattice.o get-lattice-data.o wr-lstr-orcl-b-sync.o stmp-uf-rd.o stmp-f-rd.o wr.o
ascii-wr.o ${PAPI_POST_LINK_OPTS} ${LIBLUT_POST_LINK_OPTS} -lLUT -lm
```

Q4 Run Script - STOMP Lustre Study

```
#PBS -V
#PBS -l walltime=10:00:00,size=13200
#PBS -A csc074stomp
#PBS -N lwrio-orcl-stmp
#PBS -j oe

cd ${PBS_O_WORKDIR}
time aprun -n 3300 ./xf-usr
```

Note, the reason for asking for 13,200 PEs is because to execute the oracle study (i.e. exhaustive search) might require a longer runtime than the queue policy on this machine allows for 3300 PEs. The input required to execute the study is the same as provided above for the Q2 and Q4 runs.

Q4 Runtime Environment - STOMP Lustre Study

```
Currently Loaded Modulefiles:
  1) modules/3.1.6
  2) DefApps
  3) torque/2.4.1b1-snap.200905191614
  4) moab/5.3.6
  5) /opt/cray/xt-asyncpe/default/modulefiles/xtpe-istanbul
  6) cray/MySQL/5.0.64-1.0000.2342.16.1
  7) xtpe-target-cnl
  8) xt-service/2.2.73
  9) xt-os/2.2.73
 10) xt-boot/2.2.73
 11) xt-lustre-ss/2.2_1.6.5
 12) cray/job/1.5.5-0.1_2.0202.21413.56.6
 13) cray/csa/3.0.0-1_2.0202.21426.77.6
14) cray/account/1.0.0-2.0202.19482.49.17
15) cray/projdb/1.0.0-1.0202.19483.52.1
16) Base-opts/2.2.73
17) pgi/10.9.0
18) xt-libsci/10.5.0
19) pmi/1.0-1.0000.8160.39.1.ss
20) xt-mpt/5.1.3
21) xt-pe/2.2.73
22) xt-asyncpe/4.9
23) PrgEnv-pgi/2.2.73
24) xt-papi/3.6.2
25) liblut/0.9.9
```

0.6.2 LAMMPS

The LAMMPS benchmarks have been conducted with both Intel and Portland Group Inc. (PGI) compilers in Q2. We report information for both cases. The same problem is executed regardless of the compiler suite used.
In Q4, another interaction, Lennart-Jones as described in the text, was added and the GPU accelerated algorithms were built and tested on the target architecture using GNU Compiler Collection (GCC). For consistency, the Q2 runs were re-executed with the GCC-built version of the binary to make the comparison to the host + GPU runs meaningful.

Q2 Input Settings in.eam

```
# bulk Cu lattice

variable x index 1
variable y index 1
variable z index 1

variable xx equal 20*$x
variable yy equal 20*$y
variable zz equal 20*$z

units metal
atom_style atomic

lattice fcc 3.615
region box block 0 ${xx} 0 ${yy} 0 ${zz}
create_box 1 box
create_atoms 1 box

pair_style eam
pair_coeff 1 1 Cu_u3.eam

velocity all create 1600.0 376847 loop geom

neighbor 1.0 bin
neigh_modify    every 1 delay 5 check yes

fix 1 all nve

timestep 0.005
thermo 50

run 100
```

Q2 Compilation Script Intel

```
SHELL = /bin/sh

# -----------------------------------------------------------------------
# compiler/linker settings
# specify flags and libraries needed for your compiler

CC = icpc
CCFLAGS = -fast
DEPFLAGS = -M
LINK = icpc
LINKFLAGS =
LIB =           # -lstdc++
ARCHIVE = ar
```

```
ARFLAGS = -rc
SIZE = size

# ---------------------------------------------------------------------
# LAMMPS-specific settings
# specify settings for LAMMPS features you will use

# LAMMPS ifdef options, see doc/Section_start.html

LMP_INC = -DLAMMPS_GZIP

# MPI library, can be src/STUBS dummy lib
# INC = path for mpi.h, MPI compiler settings
# PATH = path for MPI library
# LIB = name of MPI library

MPI_INC =  -I/usr/common/usg/openmpi/1.4.2/intel/include
MPI_PATH =
MPI_LIB = -L/usr/common/usg/openmpi/1.4.2/intel/lib -lmpi

# FFT library, can be -DFFT_NONE if not using PPPM from KSPACE package
# INC = -DFFT_FFTW, -DFFT_INTEL, -DFFT_NONE, etc, FFT compiler settings
# PATH = path for FFT library
# LIB = name of FFT library

# FFTW=-I/usr/common/usg/fftw/2.1.5/include -L/usr/common/usg/fftw/2.1.5/lib

FFT_INC = -DFFT_FFTW -I/usr/common/usg/fftw/2.1.5-Intel/include
FFT_PATH =
FFT_LIB = -L/usr/common/usg/fftw/2.1.5-Intel/lib -lfftw

# kr -add profiling capability for q2 omb pmm ascr bm
PAPI_INC = -I/usr/common/usg/papi/4.0.0/include
PAPI_LIB = -L/usr/common/usg/papi/4.0.0/lib -lpapi -lpfm

# additional system libraries needed by LAMMPS package libraries
# these settings are IGNORED if the corresponding LAMMPS package
#   (e.g. gpu, meam) is NOT included in the LAMMPS build
# SYSLIB = names of libraries
# SYSPATH = paths of libraries

gpu_SYSLIB =       -lcudart
meam_SYSLIB =      -lifcore -lsvml -lompstub -limf
reax_SYSLIB =      -lifcore -lsvml -lompstub -limf
user-atc_SYSLIB =  -lblas -llapack

gpu_SYSPATH =      -L/usr/local/cuda/lib64
meam_SYSPATH =     -L/opt/intel/fce/10.0.023/lib
reax_SYSPATH =     -L/opt/intel/fce/10.0.023/lib
user-atc_SYSPATH =

# ---------------------------------------------------------------------
# build rules and dependencies
# no need to edit this section

include Makefile.package
```

```
EXTRA_INC = $(LMP_INC) $(PKG_INC) $(MPI_INC) $(FFT_INC) $(PAPI_INC)
EXTRA_PATH = $(PKG_PATH) $(MPI_PATH) $(FFT_PATH) $(PKG_SYSPATH)
EXTRA_LIB = $(PKG_LIB) $(MPI_LIB) $(FFT_LIB) $(PAPI_LIB) $(PKG_SYSLIB)

# Link target

$(EXE): $(OBJ)
$(LINK) $(LINKFLAGS) $(EXTRA_PATH) $(OBJ) $(EXTRA_LIB) $(LIB) -o $(EXE)
$(SIZE) $(EXE)

# Library target

lib: $(OBJ)
$(ARCHIVE) $(ARFLAGS) $(EXE) $(OBJ)

# Compilation rules

%.o:%.cpp
$(CC) -alias=ansi $(CCFLAGS) $(EXTRA_INC) -c $<

%.d:%.cpp
$(CC) $(CCFLAGS) $(EXTRA_INC) $(DEPFLAGS) $< > $@

# Individual dependencies

DEPENDS = $(OBJ:.o=.d)
include $(DEPENDS)
```

This compilation will fail to grab the profiling routines used to instrument LAMMPS. Here is the additional step required to complete the build:

```
icc -I/usr/common/usg/papi/4.0.0/include krp-init.c ;
icc -I/usr/common/usg/papi/4.0.0/include krp-rpt.c ;

icpc angle.o angle_charmm.o angle_cosine.o angle_cosine_delta.o angle_cosine_periodic.o\
     angle_cosine_squared.o angle_harmonic.o angle_hybrid.o angle_table.o atom.o\
     atom_vec.o atom_vec_angle.o atom_vec_atomic.o atom_vec_bond.o atom_vec_charge.o\
     atom_vec_full.o atom_vec_hybrid.o atom_vec_molecular.o bond.o bond_fene.o\
     bond_fene_expand.o bond_harmonic.o bond_hybrid.o bond_morse.o bond_nonlinear.o\
     bond_quartic.o bond_table.o change_box.o comm.o compute.o compute_angle_local.o\
     compute_atom_molecule.o compute_bond_local.o compute_centro_atom.o\
     compute_cluster_atom.o compute_cna_atom.o compute_com.o compute_com_molecule.o\
     compute_coord_atom.o compute_dihedral_local.o compute_displace_atom.o\
     compute_erotate_sphere.o compute_group_group.o compute_gyration.o\
     compute_gyration_molecule.o compute_heat_flux.o compute_improper_local.o\
     compute_ke.o compute_ke_atom.o compute_msd.o compute_msd_molecule.o compute_pair.o\
     compute_pair_local.o compute_pe.o compute_pe_atom.o compute_pressure.o\
     compute_property_atom.o compute_property_local.o compute_property_molecule.o\
     compute_rdf.o compute_reduce.o compute_reduce_region.o compute_stress_atom.o\
     compute_temp.o compute_temp_com.o compute_temp_deform.o compute_temp_partial.o\
     compute_temp_profile.o compute_temp_ramp.o compute_temp_region.o\
     compute_temp_sphere.o compute_ti.o create_atoms.o create_box.o delete_atoms.o\
     delete_bonds.o dihedral.o dihedral_charmm.o dihedral_harmonic.o dihedral_helix.o\
     dihedral_hybrid.o dihedral_multi_harmonic.o dihedral_opls.o displace_atoms.o\
     displace_box.o domain.o dump.o dump_atom.o dump_cfg.o dump_custom.o dump_dcd.o\
     dump_local.o dump_xyz.o error.o ewald.o fft3d.o fft3d_wrap.o finish.o fix.o\
     fix_adapt.o fix_addforce.o fix_ave_atom.o fix_ave_correlate.o fix_ave_histo.o\
```

```
fix_ave_spatial.o fix_ave_time.o fix_aveforce.o fix_bond_break.o fix_bond_create.o\
fix_bond_swap.o fix_box_relax.o fix_deform.o fix_deposit.o fix_drag.o fix_dt_reset.o\
fix_efield.o fix_enforce2d.o fix_evaporate.o fix_external.o fix_gravity.o fix_heat.o\
fix_indent.o fix_langevin.o fix_lineforce.o fix_minimize.o fix_momentum.o fix_move.o\
fix_nh.o fix_nh_sphere.o fix_nph.o fix_nph_sphere.o fix_npt.o fix_npt_sphere.o\
fix_nve.o fix_nve_limit.o fix_nve_noforce.o fix_nve_sphere.o fix_nvt.o\
fix_nvt_sllod.o fix_nvt_sphere.o fix_orient_fcc.o fix_planeforce.o\
fix_press_berendsen.o fix_print.o fix_qeq_comb.o fix_read_restart.o fix_recenter.o\
fix_respa.o fix_rigid.o fix_rigid_nve.o fix_rigid_nvt.o fix_setforce.o fix_shake.o\
fix_shear_history.o fix_spring.o fix_spring_rg.o fix_spring_self.o fix_store_force.o\
fix_store_state.o fix_temp_berendsen.o fix_temp_rescale.o fix_thermal_conductivity.o\
fix_tmd.o fix_ttm.o fix_viscosity.o fix_viscous.o fix_wall.o fix_wall_harmonic.o\
fix_wall_lj126.o fix_wall_lj93.o fix_wall_reflect.o fix_wall_region.o force.o group.o\
improper.o improper_cvff.o improper_harmonic.o improper_hybrid.o improper_umbrella.o\
input.o integrate.o irregular.o kspace.o lammps.o lattice.o library.o main.o\
memory.o min.o min_cg.o min_fire.o min_hftn.o min_linesearch.o min_quickmin.o\
min_sd.o minimize.o modify.o neigh_bond.o neigh_derive.o neigh_full.o neigh_gran.o\
neigh_half_bin.o neigh_half_multi.o neigh_half_nsq.o neigh_list.o neigh_request.o\
neigh_respa.o neigh_stencil.o neighbor.o output.o pack.o pair.o pair_airebo.o\
pair_born.o pair_born_coul_long.o pair_buck.o pair_buck_coul_cut.o\
pair_buck_coul_long.o pair_comb.o pair_coul_cut.o pair_coul_debye.o pair_coul_long.o\
pair_dpd.o pair_dpd_tstat.o pair_eam.o pair_eam_alloy.o pair_eam_fs.o pair_eim.o\
pair_gauss.o pair_hbond_dreiding_lj.o pair_hbond_dreiding_morse.o pair_hybrid.o\
pair_hybrid_overlay.o pair_lj96_cut.o pair_lj_charmm_coul_charmm.o\
pair_lj_charmm_coul_charmm_implicit.o pair_lj_charmm_coul_long.o pair_lj_cut.o\
pair_lj_cut_coul_cut.o pair_lj_cut_coul_debye.o pair_lj_cut_coul_long.o\
pair_lj_cut_coul_long_tip4p.o pair_lj_expand.o pair_lj_gromacs.o\
pair_lj_gromacs_coul_gromacs.o pair_lj_smooth.o pair_morse.o pair_soft.o pair_sw.o\
pair_table.o pair_tersoff.o pair_tersoff_zbl.o pair_yukawa.o pppm.o pppm_tip4p.o\
random_mars.o random_park.o read_data.o read_restart.o region.o region_block.o\
region_cone.o region_cylinder.o region_intersect.o region_plane.o region_prism.o\
region_sphere.o region_union.o remap.o remap_wrap.o replicate.o respa.o run.o set.o\
special.o thermo.o timer.o universe.o update.o variable.o velocity.o verlet.o\
write_restart.o krp-init.o krp-rpt.o\
-L/usr/common/usg/openmpi/1.4.2/intel/lib -lmpi\
-L/usr/common/usg/fftw/2.1.5-Intel/lib -lfftw\
-L/usr/common/usg/papi/4.0.0/lib -lpapi -lpfm -o ../lmp_intel-papi
```

PGI

```
# pgi = Portland Group compiler, pgCC, MPICH, FFTW

SHELL = /bin/sh

# ----------------------------------------------------------------------
# compiler/linker settings
# specify flags and libraries needed for your compiler

CC = pgCC
C = pgcc
CCFLAGS = -alias=ansi -fast
DEPFLAGS = -M
LINK = pgCC
LINKFLAGS =
LIB =           -lstdc++
ARCHIVE = ar
```

```
ARFLAGS = -rc
SIZE = size

# ---------------------------------------------------------------------
# LAMMPS-specific settings
# specify settings for LAMMPS features you will use

# LAMMPS ifdef options, see doc/Section_start.html

LMP_INC = -DLAMMPS_GZIP

# MPI library, can be src/STUBS dummy lib

MPI_INC =  -I/usr/common/usg/openmpi/1.4.2/pgi/include
MPI_PATH = -L/usr/common/usg/openmpi/1.4.2/pgi/lib
MPI_LIB = -lmpi

# FFT library, can be -DFFT_NONE if not using PPPM from KSPACE package

FFT_INC = -DFFT_FFTW -I/usr/common/usg/fftw/2.1.5/include
FFT_PATH =
FFT_LIB = -L/usr/common/usg/fftw/2.1.5/lib -lfftw

# kr -add profiling capability for q2 omb pmm ascr bm

PAPI_INC = -I/usr/common/usg/papi/4.0.0/include
PAPI_LIB = -L/usr/common/usg/papi/4.0.0/lib -lpapi -lpfm

# additional system libraries needed by LAMMPS package libraries
# these settings are IGNORED if the corresponding LAMMPS package
#   (e.g. gpu, meam) is NOT included in the LAMMPS build
# SYSLIB = names of libraries
# SYSPATH = paths of libraries

gpu_SYSLIB =       -lcudart
meam_SYSLIB =      -lifcore -lsvml -lompstub -limf
reax_SYSLIB =      -lifcore -lsvml -lompstub -limf
user-atc_SYSLIB =  -lblas -llapack

gpu_SYSPATH =      -L/usr/local/cuda/lib64
meam_SYSPATH =     -L/opt/intel/fce/10.0.023/lib
reax_SYSPATH =     -L/opt/intel/fce/10.0.023/lib
user-atc_SYSPATH =

# ---------------------------------------------------------------------
# build rules and dependencies
# no need to edit this section

include Makefile.package

EXTRA_INC = $(LMP_INC) $(PKG_INC) $(MPI_INC) $(FFT_INC) $(PAPI_INC)
EXTRA_PATH = $(PKG_PATH) $(MPI_PATH) $(FFT_PATH) $(PKG_SYSPATH)
EXTRA_LIB = $(PKG_LIB) $(MPI_LIB) $(FFT_LIB) $(PAPI_LIB) $(PKG_SYSLIB)

# Link target

$(EXE): $(OBJ)
```

```
$(LINK) $(LINKFLAGS) $(EXTRA_PATH) $(OBJ) $(EXTRA_LIB) $(LIB) -o $(EXE)
$(SIZE) $(EXE)

# Library target

lib: $(OBJ)
$(ARCHIVE) $(ARFLAGS) $(EXE) $(OBJ)

# Compilation rules

%.o:%.cpp
$(CC) $(CCFLAGS) $(EXTRA_INC) -c $<

%.d:%.cpp
$(CC) $(CCFLAGS) $(EXTRA_INC) $(DEPFLAGS) $< > $@

# Individual dependencies

DEPENDS = $(OBJ:.o=.d)
include $(DEPENDS)
```

The additional step:

```
pgcc -I/usr/common/usg/papi/4.0.0/include -c krp-init.c ;
pgcc -I/usr/common/usg/papi/4.0.0/include -c krp-rpt.c ;
pgCC angle.o angle_charmm.o angle_cosine.o angle_cosine_delta.o angle_cosine_periodic.o\
     angle_cosine_squared.o angle_harmonic.o angle_hybrid.o angle_table.o atom.o\
     atom_vec.o atom_vec_angle.o atom_vec_atomic.o atom_vec_bond.o atom_vec_charge.o\
     atom_vec_full.o atom_vec_hybrid.o atom_vec_molecular.o bond.o bond_fene.o\
     bond_fene_expand.o bond_harmonic.o bond_hybrid.o bond_morse.o bond_nonlinear.o\
     bond_quartic.o bond_table.o change_box.o comm.o compute.o compute_angle_local.o\
     compute_atom_molecule.o compute_bond_local.o compute_centro_atom.o\
     compute_cluster_atom.o compute_cna_atom.o compute_com.o compute_com_molecule.o\
     compute_coord_atom.o compute_dihedral_local.o compute_displace_atom.o\
     compute_erotate_sphere.o compute_group_group.o compute_gyration.o\
     compute_gyration_molecule.o compute_heat_flux.o compute_improper_local.o\
     compute_ke.o compute_ke_atom.o compute_msd.o compute_msd_molecule.o compute_pair.o\
     compute_pair_local.o compute_pe.o compute_pe_atom.o compute_pressure.o\
     compute_property_atom.o compute_property_local.o compute_property_molecule.o\
     compute_rdf.o compute_reduce.o compute_reduce_region.o compute_stress_atom.o\
     compute_temp.o compute_temp_com.o compute_temp_deform.o compute_temp_partial.o\
     compute_temp_profile.o compute_temp_ramp.o compute_temp_region.o\
     compute_temp_sphere.o compute_ti.o create_atoms.o create_box.o delete_atoms.o\
     delete_bonds.o dihedral.o dihedral_charmm.o dihedral_harmonic.o dihedral_helix.o\
     dihedral_hybrid.o dihedral_multi_harmonic.o dihedral_opls.o displace_atoms.o\
     displace_box.o domain.o dump.o dump_atom.o dump_cfg.o dump_custom.o dump_dcd.o\
     dump_local.o dump_xyz.o error.o ewald.o fft3d.o fft3d_wrap.o finish.o fix.o\
     fix_adapt.o fix_addforce.o fix_ave_atom.o fix_ave_correlate.o fix_ave_histo.o\
     fix_ave_spatial.o fix_ave_time.o fix_aveforce.o fix_bond_break.o fix_bond_create.o\
     fix_bond_swap.o fix_box_relax.o fix_deform.o fix_deposit.o fix_drag.o fix_dt_reset.o\
     fix_efield.o fix_enforce2d.o fix_evaporate.o fix_external.o fix_gravity.o fix_heat.o\
     fix_indent.o fix_langevin.o fix_lineforce.o fix_minimize.o fix_momentum.o fix_move.o\
     fix_nh.o fix_nh_sphere.o fix_nph.o fix_nph_sphere.o fix_npt.o fix_npt_sphere.o\
     fix_nve.o fix_nve_limit.o fix_nve_noforce.o fix_nve_sphere.o fix_nvt.o\
     fix_nvt_sllod.o fix_nvt_sphere.o fix_orient_fcc.o fix_planeforce.o\
     fix_press_berendsen.o fix_print.o fix_qeq_comb.o fix_read_restart.o fix_recenter.o\
     fix_respa.o fix_rigid.o fix_rigid_nve.o fix_rigid_nvt.o fix_setforce.o fix_shake.o\
```

```
fix_shear_history.o fix_spring.o fix_spring_rg.o fix_spring_self.o fix_store_force.o\
fix_store_state.o fix_temp_berendsen.o fix_temp_rescale.o fix_thermal_conductivity.o\
fix_tmd.o fix_ttm.o fix_viscosity.o fix_viscous.o fix_wall.o fix_wall_harmonic.o\
fix_wall_lj126.o fix_wall_lj93.o fix_wall_reflect.o fix_wall_region.o force.o group.o\
improper.o improper_cvff.o improper_harmonic.o improper_hybrid.o improper_umbrella.o\
input.o integrate.o irregular.o kspace.o lammps.o lattice.o library.o main.o\
memory.o min.o min_cg.o min_fire.o min_hftn.o min_linesearch.o min_quickmin.o\
min_sd.o minimize.o modify.o neigh_bond.o neigh_derive.o neigh_full.o neigh_gran.o\
neigh_half_bin.o neigh_half_multi.o neigh_half_nsq.o neigh_list.o neigh_request.o\
neigh_respa.o neigh_stencil.o neighbor.o output.o pack.o pair.o pair_airebo.o\
pair_born.o pair_born_coul_long.o pair_buck.o pair_buck_coul_cut.o\
pair_buck_coul_long.o pair_comb.o pair_coul_cut.o pair_coul_debye.o pair_coul_long.o\
pair_dpd.o pair_dpd_tstat.o pair_eam.o pair_eam_alloy.o pair_eam_fs.o pair_eim.o\
pair_gauss.o pair_hbond_dreiding_lj.o pair_hbond_dreiding_morse.o pair_hybrid.o\
pair_hybrid_overlay.o pair_lj96_cut.o pair_lj_charmm_coul_charmm.o\
pair_lj_charmm_coul_charmm_implicit.o pair_lj_charmm_coul_long.o pair_lj_cut.o\
pair_lj_cut_coul_cut.o pair_lj_cut_coul_debye.o pair_lj_cut_coul_long.o\
pair_lj_cut_coul_long_tip4p.o pair_lj_expand.o pair_lj_gromacs.o\
pair_lj_gromacs_coul_gromacs.o pair_lj_smooth.o pair_morse.o pair_soft.o pair_sw.o\
pair_table.o pair_tersoff.o pair_tersoff_zbl.o pair_yukawa.o pppm.o pppm_tip4p.o\
random_mars.o random_park.o read_data.o read_restart.o region.o region_block.o\
region_cone.o region_cylinder.o region_intersect.o region_plane.o region_prism.o\
region_sphere.o region_union.o remap.o remap_wrap.o replicate.o respa.o run.o set.o\
special.o thermo.o timer.o universe.o update.o variable.o velocity.o verlet.o\
write_restart.o krp-init.o krp-rpt.o\
-L/usr/common/usg/openmpi/1.4.2/pgi/lib -lmpi\
-L/usr/common/usg/fftw/2.1.5/lib -lfftw\
-L/usr/common/usg/papi/4.0.0/lib -lpapi -lpfm -o ../lmp_pgi-papi
```

Q2 Run Script

Intel

```
#!/bin/bash
#PBS -V
#PBS -q dirac_special
#PBS -l nodes=44:ppn=8
#PBS -l walltime=00:10:00
#PBS -A m947
#PBS -N lmps_intel
#PBS -j oe

cd ${PBS_O_WORKDIR}
time mpirun -np 352 ./lmp_intel-papi < in.eam
```

PGI

```
#!/bin/bash
#PBS -V
#PBS -q dirac_special
#PBS -l nodes=44:ppn=8
#PBS -l walltime=00:10:00
```

```
#PBS -A m947
#PBS -N lmps_pgi
#PBS -j oe

cd ${PBS_O_WORKDIR}
time mpirun -np 352 ./lmp_pgi-papi < in.eam
```

Q2 Runtime Environment Intel

```
Currently Loaded Modulefiles:
  1) intel/11.1
  2) openmpi-intel/1.4.2
  3) papi/4.0.0
  4) fftw-Intel/2.1.5

[roche@cvrsvc06 roche]$ env
USER=roche
LOGNAME=roche
HOME=/global/homes/r/roche
PATH=/usr/common/usg/papi/4.0.0/bin:/usr/common/usg/openmpi/1.4.2/intel/bin:
     /usr/common/usg/intel/Compiler/11.1/072/bin/intel64:
     /usr/syscom/opt/moab/default/bin:/usr/syscom/opt/torque/default/bin:
     /usr/common/usg/bin:/usr/common/mss/bin:/usr/common/nsg/bin:
     /usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin
MAIL=/var/spool/mail/roche
SHELL=/bin/csh
SSH_CLIENT=67.207.117.6 65177 22
SSH_CONNECTION=67.207.117.6 65177 128.55.69.10 22
SSH_TTY=/dev/pts/26
TERM=xterm-color
HOSTTYPE=x86_64-linux
VENDOR=unknown
OSTYPE=linux
MACHTYPE=x86_64
SHLVL=1
PWD=/global/scratch/sd/roche
GROUP=roche
HOST=cvrsvc06
REMOTEHOST=67.207.117.6
HOSTNAME=cvrsvc06
INPUTRC=/etc/inputrc
G_BROKEN_FILENAMES=1
LESSOPEN=|/usr/bin/lesspipe.sh %s
NERSC_HOST=carver
MOABHOMEDIR=/usr/syscom/opt/moab/default
MODULE_VERSION=3.2.7
MODULE_VERSION_STACK=3.2.7
MODULESHOME=/usr/common/nsg/opt/Modules/3.2.7
MODULEPATH=/usr/common/nsg/opt/Modules/3.2.7/config/versions:
           /usr/common/nsg/opt/Modules/$MODULE_VERSION/modulefiles:
           /usr/common/nsg/Modules/3.2.7/config/:/opt/modules/3.1.6/modulefiles:
           /usr/common/usg/Modules/modulefiles:/usr/common/acts/Modules/modulefiles:
```

```
           /usr/common/nsg/Modules/modulefiles:/usr/common/ftg/Modules/modulefiles:
           /usr/common/graphics/Modules/modulefiles:
           /usr/common/mss/Modules/modulefiles:/usr/common/net/Modules/modulefiles:
           /usr/common/osg/Modules/modulefiles:/usr/common/osp/Modules/modulefiles
GSCRATCH=/global/scratch/sd/roche
SCRATCH=/global/scratch/sd/roche
TMPDIR=/global/scratch/sd/roche
CC=icc
FC=ifort
IDB=idb
INTEL_LICENSE_FILE=/usr/common/usg/intel/licenses/COM_L_CMPCT__F9GM-Z96WKXLH.lic
LD_LIBRARY_PATH=/usr/common/usg/papi/4.0.0/lib:/usr/common/usg/openmpi/1.4.2/intel/lib:
                /usr/common/usg/fftw/2.1.5-Intel/lib:
                /usr/common/usg/intel/Compiler/11.1/072/lib/intel64
LIBRARY_PATH=/usr/common/usg/intel/Compiler/11.1/072/lib/intel64
LOADEDMODULES=intel/11.1:fftw-Intel/2.1.5:openmpi-intel/1.4.2:papi/4.0.0
NLSPATH=/usr/common/usg/intel/Compiler/11.1/072/lib/intel64/locale/%l_%t/%N
_LMFILES_=/usr/common/usg/Modules/modulefiles/intel/11.1:
          /usr/common/usg/Modules/modulefiles/fftw-Intel/2.1.5:
          /usr/common/usg/Modules/modulefiles/openmpi-intel/1.4.2:
          /usr/common/usg/Modules/modulefiles/papi/4.0.0
FFTW=-I/usr/common/usg/fftw/2.1.5-Intel/include
-L/usr/common/usg/fftw/2.1.5-Intel/lib
FFTW_INC=/usr/common/usg/fftw/2.1.5-Intel/include
FFTW_LIBDIR=/usr/common/usg/fftw/2.1.5-Intel/lib
FFTW_ROOT=/usr/common/usg/fftw/2.1.5-Intel
INFOPATH=/usr/common/usg/fftw/2.1.5-Intel/info
OMPI_DIR=/usr/common/usg/openmpi/1.4.2/intel
PAPI_INCLUDE=/usr/common/usg/papi/4.0.0/include
PAPI_LIB=/usr/common/usg/papi/4.0.0/lib
PAPI_VERSION=4.0.0
```

PGI

```
Currently Loaded Modulefiles:
  1) pgi/10.8
  2) openmpi/1.4.2
  3) fftw/2.1.5
  4) papi/4.0.0


[roche@cvrsvc04 MAKE]$ env
USER=roche
LOGNAME=roche
HOME=/global/homes/r/roche
PATH=/usr/common/usg/papi/4.0.0/bin:/usr/common/usg/openmpi/1.4.2/pgi/bin:
     /usr/common/usg/pgi/10.8/linux86-64/10.8/bin:/usr/syscom/opt/moab/default/bin:
     /usr/syscom/opt/torque/default/bin:/usr/common/usg/bin:/usr/common/mss/bin:
     /usr/common/nsg/bin:/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin
MAIL=/var/spool/mail/roche
SHELL=/bin/csh
SSH_CLIENT=128.95.93.15 63497 22
SSH_CONNECTION=128.95.93.15 63497 128.55.69.10 22
SSH_TTY=/dev/pts/49
TERM=xterm-color
```

```
HOSTTYPE=x86_64-linux
VENDOR=unknown
OSTYPE=linux
MACHTYPE=x86_64
SHLVL=1
PWD=/global/scratch/sd/roche/lammps-15Mar11/src/MAKE
GROUP=roche
HOST=cvrsvc04
REMOTEHOST=128.95.93.15
HOSTNAME=cvrsvc04
INPUTRC=/etc/inputrc
G_BROKEN_FILENAMES=1
LESSOPEN=|/usr/bin/lesspipe.sh %s
NERSC_HOST=carver
MOABHOMEDIR=/usr/syscom/opt/moab/default
MODULE_VERSION=3.2.7
MODULE_VERSION_STACK=3.2.7
MODULESHOME=/usr/common/nsg/opt/Modules/3.2.7
MODULEPATH=/usr/common/nsg/opt/Modules/3.2.7/config/versions:
           /usr/common/nsg/opt/Modules/$MODULE_VERSION/modulefiles:
           /usr/common/nsg/Modules/3.2.7/config/:/opt/modules/3.1.6/modulefiles:
           /usr/common/usg/Modules/modulefiles:/usr/common/acts/Modules/modulefiles:
           /usr/common/nsg/Modules/modulefiles:/usr/common/ftg/Modules/modulefiles:
           /usr/common/graphics/Modules/modulefiles:
           /usr/common/mss/Modules/modulefiles:/usr/common/net/Modules/modulefiles:
           /usr/common/osg/Modules/modulefiles:/usr/common/osp/Modules/modulefiles
LOADEDMODULES=pgi/10.8:openmpi/1.4.2:fftw/2.1.5:papi/4.0.0
LD_LIBRARY_PATH=/usr/common/usg/papi/4.0.0/lib:/usr/common/usg/fftw/2.1.5/lib:
                /usr/common/usg/openmpi/1.4.2/pgi/lib:
                /usr/common/usg/pgi/10.8/linux86-64/10.8/libso
LM_LICENSE_FILE=/usr/common/usg/pgi/10.8/../license.dat
PGI=/usr/common/usg/pgi/10.8
_LMFILES_=/usr/common/usg/Modules/modulefiles/pgi/10.8:
          /usr/common/usg/Modules/modulefiles/openmpi/1.4.2:
          /usr/common/usg/Modules/modulefiles/fftw/2.1.5:
          /usr/common/usg/Modules/modulefiles/papi/4.0.0
OMPI_DIR=/usr/common/usg/openmpi/1.4.2/pgi
GSCRATCH=/global/scratch/sd/roche
SCRATCH=/global/scratch/sd/roche
TMPDIR=/global/scratch/sd/roche
FFTW=-I/usr/common/usg/fftw/2.1.5/include -L/usr/common/usg/fftw/2.1.5/lib
FFTW_INC=/usr/common/usg/fftw/2.1.5/include
FFTW_LIBDIR=/usr/common/usg/fftw/2.1.5/lib
FFTW_ROOT=/usr/common/usg/fftw/2.1.5
INFOPATH=/usr/common/usg/fftw/2.1.5/info
PAPI_INCLUDE=/usr/common/usg/papi/4.0.0/include
PAPI_LIB=/usr/common/usg/papi/4.0.0/lib
PAPI_VERSION=4.0.0
```

Q4 Input Settings

The installation is executed during an interaction session on a target compute node. First one builds the CUDA enabled libraries for LAMMPS. Next, one must build the remainder of the LAMMPS library which will necessarily includes the target packages.

There are two interactions considered in Q4: embedded atomic potential, Lennart-Jones potential. The `user-cuda` and `gpu` packages were built. Thus, there are Q4 binaries that execute only with MPI processes (`mpi`), with a single MPI process per node plus GPU enhancement (`user-cuda`), and as many

MPI processes per node as there are processor cores plus GPU enhancement (gpu). These cases are distinguished in their input decks by in.*.mpi, in.*.user-cuda, and in.*.gpu respectively. It is noted that as of Q4 FY11, the embedded atomic potential was only enhanced to execute the mpi and user-cuda packages. The Lennart-Jones potential was enhanced for mpi and gpu capabilities.

in.eam.mpi

```
# bulk Cu lattice

variable timesteps equal 1000
# variable timesteps equal 100000
# variable timesteps equal 1000000

#variable myatoms equal 108000
#variable x equal 30

#variable myatoms equal 500000
#variable x equal 50

#variable myatoms equal 1000188
#variable x equal 63

variable myatoms equal 4000000
variable x equal 100

#variable myatoms equal 8001504
#variable x equal 126

print "Simulating ${myatoms} atoms for ${timesteps} time steps."

units  metal
atom_style atomic

lattice  fcc 3.615
region  box block 0 $x 0 $x 0 $x
create_box 1 box
create_atoms 1 box

pair_style eam
pair_coeff 1 1 Cu_u3.eam

velocity all create 1600.0 376847 loop geom

neighbor 1.0 bin
neigh_modify    every 1 delay 5 check yes

fix  1 all nve

timestep 0.005
thermo  50

run  ${timesteps}
```

in.eam.user-cuda

```
# bulk Cu lattice
```

```
package  cuda gpu/node 1
variable timesteps equal 1000
# variable timesteps equal 100000
# variable timesteps equal 1000000

#variable myatoms equal 108000
#variable x equal 30

#variable myatoms equal 500000
#variable x equal 50

#variable myatoms equal 1000188
#variable x equal 63

variable myatoms equal 4000000
variable x equal 100

#variable myatoms equal 8001504
#variable x equal 126

print "Simulating ${myatoms} atoms for ${timesteps} time steps."

units  metal
atom_style atomic

lattice  fcc 3.615
region  box block 0 $x 0 $x 0 $x
create_box 1 box
create_atoms 1 box

pair_style eam/cuda
pair_coeff 1 1 Cu_u3.eam

velocity all create 1600.0 376847 loop geom

neighbor 1.0 bin
neigh_modify    every 1 delay 5 check yes

fix  1 all nve

timestep 0.005
thermo  50

run  ${timesteps}
```

in.lj.mpi

```
# 3d Lennard-Jones melt

variable timesteps equal 1000
# variable timesteps equal 100000
# variable timesteps equal 1000000

#variable myatoms equal 108000
#variable x equal 30

#variable myatoms equal 500000
```

```
#variable x equal 50

#variable myatoms equal 1000188
#variable x equal 63

# variable myatoms equal 4000000
# variable x equal 100

variable myatoms equal 8001504
variable x equal 126

print "Simulating ${myatoms} atoms for ${timesteps} time steps."

units  lj
atom_style atomic

lattice  fcc 0.8442
region  box block 0 $x 0 $x 0 $x
create_box 1 box
create_atoms 1 box
mass  1 1.0

velocity all create 1.44 87287 loop geom

pair_style lj/cut 2.5
pair_coeff 1 1 1.0 1.0 2.5

neighbor 0.3 bin
neigh_modify delay 0 every 20 check no

fix  1 all nve

run  ${timesteps}
```

in.lj.gpu

```
# 3d Lennard-Jones melt

#package gpu force/neigh 0 0 1
#package gpu force/neigh 0 0 -1
package gpu force/neigh 0 0 .95
newton  off
variable timesteps equal 1000
# variable timesteps equal 100000
# variable timesteps equal 1000000

#variable myatoms equal 108000
#variable x equal 30

#variable myatoms equal 500000
#variable x equal 50

#variable myatoms equal 1000188
#variable x equal 63

#variable myatoms equal 4000000
#variable x equal 100
```

```
variable myatoms equal 8001504
variable x equal 126

print "Simulating ${myatoms} atoms for ${timesteps} time steps."

units  lj
atom_style atomic

lattice  fcc 0.8442
region  box block 0 $x 0 $x 0 $x
create_box 1 box
create_atoms 1 box
mass  1 1.0

velocity all create 1.44 87287 loop geom

#pair_style lj/cut 2.5
pair_style lj/cut/gpu 2.5
pair_coeff 1 1 1.0 1.0 2.5

neighbor 0.3 bin
neigh_modify delay 0 every 20 check no

fix  1 all nve

run  ${timesteps}
```

Q4 Compilation Script

GCC

First one must request an interactive session so that the compute host + GPU nodes can be targeted. The environment is the same as described for the Q4 runtime. To obtain an interactive session one simply executes the following: `qsub -I -V -q dirac_int -l nodes=1:ppn=8:fermi`. The LAMMPS packages are found in `$LAMMPS_HOME/lib`. To build LAMMPS one works from `$LAMMPS_HOME/src`.

USER-CUDA Package

Makefile.common

```
#Common commandline argument interpreter for compilation with lammpscuda (USER-CUDA) installed

# make options:
# emu=1        switch to cuda emulation mode (otherwise: use gpu)
# dbg=1        print a lot of debugging output during runtime
# verbose=1    output nvcc command line during compilation
# keep=1       do not delete temporary compilation files (.ii, .cubin, ...)
# cufft=1      use cuda's fast fourier transformation lib "cufft" where possible (otherwise: use cpu fftw)
# binning=1    create virtual particle grid (neighbor-lists otherwise); currently this is not supported
# precision=1  single precision (global setting)
# precision=2  double precision (global setting)

SHELL = /bin/sh
```

```
# System-specific settings
# kr -q4
CUDA_INSTALL_PATH = /usr/common/usg/cuda/3.2
# e.g. in Gentoo
# CUDA_INSTALL_PATH = /opt/cuda


#////////////////////////////////////////////////////////////////////////////////////////
# no need to change anything below this line
#////////////////////////////////////////////////////////////////////////////////////////

#use CPU FFT if cufft=0 is requested.
FALLBACK_FFT = 1

#default settings for compiler switches
ifdef COMPILELIB
include Makefile.defaults
else
include ../../lib/cuda/Makefile.defaults
endif

shell echo "Compiling with precision = " ${precision} ", arch = " ${arch} ", cufft = " ${cufft} ",\
 dbg = " ${dbg} ", prec_timer = " ${prec_timer}

CUDA_FLAGS := -I${CUDA_INSTALL_PATH}/include -DUNIX
CUDA_USRLIB_CONDITIONAL := -L${CUDA_INSTALL_PATH}/lib -L${CUDA_INSTALL_PATH}/lib64

# debug setting
ifeq ($(strip $(dbg)), 1)
CUDA_FLAGS += -D_DEBUG -g
NVCC_FLAGS += -g -G
else
NVCC_FLAGS += --compiler-options -fno-strict-aliasing -O2
endif

# skip timing on Mac and Windows manually
ifeq ($(strip $(prec_timer)), 0)
CUDA_FLAGS += -DNO_PREC_TIMING
endif

# set fft routine
ifeq ($(strip $(cufft)), 0)
ifneq ($(FALLBACK_FFT), 1)
    FFT_INC = -DFFT_NONE
    FFT_PATH =
    FFT_LIB =
CUDA_FLAGS += -DFFT_NONE
endif
else
CUDA_FLAGS += -DFFT_CUFFT
CUDA_USRLIB_CONDITIONAL += -lcufft
endif

# make global precision setting

ifeq ($(strip $(precision)), 1)
CUDA_FLAGS += -DCUDA_PRECISION=1
```

```
else
ifeq ($(strip $(precision)), 3)
CUDA_FLAGS += -DCUDA_PRECISION=1 -DX_PRECISION=2
else
ifeq ($(strip $(precision)), 4)
CUDA_FLAGS += -DCUDA_PRECISION=1 -DX_PRECISION=2 -DV_PRECISION=2
else
CUDA_FLAGS += -DCUDA_PRECISION=2
endif
endif
endif

# make architecture settings
ifeq ($(strip $(arch)), 13)
CUDA_FLAGS += -DCUDA_ARCH=13
SMVERSIONFLAGS := -arch sm_13
else
  ifeq ($(strip $(arch)), 20)
 CUDA_FLAGS += -DCUDA_ARCH=20
 #NVCC_FLAGS += -ftz=false -prec-div=true -prec-sqrt=true
 NVCC_FLAGS += -ftz=true -prec-div=false -prec-sqrt=false
 SMVERSIONFLAGS := -arch sm_20
  else
     ifeq ($(strip $(arch)), 21)
   CUDA_FLAGS += -DCUDA_ARCH=20
   #NVCC_FLAGS += -ftz=false -prec-div=true -prec-sqrt=true
   NVCC_FLAGS += -ftz=true -prec-div=false -prec-sqrt=false
   SMVERSIONFLAGS := -arch sm_21
     else
       CUDA_FLAGS += -DCUDA_ARCH=99
       SMVERSIONFLAGS := -arch sm_13
     endif
  endif
endif



CCFLAGS := $(CCFLAGS) $(CUDA_FLAGS) \
-I$(CUDA_INSTALL_PATH)/include
```

Makefile.lammps

```
# Settings that the LAMMPS build will import when this package library is used
CUDA_FLAGS := -I/usr/common/usg/cuda/3.2/include -DUNIX -DFFT_CUFFT \
 -DCUDA_PRECISION=1 -DCUDA_ARCH=20
CUDA_USRLIB_CONDITIONAL := -L/usr/common/usg/cuda/3.2/lib \
-L/usr/common/usg/cuda/3.2/lib64 -lcufft

user-cuda_SYSINC  = ${CUDA_FLAGS}
user-cuda_SYSLIB  = -lcuda -lcudart -lrt
user-cuda_SYSPATH = ${CUDA_USRLIB_CONDITIONAL}
```

And to build the packaged library:

```
/usr/common/usg/cuda/3.2/bin/nvcc -lib atom_vec_cuda_cu.o binning_cu.o comm_cuda_cu.o \
compute_temp_cuda_cu.o compute_temp_partial_cuda_cu.o cuda_data_cu.o cuda_cu.o cuda_pair_cu.o \
```

```
cuda_wrapper_cu.o domain_cu.o fft3d_cuda_cu.o fix_addforce_cuda_cu.o fix_aveforce_cuda_cu.o \
fix_enforce2d_cuda_cu.o fix_freeze_cuda_cu.o fix_gravity_cuda_cu.o fix_nh_cuda_cu.o \
fix_nve_cuda_cu.o fix_set_force_cuda_cu.o fix_shake_cuda_cu.o fix_temp_berendsen_cuda_cu.o \
fix_temp_rescale_cuda_cu.o fix_temp_rescale_limit_cuda_cu.o fix_viscous_cuda_cu.o neighbor_cu.o \
pppm_cuda_cu.o -L/usr/common/usg/cuda/3.2/lib -L/usr/common/usg/cuda/3.2/lib64 \
-lcufft -o liblammpscuda.a
```

After this step, the appropriate paths have to be defined for the target architecture and this new package.

Makefile.package

```
PKG_INC = -I../../lib/cuda -DLMP_USER_CUDA -DUNIX -DFFT_CUFFT -I/global/scratch/sd/roche/lammps-17Sep11/lib/cuda
PKG_PATH =  -L../../lib/cuda
PKG_LIB =   -llammpscuda -lrt

PKG_SYSINC =  $(user-cuda_SYSINC)
PKG_SYSLIB =  $(user-cuda_SYSLIB)
PKG_SYSPATH = $(user-cuda_SYSPATH)
```

Makefile.package.settings

```
include ../../lib/cuda/Makefile.lammps
\scriptsize{\begin{verbatim}
```

MAKE/Makefile.linux

```
# linux = RedHat Linux box, Intel icc, MPICH2, FFTW

SHELL = /bin/sh

# ----------------------------------------------------------------------
# compiler/linker settings
# specify flags and libraries needed for your compiler
#kr
CC = g++
CCFLAGS = -O2
DEPFLAGS = -M
LINK = g++
LINKFLAGS = -O2
LIB =
ARCHIVE = ar
ARFLAGS = -rc
SIZE = size

# ----------------------------------------------------------------------
# LAMMPS-specific settings
# specify settings for LAMMPS features you will use
# if you change any -D setting, do full re-compile after "make clean"

# LAMMPS ifdef settings, OPTIONAL
# see possible settings in doc/Section_start.html#2_2 (step 4)

LMP_INC = -DLAMMPS_GZIP
```

```
# MPI library, REQUIRED
# see discussion in doc/Section_start.html#2_2 (step 5)
# can point to dummy MPI library in src/STUBS as in Makefile.serial
# INC = path for mpi.h, MPI compiler settings
# PATH = path for MPI library
# LIB = name of MPI library

#kr
#OMPI_DIR=/usr/common/usg/openmpi/1.4.2/gcc
MPI_INC =       -I/usr/common/usg/openmpi/1.4.2/gcc/include
MPI_LIB = -L/usr/common/usg/openmpi/1.4.2/gcc/lib -lmpi_cxx -lmpi

# FFT library, OPTIONAL
# see discussion in doc/Section_start.html#2_2 (step 6)
# can be left blank to use provided KISS FFT library
# INC = -DFFT setting, e.g. -DFFT_FFTW, FFT compiler settings
# PATH = path for FFT library
# LIB = name of FFT library
#kr
#FFTW=-I/usr/common/usg/fftw/2.1.5-gnu/include -L/usr/common/usg/fftw/2.1.5-gnu/lib
#FFTW_INC=/usr/common/usg/fftw/2.1.5-gnu/include
#FFTW_LIBDIR=/usr/common/usg/fftw/2.1.5-gnu/lib
#FFTW_ROOT=/usr/common/usg/fftw/2.1.5-gnu
FFT_INC =       -DFFT_FFTW -I/usr/common/usg/fftw/2.1.5-gnu/include
FFT_LIB = -L/usr/common/usg/fftw/2.1.5-gnu/lib -ldfftw -lsfftw

# JPEG library, OPTIONAL
# see discussion in doc/Section_start.html#2_2 (step 7)
# only needed if -DLAMMPS_JPEG listed with LMP_INC
# INC = path for jpeglib.h
# PATH = path for JPEG library
# LIB = name of JPEG library

JPG_INC =
JPG_PATH =
JPG_LIB =

# ----------------------------------------------------------------------
# build rules and dependencies
# no need to edit this section

#kr
#/global/scratch/sd/roche/lammps-17Sep11/src
OTHER_INC = -I/global/scratch/sd/roche/lammps-cuda32-sm20-spe/lib/cuda
include Makefile.package.settings
include Makefile.package

EXTRA_INC = $(LMP_INC) $(PKG_INC) $(MPI_INC) $(FFT_INC) $(JPG_INC) $(PKG_SYSINC)
EXTRA_PATH = $(PKG_PATH) $(MPI_PATH) $(FFT_PATH) $(JPG_PATH) $(PKG_SYSPATH)
EXTRA_LIB = $(PKG_LIB) $(MPI_LIB) $(FFT_LIB) $(JPG_LIB) $(PKG_SYSLIB)

# Link target

$(EXE): $(OBJ)
$(LINK) $(LINKFLAGS) $(EXTRA_PATH) $(OBJ) $(EXTRA_LIB) $(LIB) -o $(EXE)
$(SIZE) $(EXE)
```

```
# Library target

lib: $(OBJ)
$(ARCHIVE) $(ARFLAGS) $(EXE) $(OBJ)

# Compilation rules

%.o:%.cpp
$(CC) $(CCFLAGS) $(EXTRA_INC) -c $<

%.d:%.cpp
$(CC) $(CCFLAGS) $(EXTRA_INC) $(DEPFLAGS) $< > $@

# Individual dependencies

DEPENDS = $(OBJ:.o=.d)
include $(DEPENDS)
```

And, finally the binary is built:

```
g++ -o ../lmp_gcc_cu32_sm20 angle.o angle_charmm.o angle_cosine.o angle_cosine_delta.o \
angle_cosine_periodic.o angle_cosine_squared.o angle_harmonic.o angle_hybrid.o \
angle_table.o atom.o atom_vec.o atom_vec_angle.o atom_vec_angle_cuda.o atom_vec_atomic.o \
 atom_vec_atomic_cuda.o atom_vec_bond.o atom_vec_charge.o atom_vec_charge_cuda.o \
atom_vec_ellipsoid.o atom_vec_full.o atom_vec_full_cuda.o atom_vec_hybrid.o \
atom_vec_molecular.o atom_vec_sphere.o bond.o bond_fene.o bond_fene_expand.o \
bond_harmonic.o bond_hybrid.o bond_morse.o bond_nonlinear.o bond_quartic.o bond_table.o \
change_box.o comm.o comm_cuda.o compute.o compute_angle_local.o compute_atom_molecule.o \
compute_bond_local.o compute_centro_atom.o compute_cluster_atom.o compute_cna_atom.o \
compute_com.o compute_com_molecule.o compute_coord_atom.o compute_dihedral_local.o \
compute_displace_atom.o compute_erotate_sphere.o compute_group_group.o compute_gyration.o \
 compute_gyration_molecule.o compute_heat_flux.o compute_improper_local.o compute_ke.o \
compute_ke_atom.o compute_msd.o compute_msd_molecule.o compute_pair.o compute_pair_local.o \
 compute_pe.o compute_pe_atom.o compute_pe_cuda.o compute_pressure.o compute_pressure_cuda.o \
 compute_property_atom.o compute_property_local.o compute_property_molecule.o compute_rdf.o \
compute_reduce.o compute_reduce_region.o compute_slice.o compute_stress_atom.o compute_temp.o \
 compute_temp_com.o compute_temp_cuda.o compute_temp_deform.o compute_temp_partial.o \
 compute_temp_partial_cuda.o compute_temp_profile.o compute_temp_ramp.o compute_temp_region.o \
compute_temp_sphere.o compute_ti.o create_atoms.o create_box.o cuda.o cuda_neigh_list.o \
delete_atoms.o delete_bonds.o dihedral.o dihedral_charmm.o dihedral_harmonic.o \
dihedral_helix.o dihedral_hybrid.o dihedral_multi_harmonic.o dihedral_opls.o displace_atoms.o \
displace_box.o domain.o domain_cuda.o dump.o dump_atom.o dump_cfg.o dump_custom.o dump_dcd.o \
dump_image.o dump_local.o dump_xyz.o error.o ewald.o fft3d.o fft3d_cuda.o fft3d_wrap.o \
fft3d_wrap_cuda.o finish.o fix.o fix_adapt.o fix_addforce.o fix_addforce_cuda.o fix_ave_atom.o \
fix_ave_correlate.o fix_ave_histo.o fix_ave_spatial.o fix_ave_time.o fix_aveforce.o \
fix_aveforce_cuda.o fix_box_relax.o fix_deform.o fix_deposit.o fix_drag.o fix_dt_reset.o \
fix_efield.o fix_enforce2d.o fix_enforce2d_cuda.o fix_evaporate.o fix_external.o fix_gravity.o \
fix_gravity_cuda.o fix_heat.o fix_indent.o fix_langevin.o fix_lineforce.o fix_minimize.o \
fix_momentum.o fix_move.o fix_nh.o fix_nh_cuda.o fix_nh_sphere.o fix_nph.o fix_nph_sphere.o \
fix_npt.o fix_npt_cuda.o fix_npt_sphere.o fix_nve.o fix_nve_cuda.o fix_nve_limit.o \
fix_nve_noforce.o fix_nve_sphere.o fix_nvt.o fix_nvt_cuda.o fix_nvt_sllod.o fix_nvt_sphere.o \
fix_orient_fcc.o fix_planeforce.o fix_press_berendsen.o fix_print.o fix_qeq_comb.o \
fix_read_restart.o fix_recenter.o fix_respa.o fix_restrain.o fix_rigid.o fix_rigid_nve.o \
fix_rigid_nvt.o fix_set_force_cuda.o fix_setforce.o fix_shake.o fix_shake_cuda.o \
fix_shear_history.o fix_spring.o fix_spring_rg.o fix_spring_self.o fix_store_force.o \
fix_store_state.o fix_temp_berendsen.o fix_temp_berendsen_cuda.o fix_temp_rescale.o \
```

```
fix_temp_rescale_cuda.o fix_temp_rescale_limit_cuda.o fix_thermal_conductivity.o fix_tmd.o \
fix_ttm.o fix_viscosity.o fix_viscous.o fix_viscous_cuda.o fix_wall.o fix_wall_harmonic.o \
fix_wall_lj126.o fix_wall_lj93.o fix_wall_reflect.o fix_wall_region.o force.o group.o \
improper.o improper_cvff.o improper_harmonic.o improper_hybrid.o improper_umbrella.o \
input.o integrate.o irregular.o kspace.o lammps.o lattice.o library.o main.o math_extra.o \
memory.o min.o min_cg.o min_fire.o min_hftn.o min_linesearch.o min_quickmin.o min_sd.o \
minimize.o modify.o modify_cuda.o neigh_bond.o neigh_derive.o neigh_full.o neigh_full_cuda.o \
neigh_gran.o neigh_half_bin.o neigh_half_multi.o neigh_half_nsq.o neigh_list.o neigh_request.o \
neigh_respa.o neigh_stencil.o neighbor.o neighbor_cuda.o output.o pair.o pair_adp.o \
pair_airebo.o pair_born.o pair_born_coul_long.o pair_born_coul_long_cuda.o pair_buck.o \
pair_buck_coul_cut.o pair_buck_coul_cut_cuda.o pair_buck_coul_long.o pair_buck_coul_long_cuda.o \
pair_buck_cuda.o pair_comb.o pair_coul_cut.o pair_coul_debye.o pair_coul_long.o pair_dpd.o \
pair_dpd_tstat.o pair_eam.o pair_eam_alloy.o pair_eam_alloy_cuda.o pair_eam_cuda.o \
pair_eam_fs.o pair_eam_fs_cuda.o pair_eim.o pair_gauss.o pair_hbond_dreiding_lj.o \
pair_hbond_dreiding_morse.o pair_hybrid.o pair_hybrid_overlay.o pair_lj96_cut.o \
pair_lj96_cut_cuda.o pair_lj_charmm_coul_charmm.o pair_lj_charmm_coul_charmm_cuda.o \
pair_lj_charmm_coul_charmm_implicit.o pair_lj_charmm_coul_charmm_implicit_cuda.o \
pair_lj_charmm_coul_long.o pair_lj_charmm_coul_long_cuda.o pair_lj_cubic.o pair_lj_cut.o \
pair_lj_cut_coul_cut.o pair_lj_cut_coul_cut_cuda.o pair_lj_cut_coul_debye.o \
pair_lj_cut_coul_debye_cuda.o pair_lj_cut_coul_long.o pair_lj_cut_coul_long_cuda.o \
pair_lj_cut_coul_long_tip4p.o pair_lj_cut_cuda.o pair_lj_cut_experimental_cuda.o \
pair_lj_expand.o pair_lj_expand_cuda.o pair_lj_gromacs.o pair_lj_gromacs_coul_gromacs.o \
pair_lj_gromacs_coul_gromacs_cuda.o pair_lj_gromacs_cuda.o pair_lj_smooth.o \
pair_lj_smooth_cuda.o pair_morse.o pair_morse_cuda.o pair_rebo.o pair_soft.o pair_sw.o \
pair_table.o pair_tersoff.o pair_tersoff_zbl.o pair_yukawa.o pppm.o pppm_cg.o pppm_cuda.o \
pppm_tip4p.o random_mars.o random_park.o read_data.o read_restart.o region.o region_block.o \
region_cone.o region_cylinder.o region_intersect.o region_plane.o region_prism.o \
region_sphere.o region_union.o remap.o remap_wrap.o replicate.o respa.o run.o set.o special.o \
thermo.o timer.o universe.o update.o variable.o velocity.o verlet.o verlet_cuda.o \
write_restart.o -L/usr/common/usg/fftw/2.1.5-gnu/lib -ldfftw -lsfftw \
-L/global/scratch/sd/roche/lammps-cuda32-sm20-spe/lib/cuda -llammpscuda \
-L/usr/common/usg/cuda/3.2/lib64 -L/usr/common/usg/cuda/3.2/lib -lcudart -lcufft \
-L/usr/common/usg/openmpi/1.4.2/gcc/lib -lmpi_cxx -lmpi -lrt
```

Basically, the same sequence of events must take place for the GPU Package build.

GPU Package

```
[roche@cvrsvc04 gpu]$ cat Makefile.linux
# /* ----------------------------------------------------------------------
#    LAMMPS - Large-scale Atomic/Molecular Massively Parallel Simulator
#    http://lammps.sandia.gov, Sandia National Laboratories
#    Steve Plimpton, sjplimp@sandia.gov
#
#    Copyright (2003) Sandia Corporation.  Under the terms of Contract
#    DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
#    certain rights in this software.  This software is distributed under
#    the GNU General Public License.
#
#    See the README file in the top-level LAMMPS directory.
# ------------------------------------------------------------------------- */
#
# /* ----------------------------------------------------------------------
#    Contributing authors: Mike Brown (ORNL), brownw@ornl.gov
#                          Peng Wang (Nvidia), penwang@nvidia.com
#                          Paul Crozier (SNL), pscrozi@sandia.gov
```

```
# ------------------------------------------------------------------------ */

#kr
#CUDA_INCLUDE=-I/usr/common/usg/cuda/3.2/include
#CUDA_LIB=-L/usr/common/usg/cuda/3.2/lib
#CUDA_LIB64=-L/usr/common/usg/cuda/3.2/lib64
#NVIDIA_DRIVER_LIB=-L/usr/common/usg/nvidia-driver-util/3.2/lib64


CUDA_HOME = /usr/common/usg/cuda/3.2
NVCC = nvcc

# newer CUDA
CUDA_ARCH = -arch=sm_13
# older CUDA
#CUDA_ARCH = -arch=sm_10 -DCUDA_PRE_THREE

CUDA_PRECISION = -D_SINGLE_SINGLE
CUDA_INCLUDE = -I$(CUDA_HOME)/include
CUDA_LIB = -L$(CUDA_HOME)/lib64
CUDA_OPTS = -DUNIX -O3 -Xptxas -v --use_fast_math

CUDR_CPP = mpic++ -DMPI_GERYON -DUCL_NO_EXIT -DMPICH_IGNORE_CXX_SEEK
CUDR_OPTS = -O2 # -xHost -no-prec-div -ansi-alias

BIN_DIR = ./
OBJ_DIR = ./
LIB_DIR = ./
AR = ar
BSH = /bin/sh

include Nvidia.makefile

[roche@cvrsvc04 gpu]$ cat Nvidia.makefile
# /* ----------------------------------------------------------------------
#    LAMMPS - Large-scale Atomic/Molecular Massively Parallel Simulator
#    http://lammps.sandia.gov, Sandia National Laboratories
#    Steve Plimpton, sjplimp@sandia.gov
#
#    Copyright (2003) Sandia Corporation.  Under the terms of Contract
#    DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
#    certain rights in this software.  This software is distributed under
#    the GNU General Public License.
#
#    See the README file in the top-level LAMMPS directory.
# ------------------------------------------------------------------------ */
#
# /* ----------------------------------------------------------------------
#    Contributing authors: Mike Brown (ORNL), brownw@ornl.gov
#                          Peng Wang (Nvidia), penwang@nvidia.com
#                          Inderaj Bains (NVIDIA), ibains@nvidia.com
#                          Paul Crozier (SNL), pscrozi@sandia.gov
# ------------------------------------------------------------------------ */

CUDA  = $(NVCC) $(CUDA_INCLUDE) $(CUDA_OPTS) -Icudpp_mini $(CUDA_ARCH) \
            $(CUDA_PRECISION)
CUDR  = $(CUDR_CPP) $(CUDR_OPTS) $(CUDA_PRECISION) $(CUDA_INCLUDE) \
```

```
        -Icudpp_mini
CUDA_LINK = $(CUDA_LIB) -lcudart

GPU_LIB = $(LIB_DIR)/libgpu.a

# Headers for Geryon
UCL_H  = $(wildcard ./geryon/ucl*.h)
NVC_H  = $(wildcard ./geryon/nvc*.h) $(UCL_H)
NVD_H  = $(wildcard ./geryon/nvd*.h) $(UCL_H) nv_kernel_def.h
# Headers for Pair Stuff
PAIR_H  = pair_gpu_atom.h pair_gpu_ans.h pair_gpu_nbor_shared.h \
          pair_gpu_nbor.h pair_gpu_precision.h pair_gpu_device.h \
          pair_gpu_balance.h pppm_gpu_memory.h

ALL_H = $(NVD_H) $(PAIR_H)

EXECS = $(BIN_DIR)/nvc_get_devices
CUDPP = $(OBJ_DIR)/cudpp.o $(OBJ_DIR)/cudpp_plan.o \
        $(OBJ_DIR)/cudpp_maximal_launch.o $(OBJ_DIR)/cudpp_plan_manager.o \
        $(OBJ_DIR)/radixsort_app.cu_o $(OBJ_DIR)/scan_app.cu_o
OBJS = $(OBJ_DIR)/pair_gpu_atom.o $(OBJ_DIR)/pair_gpu_ans.o \
       $(OBJ_DIR)/pair_gpu_nbor.o $(OBJ_DIR)/pair_gpu_nbor_shared.o \
       $(OBJ_DIR)/pair_gpu_device.o $(OBJ_DIR)/atomic_gpu_memory.o \
       $(OBJ_DIR)/charge_gpu_memory.o $(OBJ_DIR)/base_ellipsoid.o \
       $(OBJ_DIR)/pppm_gpu_memory.o $(OBJ_DIR)/pppm_l_gpu.o \
       $(OBJ_DIR)/gayberne.o $(OBJ_DIR)/gayberne_ext.o \
       $(OBJ_DIR)/re_squared.o $(OBJ_DIR)/re_squared_ext.o \
       $(OBJ_DIR)/lj_cut_gpu_memory.o $(OBJ_DIR)/lj_cut_gpu.o \
       $(OBJ_DIR)/lj96_cut_gpu_memory.o $(OBJ_DIR)/lj96_cut_gpu.o \
       $(OBJ_DIR)/lj_expand_gpu_memory.o $(OBJ_DIR)/lj_expand_gpu.o \
       $(OBJ_DIR)/ljc_cut_gpu_memory.o $(OBJ_DIR)/ljc_cut_gpu.o \
       $(OBJ_DIR)/ljcl_cut_gpu_memory.o $(OBJ_DIR)/ljcl_cut_gpu.o \
       $(OBJ_DIR)/lj_class2_long.o $(OBJ_DIR)/lj_class2_long_ext.o \
       $(OBJ_DIR)/coul_long_gpu_memory.o $(OBJ_DIR)/coul_long_gpu.o \
       $(OBJ_DIR)/morse_gpu_memory.o $(OBJ_DIR)/morse_gpu.o \
       $(OBJ_DIR)/crml_gpu_memory.o $(OBJ_DIR)/crml_gpu.o \
       $(OBJ_DIR)/cmm_cut_gpu_memory.o $(OBJ_DIR)/cmm_cut_gpu.o \
       $(OBJ_DIR)/cmmc_long_gpu_memory.o $(OBJ_DIR)/cmmc_long_gpu.o \
       $(OBJ_DIR)/cmmc_msm_gpu_memory.o $(OBJ_DIR)/cmmc_msm_gpu.o \
       $(CUDPP)
PTXS = $(OBJ_DIR)/pair_gpu_dev_kernel.ptx \
       $(OBJ_DIR)/pair_gpu_atom_kernel.ptx $(OBJ_DIR)/pair_gpu_atom_ptx.h \
       $(OBJ_DIR)/pair_gpu_nbor_kernel.ptx $(OBJ_DIR)/pair_gpu_nbor_ptx.h \
       $(OBJ_DIR)/pair_gpu_build_kernel.ptx $(OBJ_DIR)/pair_gpu_build_ptx.h \
       $(OBJ_DIR)/pppm_f_gpu_kernel.ptx $(OBJ_DIR)/pppm_f_gpu_ptx.h \
       $(OBJ_DIR)/pppm_d_gpu_kernel.ptx $(OBJ_DIR)/pppm_d_gpu_ptx.h \
       $(OBJ_DIR)/ellipsoid_nbor.ptx $(OBJ_DIR)/ellipsoid_nbor_ptx.h \
       $(OBJ_DIR)/gayberne.ptx $(OBJ_DIR)/gayberne_lj.ptx \
       $(OBJ_DIR)/gayberne_ptx.h $(OBJ_DIR)/re_squared.ptx \
       $(OBJ_DIR)/re_squared_lj.ptx $(OBJ_DIR)/re_squared_ptx.h \
       $(OBJ_DIR)/lj_cut_gpu_kernel.ptx $(OBJ_DIR)/lj_cut_gpu_ptx.h \
       $(OBJ_DIR)/lj96_cut_gpu_kernel.ptx $(OBJ_DIR)/lj96_cut_gpu_ptx.h \
       $(OBJ_DIR)/lj_expand_gpu_kernel.ptx $(OBJ_DIR)/lj_expand_gpu_ptx.h \
       $(OBJ_DIR)/ljc_cut_gpu_kernel.ptx $(OBJ_DIR)/ljc_cut_gpu_ptx.h \
       $(OBJ_DIR)/ljcl_cut_gpu_kernel.ptx $(OBJ_DIR)/ljcl_cut_gpu_ptx.h \
       $(OBJ_DIR)/lj_class2_long.ptx $(OBJ_DIR)/lj_class2_long_ptx.h \
       $(OBJ_DIR)/coul_long_gpu_kernel.ptx $(OBJ_DIR)/coul_long_gpu_ptx.h \
```

```
        $(OBJ_DIR)/morse_gpu_kernel.ptx $(OBJ_DIR)/morse_gpu_ptx.h \
        $(OBJ_DIR)/crml_gpu_kernel.ptx $(OBJ_DIR)/crml_gpu_ptx.h \
        $(OBJ_DIR)/cmm_cut_gpu_kernel.ptx $(OBJ_DIR)/cmm_cut_gpu_ptx.h \
        $(OBJ_DIR)/cmmc_long_gpu_kernel.ptx $(OBJ_DIR)/cmmc_long_gpu_ptx.h \
        $(OBJ_DIR)/cmmc_msm_gpu_kernel.ptx $(OBJ_DIR)/cmmc_msm_gpu_ptx.h

all: $(GPU_LIB) $(EXECS)

$(OBJ_DIR)/cudpp.o: cudpp_mini/cudpp.cpp
$(CUDR) -o $@ -c cudpp_mini/cudpp.cpp -Icudpp_mini

$(OBJ_DIR)/cudpp_plan.o: cudpp_mini/cudpp_plan.cpp
$(CUDR) -o $@ -c cudpp_mini/cudpp_plan.cpp -Icudpp_mini

$(OBJ_DIR)/cudpp_maximal_launch.o: cudpp_mini/cudpp_maximal_launch.cpp
$(CUDR) -o $@ -c cudpp_mini/cudpp_maximal_launch.cpp -Icudpp_mini

$(OBJ_DIR)/cudpp_plan_manager.o: cudpp_mini/cudpp_plan_manager.cpp
$(CUDR) -o $@ -c cudpp_mini/cudpp_plan_manager.cpp -Icudpp_mini

$(OBJ_DIR)/radixsort_app.cu_o: cudpp_mini/radixsort_app.cu
$(CUDA) -o $@ -c cudpp_mini/radixsort_app.cu

$(OBJ_DIR)/scan_app.cu_o: cudpp_mini/scan_app.cu
$(CUDA) -o $@ -c cudpp_mini/scan_app.cu

$(OBJ_DIR)/pair_gpu_atom_kernel.ptx: pair_gpu_atom_kernel.cu
$(CUDA) --ptx -DNV_KERNEL -o $@ pair_gpu_atom_kernel.cu

$(OBJ_DIR)/pair_gpu_atom_ptx.h: $(OBJ_DIR)/pair_gpu_atom_kernel.ptx
$(BSH) ./geryon/file_to_cstr.sh $(OBJ_DIR)/pair_gpu_atom_kernel.ptx $(OBJ_DIR)/pair_gpu_atom_ptx.h

$(OBJ_DIR)/pair_gpu_atom.o: pair_gpu_atom.cpp pair_gpu_atom.h $(NVD_H) $(OBJ_DIR)/pair_gpu_atom_ptx.h
$(CUDR) -o $@ -c pair_gpu_atom.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/pair_gpu_ans.o: pair_gpu_ans.cpp pair_gpu_ans.h $(NVD_H)
$(CUDR) -o $@ -c pair_gpu_ans.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/pair_gpu_nbor_kernel.ptx: pair_gpu_nbor_kernel.cu
$(CUDA) --ptx -DNV_KERNEL -o $@ pair_gpu_nbor_kernel.cu

$(OBJ_DIR)/pair_gpu_nbor_ptx.h: $(OBJ_DIR)/pair_gpu_nbor_kernel.ptx
$(BSH) ./geryon/file_to_cstr.sh $(OBJ_DIR)/pair_gpu_nbor_kernel.ptx $(OBJ_DIR)/pair_gpu_nbor_ptx.h

$(OBJ_DIR)/pair_gpu_build_kernel.ptx: pair_gpu_build_kernel.cu
$(CUDA) --ptx -DNV_KERNEL -o $@ pair_gpu_build_kernel.cu

$(OBJ_DIR)/pair_gpu_build_ptx.h: $(OBJ_DIR)/pair_gpu_build_kernel.ptx
$(BSH) ./geryon/file_to_cstr.sh $(OBJ_DIR)/pair_gpu_build_kernel.ptx $(OBJ_DIR)/pair_gpu_build_ptx.h

$(OBJ_DIR)/pair_gpu_nbor_shared.o: pair_gpu_nbor_shared.cpp pair_gpu_nbor_shared.h \
$(OBJ_DIR)/pair_gpu_nbor_ptx.h $(OBJ_DIR)/pair_gpu_build_ptx.h $(NVD_H)
$(CUDR) -o $@ -c pair_gpu_nbor_shared.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/pair_gpu_nbor.o: pair_gpu_nbor.cpp pair_gpu_nbor.h pair_gpu_nbor_shared.h $(NVD_H)
$(CUDR) -o $@ -c pair_gpu_nbor.cpp -I$(OBJ_DIR)
```

```
$(OBJ_DIR)/pair_gpu_dev_kernel.ptx: pair_gpu_dev_kernel.cu
$(CUDA) --ptx -DNV_KERNEL -o $@ pair_gpu_dev_kernel.cu

$(OBJ_DIR)/pair_gpu_dev_ptx.h: $(OBJ_DIR)/pair_gpu_dev_kernel.ptx
$(BSH) ./geryon/file_to_cstr.sh $(OBJ_DIR)/pair_gpu_dev_kernel.ptx $(OBJ_DIR)/pair_gpu_dev_ptx.h

$(OBJ_DIR)/pair_gpu_device.o: pair_gpu_device.cpp pair_gpu_device.h $(ALL_H) $(OBJ_DIR)/pair_gpu_dev_ptx.h
$(CUDR) -o $@ -c pair_gpu_device.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/atomic_gpu_memory.o: $(ALL_H) atomic_gpu_memory.h atomic_gpu_memory.cpp
$(CUDR) -o $@ -c atomic_gpu_memory.cpp

$(OBJ_DIR)/charge_gpu_memory.o: $(ALL_H) charge_gpu_memory.h charge_gpu_memory.cpp
$(CUDR) -o $@ -c charge_gpu_memory.cpp

$(OBJ_DIR)/base_ellipsoid.o: $(ALL_H) base_ellipsoid.h base_ellipsoid.cpp $(OBJ_DIR)/ellipsoid_nbor_ptx.h
$(CUDR) -o $@ -c base_ellipsoid.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/pppm_f_gpu_kernel.ptx: pppm_gpu_kernel.cu pair_gpu_precision.h
$(CUDA) --ptx -DNV_KERNEL -Dgrdtyp=float -Dgrdtyp4=float4 -o $@ pppm_gpu_kernel.cu

$(OBJ_DIR)/pppm_f_gpu_ptx.h: $(OBJ_DIR)/pppm_f_gpu_kernel.ptx
$(BSH) ./geryon/file_to_cstr.sh $(OBJ_DIR)/pppm_f_gpu_kernel.ptx $(OBJ_DIR)/pppm_f_gpu_ptx.h

$(OBJ_DIR)/pppm_d_gpu_kernel.ptx: pppm_gpu_kernel.cu pair_gpu_precision.h
$(CUDA) --ptx -DNV_KERNEL -Dgrdtyp=double -Dgrdtyp4=double4 -o $@ pppm_gpu_kernel.cu

$(OBJ_DIR)/pppm_d_gpu_ptx.h: $(OBJ_DIR)/pppm_d_gpu_kernel.ptx
$(BSH) ./geryon/file_to_cstr.sh $(OBJ_DIR)/pppm_d_gpu_kernel.ptx $(OBJ_DIR)/pppm_d_gpu_ptx.h

$(OBJ_DIR)/pppm_gpu_memory.o: $(ALL_H) pppm_gpu_memory.h pppm_gpu_memory.cpp \
$(OBJ_DIR)/pppm_f_gpu_ptx.h $(OBJ_DIR)/pppm_d_gpu_ptx.h
$(CUDR) -o $@ -c pppm_gpu_memory.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/pppm_l_gpu.o: $(ALL_H) pppm_gpu_memory.h pppm_l_gpu.cpp
$(CUDR) -o $@ -c pppm_l_gpu.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/ellipsoid_nbor.ptx: ellipsoid_nbor.cu pair_gpu_precision.h
$(CUDA) --ptx -DNV_KERNEL -o $@ ellipsoid_nbor.cu

$(OBJ_DIR)/ellipsoid_nbor_ptx.h: $(OBJ_DIR)/ellipsoid_nbor.ptx
$(BSH) ./geryon/file_to_cstr.sh $(OBJ_DIR)/ellipsoid_nbor.ptx $(OBJ_DIR)/ellipsoid_nbor_ptx.h

$(OBJ_DIR)/gayberne.ptx: gayberne.cu pair_gpu_precision.h ellipsoid_extra.h
$(CUDA) --ptx -DNV_KERNEL -o $@ gayberne.cu

$(OBJ_DIR)/gayberne_lj.ptx: gayberne_lj.cu pair_gpu_precision.h ellipsoid_extra.h
$(CUDA) --ptx -DNV_KERNEL -o $@ gayberne_lj.cu

$(OBJ_DIR)/gayberne_ptx.h: $(OBJ_DIR)/gayberne.ptx $(OBJ_DIR)/gayberne_lj.ptx
$(BSH) ./geryon/file_to_cstr.sh $(OBJ_DIR)/gayberne.ptx $(OBJ_DIR)/gayberne_lj.ptx \
$(OBJ_DIR)/gayberne_ptx.h

$(OBJ_DIR)/gayberne.o: $(ALL_H) gayberne.h gayberne.cpp $(OBJ_DIR)/gayberne_ptx.h $(OBJ_DIR)/base_ellipsoid.o
$(CUDR) -o $@ -c gayberne.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/gayberne_ext.o: $(ALL_H) $(OBJ_DIR)/gayberne.o gayberne_ext.cpp
```

```
$(CUDR) -o $@ -c gayberne_ext.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/re_squared.ptx: re_squared.cu pair_gpu_precision.h ellipsoid_extra.h
$(CUDA) --ptx -DNV_KERNEL -o $@ re_squared.cu

$(OBJ_DIR)/re_squared_lj.ptx: re_squared_lj.cu pair_gpu_precision.h ellipsoid_extra.h
$(CUDA) --ptx -DNV_KERNEL -o $@ re_squared_lj.cu

$(OBJ_DIR)/re_squared_ptx.h: $(OBJ_DIR)/re_squared.ptx $(OBJ_DIR)/re_squared_lj.ptx
$(BSH) ./geryon/file_to_cstr.sh $(OBJ_DIR)/re_squared.ptx $(OBJ_DIR)/re_squared_lj.ptx \
$(OBJ_DIR)/re_squared_ptx.h

$(OBJ_DIR)/re_squared.o: $(ALL_H) re_squared.h re_squared.cpp $(OBJ_DIR)/re_squared_ptx.h \
$(OBJ_DIR)/base_ellipsoid.o
$(CUDR) -o $@ -c re_squared.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/re_squared_ext.o: $(ALL_H) $(OBJ_DIR)/re_squared.o re_squared_ext.cpp
$(CUDR) -o $@ -c re_squared_ext.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/lj_cut_gpu_kernel.ptx: lj_cut_gpu_kernel.cu pair_gpu_precision.h
$(CUDA) --ptx -DNV_KERNEL -o $@ lj_cut_gpu_kernel.cu

$(OBJ_DIR)/lj_cut_gpu_ptx.h: $(OBJ_DIR)/lj_cut_gpu_kernel.ptx $(OBJ_DIR)/lj_cut_gpu_kernel.ptx
$(BSH) ./geryon/file_to_cstr.sh $(OBJ_DIR)/lj_cut_gpu_kernel.ptx $(OBJ_DIR)/lj_cut_gpu_ptx.h

$(OBJ_DIR)/lj_cut_gpu_memory.o: $(ALL_H) lj_cut_gpu_memory.h lj_cut_gpu_memory.cpp $(OBJ_DIR)/lj_cut_gpu_ptx.h \
$(OBJ_DIR)/atomic_gpu_memory.o
$(CUDR) -o $@ -c lj_cut_gpu_memory.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/lj_cut_gpu.o: $(ALL_H) lj_cut_gpu_memory.h lj_cut_gpu.cpp atomic_gpu_memory.h
$(CUDR) -o $@ -c lj_cut_gpu.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/ljc_cut_gpu_kernel.ptx: ljc_cut_gpu_kernel.cu pair_gpu_precision.h
$(CUDA) --ptx -DNV_KERNEL -o $@ ljc_cut_gpu_kernel.cu

$(OBJ_DIR)/ljc_cut_gpu_ptx.h: $(OBJ_DIR)/ljc_cut_gpu_kernel.ptx $(OBJ_DIR)/ljc_cut_gpu_kernel.ptx
$(BSH) ./geryon/file_to_cstr.sh $(OBJ_DIR)/ljc_cut_gpu_kernel.ptx $(OBJ_DIR)/ljc_cut_gpu_ptx.h

$(OBJ_DIR)/ljc_cut_gpu_memory.o: $(ALL_H) ljc_cut_gpu_memory.h ljc_cut_gpu_memory.cpp \
$(OBJ_DIR)/ljc_cut_gpu_ptx.h $(OBJ_DIR)/charge_gpu_memory.o
$(CUDR) -o $@ -c ljc_cut_gpu_memory.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/ljc_cut_gpu.o: $(ALL_H) ljc_cut_gpu_memory.h ljc_cut_gpu.cpp charge_gpu_memory.h
$(CUDR) -o $@ -c ljc_cut_gpu.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/lj_class2_long.ptx: lj_class2_long.cu pair_gpu_precision.h
$(CUDA) --ptx -DNV_KERNEL -o $@ lj_class2_long.cu

$(OBJ_DIR)/lj_class2_long_ptx.h: $(OBJ_DIR)/lj_class2_long.ptx $(OBJ_DIR)/lj_class2_long.ptx
$(BSH) ./geryon/file_to_cstr.sh $(OBJ_DIR)/lj_class2_long.ptx $(OBJ_DIR)/lj_class2_long_ptx.h

$(OBJ_DIR)/lj_class2_long.o: $(ALL_H) lj_class2_long.h lj_class2_long.cpp $(OBJ_DIR)/lj_class2_long_ptx.h \
$(OBJ_DIR)/charge_gpu_memory.o
$(CUDR) -o $@ -c lj_class2_long.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/lj_class2_long_ext.o: $(ALL_H) lj_class2_long.h lj_class2_long_ext.cpp charge_gpu_memory.h
$(CUDR) -o $@ -c lj_class2_long_ext.cpp -I$(OBJ_DIR)
```

```
$(OBJ_DIR)/ljcl_cut_gpu_kernel.ptx: ljcl_cut_gpu_kernel.cu pair_gpu_precision.h
$(CUDA) --ptx -DNV_KERNEL -o $@ ljcl_cut_gpu_kernel.cu

$(OBJ_DIR)/ljcl_cut_gpu_ptx.h: $(OBJ_DIR)/ljcl_cut_gpu_kernel.ptx $(OBJ_DIR)/ljcl_cut_gpu_kernel.ptx
$(BSH) ./geryon/file_to_cstr.sh $(OBJ_DIR)/ljcl_cut_gpu_kernel.ptx $(OBJ_DIR)/ljcl_cut_gpu_ptx.h

$(OBJ_DIR)/ljcl_cut_gpu_memory.o: $(ALL_H) ljcl_cut_gpu_memory.h ljcl_cut_gpu_memory.cpp \
$(OBJ_DIR)/ljcl_cut_gpu_ptx.h $(OBJ_DIR)/charge_gpu_memory.o
$(CUDR) -o $@ -c ljcl_cut_gpu_memory.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/ljcl_cut_gpu.o: $(ALL_H) ljcl_cut_gpu_memory.h ljcl_cut_gpu.cpp charge_gpu_memory.h
$(CUDR) -o $@ -c ljcl_cut_gpu.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/coul_long_gpu_kernel.ptx: coul_long_gpu_kernel.cu pair_gpu_precision.h
$(CUDA) --ptx -DNV_KERNEL -o $@ coul_long_gpu_kernel.cu

$(OBJ_DIR)/coul_long_gpu_ptx.h: $(OBJ_DIR)/coul_long_gpu_kernel.ptx $(OBJ_DIR)/coul_long_gpu_kernel.ptx
$(BSH) ./geryon/file_to_cstr.sh $(OBJ_DIR)/coul_long_gpu_kernel.ptx $(OBJ_DIR)/coul_long_gpu_ptx.h

$(OBJ_DIR)/coul_long_gpu_memory.o: $(ALL_H) coul_long_gpu_memory.h coul_long_gpu_memory.cpp \
$(OBJ_DIR)/coul_long_gpu_ptx.h $(OBJ_DIR)/charge_gpu_memory.o
$(CUDR) -o $@ -c coul_long_gpu_memory.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/coul_long_gpu.o: $(ALL_H) coul_long_gpu_memory.h coul_long_gpu.cpp charge_gpu_memory.h
$(CUDR) -o $@ -c coul_long_gpu.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/morse_gpu_kernel.ptx: morse_gpu_kernel.cu pair_gpu_precision.h
$(CUDA) --ptx -DNV_KERNEL -o $@ morse_gpu_kernel.cu

$(OBJ_DIR)/morse_gpu_ptx.h: $(OBJ_DIR)/morse_gpu_kernel.ptx $(OBJ_DIR)/morse_gpu_kernel.ptx
$(BSH) ./geryon/file_to_cstr.sh $(OBJ_DIR)/morse_gpu_kernel.ptx $(OBJ_DIR)/morse_gpu_ptx.h

$(OBJ_DIR)/morse_gpu_memory.o: $(ALL_H) morse_gpu_memory.h morse_gpu_memory.cpp $(OBJ_DIR)/morse_gpu_ptx.h \
 $(OBJ_DIR)/atomic_gpu_memory.o
$(CUDR) -o $@ -c morse_gpu_memory.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/morse_gpu.o: $(ALL_H) morse_gpu_memory.h morse_gpu.cpp atomic_gpu_memory.h
$(CUDR) -o $@ -c morse_gpu.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/crml_gpu_kernel.ptx: crml_gpu_kernel.cu pair_gpu_precision.h
$(CUDA) --ptx -DNV_KERNEL -o $@ crml_gpu_kernel.cu

$(OBJ_DIR)/crml_gpu_ptx.h: $(OBJ_DIR)/crml_gpu_kernel.ptx $(OBJ_DIR)/crml_gpu_kernel.ptx
$(BSH) ./geryon/file_to_cstr.sh $(OBJ_DIR)/crml_gpu_kernel.ptx $(OBJ_DIR)/crml_gpu_ptx.h

$(OBJ_DIR)/crml_gpu_memory.o: $(ALL_H) crml_gpu_memory.h crml_gpu_memory.cpp $(OBJ_DIR)/crml_gpu_ptx.h \
$(OBJ_DIR)/charge_gpu_memory.o
$(CUDR) -o $@ -c crml_gpu_memory.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/crml_gpu.o: $(ALL_H) crml_gpu_memory.h crml_gpu.cpp charge_gpu_memory.h
$(CUDR) -o $@ -c crml_gpu.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/lj96_cut_gpu_kernel.ptx: lj96_cut_gpu_kernel.cu pair_gpu_precision.h
$(CUDA) --ptx -DNV_KERNEL -o $@ lj96_cut_gpu_kernel.cu

$(OBJ_DIR)/lj96_cut_gpu_ptx.h: $(OBJ_DIR)/lj96_cut_gpu_kernel.ptx $(OBJ_DIR)/lj96_cut_gpu_kernel.ptx
```

```
$(BSH) ./geryon/file_to_cstr.sh $(OBJ_DIR)/lj96_cut_gpu_kernel.ptx $(OBJ_DIR)/lj96_cut_gpu_ptx.h

$(OBJ_DIR)/lj96_cut_gpu_memory.o: $(ALL_H) lj96_cut_gpu_memory.h lj96_cut_gpu_memory.cpp \
$(OBJ_DIR)/lj96_cut_gpu_ptx.h $(OBJ_DIR)/atomic_gpu_memory.o
$(CUDR) -o $@ -c lj96_cut_gpu_memory.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/lj96_cut_gpu.o: $(ALL_H) lj96_cut_gpu_memory.h lj96_cut_gpu.cpp atomic_gpu_memory.h
$(CUDR) -o $@ -c lj96_cut_gpu.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/lj_expand_gpu_kernel.ptx: lj_expand_gpu_kernel.cu pair_gpu_precision.h
$(CUDA) --ptx -DNV_KERNEL -o $@ lj_expand_gpu_kernel.cu

$(OBJ_DIR)/lj_expand_gpu_ptx.h: $(OBJ_DIR)/lj_expand_gpu_kernel.ptx $(OBJ_DIR)/lj_expand_gpu_kernel.ptx
$(BSH) ./geryon/file_to_cstr.sh $(OBJ_DIR)/lj_expand_gpu_kernel.ptx $(OBJ_DIR)/lj_expand_gpu_ptx.h

$(OBJ_DIR)/lj_expand_gpu_memory.o: $(ALL_H) lj_expand_gpu_memory.h lj_expand_gpu_memory.cpp \
$(OBJ_DIR)/lj_expand_gpu_ptx.h $(OBJ_DIR)/atomic_gpu_memory.o
$(CUDR) -o $@ -c lj_expand_gpu_memory.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/lj_expand_gpu.o: $(ALL_H) lj_expand_gpu_memory.h lj_expand_gpu.cpp atomic_gpu_memory.h
$(CUDR) -o $@ -c lj_expand_gpu.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/cmm_cut_gpu_kernel.ptx: cmm_cut_gpu_kernel.cu pair_gpu_precision.h
$(CUDA) --ptx -DNV_KERNEL -o $@ cmm_cut_gpu_kernel.cu

$(OBJ_DIR)/cmm_cut_gpu_ptx.h: $(OBJ_DIR)/cmm_cut_gpu_kernel.ptx $(OBJ_DIR)/cmm_cut_gpu_kernel.ptx
$(BSH) ./geryon/file_to_cstr.sh $(OBJ_DIR)/cmm_cut_gpu_kernel.ptx $(OBJ_DIR)/cmm_cut_gpu_ptx.h

$(OBJ_DIR)/cmm_cut_gpu_memory.o: $(ALL_H) cmm_cut_gpu_memory.h cmm_cut_gpu_memory.cpp \
$(OBJ_DIR)/cmm_cut_gpu_ptx.h $(OBJ_DIR)/atomic_gpu_memory.o
$(CUDR) -o $@ -c cmm_cut_gpu_memory.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/cmm_cut_gpu.o: $(ALL_H) cmm_cut_gpu_memory.h cmm_cut_gpu.cpp atomic_gpu_memory.h
$(CUDR) -o $@ -c cmm_cut_gpu.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/cmmc_long_gpu_kernel.ptx: cmmc_long_gpu_kernel.cu pair_gpu_precision.h
$(CUDA) --ptx -DNV_KERNEL -o $@ cmmc_long_gpu_kernel.cu

$(OBJ_DIR)/cmmc_long_gpu_ptx.h: $(OBJ_DIR)/cmmc_long_gpu_kernel.ptx $(OBJ_DIR)/cmmc_long_gpu_kernel.ptx
$(BSH) ./geryon/file_to_cstr.sh $(OBJ_DIR)/cmmc_long_gpu_kernel.ptx $(OBJ_DIR)/cmmc_long_gpu_ptx.h

$(OBJ_DIR)/cmmc_long_gpu_memory.o: $(ALL_H) cmmc_long_gpu_memory.h cmmc_long_gpu_memory.cpp \
$(OBJ_DIR)/cmmc_long_gpu_ptx.h $(OBJ_DIR)/atomic_gpu_memory.o
$(CUDR) -o $@ -c cmmc_long_gpu_memory.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/cmmc_long_gpu.o: $(ALL_H) cmmc_long_gpu_memory.h cmmc_long_gpu.cpp charge_gpu_memory.h
$(CUDR) -o $@ -c cmmc_long_gpu.cpp -I$(OBJ_DIR)

$(OBJ_DIR)/cmmc_msm_gpu_kernel.ptx: cmmc_msm_gpu_kernel.cu pair_gpu_precision.h
$(CUDA) --ptx -DNV_KERNEL -o $@ cmmc_msm_gpu_kernel.cu

$(OBJ_DIR)/cmmc_msm_gpu_ptx.h: $(OBJ_DIR)/cmmc_msm_gpu_kernel.ptx $(OBJ_DIR)/cmmc_msm_gpu_kernel.ptx
$(BSH) ./geryon/file_to_cstr.sh $(OBJ_DIR)/cmmc_msm_gpu_kernel.ptx $(OBJ_DIR)/cmmc_msm_gpu_ptx.h

$(OBJ_DIR)/cmmc_msm_gpu_memory.o: $(ALL_H) cmmc_msm_gpu_memory.h cmmc_msm_gpu_memory.cpp \
$(OBJ_DIR)/cmmc_msm_gpu_ptx.h $(OBJ_DIR)/atomic_gpu_memory.o
$(CUDR) -o $@ -c cmmc_msm_gpu_memory.cpp -I$(OBJ_DIR)
```

```
$(OBJ_DIR)/cmmc_msm_gpu.o: $(ALL_H) cmmc_msm_gpu_memory.h cmmc_msm_gpu.cpp charge_gpu_memory.h
$(CUDR) -o $@ -c cmmc_msm_gpu.cpp -I$(OBJ_DIR)

$(BIN_DIR)/nvc_get_devices: ./geryon/ucl_get_devices.cpp $(NVC_H)
$(CUDR) -o $@ ./geryon/ucl_get_devices.cpp -DUCL_CUDART $(CUDA_LINK)

$(GPU_LIB): $(OBJS)
$(AR) -crusv $(GPU_LIB) $(OBJS)

clean:
rm -f $(EXECS) $(GPU_LIB) $(OBJS) $(PTXS) *.linkinfo

veryclean: clean
rm -rf *~ *.linkinfo
```

Due to the fact that during this build process the GPU is probed and a parameter optimization phase is invoked, the build itself is not reported here for brevity. Further, the remainder of the process is nearly identical to the above description. The following are needed to complete the compilation:

Makefile.package

```
PKG_INC =
PKG_PATH =  -L../../lib/gpu
PKG_LIB =   -lgpu

PKG_SYSINC =  $(gpu_SYSINC)
PKG_SYSLIB =  $(gpu_SYSLIB)
PKG_SYSPATH = $(gpu_SYSPATH)


g++   angle.o angle_charmm.o angle_cosine.o angle_cosine_delta.o angle_cosine_periodic.o \
 angle_cosine_squared.o angle_harmonic.o angle_hybrid.o angle_table.o atom.o atom_vec.o \
atom_vec_angle.o atom_vec_atomic.o atom_vec_bond.o atom_vec_charge.o atom_vec_ellipsoid.o \
atom_vec_full.o atom_vec_hybrid.o atom_vec_molecular.o atom_vec_sphere.o bond.o bond_fene.o \
bond_fene_expand.o bond_harmonic.o bond_hybrid.o bond_morse.o bond_nonlinear.o \
bond_quartic.o bond_table.o change_box.o comm.o compute.o compute_angle_local.o \
compute_atom_molecule.o compute_bond_local.o compute_centro_atom.o compute_cluster_atom.o \
compute_cna_atom.o compute_com.o compute_com_molecule.o compute_coord_atom.o \
compute_dihedral_local.o compute_displace_atom.o compute_erotate_sphere.o \
compute_group_group.o compute_gyration.o compute_gyration_molecule.o compute_heat_flux.o \
compute_improper_local.o compute_ke.o compute_ke_atom.o compute_msd.o compute_msd_molecule.o \
compute_pair.o compute_pair_local.o compute_pe.o compute_pe_atom.o compute_pressure.o \
compute_property_atom.o compute_property_local.o compute_property_molecule.o compute_rdf.o \
compute_reduce.o compute_reduce_region.o compute_slice.o compute_stress_atom.o compute_temp.o \
compute_temp_com.o compute_temp_deform.o compute_temp_partial.o compute_temp_profile.o \
compute_temp_ramp.o compute_temp_region.o compute_temp_sphere.o compute_ti.o create_atoms.o \
create_box.o delete_atoms.o delete_bonds.o dihedral.o dihedral_charmm.o dihedral_harmonic.o \
dihedral_helix.o dihedral_hybrid.o dihedral_multi_harmonic.o dihedral_opls.o displace_atoms.o \
displace_box.o domain.o dump.o dump_atom.o dump_cfg.o dump_custom.o dump_dcd.o dump_image.o \
dump_local.o dump_xyz.o error.o ewald.o fft3d.o fft3d_wrap.o finish.o fix.o fix_adapt.o \
fix_addforce.o fix_ave_atom.o fix_ave_correlate.o fix_ave_histo.o fix_ave_spatial.o \
fix_ave_time.o fix_aveforce.o fix_box_relax.o fix_deform.o fix_deposit.o fix_drag.o \
fix_dt_reset.o fix_efield.o fix_enforce2d.o fix_evaporate.o fix_external.o fix_gpu.o \
fix_gravity.o fix_heat.o fix_indent.o fix_langevin.o fix_lineforce.o fix_minimize.o \
fix_momentum.o fix_move.o fix_nh.o fix_nh_sphere.o fix_nph.o fix_nph_sphere.o fix_npt.o \
fix_npt_sphere.o fix_nve.o fix_nve_limit.o fix_nve_noforce.o fix_nve_sphere.o fix_nvt.o \
fix_nvt_sllod.o fix_nvt_sphere.o fix_orient_fcc.o fix_planeforce.o fix_press_berendsen.o \
```

```
fix_print.o fix_qeq_comb.o fix_read_restart.o fix_recenter.o fix_respa.o fix_restrain.o \
fix_rigid.o fix_rigid_nve.o fix_rigid_nvt.o fix_setforce.o fix_shake.o fix_shear_history.o \
fix_spring.o fix_spring_rg.o fix_spring_self.o fix_store_force.o fix_store_state.o \
fix_temp_berendsen.o fix_temp_rescale.o fix_thermal_conductivity.o fix_tmd.o fix_ttm.o \
fix_viscosity.o fix_viscous.o fix_wall.o fix_wall_harmonic.o fix_wall_lj126.o fix_wall_lj93.o \
fix_wall_reflect.o fix_wall_region.o force.o group.o improper.o improper_cvff.o \
improper_harmonic.o improper_hybrid.o improper_umbrella.o input.o integrate.o irregular.o \
kspace.o lammps.o lattice.o library.o main.o math_extra.o memory.o min.o min_cg.o min_fire.o \
min_hftn.o min_linesearch.o min_quickmin.o min_sd.o minimize.o modify.o neigh_bond.o \
neigh_derive.o neigh_full.o neigh_gran.o neigh_half_bin.o neigh_half_multi.o neigh_half_nsq.o \
neigh_list.o neigh_request.o neigh_respa.o neigh_stencil.o neighbor.o output.o pair.o \
pair_adp.o pair_airebo.o pair_born.o pair_born_coul_long.o pair_buck.o pair_buck_coul_cut.o \
pair_buck_coul_long.o pair_comb.o pair_coul_cut.o pair_coul_debye.o pair_coul_long.o \
pair_coul_long_gpu.o pair_dpd.o pair_dpd_tstat.o pair_eam.o pair_eam_alloy.o pair_eam_fs.o \
pair_eim.o pair_gauss.o pair_hbond_dreiding_lj.o pair_hbond_dreiding_morse.o pair_hybrid.o \
pair_hybrid_overlay.o pair_lj96_cut.o pair_lj96_cut_gpu.o pair_lj_charmm_coul_charmm.o \
pair_lj_charmm_coul_charmm_implicit.o pair_lj_charmm_coul_long.o \
pair_lj_charmm_coul_long_gpu.o pair_lj_cubic.o pair_lj_cut.o pair_lj_cut_coul_cut.o \
pair_lj_cut_coul_cut_gpu.o pair_lj_cut_coul_debye.o pair_lj_cut_coul_long.o \
pair_lj_cut_coul_long_gpu.o pair_lj_cut_coul_long_tip4p.o pair_lj_cut_gpu.o pair_lj_cut_tgpu.o \
pair_lj_expand.o pair_lj_expand_gpu.o pair_lj_gromacs.o pair_lj_gromacs_coul_gromacs.o \
pair_lj_smooth.o pair_morse.o pair_morse_gpu.o pair_omp_gpu.o pair_rebo.o pair_soft.o \
pair_sw.o pair_table.o pair_tersoff.o pair_tersoff_zbl.o pair_yukawa.o pppm.o pppm_cg.o \
pppm_gpu.o pppm_tip4p.o random_mars.o random_park.o read_data.o read_restart.o region.o \
region_block.o region_cone.o region_cylinder.o region_intersect.o region_plane.o \
region_prism.o region_sphere.o region_union.o remap.o remap_wrap.o replicate.o respa.o \
run.o set.o special.o thermo.o timer.o universe.o update.o variable.o velocity.o verlet.o \
write_restart.o -L../../lib/gpu -lgpu  \
-L/usr/common/usg/openmpi/1.4.2/gcc/lib -lmpi_cxx -lmpi \
-L/usr/common/usg/fftw/2.1.5-gnu/lib -ldfftw -ldfftw_mpi -lsfftw -lsfftw_mpi \
-L/usr/common/usg/cuda/3.2/lib -L/usr/common/usg/cuda/3.2/lib64 -lcudart -lcuda  \
-lstdc++ -o ../lmp_linux
```

Q4 Run Script

The run scripts are basically the same in each case but the binaries, though sharing the same name, were built differently and executed in different directories (as noted above). The name given to each job is descriptive of the case executed. For instance, *lmps-eam-mpi-1Ma-64p8n* indicates that the LAMMPS binary that only utilizes MPI processes is used to evaluate the embedded atomic potential interaction for one million Cu atoms on eight nodes employing a total of 64 MPI processes. To study other systems, the input files are modified and the name changed to match the problem instance.

lmp-q4-eam-mpi.sc

```
#!/bin/bash
#PBS -V
#PBS -q dirac_special
#PBS -l nodes=8:ppn=8:fermi
#PBS -l walltime=00:15:00
#PBS -A m947
#PBS -N lmps-eam-mpi-1Ma-64p8n
#PBS -j oe

cd ${PBS_O_WORKDIR}
time mpirun -n 64 ./lmp_linux < in.eam.mpi
```

lmp-q4-eam-user_cuda.sc

```
#!/bin/bash
#PBS -V
#PBS -q dirac_special
#PBS -l nodes=8:ppn=8:fermi
#PBS -l walltime=00:15:00
#PBS -A m947
#PBS -N lmps-eam-cuda-1Ma-8p8n
#PBS -j oe

cd ${PBS_O_WORKDIR}
time mpirun -n 8 ./lmp_linux < in.eam.user-cuda
```

Note the request for 8 nodes and the execution of a single MPI process per node in the above script. The `user-cuda` package cannot support more than a single MPI process working in coordination with the GPU in this implementation.

lmp-q4-lj-gpu.sc

```
#!/bin/bash
#PBS -V
#PBS -q dirac_special
#PBS -l nodes=8:ppn=8:fermi
#PBS -l walltime=00:15:00
#PBS -A m947
#PBS -N lmps-lj-gpu-4Ma-64p8n
#PBS -j oe

cd ${PBS_O_WORKDIR}
time mpirun -n 64 ./lmp_linux < in.lj.gpu
```

lmp-q4-lj-mpi.sc

```
#!/bin/bash
#PBS -V
#PBS -q dirac_special
#PBS -l nodes=8:ppn=8:fermi
#PBS -l walltime=00:15:00
#PBS -A m947
#PBS -N lmps-lj-mpi-4Ma-64p8n
#PBS -j oe

cd ${PBS_O_WORKDIR}
time mpirun -n 64 ./lmp_linux < in.lj.mpi
```

There exists a nice tool that greatly simplifies profiling on the GPU cluster at NERSC. The Integrated Performance Monitoring tool, IPM, was used during debugging and for optimization of algorithmic parameters. One can learn more about IPM at . The process for utilizing IPM is fairly non-intrusive. The run script is provided here. To invoke a comprehensive set of predefined hardware monitors, one must define a couple of environment variables and simply preload the correct library version at runtime, ie for MPI only runs or those that include CUDA based GPU instructions. For instance, `time mpirun -n 1 -x LD_PRELOAD=/usr/common/usg/ipm/2.00/lib/libipm.so ./lmp_linux < in.lj.gpu`, or `time mpirun -n 1 -x LD_PRELOAD=/usr/common/usg/ipm/2.00/lib/libipm_cuda.so ./lmp_linux < in.lj.gpu`. In the first case, only the host instructions will be monitored. In the latter case, both host and GPU instructions will be captured.

An example script is provided here:

lmp-q4-ipm.sc

```
#!/bin/bash
#PBS -V
#PBS -q dirac_special
#PBS -l nodes=30:ppn=1:fermi
#PBS -l walltime=00:20:00
#PBS -A m947
#PBS -N lmps_cu32_sm20-4Ma30gpu-ipm
#PBS -j oe

cd ${PBS_O_WORKDIR}
export IPM_REPORT=full
export IPM_LOG=full
time mpirun -np 30 -npernode 1 -x LD_PRELOAD=/usr/common/usg/ipm/2.00/lib/libipm_cuda.so
./lmp_gcc_cu32_sm20 -cuda on < in.eam.q4
```

Q4 Runtime Environment

The environment requires that the proper version of CUDA and NVIDIA drivers be installed. While CUDA 4.0 was tested on the target system. Only CUDA 3.2 was built on a significant fraction of the nodes and thus the logical choice for testing the LAMMPS algorithms for weak and strong scaling.

```
> cat lmp-env-gnu-3.2.sc
module load cuda/3.2;
module load nvidia-driver-util/3.2;
module swap pgi gcc/4.4.2;
module swap openmpi/1.4.2 openmpi-gnu/1.4.2;
module load fftw-gnu/2.1.5;
module load papi

> source lmp-env-gnu-3.2.sc
> module list
Currently Loaded Modulefiles:
  1) gcc/4.4.2                3) cuda/3.2                 5) fftw-gnu/2.1.5
  2) openmpi-gnu/1.4.2        4) nvidia-driver-util/3.2   6) papi/4.0.0

> env
USER=roche
LOGNAME=roche
HOME=/global/homes/r/roche
PATH=/usr/common/usg/papi/4.0.0/bin:/usr/common/usg/nvidia-driver-util/3.2/bin: \
/usr/common/usg/cuda/3.2/bin:/usr/common/usg/openmpi/1.4.2/gcc/bin:/usr/common/usg/gcc/4.4.2/bin: \
/usr/syscom/opt/moab/default/bin:/usr/syscom/opt/torque/default/bin:/usr/common/usg/bin: \
/usr/common/mss/bin:/usr/common/nsg/bin:/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin
MAIL=/var/spool/mail/roche
SHELL=/bin/csh
SSH_CLIENT=166.187.248.90 57260 22
SSH_CONNECTION=166.187.248.90 57260 128.55.69.10 22
SSH_TTY=/dev/pts/9
TERM=xterm-color
HOSTTYPE=x86_64-linux
VENDOR=unknown
OSTYPE=linux
MACHTYPE=x86_64
SHLVL=1
PWD=/global/scratch/sd/roche
```

```
GROUP=roche
HOST=cvrsvc01
REMOTEHOST=166.187.248.90
HOSTNAME=cvrsvc01
INPUTRC=/etc/inputrc
G_BROKEN_FILENAMES=1
LESSOPEN=|/usr/bin/lesspipe.sh %s
NERSC_HOST=carver
MOABHOMEDIR=/usr/syscom/opt/moab/default
MODULE_VERSION=3.2.7
MODULE_VERSION_STACK=3.2.7
MODULESHOME=/usr/common/nsg/opt/Modules/3.2.7
MODULEPATH=/usr/common/nsg/opt/Modules/3.2.7/config/versions: \
/usr/common/nsg/opt/Modules/$MODULE_VERSION/modulefiles:/usr/common/nsg/Modules/3.2.7/config/: \
/opt/modules/3.1.6/modulefiles:/usr/common/usg/Modules/modulefiles: \
/usr/common/acts/Modules/modulefiles:/usr/common/nsg/Modules/modulefiles: \
/usr/common/ftg/Modules/modulefiles:/usr/common/graphics/Modules/modulefiles: \
/usr/common/mss/Modules/modulefiles:/usr/common/net/Modules/modulefiles: \
/usr/common/osg/Modules/modulefiles:/usr/common/osp/Modules/modulefiles
LOADEDMODULES=gcc/4.4.2:openmpi-gnu/1.4.2:cuda/3.2:nvidia-driver-util/3.2:fftw-gnu/2.1.5:papi/4.0.0
LD_LIBRARY_PATH=/usr/common/usg/papi/4.0.0/lib:/usr/common/usg/fftw/2.1.5-gnu/lib: \
/usr/common/usg/nvidia-driver-util/3.2/lib64:/usr/common/usg/nvidia-driver-util/3.2/lib: \
/usr/common/usg/cuda/3.2/lib64:/usr/common/usg/cuda/3.2/lib:/usr/syscom/opt/torque/default/lib: \
/usr/common/usg/openmpi/1.4.2/gcc/lib:/usr/common/usg/gmp/4.3.1/lib:/usr/common/usg/mpfr/2.4.1/lib: \
/usr/common/usg/gcc/4.4.2/lib64:/usr/common/usg/gcc/4.4.2/lib
_LMFILES_=/usr/common/usg/Modules/modulefiles/gcc/4.4.2: \
/usr/common/usg/Modules/modulefiles/openmpi-gnu/1.4.2:/usr/common/usg/Modules/modulefiles/cuda/3.2: \
/usr/common/usg/Modules/modulefiles/nvidia-driver-util/3.2: \
/usr/common/usg/Modules/modulefiles/fftw-gnu/2.1.5:/usr/common/usg/Modules/modulefiles/papi/4.0.0
OMPI_DIR=/usr/common/usg/openmpi/1.4.2/gcc
GSCRATCH=/global/scratch/sd/roche
SCRATCH=/global/scratch/sd/roche
TMPDIR=/global/scratch/sd/roche
CUDA_INCLUDE=-I/usr/common/usg/cuda/3.2/include
CUDA_LIB=-L/usr/common/usg/cuda/3.2/lib
CUDA_LIB64=-L/usr/common/usg/cuda/3.2/lib64
NVIDIA_DRIVER_LIB=-L/usr/common/usg/nvidia-driver-util/3.2/lib64
CC=gcc
CXX=g++
F77=gfortran
F90=gfortran
FC=gfortran
FFTW=-I/usr/common/usg/fftw/2.1.5-gnu/include -L/usr/common/usg/fftw/2.1.5-gnu/lib
FFTW_INC=/usr/common/usg/fftw/2.1.5-gnu/include
FFTW_LIBDIR=/usr/common/usg/fftw/2.1.5-gnu/lib
FFTW_ROOT=/usr/common/usg/fftw/2.1.5-gnu
INFOPATH=/usr/common/usg/fftw/2.1.5-gnu/info
PAPI_INCLUDE=/usr/common/usg/papi/4.0.0/include
PAPI_LIB=/usr/common/usg/papi/4.0.0/lib
PAPI_VERSION=4.0.0
```

In the event that one chooses to use IPM, then the additional step required is simply:

```
> module load ipm
> module list
Currently Loaded Modulefiles:
```

```
  1) gcc/4.4.2                3) papi/4.0.0               5) cuda/3.2                 7) fftw-gnu/2.1.5
  2) openmpi-gnu/1.4.2        4) ipm/2.00                 6) nvidia-driver-util/3.2
```

The only changes to the environment that result from loading module IPM follow. Note that the environment variables included in the run script are not defined simply by loading the module.

```
> env | grep IPM
IPM_HPM=PAPI_DP_OPS,PAPI_FP_OPS,PAPI_SP_OPS,PAPI_TOT_INS
IPM_LOGDIR=/project/projectdirs/ipm/log/carver
> env | grep ipm
PATH=/usr/common/usg/nvidia-driver-util/3.2/bin:/usr/common/usg/cuda/3.2/bin:
/usr/common/usg/papi/4.0.0/bin:/usr/common/usg/ipm/2.00/bin:/usr/common/usg/openmpi/1.4.2/gcc/bin:
/usr/common/usg/gcc/4.4.2/bin:/usr/syscom/opt/moab/default/bin:/usr/syscom/opt/torque/default/bin:
/usr/common/usg/bin:/usr/common/mss/bin:/usr/common/nsg/bin:/usr/kerberos/bin:/usr/local/bin:/bin:
/usr/bin:/usr/X11R6/bin
LOADEDMODULES=gcc/4.4.2:openmpi-gnu/1.4.2:papi/4.0.0:ipm/2.00:cuda/3.2:nvidia-driver-util/3.2:fftw-gnu/2.1.5
_LMFILES_=/usr/common/usg/Modules/modulefiles/gcc/4.4.2:
/usr/common/usg/Modules/modulefiles/openmpi-gnu/1.4.2:/usr/common/usg/Modules/modulefiles/papi/4.0.0:
/usr/common/usg/Modules/modulefiles/ipm/2.00:/usr/common/usg/Modules/modulefiles/cuda/3.2:
/usr/common/usg/Modules/modulefiles/nvidia-driver-util/3.2:/usr/common/usg/Modules/modulefiles/fftw-gnu/2.1.5
IPM_LOGDIR=/project/projectdirs/ipm/log/carver
```

0.6.3 OMEN, NEMO 5

Q2 Input Settings input_file_omen.txt

```
mat_name                = Si_new;
a0                      = 0.543;
first_atom              = cation;

poisson_criterion       = 1e-3;
poisson_iteration       = 1;
poisson_inner_criterion = 1e-2;

NDim                    = 2;
Nkz                     = 25;

tb                      = 10;
dsp3                    = 30;

Temp                    = 300;

n_of_modes              = 16;
Nk                      = 251;

max_bond_def            = 0.1;

last_first              = 0;

x                       = [1 0 1];
y                       = [0 1 0];
z                       = [-1 0 1];
```

```
eta_res                 = 1e-10;
eta                     = 0.0;

Elimit                  = 50e-3;
Emin_tail               = 4.0e-3;
EOffset                 = 25*UT;
dE_in                   = 7.5e-4;
dE_f                    = 7.5e-4;
dE_sep                  = 1.0e-4;
NEmax                   = 20;
regular_mesh            = 1;

CPU_per_vg_point        = all;
CPU_per_kz_point        = 3408;
CPU_per_wire            = 568;
CPU_ppoint              = 4;
spec_decomp             = 2;
NPCS                    = 2;

strain.on               = 1;
strain.Eps_xx           = 0.01;
strain.Eps_yy           = -6.2274e-04;
strain.Eps_zz           = -0.003616;
strain.zeta             = 0.53;

Eps_wire                = 11.9;
Eps_ox                  = 20.0;
Xi_wire                 = 4.05;
phi_m                   = 4.20;

NVG                     = 1;
Vgmin                   = 0.5;
Vgmax                   = 0.5;

NVS                     = 1;
Vsmin                   = 0.0;
Vsmax                   = 0.0;

NVD                     = 1;
Vdmin                   = 0.7;
Vdmax                   = 0.7;

no_ph_energy            = 10;
no_ph_k                 = 101;
sc_dist_dep             = 1;
//sc_restart            = 3;
sc_max_iteration        = 20;

incoherent_injection    = 1;

restart                 = [2 0 0 0];
vact_file               = vact_dat;

update_fitness          = 0;
fitness_file            = FitMat_dat;
```

```
no_mat                  = 3;
no_channel_mat          = 1;
no_oxide_mat            = 2;

Lc                      = 12.0;
Ls                      = 10.0;
Ld                      = 10.0;
tc                      = 5.0;
tox                     = 2.9;
x0                      = 0.0;
y0                      = 0.0;

mat_type(1)             = square;
mat_cs(1)               = yes;
mat_coord(1,1)          = [x0 y0];
mat_coord(1,2)          = [Ls+Lc+Ld y0];
mat_coord(1,3)          = [Ls+Lc+Ld tc];
mat_coord(1,4)          = [x0 tc];

ox_type(1)              = square;
ox_cs(1)                = no;
ox_coord(1,1)           = [x0 y0-tox];
ox_coord(1,2)           = [Ls+Lc+Ld y0-tox];
ox_coord(1,3)           = [Ls+Lc+Ld y0];
ox_coord(1,4)           = [x0 y0];

ox_type(2)              = square;
ox_cs(2)                = no;
ox_coord(2,1)           = [x0 tc];
ox_coord(2,2)           = [Ls+Lc+Ld tc];
ox_coord(2,3)           = [Ls+Lc+Ld tc+tox];
ox_coord(2,4)           = [x0 tc+tox];

no_gate                 = 2;

gate_type(1)            = square;
gate_coord(1,1)         = [Ls y0-tox];
gate_coord(1,2)         = [Ls+Lc y0-tox];

gate_type(2)            = square;
gate_coord(2,1)         = [Ls tc+tox];
gate_coord(2,2)         = [Ls+Lc tc+tox];

no_doping               = 2;

ND_S                    = 1e26;
NA_S                    = 1e12;
ND_D                    = 1e26;
NA_D                    = 1e12;

doping_type(1)          = square;
doping_ND(1)            = ND_S;
doping_NA(1)            = NA_S;
doping_coord(1,1)       = [x0 y0];
doping_coord(1,2)       = [Ls y0];
doping_coord(1,3)       = [Ls tc];
doping_coord(1,4)       = [x0 tc];
```

```
doping_type(2)          = square;
doping_ND(2)            = ND_D;
doping_NA(2)            = NA_D;
doping_coord(2,1)       = [Ls+Lc y0];
doping_coord(2,2)       = [Ls+Lc+Ld y0];
doping_coord(2,3)       = [Ls+Lc+Ld tc];
doping_coord(2,4)       = [Ls+Lc tc];

command(1)              = Write_Grid_Matrix;
command(2)              = EL_SC_SCATT;
```

eigensolver.in

```
Structure
{
Material
{
name = InAs
tag = dot
crystal_structure = zincblende
Bands:TB:sp3d5sstar_SO:param_set = param_Klimeck

regions = (2)
}

Material
{
name = GaAs
tag =  barrier
crystal_structure = zincblende
Bands:TB:sp3d5sstar_SO:param_set = param_Klimeck

regions = (1)
}

Domain
{
name = atomic_structure
type = pseudomorphic
base_material = barrier
dimension = (@dim@,@dim@,@dim@)

periodic = (false,false,false)
passivate = true

crystal_direction1 = (1,0,0)
crystal_direction2 = (0,1,0)
crystal_direction3 = (0,0,1)

space_orientation_dir1 =  (1.0,0.0,0.0)
space_orientation_dir2 =  (0.0,1.0,0.0)

origin = (0, 0, 0)

regions = (1,2)
```

```
output = ()
// FEM_mesh_creation = false
}

Geometry
{
Region
{
shape         = cuboid
region_number = 1
priority      = 1
min           = (-1,-1,-1)
max           = (1000,1000,1000)
}
Region
{
shape         = cuboid
region_number = 2
priority      = 2
min           = ( @x1@, @x1@, @x1@)
max           = ( @x2@, @x2@, @x2@)
}

Partitioning
{
x_extension = (-0.01, @xmax@)
y_extension = (-0.01, @xmax@)
z_extension = (-0.01, @xmax@)
}
}
}

Solvers
{
solver
{
name           = qd_@PROCS@
type           = Schroedinger
domain         = atomic_structure
active_regions = (1,2)
job_list       = (passivate_H, calculate_band_structure)
output         = (energies, eigenfunctions_Silo_k0)
//output         = (energies)
output_precision = 7

tb_basis = sp3d5sstar_SO

number_of_eigenvalues = 6
shift                 = 1.0
max_number_iterations = 5000
convergence_limit     = 1e-7

// ncv=40 gives >=6 evals for nprocs<=384, nprocs=768 only finds 3 evals
ncv                   = 60

// fix scalability problem of AOCreateMapping: do not use it
// (dirty hack that works only when all atoms are active and only for Schroedinger)
```

```
assume_all_atom_active = true

eigen_values_solver   = krylovschur

 solver_transformation_type = shift

linear_solver         = gmres
//linear_solver         = bcgs
//linear_solver         = bicg
//linear_solver         = tcqmr

//preconditioner        = asm

monitor_convergence   = true

number_of_nodes = (1)
k_space_basis = cartesian
k_points      = [ (0.0,0.0,0.0) ]
}
}

Global
{
solve = (qd_@PROCS@)
database = ./all.mat

// output is now piped to logfile
//logfile = eigensolver_@PROCS@.log
messaging_level = 3

petsc_double_report = true
petsc_complex_report = true
}
```

UTB.in

```
// ---------------------------------------------------------------------------
// Ballistic transport in an atomistic  Si ultra-thin body (UTB)
// device dimensions: 42x8.15nm (Poisson), 42x5nm (Quantum)
// 1e20cm-3 doping in regions [0,10nm] and [32nm,42nm]
// transport direction is [100] - 3rd dimension is 1 (cubic) unit cell
// band model: sp3d5sstar (10-band)
// voltages: Vs=0V, Vd=0.001V, Vg=0V...1.1V in 0.1V steps (12 bias points)
//
// parallelization: V-E-k-x, NV=12 always completely parallelized,
// NE/Nk/Nx parallelization self-determined (top-down)
// NE=576 (~same as avg of OMEN scaling), Nk=15, Nx=2
// run on jaguar using the following:
// 768, 1152, 2304, 4608, 6912, 9216, 18432, 25920, 51840, 77760, 103680, 207360
// ---------------------------------------------------------------------------

Structure
{
Material
{
tag = channel
```

```
name = Si
crystal_structure = diamond
regions = (2)

Bands:TB:sp3d5sstar:param_set = param_Boykin
Bands:TB:sp3sstar:param_set = param_Klimeck
}


Material
{
tag = lead

name = Si
crystal_structure = diamond
regions = (1, 3)

doping_type = N
doping_density = 1e20

Bands:TB:sp3d5sstar:param_set = param_Boykin
Bands:TB:sp3sstar:param_set = param_Klimeck
}

Material
{
tag = oxide

name = Si
crystal_structure = diamond
regions = (4)

Bands:TB:sp3d5sstar:param_set = param_Boykin
Bands:TB:sp3sstar:param_set = param_Klimeck
}

Domain
{
name = device
type = pseudomorphic
base_material = channel
regions = (1, 2, 3, 4)
passivation_regions = (4)

// crystal growth direction
crystal_direction1 = (1, 0, 0)
crystal_direction2 = (0, 1, 0)
crystal_direction3 = (0, 0, 1)

// extrude along "crystal_direction"
dimension = (77, 15, 1)

// periodicity of "dimension"
periodic = (false, false, true)

// integer coordinate of starting cell of domain
```

```
starting_cell_coordinate = (0, 0, 0)

// real direction of "crystal_direction"
space_orientation_dir1 =  (1, 0, 0)
space_orientation_dir2 =  (0, 1, 0)

// position of the first atom of starting cell
origin = (0, 0, 0)

// names of leads domain
leads = (source_contact, drain_contact)

geometry_description = simple_shapes

projection_direction = (1, 0, 0)
}


Domain
{
name = source_contact
type = pseudomorphic
base_material = channel
regions = (1)

crystal_direction1 = (1, 0, 0)
crystal_direction2 = (0, 1, 0)
crystal_direction3 = (0, 0, 1)

dimension = (1, 15, 1)
periodic = (true, false, true)
starting_cell_coordinate = (-1, 0, 0)

space_orientation_dir1 =  (1, 0, 0)
space_orientation_dir2 =  (0, 1, 0)

lead_direction = -1
lead_type = N

geometry_description = simple_shapes
}


Domain
{
name = drain_contact
type = pseudomorphic
base_material = channel
regions = (3)

crystal_direction1 = (1, 0, 0)
crystal_direction2 = (0, 1, 0)
crystal_direction3 = (0, 0, 1)

dimension = (1, 15, 1)
periodic = (true, false, true)
starting_cell_coordinate = (77, 0, 0)
```

```
space_orientation_dir1 =  (1, 0, 0)
space_orientation_dir2 =  (0, 1, 0)

lead_direction = 1
lead_type = N

geometry_description = simple_shapes
}


Domain
{
name = continuum
type = finite_elements
mesh_from_domain = device
}


Geometry
{
Region // source silicon
{
  shape         = cuboid
  region_number = 1
  priority      = 2
min           = (-100, 1.6, -100)
  max           = ( 9.7, 6.6, 100)
}

Region // channel silicon
{
  shape         = cuboid
  region_number = 2
  priority      = 2
min           = (9.7, 1.6, -100)
  max           = ( 32, 6.6, 100)
}

Region // drain silicon
{
  shape         = cuboid
  region_number = 3
  priority      = 2
min           = ( 32, 1.6, -100)
  max           = (100, 6.6, 100)
}

Region // SiO2
{
  shape         = cuboid
  region_number = 4
  priority      = 1
min           = (-100, -100, -100)
  max           = ( 100,  100,  100)
}
```

```
Boundary_region // top
{
shape = cuboid
region_number = 1
priority = 1
min = (9.7, 8.1, -100)
max = (32.1, 100, 100)
}

Boundary_region // bottom
{
shape = cuboid
region_number = 2
priority = 1
min = (9.7, -100, -100)
max = (32.1, 0.1, 100)
}
}
}



Solvers
{
solver
{
name = UTB
type = Ramper
domain = device

density_solver = density_solver

contacts = (source_contact, drain_contact, gate)
source_contact = (0.0  , 0.0  , 0.0  , 0.0  , 0.0  , 0.0  , 0.0  ,   0.0,   0.0,   0.0,   0.0,   0.0)
drain_contact =  (0.001, 0.001, 0.001, 0.001, 0.001, 0.001, 0.001, 0.001, 0.001, 0.001, 0.001, 0.001)
//drain_contact =  (1.000, 1.000, 1.000, 1.000, 1.000, 1.000, 1.000, 1.000, 1.000, 1.000, 1.000, 1.000)
gate =          (0.0  , 0.1  , 0.2  , 0.3  , 0.4  , 0.5  , 0.6  ,   0.7,   0.8,   0.9,   1.0,   1.1)
parallelization = 12 // parallelize all V-points

output_name = UTB_@PROCS@_ramper
}


solver
{
name = potential_solver
type = NonlinearPoisson
domain = continuum

ksp_type = gmres
pc_type = lu

Newton_step_only = true
density_solver = (density_solver)

charge_model = electron_hole
equilibrium_el_chem_pot = 0.0
```

```
set_initial_potential = true

// this statement should be placed at the end
boundary_condition
{
name = gate
type = ElectrostaticContact
boundary_regions = (1, 2)
//voltage = 0.0 // want to apply biases in Ramper
}
}

solver
{
name           = semiclass_dens
type           = Semiclassical
domain         = device
active_regions = (1, 2, 3, 4)

potential_solver = potential_solver
fermilevel_solver = density_solver // control option for linear drop

//small_coordinate_contact = source_contact
                // [linear drop] model contact with smaller coordinate in linear drop direction

equilibrium_contacts = (source_contact, drain_contact, gate)  //[shared] same as in ramper
equilibrium_regions = [(1), (3), (-1)]   // [shared] should include contacts with no region as ()
contacts_control = (1.0, 1.0, 0.0)       // [optional] positive number. gate contact has 0 influence

Ef_approximation = average  // could be "x_linear_drop", "y_linear_drop", "z_linear_drop" and "average"

dimensionality = 2
fermilevel  = 0.0 // global fermi level, irrelevant here?
temperature = 300
}

solver
{
name = density_solver
type = WF
domain = device
active_regions = (1, 2, 3) // note that the atoms of lead domains are all active

// ---------------
// physics
// ---------------
tb_basis = sp3d5sstar
//tb_basis = sp3sstar

//scattering = true
num_asection = 1

// ---------------
// numerics
// ---------------
```

```
//symmetric = true

//Emin = -0.5 // set automatically according to band edges, doping
//Emax = -0.4
Enum = 574 // default: 2 meV
//Edif = 0.002   // (default: 2meV) initial energy interval size
//use_adaptive_grid = false // adaptive energy grid. default: true
use_adaptive_grid1  = true
adaptive_grid1_criteria = 0.0002 // default: 0.1/Enum
adaptive_grid_low   = 2.0
adaptive_grid_high  = -1.0
adaptive_grid_Fermi = -1.0

Korigin = (0.0, 0.0, 0.5)
Kradius = 0.5
Knum = 7
Kdeg = 2

option2_value = mumps // default: superlu_dist

// ---------------
// other solvers
// ---------------

ramper_name = UTB

// 1. solve semiclassical+poisson self-consistently
// 2. solve quantum transport non-selfconsistently using previous potential (1-step calculation)
self_consistent  = true          // setting this to true will enable quantum self-consistency
Semiclassical    = semiclass_dens
potential_solver = potential_solver
potential_criteria = 0.001 // convergence criterion for norm of potential update vector
add_constant_potential = 0.05
                // add 0.05V (-0.05eV) to the entire energy landscape, in addition to electrostatic potential

// FOR BENCHMARKS ONLY. SIMULATION WILL NOT CONVERGE!
potential_iteration = 2  // max. number of outer Poisson-NEGF iterations

// ---------------
// output
// ---------------

output_name = UTB_@PROCS@_dens
output = (transmission, potential, free_charge_cm-3)
}

   solver
   {
     name = struct_out
     type = Structure
     domain = device
     active_atoms_only = true

output_format  = vtk
structure_file = UTB_@PROCS@.vtk
   }
}
```

```
Global
{
solve = (UTB)

database = ./all.mat

// output is now piped to .log-file
//logfile = UTB_@PROCS@.log
//output =()

messaging_level = 3

    //the following gives a segmentation fault on Jaguar!
//petsc_double_report = true
//petsc_complex_report = true
//disable_fp_trapping = true // floating point exception trapping
}
```

qd.in

```
Structure
{
Material
{
name              = GaAs
tag               = substrate
crystal_structure = zincblende
regions           = (1)
}
Material
{
name              = InAs
tag               = dot
crystal_structure = zincblende
regions           = (2)
}

Domain
{
name          = atomic_structure
type          = pseudomorphic
base_material = substrate
dimension     = (@dim@,@dim@,@dim@)
periodic      = (true,true,false)

passivate              = false  // do not add H atoms
//conformal_partitioning = false  // break unit cells

crystal_direction1 = (1,0,0)
crystal_direction2 = (0,1,0)
crystal_direction3 = (0,0,1)

space_orientation_dir1 = (1, 0, 0)
```

```
space_orientation_dir2 = (0, 1, 0)

origin  = (0,0,0)
output  = ()
regions = (1,2)
}

Geometry
{
Region
{
shape         = cuboid
region_number = 1
priority      = 1
min           = (-1,-1,-1)
max           = (1000,1000,1000)
}
Region
{
shape         = cuboid
region_number = 2
priority      = 2
min           = ( @x1@, @x1@, @x1@)
max           = ( @x2@, @x2@, @x2@)
}

Boundary_region
{
shape         = cuboid
region_number = 1
priority      = 1
min           = ( -1, -1,-1)
max           = ( 0.15, 1000,1000)
}

Partitioning
{
x_extension = (-0.01, @xmax@)
y_extension = (-0.01, @xmax@)
z_extension = (-0.01, @xmax@)
}
}
}

Solvers
{
solver
{
name           = strain
type           = KeatingStrain
domain         = atomic_structure
active_regions = (1,2)

models         = (harmonic)

num_harmonic_iters = 1
hessian_info   = true
```

```
silo_file       = qd_@PROCS@.silo

linear_solver  = gmres // default: preonly; gmres works well for harmonic

// please note; asm preconditioner is not scalable
//preconditioner = none   // solve time becomes much longer for >2000 cores
//preconditioner = asm   // not really scalable
preconditioner = bjacobi
asm_overlap    = 0
max_num_iters  = 20    // default: 200  (#Newton iterations)
absolute_tol   = 1e-6  // default: 1e-8
relative_tol   = 1e-6  // default: 1e-8
linsolver_max_iters = 100000 // default: 1000
linsolver_monitor   = true

// fix scalability problem of AOCreateMapping: do not use it
// (dirty hack that works only when all atoms are active and only for Schroedinger)
assume_all_atom_active = true
}
}

Global
{
solve = (strain)
database = ./all.mat

// output is now piped to logfile
//logfile  =  qd_@PROCS@.log
messaging_level = 5

petsc_double_report = true
petsc_complex_report = true

petsc_memory_reports = true
}
```

Q2 Compilation Script compilation_omen.txt

```
luisier@jaguarpf-login8:/ccs/home/luisier/OMEN_JOULE/OMEN> make -j2 OMEN
CC -c CSR.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK       -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/

CC -c LinkedList.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK  -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
```

```
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used

CC -c Utilities.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK    -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/

CC -c WireGenerator.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used

CC -c Material.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK      -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/

CC -c Arpack.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK       -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used

CC -c PhDynamic.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK    -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
```

```
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/
-I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used

CC -c PhDispersion.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used

CC -c GreenFunction.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK  -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/

CC -c RGF.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK       -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used

CC -c PDIV.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK       -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used

CC -c Phonon.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK     -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
```

```
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/

CC -c ScattGreenFunction.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used

CC -c Scattering.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK     -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/

CC -c PDIVScatt.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK       -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used

CC -c FEMGrid.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK       -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/

CC -c Poisson.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK       -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/
```

```
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used

CC -c Fermi.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK       -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/

CC -c InputParameter.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used

bison -d -b y parser.y

CC -c Macopt.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK       -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/

CC -c AtomStrain.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK    -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used

CC -c GreenSolver.C -fast -tp istanbul-64 -DMPICH_IGNORE_CXX_SEEK   -DAdd_ -DSOLVE_CRAY -DSPIN_COMP
-I/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -I/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/
-I/opt/pgi/default/linux86-64/default//include -I/autofs/na1_home/luisier/OMEN_JOULE//UFconfig/
-I/autofs/na1_home/luisier/OMEN_JOULE//AMD/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Include/
-I/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/SRC
-I/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/include/ -I/include/
-I/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -I/autofs/na1_home/luisier/OMEN_JOULE//JOULE/
```

```
-I/opt/xt-tools/papi/3.6.2/v23/xt-cnl//include/
flex parser.lex

cc -c y.tab.c -O2 -tp istanbul-64
/opt/cray/xt-asyncpe/3.7/bin/cc: INFO: linux target is being used
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used

cc -c lex.yy.c -O2 -tp istanbul-64
/opt/cray/xt-asyncpe/3.7/bin/cc: INFO: linux target is being used

CC -fast -tp istanbul-64 CSR.o LinkedList.o Utilities.o WireGenerator.o Material.o Arpack.o
PhDynamic.o PhDispersion.o GreenFunction.o RGF.o PDIV.o Phonon.o ScattGreenFunction.o Scattering.o
PDIVScatt.o FEMGrid.o Poisson.o Fermi.o InputParameter.o lex.yy.o y.tab.o Macopt.o AtomStrain.o
GreenSolver.o -L/autofs/na1_home/luisier/OMEN_JOULE//QHULL/src/ -lqhull
-L/autofs/na1_home/luisier/OMEN_JOULE//AZTEC/lib/ -laztec
-L/autofs/na1_home/luisier/OMEN_JOULE//ARPACK/ -larpack
-L/autofs/na1_home/luisier/OMEN_JOULE//UMFPACK/Lib/ -lumfpack
-L/autofs/na1_home/luisier/OMEN_JOULE//AMD/Lib/ -lamd
-L/autofs/na1_home/luisier/OMEN_JOULE//SuperLU_DIST_2.0/Lib/ -lsuperlu_dist_2.0
-L/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/lib/ -lzmumps
-L/autofs/na1_home/luisier/OMEN_JOULE//MUMPS/lib/ -lpord
-L/autofs/na1_home/luisier/OMEN_JOULE//Metis/ -lmetis
-L/autofs/na1_home/luisier/OMEN_JOULE//PDIV/ -lpdiv
-L/autofs/na1_home/luisier/OMEN_JOULE//JOULE/ -ljoule
-L/opt/xt-tools/papi/3.6.2/v23/xt-cnl//lib64/ -lpapi -rpath
/opt/xt-tools/papi/3.6.2/v23/xt-cnl//lib64
-L/opt/xt-libsci/default/pgi/lib -lsci_istanbul
-lpgftnrtl -lpgmp -lpgc -lpghpf -lpgf90 -lpgf90rtl -lpghpf -lpghpf_rpm1 -lpgf902 -mp
-L/lib  -lm -o OMEN_Jaguar-pgi64-XT5
/opt/cray/xt-asyncpe/3.7/bin/CC: INFO: linux target is being used

luisier@jaguarpf-login8:/ccs/home/luisier/OMEN_JOULE/OMEN>
```

compilation_nemo5.txt

```
steiger@jaguarpf-login1:~/src/app-nemo/NEMO/prototype$ make
Compiling C++ (in optimized mode) Atom.cpp ...
Compiling C++ (in optimized mode) AtomicStructure.cpp ...
Compiling C++ (in optimized mode) AtomisticDomain.cpp ...
Compiling C++ (in optimized mode) AtomisticFemBase.cpp ...
Compiling C++ (in optimized mode) BCCCrystal.cpp ...
Compiling C++ (in optimized mode) Bi2Te3Crystal.cpp ...
Compiling C++ (in optimized mode) CaesiumChlorideCrystal.cpp ...
Compiling C++ (in optimized mode) Crystal.cpp ...
Compiling C++ (in optimized mode) DiamondCrystal.cpp ...
Compiling C++ (in optimized mode) DisorderBuilder.cpp ...
Compiling C++ (in optimized mode) DOFmap.cpp ...
Compiling C++ (in optimized mode) Domain.cpp ...
Compiling C++ (in optimized mode) FDDomain.cpp ...
Compiling C++ (in optimized mode) FEMdomain.cpp ...
Compiling C++ (in optimized mode) Geometry.cpp ...
Compiling C++ (in optimized mode) GrapheneCrystal.cpp ...
Compiling C++ (in optimized mode) Material.cpp ...
Compiling C++ (in optimized mode) MaterialProperties.cpp ...
Compiling C++ (in optimized mode) MPIVariable.cpp ...
```

```
Compiling C++ (in optimized mode) Nemo.cpp ...
Compiling C++ (in optimized mode) NemoMesh.cpp ...
Compiling C++ (in optimized mode) NemoUtils.cpp ...
Compiling C++ (in optimized mode) PseudomorphicDomain.cpp ...
Compiling C++ (in optimized mode) RhombohedralCrystal.cpp ...
Compiling C++ (in optimized mode) SimpleCubicCrystal.cpp ...
Compiling C++ (in optimized mode) SimpleShapes.cpp ...
Compiling C++ (in optimized mode) Simulation.cpp ...
Compiling C++ (in optimized mode) Tetrahedralizer.cpp ...
Compiling C++ (in optimized mode) TotallyRandomAlloy.cpp ...
Compiling C++ (in optimized mode) TotallyRandomDopant.cpp ...
Compiling C++ (in optimized mode) WurtziteCrystal.cpp ...
Compiling C++ (in optimized mode) ZincBlendeCrystal.cpp ...
Compiling C++ (in optimized mode) InputOptions.cpp ...
Compiling C++ (in optimized mode) InputVTK.cpp ...
In file included from /opt/gcc/4.4.2/snos/include/g++/backward/strstream:46,
                 from /ccs/home/steiger/src/app-nemo/NEMO/libs/vtk/include/vtk-5.6/vtkIOStream.h:112,
                 from /ccs/home/steiger/src/app-nemo/NEMO/libs/vtk/include/vtk-5.6/vtkSystemIncludes.h:40,
                 from /ccs/home/steiger/src/app-nemo/NEMO/libs/vtk/include/vtk-5.6/vtkIndent.h:24,
                 from /ccs/home/steiger/src/app-nemo/NEMO/libs/vtk/include/vtk-5.6/vtkObjectBase.h:43,
                 from /ccs/home/steiger/src/app-nemo/NEMO/libs/vtk/include/vtk-5.6/vtkObject.h:41,
                 from /ccs/home/steiger/src/app-nemo/NEMO/libs/vtk/include/vtk-5.6/vtkCellArray.h:35,
                 from InputVTK.cpp:31:
/opt/gcc/4.4.2/snos/include/g++/backward/backward_warning.h:28:2: warning: #warning This file includes at
least one deprecated or antiquated header which may be removed without further notice at a future date.
Please use a non-deprecated interface with equivalent functionality instead. For a listing of replacement
headers and interfaces, consult the file backward_warning.h. To disable this warning use -Wno-deprecated.
Compiling C++ (in optimized mode) NemoOutput.cpp ...
Compiling C++ (in optimized mode) OutputDX.cpp ...
Compiling C++ (in optimized mode) OutputPDB.cpp ...
Compiling C++ (in optimized mode) OutputSilo.cpp ...
In file included from OutputSilo.cpp:74:
/sw/xt5/silo/4.7/cnl2.1_gnu4.2.0/include/pmpio.h:
In function 'void* PMPIO_WaitForBaton(PMPIO_baton_t*, const char*, const char*)':
/sw/xt5/silo/4.7/cnl2.1_gnu4.2.0/include/pmpio.h:336:
warning: comparison between signed and unsigned integer expressions
/sw/xt5/silo/4.7/cnl2.1_gnu4.2.0/include/pmpio.h: At global scope:
/sw/xt5/silo/4.7/cnl2.1_gnu4.2.0/include/pmpio.h:448:
warning: 'void* PMPIO_DefaultCreate(const char*, const char*, void*)'
defined but not used
/sw/xt5/silo/4.7/cnl2.1_gnu4.2.0/include/pmpio.h:470: warning: 'void* PMPIO_DefaultOpen(const char*,
const char*, PMPIO_iomode_t, void*)' defined but not used
/sw/xt5/silo/4.7/cnl2.1_gnu4.2.0/include/pmpio.h:490:
warning: 'void PMPIO_DefaultClose(void*, void*)' defined but not used
Compiling C++ (in optimized mode) OutputVTK.cpp ...
In file included from /opt/gcc/4.4.2/snos/include/g++/backward/strstream:46,
                 from /ccs/home/steiger/src/app-nemo/NEMO/libs/vtk/include/vtk-5.6/vtkIOStream.h:112,
                 from /ccs/home/steiger/src/app-nemo/NEMO/libs/vtk/include/vtk-5.6/vtkSystemIncludes.h:40,
                 from /ccs/home/steiger/src/app-nemo/NEMO/libs/vtk/include/vtk-5.6/vtkIndent.h:24,
                 from /ccs/home/steiger/src/app-nemo/NEMO/libs/vtk/include/vtk-5.6/vtkObjectBase.h:43,
                 from /ccs/home/steiger/src/app-nemo/NEMO/libs/vtk/include/vtk-5.6/vtkObject.h:41,
                 from /ccs/home/steiger/src/app-nemo/NEMO/libs/vtk/include/vtk-5.6/vtkPoints.h:23,
                 from OutputVTK.cpp:33:
/opt/gcc/4.4.2/snos/include/g++/backward/backward_warning.h:28:2: warning: #warning This file includes at
least one deprecated or antiquated header which may be removed without further notice at a future date.
Please use a non-deprecated interface with equivalent functionality instead. For a listing of replacement
headers and interfaces, consult the file backward_warning.h. To disable this warning use -Wno-deprecated.
```

```
Compiling C++ (in optimized mode) Parser.cpp ...
Compiling C++ (in optimized mode) EigensolverSlepcDouble.cpp ...
EigensolverSlepcDouble.cpp: In member function 'void EigensolverSlepc<T>::solve() [with T = double]':
EigensolverSlepcDouble.cpp:209: warning: unused variable 'ncv'
Compiling C++ (in optimized mode) LinearSolverPetscDouble.cpp ...
Compiling C++ (in optimized mode) NonlinearPoissonPetscDouble.cpp ...
Compiling C++ (in optimized mode) Partitioner.cpp ...
Compiling C++ (in optimized mode) PetscAODouble.cpp ...
Compiling C++ (in optimized mode) PetscGhostVectorDouble.cpp ...
Compiling C++ (in optimized mode) PetscMatrixBlockDouble.cpp ...
Compiling C++ (in optimized mode) PetscMatrixDouble.cpp ...
Compiling C++ (in optimized mode) PetscMatrixParallelDouble.cpp ...
PetscMatrixParallelDouble.cpp:
In member function 'bool PetscMatrixParallel<T>::matrix_assembled() [with T = double]':
PetscMatrixParallelDouble.cpp:222: warning: 'assembled' is used uninitialized in this function
Compiling C++ (in optimized mode) PetscMatrixSequentialDouble.cpp ...
Compiling C++ (in optimized mode) PetscNewtonSolverDouble.cpp ...
Compiling C++ (in optimized mode) PetscVectorNemoDouble.cpp ...
CC -dynamic -shared -o /ccs/home/steiger/src/app-nemo/NEMO/prototype/lib/libnemo_petsc_double.so
-Wl,--version-script=versioning.txt
EigensolverSlepcDouble.opt.o  LinearSolverPetscDouble.opt.o  NonlinearPoissonPetscDouble.opt.o
Partitioner.opt.o  PetscAODouble.opt.o PetscGhostVectorDouble.opt.o  PetscMatrixBlockDouble.opt.o
PetscMatrixDouble.opt.o  PetscMatrixParallelDouble.opt.o
PetscMatrixSequentialDouble.opt.o  PetscNewtonSolverDouble.opt.o  PetscVectorNemoDouble.opt.o \
  -Wl,-Bstatic -L/ccs/home/steiger/src/app-nemo/NEMO/libs/slepc/linux/lib -lslepc
-L../../../../libs/ARPACK -lparpack -larpack \
  -Wl,-Bstatic -L/ccs/home/steiger/src/app-nemo/NEMO/libs/petsc/linux/lib -lpetscts \
-lpetscksp -lpetscdm -lpetscmat -lpetscvec -lpetscsnes -lpetscsys -lsuperlu_dist -lHYPRE \
-lzmumps -ldmumps -lmumps_common -lscalapack -lblacs -lpord -lparmetis -lmetis  \
  -Wl,-Bdynamic  -Wl,-rpath,/usr/X11R6/lib64/ -L/usr/X11R6/lib64 -lX11
ar ruv /ccs/home/steiger/src/app-nemo/NEMO/prototype/lib/libnemo_petsc_double.a  EigensolverSlepcDouble.opt.o
LinearSolverPetscDouble.opt.o  NonlinearPoissonPetscDouble.opt.o  Partitioner.opt.o  PetscAODouble.opt.o
PetscGhostVectorDouble.opt.o  PetscMatrixBlockDouble.opt.o  PetscMatrixDouble.opt.o  PetscMatrixParallelDouble.opt.o
PetscMatrixSequentialDouble.opt.o  PetscNewtonSolverDouble.opt.o  PetscVectorNemoDouble.opt.o
r - EigensolverSlepcDouble.opt.o
r - LinearSolverPetscDouble.opt.o
r - NonlinearPoissonPetscDouble.opt.o
r - Partitioner.opt.o
r - PetscAODouble.opt.o
r - PetscGhostVectorDouble.opt.o
r - PetscMatrixBlockDouble.opt.o
r - PetscMatrixDouble.opt.o
r - PetscMatrixParallelDouble.opt.o
r - PetscMatrixSequentialDouble.opt.o
r - PetscNewtonSolverDouble.opt.o
r - PetscVectorNemoDouble.opt.o
ranlib /ccs/home/steiger/src/app-nemo/NEMO/prototype/lib/libnemo_petsc_double.a
Compiling C++ (in optimized mode) EigensolverSlepcComplex.cpp ...
Compiling C++ (in optimized mode) LinearSolverPetscComplex.cpp ...
Compiling C++ (in optimized mode) PetscMatrixBlockComplex.cpp ...
Compiling C++ (in optimized mode) PetscMatrixComplex.cpp ...
Compiling C++ (in optimized mode) PetscMatrixParallelComplex.cpp ...
PetscMatrixParallelComplex.cpp:
In member function 'void PetscMatrixParallel<T>::get_diagonal(std::vector<std::complex<double>,
std::allocator<std::complex<double> > >*) [with T = std::complex<double>]':
PetscMatrixParallelComplex.cpp:391: warning: comparison between signed and unsigned integer expressions
Compiling C++ (in optimized mode) PetscMatrixSequentialComplex.cpp ...
```

```
Compiling C++ (in optimized mode) PetscNewtonSolverComplex.cpp ...
Compiling C++ (in optimized mode) PetscVectorNemoComplex.cpp ...
CC -dynamic -shared -o /ccs/home/steiger/src/app-nemo/NEMO/prototype/lib/libnemo_petsc_complex.so
-Wl,--version-script=versioning.txt
EigensolverSlepcComplex.opt.o  LinearSolverPetscComplex.opt.o  PetscMatrixBlockComplex.opt.o
PetscMatrixComplex.opt.o  PetscMatrixParallelComplex.opt.o  PetscMatrixSequentialComplex.opt.o
PetscNewtonSolverComplex.opt.o  PetscVectorNemoComplex.opt.o \
  -Wl,-Bstatic -L/ccs/home/steiger/src/app-nemo/NEMO/libs/slepc/linux-complex/lib -lslepc
-L../../../../libs/ARPACK -lparpack -larpack \
  -Wl,-Bstatic -L/ccs/home/steiger/src/app-nemo/NEMO/libs/petsc/linux-complex/lib -lpetscts
-lpetscksp -lpetscdm -lpetscmat -lpetscvec -lpetscsnes -lpetscsys -lsuperlu_dist -lzmumps -ldmumps
-lmumps_common -lscalapack -lblacs -lpord -lparmetis -lmetis \
  -Wl,-Bdynamic -Wl,-rpath,/usr/X11R6/lib64/ -L/usr/X11R6/lib64 -lX11
ar ruv /ccs/home/steiger/src/app-nemo/NEMO/prototype/lib/libnemo_petsc_complex.a  EigensolverSlepcComplex.opt.o
LinearSolverPetscComplex.opt.o  PetscMatrixBlockComplex.opt.o  PetscMatrixComplex.opt.o
PetscMatrixParallelComplex.opt.o  PetscMatrixSequentialComplex.opt.o  PetscNewtonSolverComplex.opt.o
PetscVectorNemoComplex.opt.o
r - EigensolverSlepcComplex.opt.o
r - LinearSolverPetscComplex.opt.o
r - PetscMatrixBlockComplex.opt.o
r - PetscMatrixComplex.opt.o
r - PetscMatrixParallelComplex.opt.o
r - PetscMatrixSequentialComplex.opt.o
r - PetscNewtonSolverComplex.opt.o
r - PetscVectorNemoComplex.opt.o
ranlib /ccs/home/steiger/src/app-nemo/NEMO/prototype/lib/libnemo_petsc_complex.a
Compiling C++ (in optimized mode) BandedMatrix.cpp ...
Compiling C++ (in optimized mode) DenseMatrixNemo.cpp ...
Compiling C++ (in optimized mode) EigensolverArpack.cpp ...
Compiling C++ (in optimized mode) Eigensolver.cpp ...
Compiling C++ (in optimized mode) EigensolverLapack.cpp ...
Compiling C++ (in optimized mode) Integrator.cpp ...
Compiling C++ (in optimized mode) LinearSolver.cpp ...
Compiling C++ (in optimized mode) LinearSolverLapack.cpp ...
Compiling C++ (in optimized mode) LinearSolverUmfpack.cpp ...
Compiling C++ (in optimized mode) Matrix.cpp ...
Compiling C++ (in optimized mode) NemoMath.cpp ...
Compiling C++ (in optimized mode) PetscMatrix.cpp ...
Compiling C++ (in optimized mode) PetscMatrixParallel.cpp ...
Compiling C++ (in optimized mode) PetscNewtonSolver.cpp ...
Compiling C++ (in optimized mode) SparseMatrix.cpp ...
Compiling C++ (in optimized mode) Absorption.cpp ...
Compiling C++ (in optimized mode) Espace.cpp ...
Compiling C++ (in optimized mode) Kspace.cpp ...
Compiling C++ (in optimized mode) MatrixElementCalculator.cpp ...
Compiling C++ (in optimized mode) NemoPhys.cpp ...
Compiling C++ (in optimized mode) Symmetry2D.cpp ...
Compiling C++ (in optimized mode) UtilityFunctionsSymmetry2D.cpp ...
Compiling C++ (in optimized mode) WallPaper_Group.cpp ...
Compiling C++ (in optimized mode) WallPaper_Groups_DB.cpp ...
Compiling C++ (in optimized mode) Brillouin.cpp ...
Compiling C++ (in optimized mode) PhononParameters.cpp ...
Compiling C++ (in optimized mode) Phonons.cpp ...
Compiling C++ (in optimized mode) Structure.cpp ...
Compiling C++ (in optimized mode) StrainParameters.cpp ...
Compiling C++ (in optimized mode) StrainVFF.cpp ...
Compiling C++ (in optimized mode) VFFAllModels.cpp ...
```

```
Compiling C++ (in optimized mode) VFFAnharmonic.cpp ...
Compiling C++ (in optimized mode) VFFCoulomb.cpp ...
VFFCoulomb.cpp: In member function 'void VFFAllModels::add_coulomb(double*)':
VFFCoulomb.cpp:827: warning: unused variable 'atom_j_dofs'
VFFCoulomb.cpp:863: warning: unused variable 'atom_j_dofs'
VFFCoulomb.cpp: In static member function
'static void VFFAllModels::first_derivatives_el2d1(std::vector<double, std::allocator<double> >&,
std::vector<double, std::allocator<double> >&, std::vector<double, std::allocator<double> >&,
std::vector<double, std::allocator<double> >&,
double, double, double, double, std::vector<std::vector<double, std::allocator<double> >,
std::allocator<std::vector<double, std::allocator<double> > > >*)':
VFFCoulomb.cpp:938: warning: unused variable 'l0s2'
VFFCoulomb.cpp:940: warning: unused variable 'l0s4'
VFFCoulomb.cpp:942: warning: unused variable 'l0s6'
VFFCoulomb.cpp:944: warning: unused variable 'l1s2'
VFFCoulomb.cpp:945: warning: unused variable 'l1s3'
VFFCoulomb.cpp:946: warning: unused variable 'l1s4'
VFFCoulomb.cpp:950: warning: unused variable 'l3s1'
VFFCoulomb.cpp:967: warning: unused variable 'l9s3'
VFFCoulomb.cpp:968: warning: unused variable 'l9s4'
VFFCoulomb.cpp:976: warning: unused variable 'l11s3'
VFFCoulomb.cpp: In static member function
'static void VFFAllModels::first_derivatives_el2d2(std::vector<double, std::allocator<double> >&,
std::vector<double, std::allocator<double> >&, std::vector<double, std::allocator<double> >&,
double, double, double, std::vector<std::vector<double, std::allocator<double> >,
std::allocator<std::vector<double, std::allocator<double> > > >*)':
VFFCoulomb.cpp:1036: warning: unused variable 'l1s1'
VFFCoulomb.cpp:1038: warning: unused variable 'l1s3'
VFFCoulomb.cpp:1040: warning: unused variable 'l1s5'
VFFCoulomb.cpp: In static member function
'static void VFFAllModels::second_derivatives_el2d1(std::vector<double, std::allocator<double> >&,
std::vector<double, std::allocator<double> >&, std::vector<double, std::allocator<double> >&,
std::vector<double, std::allocator<double> >&, double, double, double, double,
std::vector<std::vector<std::vector<std::vector<double, std::allocator<double> >,
std::allocator<std::vector<double, std::allocator<double> > > >,
std::allocator<std::vector<std::vector<double, std::allocator<double> >,
std::allocator<std::vector<double, std::allocator<double> > > > > >,
std::allocator<std::vector<std::vector<std::vector<double, std::allocator<double> >,
std::allocator<std::vector<double, std::allocator<double> > > >,
std::allocator<std::vector<std::vector<double, std::allocator<double> >,
std::allocator<std::vector<double, std::allocator<double> > > > > > >*)':
VFFCoulomb.cpp:1101: warning: unused variable 'l0s2'
VFFCoulomb.cpp:1103: warning: unused variable 'l0s4'
VFFCoulomb.cpp:1105: warning: unused variable 'l0s6'
VFFCoulomb.cpp:1109: warning: unused variable 'l2s1'
VFFCoulomb.cpp:1120: warning: unused variable 'l6s1'
VFFCoulomb.cpp:1151: warning: unused variable 'l15s1'
VFFCoulomb.cpp:1152: warning: unused variable 'l15s2'
VFFCoulomb.cpp:1153: warning: unused variable 'l15s3'
VFFCoulomb.cpp:1154: warning: unused variable 'l15s4'
VFFCoulomb.cpp:1159: warning: unused variable 'l15s9'
VFFCoulomb.cpp:1160: warning: unused variable 'l15s10'
VFFCoulomb.cpp:1161: warning: unused variable 'l15s11'
VFFCoulomb.cpp:1162: warning: unused variable 'l15s12'
VFFCoulomb.cpp:1165: warning: unused variable 'l15s15'
VFFCoulomb.cpp:1166: warning: unused variable 'l15s16'
VFFCoulomb.cpp:1167: warning: unused variable 'l15s17'
```

```
VFFCoulomb.cpp:1168: warning: unused variable 'l15s18'
VFFCoulomb.cpp:1169: warning: unused variable 'l15s19'
VFFCoulomb.cpp:1170: warning: unused variable 'l15s20'
VFFCoulomb.cpp:1173: warning: unused variable 'l15s23'
VFFCoulomb.cpp:1174: warning: unused variable 'l15s24'
VFFCoulomb.cpp:1180: warning: unused variable 'l16s6'
VFFCoulomb.cpp:1182: warning: unused variable 'l16s8'
VFFCoulomb.cpp:1186: warning: unused variable 'l17s2'
VFFCoulomb.cpp:1187: warning: unused variable 'l17s3'
VFFCoulomb.cpp:1188: warning: unused variable 'l17s4'
VFFCoulomb.cpp:1189: warning: unused variable 'l17s5'
VFFCoulomb.cpp:1190: warning: unused variable 'l17s6'
VFFCoulomb.cpp:1191: warning: unused variable 'l17s7'
VFFCoulomb.cpp:1192: warning: unused variable 'l17s8'
VFFCoulomb.cpp:1193: warning: unused variable 'l17s9'
VFFCoulomb.cpp:1194: warning: unused variable 'l17s10'
VFFCoulomb.cpp:1195: warning: unused variable 'l17s11'
VFFCoulomb.cpp:1196: warning: unused variable 'l17s12'
VFFCoulomb.cpp:1197: warning: unused variable 'l17s13'
VFFCoulomb.cpp:1199: warning: unused variable 'l17s15'
VFFCoulomb.cpp:1200: warning: unused variable 'l17s16'
VFFCoulomb.cpp:1201: warning: unused variable 'l17s17'
VFFCoulomb.cpp:1203: warning: unused variable 'l17s19'
VFFCoulomb.cpp:1204: warning: unused variable 'l17s20'
VFFCoulomb.cpp:1207: warning: unused variable 'l17s23'
VFFCoulomb.cpp:1208: warning: unused variable 'l17s24'
VFFCoulomb.cpp:1209: warning: unused variable 'l17s25'
VFFCoulomb.cpp:1210: warning: unused variable 'l17s26'
VFFCoulomb.cpp:1213: warning: unused variable 'l17s29'
VFFCoulomb.cpp:1214: warning: unused variable 'l17s30'
VFFCoulomb.cpp:1216: warning: unused variable 'l17s32'
VFFCoulomb.cpp:1226: warning: unused variable 'l18s7'
VFFCoulomb.cpp:1227: warning: unused variable 'l18s8'
VFFCoulomb.cpp:1235: warning: unused variable 'l19s8'
VFFCoulomb.cpp:1236: warning: unused variable 'l19s9'
VFFCoulomb.cpp:1240: warning: unused variable 'l20s4'
VFFCoulomb.cpp:1241: warning: unused variable 'l20s5'
VFFCoulomb.cpp:1247: warning: unused variable 'l22s2'
VFFCoulomb.cpp:1262: warning: unused variable 'l34s2'
VFFCoulomb.cpp: In static member function
'static void VFFAllModels::second_derivatives_el2d2(std::vector<double, std::allocator<double> >&,
std::vector<double, std::allocator<double> >&, std::vector<double, std::allocator<double> >&,
double, double, double, std::vector<std::vector<std::vector<std::vector<double,
std::allocator<double> >, std::allocator<std::vector<double, std::allocator<double> > > >,
std::allocator<std::vector<std::vector<double, std::allocator<double> >,
std::allocator<std::vector<double, std::allocator<double> > > > > >,
std::allocator<std::vector<std::vector<std::vector<double, std::allocator<double> >,
std::allocator<std::vector<double, std::allocator<double> > > >,
std::allocator<std::vector<std::vector<double, std::allocator<double> >,
std::allocator<std::vector<double, std::allocator<double> > > > > > >*)':
VFFCoulomb.cpp:1345: warning: unused variable 'l0s2'
VFFCoulomb.cpp:1347: warning: unused variable 'l0s4'
VFFCoulomb.cpp:1349: warning: unused variable 'l0s6'
VFFCoulomb.cpp:1354: warning: unused variable 'l2s2'
VFFCoulomb.cpp:1358: warning: unused variable 'l4s2'
VFFCoulomb.cpp:1367: warning: unused variable 'l10s2'
Compiling C++ (in optimized mode) MatrixElements.cpp ...
```

```
make[3]: Nothing to be done for ‘all’.
Compiling C++ (in optimized mode) Selfconsistent.cpp ...
Compiling C++ (in optimized mode) DielectricProperties.cpp ...
Compiling C++ (in optimized mode) Doping.cpp ...
Compiling C++ (in optimized mode) DopingRandom.cpp ...
Compiling C++ (in optimized mode) DopingVCA.cpp ...
Compiling C++ (in optimized mode) ElectrostaticContact.cpp ...
Compiling C++ (in optimized mode) NonlinearPoisson.cpp ...
NonlinearPoisson.cpp: In member function ’void NonlinearPoisson::execute_solver()’:
NonlinearPoisson.cpp:198: warning: unused variable ’do_set_initial_potential’
Compiling C++ (in optimized mode) Poisson.cpp ...
Compiling C++ (in optimized mode) PoissonEquationInterface.cpp ...
Compiling C++ (in optimized mode) PoissonMaterialProperties.cpp ...
Compiling C++ (in optimized mode) PotentialFromSolver.cpp ...
Compiling C++ (in optimized mode) EMHamiltonConstructor.cpp ...
Compiling C++ (in optimized mode) HamiltonConstructor.cpp ...
Compiling C++ (in optimized mode) Schroedinger.cpp ...
Compiling C++ (in optimized mode) TBHamiltonConstructor.cpp ...
Compiling C++ (in optimized mode) TBSchroedinger.cpp ...
Compiling C++ (in optimized mode) Contacts.cpp ...
Compiling C++ (in optimized mode) NEGF.cpp ...
NEGF.cpp: In member function ’void NEGF::solve_recursive_Green_function(int, int)’:
NEGF.cpp:304: warning: comparison between signed and unsigned integer expressions
NEGF.cpp:567: warning: comparison between signed and unsigned integer expressions
Compiling C++ (in optimized mode) Transport.cpp ...
Transport.cpp: In member function ’virtual void Transport::do_solve()’:
Transport.cpp:389: warning: unused variable ’tstart’
Transport.cpp: In member function
’virtual void Transport::get_data(const std::string&,
std::vector<unsigned int, std::allocator<unsigned int> >&)’:
Transport.cpp:2333: warning: comparison between signed and unsigned integer expressions
Transport.cpp:2337: warning: comparison between signed and unsigned integer expressions
Compiling C++ (in optimized mode) WF.cpp ...
Compiling C++ (in optimized mode) ContactFermilevel.cpp ...
Compiling C++ (in optimized mode) Current.cpp ...
Compiling C++ (in optimized mode) GreenFunctions.cpp ...
Compiling C++ (in optimized mode) Hamiltonian.cpp ...
Compiling C++ (in optimized mode) InnerLoop.cpp ...
Compiling C++ (in optimized mode) InterfaceEffMass.cpp ...
Compiling C++ (in optimized mode) InterfaceEffMassOrtho2Band.cpp ...
Compiling C++ (in optimized mode) InterfaceEffMassOrtho.cpp ...
Compiling C++ (in optimized mode) NEGFObject.cpp ...
Compiling C++ (in optimized mode) OuterLoop.cpp ...
Compiling C++ (in optimized mode) PostProcessing.cpp ...
Compiling C++ (in optimized mode) SEContacts.cpp ...
Compiling C++ (in optimized mode) SelfEnergies.cpp ...
Compiling C++ (in optimized mode) SelfEnergy.cpp ...
Compiling C++ (in optimized mode) TdkpInfoDesk.cpp ...
Compiling C++ (in optimized mode) PoissonFEM1D.cpp ...
Compiling C++ (in optimized mode) PoissonProblem.cpp ...
PoissonProblem.cpp: In member function ’void angel::PoissonProblem::solve_one_step()’:
PoissonProblem.cpp:276: warning: unused variable ’max_update’
Compiling C++ (in optimized mode) StrainPolarization.cpp ...
Compiling C++ (in optimized mode) Angel.cpp ...
Compiling C++ (in optimized mode) AngelEspace.cpp ...
Compiling C++ (in optimized mode) AngelHeaders.cpp ...
Compiling C++ (in optimized mode) AngelKspace.cpp ...
```

```
Compiling C++ (in optimized mode) AngelMatDB.cpp ...
Compiling C++ (in optimized mode) AngelMPI.cpp ...
Compiling C++ (in optimized mode) AngelOutput.cpp ...
Compiling C++ (in optimized mode) Filenames.cpp ...
Compiling C++ (in optimized mode) Overlap.cpp ...
Compiling C++ (in optimized mode) Ramper.cpp ...
Compiling C++ (in optimized mode) ResonanceFinder.cpp ...
Compiling C++ (in optimized mode) Semiclassical.cpp ...
Compiling C++ (in optimized mode) SemiclassicalParams.cpp ...
Compiling C++ (in optimized mode) Greensolver.cpp ...
Compiling C++ (in optimized mode) Propagation.cpp ...
Compiling C++ (in optimized mode) Self_energy.cpp ...
Self_energy.cpp: In member function
'virtual void Self_energy::direct_iterative_leads(const Domain*, const std::vector<NemoMeshPoint,
std::allocator<NemoMeshPoint> >&, PetscMatrixParallel<std::complex<double> >*&)':
Self_energy.cpp:402: warning: comparison between signed and unsigned integer expressions
Self_energy.cpp:407: warning: comparison between signed and unsigned integer expressions
Self_energy.cpp:348: warning: unused variable 'atomistic_neighbor_domain'
Self_energy.cpp:349: warning: unused variable 'domain'
Self_energy.cpp:351: warning: unused variable 'this_domain_DOFmap'
Self_energy.cpp:364: warning: unused variable 'neighbor_domain_DOFmap'
[some error messages about not being able to link the dynamic library]

steiger@jaguarpf-login1:~/src/app-nemo/NEMO/prototype$ make static
Building special PETSc library with renamed symbols...
(cd src/math/libnemo_petsc_complex; ./trick.sh)
Extracting libpetsc.a...
    adding to libtmp.a...
Extracting libpetscdm.a...
    adding to libtmp.a...
Extracting libpetscksp.a...
    adding to libtmp.a...
Extracting libpetscmat.a...
    adding to libtmp.a...
Extracting libpetscsnes.a...
    adding to libtmp.a...
Extracting libpetscsys.a...
    adding to libtmp.a...
Extracting libpetscts.a...
    adding to libtmp.a...
Extracting libpetscvec.a...
    adding to libtmp.a...
Extracting libslepc.a...
    adding to libtmp.a...
Creating substitution file (--> sub.txt)...
Eliminating duplicates (--> sub2.txt)...
Replacement rules (--> sub3.txt)...
PERFORM THE REPLACEMENT...
objcopy --redefine-syms=sub3.txt libtmp.a
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/libnemo_petsc_complex/../../../../
prototype/lib/libPETScSLEPcCPLX.a
objcopy --redefine-syms=sub3.txt
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/libnemo_petsc_complex/../../../../prototype/
lib/libnemo_petsc_complex.a
Building static library...
ar ruv lib/libnemo.a /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/physics/Absorption.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/physics/Espace.opt.o
```

```
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/physics/Kspace.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/physics/MatrixElementCalculator.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/physics/NemoPhys.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/physics/Symmetry2D.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/physics/UtilityFunctionsSymmetry2D.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/physics/WallPaper_Group.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/physics/WallPaper_Groups_DB.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/io/InputOptions.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/io/InputVTK.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/io/NemoOutput.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/io/OutputDX.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/io/OutputPDB.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/io/OutputSilo.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/io/OutputVTK.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/io/Parser.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Brillouin/Brillouin.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Phonons/PhononParameters.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Phonons/Phonons.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Structure/Structure.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/strain/StrainParameters.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/strain/StrainVFF.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/strain/VFFAllModels.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/strain/VFFAnharmonic.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/strain/VFFCoulomb.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/MatrixElements/MatrixElements.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Selfconsistent/Selfconsistent.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Poisson/DielectricProperties.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Poisson/Doping.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Poisson/DopingRandom.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Poisson/DopingVCA.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Poisson/ElectrostaticContact.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Poisson/NonlinearPoisson.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Poisson/Poisson.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Poisson/PoissonEquationInterface.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Poisson/PoissonMaterialProperties.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Poisson/PotentialFromSolver.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Schroedinger/EMHamiltonConstructor.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Schroedinger/HamiltonConstructor.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Schroedinger/Schroedinger.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Schroedinger/TBHamiltonConstructor.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Schroedinger/TBSchroedinger.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Contacts/Contacts.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Transport/NEGF.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Transport/Transport.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Transport/WF.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/ContactFermilevel.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/Current.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/GreenFunctions.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/Hamiltonian.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/InnerLoop.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/InterfaceEffMass.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/InterfaceEffMassOrtho2Band.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/InterfaceEffMassOrtho.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/NEGFObject.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/OuterLoop.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/PostProcessing.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/SEContacts.opt.o
```

```
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/SelfEnergies.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/SelfEnergy.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/TdkpInfoDesk.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/poisson/PoissonFEM1D.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/poisson/PoissonProblem.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/poisson/StrainPolarization.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/Angel.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/AngelEspace.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/AngelHeaders.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/AngelKspace.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/AngelMatDB.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/AngelMPI.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/AngelOutput.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/Filenames.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/Overlap.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Ramper/Ramper.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/ResonanceFinder/ResonanceFinder.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Semiclassical/Semiclassical.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Semiclassical/SemiclassicalParams.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Propagation/Greensolver.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Propagation/Propagation.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Propagation/Self_energy.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/Atom.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/AtomicStructure.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/AtomisticDomain.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/AtomisticFemBase.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/BCCCrystal.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/Bi2Te3Crystal.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/CaesiumChlorideCrystal.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/Crystal.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/DiamondCrystal.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/DisorderBuilder.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/DOFmap.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/Domain.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/FDDomain.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/FEMdomain.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/Geometry.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/GrapheneCrystal.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/Material.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/MaterialProperties.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/MPIVariable.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/Nemo.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/NemoMesh.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/NemoUtils.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/PseudomorphicDomain.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/RhombohedralCrystal.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/SimpleCubicCrystal.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/SimpleShapes.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/Simulation.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/Tetrahedralizer.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/TotallyRandomAlloy.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/TotallyRandomDopant.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/WurtziteCrystal.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/ZincBlendeCrystal.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/BandedMatrix.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/DenseMatrixNemo.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/EigensolverArpack.opt.o
```

```
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/Eigensolver.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/EigensolverLapack.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/Integrator.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/LinearSolver.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/LinearSolverLapack.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/LinearSolverUmfpack.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/Matrix.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/NemoMath.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/PetscMatrix.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/PetscMatrixParallel.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/PetscNewtonSolver.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/SparseMatrix.opt.o
/ccs/home/steiger/src/app-nemo/NEMO/prototype/src/nemo/main.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/physics/Absorption.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/physics/Espace.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/physics/Kspace.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/physics/MatrixElementCalculator.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/physics/NemoPhys.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/physics/Symmetry2D.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/physics/UtilityFunctionsSymmetry2D.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/physics/WallPaper_Group.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/physics/WallPaper_Groups_DB.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/io/InputOptions.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/io/InputVTK.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/io/NemoOutput.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/io/OutputDX.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/io/OutputPDB.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/io/OutputSilo.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/io/OutputVTK.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/io/Parser.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Brillouin/Brillouin.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Phonons/PhononParameters.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Phonons/Phonons.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Structure/Structure.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/strain/StrainParameters.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/strain/StrainVFF.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/strain/VFFAllModels.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/strain/VFFAnharmonic.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/strain/VFFCoulomb.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/MatrixElements/MatrixElements.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Selfconsistent/Selfconsistent.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Poisson/DielectricProperties.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Poisson/Doping.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Poisson/DopingRandom.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Poisson/DopingVCA.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Poisson/ElectrostaticContact.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Poisson/NonlinearPoisson.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Poisson/Poisson.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Poisson/PoissonEquationInterface.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Poisson/PoissonMaterialProperties.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Poisson/PotentialFromSolver.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Schroedinger/EMHamiltonConstructor.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Schroedinger/HamiltonConstructor.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Schroedinger/Schroedinger.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Schroedinger/TBHamiltonConstructor.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Schroedinger/TBSchroedinger.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Contacts/Contacts.opt.o
```

```
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Transport/NEGF.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Transport/Transport.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Transport/WF.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/ContactFermilevel.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/Current.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/GreenFunctions.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/Hamiltonian.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/InnerLoop.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/InterfaceEffMass.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/InterfaceEffMassOrtho2Band.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/InterfaceEffMassOrtho.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/NEGFObject.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/OuterLoop.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/PostProcessing.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/SEContacts.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/SelfEnergies.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/SelfEnergy.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/negf/TdkpInfoDesk.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/poisson/PoissonFEM1D.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/poisson/PoissonProblem.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/poisson/StrainPolarization.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/Angel.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/AngelEspace.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/AngelHeaders.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/AngelKspace.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/AngelMatDB.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/AngelMPI.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/AngelOutput.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/Filenames.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Angel/Overlap.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Ramper/Ramper.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/ResonanceFinder/ResonanceFinder.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Semiclassical/Semiclassical.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Semiclassical/SemiclassicalParams.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Propagation/Greensolver.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Propagation/Propagation.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/simulations/Propagation/Self_energy.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/Atom.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/AtomicStructure.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/AtomisticDomain.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/AtomisticFemBase.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/BCCCrystal.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/Bi2Te3Crystal.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/CaesiumChlorideCrystal.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/Crystal.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/DiamondCrystal.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/DisorderBuilder.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/DOFmap.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/Domain.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/FDDomain.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/FEMdomain.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/Geometry.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/GrapheneCrystal.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/Material.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/MaterialProperties.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/MPIVariable.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/Nemo.opt.o
```

```
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/NemoMesh.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/NemoUtils.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/PseudomorphicDomain.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/RhombohedralCrystal.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/SimpleCubicCrystal.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/SimpleShapes.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/Simulation.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/Tetrahedralizer.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/TotallyRandomAlloy.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/TotallyRandomDopant.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/WurtziteCrystal.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/common/ZincBlendeCrystal.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/BandedMatrix.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/DenseMatrixNemo.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/EigensolverArpack.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/Eigensolver.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/EigensolverLapack.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/Integrator.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/LinearSolver.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/LinearSolverLapack.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/LinearSolverUmfpack.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/Matrix.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/NemoMath.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/PetscMatrix.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/PetscMatrixParallel.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/PetscNewtonSolver.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/math/SparseMatrix.opt.o
r - /ccs/home/steiger/src/app-nemo/NEMO/prototype/src/nemo/main.opt.o
Building statically linked executable...
make[1]: Nothing to be done for ‘all’.
CC -o bin/nemostatic -static  src/nemo/main.opt.o
-Wl,--start-group -Llib/ -lnemo -lnemo_petsc_double -lnemo_petsc_complex -Wl,--end-group
-Llib/ -lPETScSLEPcCPLX -L/ccs/home/steiger/src/app-nemo/NEMO/libs/slepc/linux/lib -lslepc
-Wl,-rpath,/ccs/home/steiger/src/app-nemo/NEMO/prototype/lib
-L/ccs/home/steiger/src/app-nemo/NEMO/prototype/lib  -lnemo_petsc_double -lnemo_petsc_complex
-Wl,-rpath,/ccs/home/steiger/src/app-nemo/NEMO/libs/tensor3D
-L/ccs/home/steiger/src/app-nemo/NEMO/libs/tensor3D -ltensor
-Wl,-rpath,/ccs/home/steiger/src/app-nemo/NEMO/libs/../MaterialDB/NEMat
-L/ccs/home/steiger/src/app-nemo/NEMO/libs/../MaterialDB/NEMat -lNEMat-1.00
-Wl,-rpath,/ccs/home/steiger/src/app-nemo/NEMO/libs/../MaterialDB/nml/lib
-L/ccs/home/steiger/src/app-nemo/NEMO/libs/../MaterialDB/nml/lib -lnml
-Wl,-rpath,/ccs/home/steiger/src/app-nemo/NEMO/libs/libmesh/libmesh/lib/x86_64-unknown-linux-gnu_opt
-L/ccs/home/steiger/src/app-nemo/NEMO/libs/libmesh/libmesh/lib/x86_64-unknown-linux-gnu_opt -lmesh
-Wl,-rpath,/ccs/home/steiger/src/app-nemo/NEMO/libs/libmesh/libmesh/contrib/lib/x86_64-unknown-linux-gnu_opt
-L/ccs/home/steiger/src/app-nemo/NEMO/libs/libmesh/libmesh/contrib/lib/x86_64-unknown-linux-gnu_opt
-lexodusii -lgmv -lgzstream -lnetcdf -lsfcurves -ltriangle -llaspack
-Wl,-rpath,/ccs/home/steiger/src/app-nemo/NEMO/libs/slepc/linux/lib
-L/ccs/home/steiger/src/app-nemo/NEMO/libs/slepc/linux/lib -lslepc
-Wl,-rpath,/ccs/home/steiger/src/app-nemo/NEMO/libs/petsc/linux/lib
-L/ccs/home/steiger/src/app-nemo/NEMO/libs/petsc/linux/lib
-lpetscts -lpetscksp -lpetscdm -lpetscmat -lpetscvec -lpetscsnes -lpetscsys -lsuperlu_dist -lHYPRE
-lzmumps -ldmumps -lmumps_common -lscalapack -lblacs -lpord -lparmetis -lmetis
/ccs/home/steiger/src/app-nemo/NEMO/libs/libmesh/libmesh/contrib/tecplot/lib/x86_64-unknown-linux-gnu/tecio.a
-Wl,-rpath,/ccs/home/steiger/src/app-nemo/NEMO/libs/vtk/lib/vtk-5.6
-L/ccs/home/steiger/src/app-nemo/NEMO/libs/vtk/lib/vtk-5.6 -lvtkFiltering -lvtkIO -lvtkCommon -lvtksys
-lvtkDICOMParser -lvtkNetCDF -lvtkmetaio -lvtksqlite -lvtkpng -lvtkjpeg -lvtktiff -lvtkexpat -lvtkzlib
-Wl,-rpath,/sw/xt5/silo/4.7/cnl2.1_gnu4.2.0/lib -L/sw/xt5/silo/4.7/cnl2.1_gnu4.2.0/lib -lsiloh5
```

```
-Wl,-rpath,/sw/xt5/hdf5/1.6.7/cnl2.1_gnu4.2.4/lib -L/sw/xt5/hdf5/1.6.7/cnl2.1_gnu4.2.4/lib -lhdf5
-Wl,-rpath,/sw/xt5/szip/2.1/sles10.1_gnu4.2.4/lib -L/sw/xt5/szip/2.1/sles10.1_gnu4.2.4/lib -lsz
-Wl,-rpath,/ccs/home/steiger/src/app-nemo/NEMO/libs/qhull-2010.1/lib
-L/ccs/home/steiger/src/app-nemo/NEMO/libs/qhull-2010.1/lib -lqhull++ -lqhull
-Wl,-rpath,/ccs/home/steiger/src/app-nemo/NEMO/libs/tetgen1.4.3
-L/ccs/home/steiger/src/app-nemo/NEMO/libs/tetgen1.4.3 -ltet
-Wl,-rpath,/ccs/home/steiger/src/app-nemo/NEMO/libs/UMFPACK/Lib/
-L/ccs/home/steiger/src/app-nemo/NEMO/libs/UMFPACK/Lib/ -lumfpack
-Wl,-rpath,/ccs/home/steiger/src/app-nemo/NEMO/libs/AMD/Lib/
-L/ccs/home/steiger/src/app-nemo/NEMO/libs/AMD/Lib/ -lamd
-Wl,-rpath,/ccs/home/steiger/src/app-nemo/NEMO/libs/ARPACK/
-L/ccs/home/steiger/src/app-nemo/NEMO/libs/ARPACK/ -lparpack -larpack
-Wl,-rpath,/sw/xt5/boost/1.44.0/cnl2.2_gnu4.4.4/lib
-L/sw/xt5/boost/1.44.0/cnl2.2_gnu4.4.4/lib -lboost_filesystem -lboost_system
/ccs/home/steiger/src/app-nemo/NEMO/libs/krp/krp-init.o
/ccs/home/steiger/src/app-nemo/NEMO/libs/krp/krp-rpt.o
/ccs/home/steiger/src/app-nemo/NEMO/libs/krp/krp-rpt-init.o -L/opt/xt-tools/papi/default/v23/xt-cnl/lib64 -lpapi
-L/usr/X11R6/lib64 -lX11 -lXext -lXdmcp -lXau -lz -ldl -pthread
lib//libnemo.a(Nemo.opt.o): In function ‘Nemo::create_new_simulation(InputOptions const&,
std::vector<InputOptions, std::allocator<InputOptions> >*)’:
Nemo.cpp:(.text+0x20fe): warning: Using ’dlopen’ in statically linked applications requires at runtime the
shared libraries from the glibc version used for linking
/ccs/home/steiger/src/app-nemo/NEMO/libs/vtk/lib/vtk-5.6/libvtksys.a(SystemTools.o):
In function ‘vtksys::SystemTools::ConvertToUnixSlashes(std::basic_string<char,
std::char_traits<char>, std::allocator<char> >&)’:
SystemTools.cxx:(.text+0x235b): warning: Using ’getpwnam’ in statically linked applications
requires at runtime the
shared libraries from the glibc version used for linking
lib//libPETScSLEPcCPLX.a(fuser.o): In function ‘PetscGetUserName(char*, unsigned long)’:
/ccs/home/steiger/src/app-nemo/NEMO/libs/petsc/build-cplx/src/sys/utils/fuser.c:66:
warning: Using ’getpwuid’ in statically linked applications requires at runtime the shared libraries from
the glibc version used for linking
/usr/X11R6/lib64/libX11.a(x11trans.o): In function ‘_X11TransSocketOpen’:
(.text+0x481): warning: Using ’getaddrinfo’ in statically linked applications requires at runtime the
shared libraries from the glibc version used for linking
lib//libPETScSLEPcCPLX.a(send.o): In function ‘PetscOpenSocket(char*, int, int*)’:
/ccs/home/steiger/src/app-nemo/NEMO/libs/petsc/build-cplx/src/sys/viewer/impls/socket/send.c:103: warning:
Using ’gethostbyname’ in statically linked applications requires at runtime the shared
libraries from the glibc version used for linking
/usr/X11R6/lib64/libX11.a(x11trans.o): In function ‘_X11TransSocketINETConnect’:
(.text+0x1b5f): warning: Using ’getservbyname’ in statically linked applications requires at runtime
the shared libraries from the glibc version used for linking
DONE. You now have a static executable.
steiger@jaguarpf-login1:~/src/app-nemo/NEMO/prototype$
```

Q2 and Q4 Run Scripts

The following scripts correspond to the respective input decks. The Q4 runs are conducted with simple modifications of these same scripts.

pbs_scrip_omen.txt

```
#!/bin/csh
#PBS -A csc074omen
#PBS -l size=85200
```

```
#PBS -l walltime=03:00:00
#PBS -o /lustre/widow3/scratch/luisier/Joule/scattering/Q2metric/run1/utb.out
#PBS -e /lustre/widow3/scratch/luisier/Joule/scattering/Q2metric/run1/utb.err
#PBS -N si_scatt_q2_run1

set echo

cd /lustre/widow3/scratch/luisier/Joule/scattering/Q2metric/run1/
setenv OMP_NUM_THREADS 1
setenv MPICH_PTL_UNEX_EVENTS 400000
setenv MPICH_PTL_OTHER_EVENTS 100000

aprun -n 85200 OMEN_Jaguar-pgi64-XT5 UTB.cmd
```

eigensolver.jaguar

```
#PBS -N NEMO5
#PBS -S /bin/bash
#PBS -l size=@PROCS@,walltime=2:00:00
#PBS -A CSC074omen
#PBS -q batch
# use batch or debug

source /etc/profile
cd $WORKDIR/scaling/Hamiltonian/eigensolver
time aprun -n @PROCS@ ./nemostatic eigensolver_@PROCS@.in &> eigensolver_@PROCS@.log

# extract time needed for matrix-vector multiplication and save to file
time_needed=`grep "\[Schroedinger\] do_solve finished in" eigensolver_@PROCS@.log
                                                          | awk --field-separator=' ' '{print $5}'`
dofs=`grep "problem size (DOFs):" eigensolver_@PROCS@.log | awk --field-separator=' ' '{print $5}'`
echo "@PROCS@ $dofs $time_needed" >> eigensolver_jaguar.txt
```

UTB.jaguar

```
#PBS -N NEMO5
#PBS -S /bin/bash
#PBS -l size=@PROCS@,walltime=@TIME@
#PBS -A CSC074omen
#PBS -q batch
# use batch or debug

# please note: jobs <2003 cores have max. time 2hrs

source /etc/profile
cd $WORKDIR/scaling/Transport/UTB
time aprun -n @PROCS@ ./nemostatic UTB_@PROCS@.in &> UTB_@PROCS@.log

# extract time needed for matrix-vector multiplication and save to file
time_needed=`grep '\[Nemo\] simulation "UTB" took' UTB_@PROCS@.log | awk --field-separator=' ' '{print $5}'`
echo "@PROCS@ $time_needed" >> UTB_jaguar.txt
```

qd.jaguar

```
#PBS -N NEMO5
#PBS -S /bin/bash
#PBS -l size=@PROCS@,walltime=@TIME@
####PBS -A NEL001
#PBS -A CSC074omen
#PBS -q batch
# use batch or debug

source /etc/profile
cd $WORKDIR/scaling/Strain/weak
aprun -n @PROCS@ ./nemostatic @NEWFILE@.in > @NEWFILE@.log

# extract time needed for matrix-vector multiplication and save to file
time_needed=`grep 'Time for strain-solve' @NEWFILE@.log | awk --field-separator=' ' '{print $14}'`
dofs=`grep "total problem size (DOFs)" @NEWFILE@.log | awk --field-separator ' ' '{print $6}'`
echo "@PROCS@ $time_needed $dofs" >> QD_jaguar.txt
```

Submission of batch processes:

submit.sh

```
#!/bin/bash

# get the name of .in, .job file
file="eigensolver"

# how many MPI processes in total
# use 1, 8, 27, 64, 125, 216, 343, 512, 729, 1000
procs=$1

# load (#cells per dimension) when run using 1 CPU
dim_1cpu=4

#load (#cells per dimension) when run using N CPUs
dim=`echo "scale=20; e(l($procs * $dim_1cpu*$dim_1cpu*$dim_1cpu) / 3)" | bc -l`
echo ""
#echo "load: $load"
dim=`printf "%.0f\n" $dim`
echo "rounded dimension: $dim"

# how many MPI processes per node
thishost=`hostname`
echo "thishost=$thishost"
if [[ "$thishost" == *coates* ]]; then
   ppn=8
elif [[ "$thishost" == *rossmann* ]]; then
   ppn=24
elif [[ "$thishost" == *jaguar* ]]; then
   ppn=12
else
   ppn=8
fi
if [[ "$procs" -lt "$ppn" ]]; then
   ppn=$procs
fi
echo "ppn=$ppn"
```

```
# name of the queue
queue=ncn-par

# maximum time
time=4:00:00

# lattice constant
alatt=0.56532

dx=`echo "scale=20; $alatt*$dim * 1/3" | bc -l`
if [ $(echo "$dx > $5.0"|bc) -eq 1 ]; then
   dx=5.0;
fi
xmid=`echo "scale=20; $alatt*$dim * 1/2" | bc -l`
x1=`echo "scale=20; $xmid - $dx/2" | bc -l`
x2=`echo "scale=20; $xmid + $dx/2" | bc -l`
xmax=`echo "scale=20; $alatt*$dim + 0.3" | bc -l`

# ==========================================

nodes=$(($procs/$ppn))
if [[ "$nodes" -lt 1 ]]; then
   nodes=1;
fi

# set up new files
newfile="$file""_$procs"
cp $file.in  $newfile.in
if [[ "$thishost" == *jaguar* ]]; then
  cp $file.jaguar $newfile.job
else
  cp $file.job $newfile.job
fi

sed -i "s/@NODES@/$nodes/g"     $newfile.job
sed -i "s/@PROCS@/$procs/g"     $newfile.job
sed -i "s/@PPN@/$ppn/g"         $newfile.job
sed -i "s/@NEWFILE@/$newfile/g" $newfile.job
sed -i "s/@QUEUE@/$queue/g"     $newfile.job
sed -i "s/@TIME@/$time/g"       $newfile.job

sed -i "s/@NODES@/$nodes/g"     $newfile.in
sed -i "s/@PROCS@/$procs/g"     $newfile.in
sed -i "s/@PPN@/$ppn/g"         $newfile.in
sed -i "s/@NEWFILE@/$newfile/g" $newfile.in
sed -i "s/@xmax@/$xmax/g"       $newfile.in
sed -i "s/@x1@/$x1/g"           $newfile.in
sed -i "s/@x2@/$x2/g"           $newfile.in
sed -i "s/@dim@/$dim/g"         $newfile.in

# PBS submission
qsub $newfile.job
```

Q2 Runtime Environment

The environment for NEMO5 and the module listing for OMEN are presented for Q2. For OMEN,

the same environment is used in Q4. The Q4 environment for NEMO5 is presented in a subsequent section.

OMEN

```
luisier@jaguarpf-login8:/ccs/home/luisier/OMEN_JOULE/OMEN> module list
Currently Loaded Modulefiles:
  1) modules/3.1.6
  2) DefApps
  3) torque/2.4.1b1-snap.200905191614
  4) moab/5.3.6
  5) /opt/cray/xt-asyncpe/default/modulefiles/xtpe-istanbul
  6) cray/MySQL/5.0.64-1.0000.2342.16.1
  7) xtpe-target-cnl
  8) xt-service/2.2.73
  9) xt-os/2.2.73
 10) xt-boot/2.2.73
 11) xt-lustre-ss/2.2.73_1.6.5
 12) cray/job/1.5.5-0.1_2.0202.21413.56.6
 13) cray/csa/3.0.0-1_2.0202.21426.77.6
 14) cray/account/1.0.0-2.0202.19482.49.17
 15) cray/projdb/1.0.0-1.0202.19483.52.1
 16) Base-opts/2.2.73
 17) pgi/10.3.0
 18) xt-libsci/10.4.4
 19) pmi/1.0-1.0000.7628.10.2.ss
 20) xt-mpt/4.0.0
 21) xt-pe/2.2.73
 22) xt-asyncpe/3.7
 23) PrgEnv-pgi/2.2.73
 24) xt-papi/3.6.2
```

NEMO 5

```
LESSKEY=/etc/lesskey.bin
MODULE_VERSION_STACK=3.1.6
INFODIR=/usr/local/info:/usr/share/info:/usr/info
NNTPSERVER=news
HOSTNAME=jaguarpf-batch3
PBS_VERSION=TORQUE-2.4.8-snap.201004151117
XKEYSYMDB=/usr/X11R6/lib/X11/XKeysymDB
GNOME2_PATH=/usr/local:/opt/gnome:/usr
_MODULESBEGINENV_=/ccs/home/steiger/.modulesbeginenv
PE_ENV=PGI
SHELL=/bin/bash
HOST=jaguarpf-batch3
BATCH_ALLOC_COOKIE=0
HISTSIZE=1000
PROFILEREAD=true
XTOS_VERSION=2.2.73
PBS_JOBNAME=NEMO5
MPT_DIR=/opt/cray/mpt/4.0.0/xt/seastar
BATCH_JOBID=384072
PBS_ENVIRONMENT=PBS_BATCH
MORE=-sl
```

```
OLDPWD=/autofs/na1_home/steiger
QTDIR=/usr/lib/qt3
BOOT_DIR=/opt/xt-boot/2.2.73
PBS_O_WORKDIR=/tmp/work/steiger/scaling/Strain/weak
PRGENV_DIR=/opt/xt-prgenv/2.2.73
USER=steiger
PBS_TASKNUM=1
JRE_HOME=/usr/lib/jvm/jre
GROFF_NO_SGR=yes
INTEL_PRE_COMPILE_OPTS= -msse3
BUILD_OPTS=/opt/cray/xt-asyncpe/3.7/bin/build-opts
ASYNCPE_DIR=/opt/cray/xt-asyncpe/3.7
LS_COLORS=
LD_LIBRARY_PATH=/opt/xt-pe/2.2.73/lib:/opt/cray/pmi/1.0-1.0000.7628.10.2.ss/lib64:
                /opt/cray/mpt/4.0.0/xt/seastar/mpich2-pgi/lib:
                /opt/cray/mpt/4.0.0/xt/sma/lib:/opt/pgi/10.3.0/linux86-64/10.3/libso:
                /opt/pgi/10.3.0/linux86-64/10.3/lib:
                /opt/cray/projdb/1.0.0-1.0202.19483.52.1/lib64:
                /opt/cray/csa/3.0.0-1_2.0202.21426.77.6/lib64:
                /opt/cray/job/1.5.5-0.1_2.0202.21413.56.6/lib64:
                /opt/cray/MySQL/5.0.64-1.0000.2342.16.1/lib64/mysql:
                /opt/cray/MySQL/5.0.64-1.0000.2342.16.1/lib64
PBS_O_HOME=/ccs/home/steiger
LC_PE_ENV=pgi
XNLSPATH=/usr/X11R6/lib/X11/nls
TVDSVRLAUNCHCMD=ssh
WORKDIR=/tmp/work/steiger
PBS_NNODES=24
ENV=/etc/bash.bashrc
MPICH_DIR=/opt/cray/mpt/4.0.0/xt/seastar/mpich2-pgi
PGI_VERS_STR=10.3.0
HOSTTYPE=x86_64
RCLOCAL_PRGENV=true
PBS_MOMPORT=15003
FROM_HEADER=
PE_PRODUCT_LIST=ASYNCPE:PE:CRAY_PMI:XTMPT:LIBSCI:PGI:LUSTRE:XTPE_ISTANBUL
MPT_VERSION=4.0.0
PAGER=less
FFTW_SYSTEM_WISDOM_DIR=/opt/xt-libsci/10.4.4
PGI_VERSION=10.3
CSHEDIT=emacs
OS_DIR=/opt/xt-os/2.2.73
PBS_O_QUEUE=batch
XDG_CONFIG_DIRS=/usr/local/etc/xdg/:/etc/xdg/:/etc/opt/gnome/xdg/
MPICHBASEDIR=/opt/cray/mpt/4.0.0/xt
MINICOM=-c on
PGI=/opt/pgi/10.3.0
PATH=/opt/cray/xt-asyncpe/3.7/bin:/opt/cray/pmi/1.0-1.0000.7628.10.2.ss/bin:
     /opt/pgi/10.3.0/linux86-64/10.3/bin:/opt/cray/projdb/1.0.0-1.0202.19483.52.1/bin:
     /opt/cray/account/1.0.0-2.0202.19482.49.17/bin:
     /opt/cray/csa/3.0.0-1_2.0202.21426.77.6/sbin:
     /opt/cray/csa/3.0.0-1_2.0202.21426.77.6/bin:
     /opt/cray/job/1.5.5-0.1_2.0202.21413.56.6/bin:
     /opt/xt-lustre-ss/2.2.73_1.6.5/usr/sbin:/opt/xt-lustre-ss/2.2.73_1.6.5/usr/bin:
     /opt/xt-boot/2.2.73/bin/snos64:/opt/xt-os/2.2.73/bin/snos64:
     /opt/xt-service/2.2.73/bin/snos64:/opt/xt-prgenv/2.2.73/bin:
     /opt/cray/MySQL/5.0.64-1.0000.2342.16.1/sbin:
```

```
     /opt/cray/MySQL/5.0.64-1.0000.2342.16.1/bin:/opt/moab/default/bin:
     /opt/torque/default/bin:/sw/xt5/bin:/opt/modules/3.1.6/bin:/usr/local/bin:/usr/bin:
     /usr/X11R6/bin:/bin:/usr/games:/opt/bin:/opt/gnome/bin:/opt/kde3/bin:
     /usr/lib/jvm/jre/bin:/usr/lib/mit/bin:/usr/lib/mit/sbin:/opt/pathscale/bin:
     .:/usr/lib/qt3/bin:/opt/bin:/opt/public/bin:/opt/bin:/opt/public/bin
PBS_O_LOGNAME=steiger
MAIL=/var/spool/mail/steiger
MODULE_VERSION=3.1.6
APRUN_XFER_LIMITS=0
YOD_LOGFILE=syslog
PBS_O_LANG=en_US.UTF-8
CPU=x86_64
PBS_JOBCOOKIE=883FCA8FFE600598A16338635F8F2587
JAVA_BINDIR=/usr/lib/jvm/jre/bin
ASYNCPE_VERSION=3.7
PWD=/tmp/work/steiger/scaling/Strain/weak
INPUTRC=/etc/inputrc
RCLOCAL_MYSQL=true
XTPE_COMPILE_TARGET=linux
JAVA_HOME=/usr/lib/jvm/jre
_LMFILES_=/opt/modulefiles/modules/3.1.6:/sw/xt5/modulefiles/DefApps:
          /opt/modulefiles/torque/2.4.1b1-snap.200905191614:
          /opt/modulefiles/moab/5.3.6:
          /opt/cray/xt-asyncpe/default/modulefiles/xtpe-istanbul:
          /opt/modulefiles/cray/MySQL/5.0.64-1.0000.2342.16.1:
          /opt/modulefiles/xtpe-target-cnl:/opt/modulefiles/xt-service/2.2.73:
          /opt/modulefiles/xt-os/2.2.73:/opt/modulefiles/xt-boot/2.2.73:
          /opt/modulefiles/xt-lustre-ss/2.2.73_1.6.5:
          /opt/modulefiles/cray/job/1.5.5-0.1_2.0202.21413.56.6:
          /opt/modulefiles/cray/csa/3.0.0-1_2.0202.21426.77.6:
          /opt/modulefiles/cray/account/1.0.0-2.0202.19482.49.17:
          /opt/modulefiles/cray/projdb/1.0.0-1.0202.19483.52.1:
          /opt/modulefiles/Base-opts/2.2.73:/opt/modulefiles/pgi/10.3.0:
          /opt/modulefiles/xt-libsci/10.4.4:
          /opt/cray/modulefiles/pmi/1.0-1.0000.7628.10.2.ss:
          /opt/modulefiles/xt-mpt/4.0.0:/opt/modulefiles/xt-pe/2.2.73:
          /opt/modulefiles/xt-asyncpe/3.7:/opt/modulefiles/PrgEnv-pgi/2.2.73
LANG=en_US.UTF-8
PBS_NODENUM=0
SYSTEM_USERDIR=/tmp/work/steiger
PGROUPD_LICENSE_FILE=/opt/pgi/license.dat
PYTHONSTARTUP=/etc/pythonstart
MODULEPATH=/opt/cray/xt-asyncpe/3.7/modulefiles:/opt/modulefiles:
           /opt/cray/modulefiles:/opt/modules/3.1.6:/sw/xt5/modulefiles
LOADEDMODULES=modules/3.1.6:DefApps:torque/2.4.1b1-snap.200905191614:
              moab/5.3.6:xtpe-istanbul:cray/MySQL/5.0.64-1.0000.2342.16.1:
              xtpe-target-cnl:xt-service/2.2.73:xt-os/2.2.73:xt-boot/2.2.73:
              xt-lustre-ss/2.2.73_1.6.5:cray/job/1.5.5-0.1_2.0202.21413.56.6:
              cray/csa/3.0.0-1_2.0202.21426.77.6:
              cray/account/1.0.0-2.0202.19482.49.17:
              cray/projdb/1.0.0-1.0202.19483.52.1:
              Base-opts/2.2.73:pgi/10.3.0:xt-libsci/10.4.4:1.0-1.0000.7628.10.2.ss:
              xt-mpt/4.0.0:xt-pe/2.2.73:xt-asyncpe/3.7:PrgEnv-pgi/2.2.73
PATHSCALE_PRE_COMPILE_OPTS= -march=barcelona
PBS_O_SHELL=/bin/bash
PBS_SERVER=pbs
PBS_JOBID=384072.nid00004
```

```
ENVIRONMENT=BATCH
CRAY_PMI_POST_LINK_OPTS=$(pkg-config cray-pmi --libs-only-L)
                        -Wl,-rpath=/opt/cray/pmi/1.0-1.0000.7628.10.2.ss/lib64
TEXINPUTS=::/ccs/home/steiger/.TeX:/usr/share/doc/.TeX:/usr/doc/.TeX:
          /ccs/home/steiger/.TeX:/usr/share/doc/.TeX:/usr/doc/.TeX
HOME=/ccs/home/steiger
SHLVL=2
QT_SYSTEM_DIR=/usr/share/desktop-data
OSTYPE=linux
LESS_ADVANCED_PREPROCESSOR=no
PGI_PATH=/opt/pgi/10.3.0
PTL_SNOS_NAL=SS
PBS_O_HOST=jaguarpf-login6.ccs.ornl.gov
XCURSOR_THEME=Industrial
LS_OPTIONS=-N --color=none -T 0
XTPE_ISTANBUL_ENABLED=ON
LIBLUSTRE_DEBUG_CONSOLE=0
CRAY_PMI_INCLUDE_OPTS=$(pkg-config cray-pmi --cflags)
SE_DIR=/opt/xt-service/2.2.73
WINDOWMANAGER=
PBS_VNODENUM=0
GTK_PATH=/usr/local/lib/gtk-2.0:/opt/gnome/lib/gtk-2.0:/usr/lib/gtk-2.0
LOGNAME=steiger
MACHTYPE=x86_64-suse-linux
LESS=-M -I
G_FILENAME_ENCODING=@locale,UTF-8,ISO-8859-15,CP1252
CVS_RSH=ssh
GTK_PATH64=/usr/local/lib64/gtk-2.0:/opt/gnome/lib64/gtk-2.0:/usr/lib64/gtk-2.0
PBS_QUEUE=batch
ACLOCAL_FLAGS=-I /opt/gnome/share/aclocal
XDG_DATA_DIRS=/usr/local/share/:/usr/share/:/etc/opt/kde3/share/:/opt/kde3/share/:
              /opt/gnome/share/
MODULESHOME=/opt/modules/3.1.6
PE_POST_LINK_OPTS=-Wl,-rpath=/opt/xt-pe/2.2.73/lib
PBS_O_MAIL=/var/mail/steiger
LESSOPEN=lessopen.sh %s
PKG_CONFIG_PATH=/opt/cray/pmi/1.0-1.0000.7628.10.2.ss/lib64/pkgconfig:
                /opt/cray/projdb/1.0.0-1.0202.19483.52.1/lib64/pkgconfig:
                /opt/cray/MySQL/5.0.64-1.0000.2342.16.1/lib64/pkgconfig:
                /usr/local/lib/pkgconfig:/usr/local/share/pkgconfig:
                /usr/lib64/pkgconfig:/usr/share/pkgconfig:
                /opt/kde3/lib64/pkgconfig:/opt/gnome/lib64/pkgconfig:
                /opt/gnome/lib64/pkgconfig:/opt/gnome/share/pkgconfig
LIBSCI_BASE_DIR=/opt/xt-libsci/10.4.4
INFOPATH=/usr/local/info:/usr/share/info:/usr/info:/opt/gnome/share/info
LIBSCI_VERSION=10.4.4
LESSCLOSE=lessclose.sh %s %s
GNU_PRE_COMPILE_OPTS= -march=amdfam10
PE_DIR=/opt/xt-pe/2.2.73
PBS_NODEFILE=/var/spool/torque/aux//384072.nid00004
G_BROKEN_FILENAMES=1
PBS_O_PATH=/opt/cray/xt-asyncpe/3.7/bin:/opt/cray/pmi/1.0-1.0000.7628.10.2.ss/bin:
           /opt/gcc/4.4.2/bin:/sw/xt5/nedit/5.5/sles10.1_binary/bin:
           /sw/xt5/cmake/2.8.0/sles10.1_gnu4.1.2/bin:/opt/fftw/3.2.2.1/bin:
           /lustre/widow/sw/visit/bin:
           /sw/xt5/valgrind/3.4.1/sles10.1_gnu4.2.0.quadcore/bin:
           /sw/xt5/subversion/1.5.0/sles10.1_gnu4.2.4/bin:
```

```
            /opt/cray/projdb/1.0.0-1.0202.19483.52.1/bin:
            /opt/cray/account/1.0.0-2.0202.19482.49.17/bin:
            /opt/cray/csa/3.0.0-1_2.0202.21426.77.6/sbin:
            /opt/cray/csa/3.0.0-1_2.0202.21426.77.6/bin:
            /opt/cray/job/1.5.5-0.1_2.0202.21413.56.6/bin:
            /opt/xt-lustre-ss/2.2.73_1.6.5/usr/sbin:/opt/xt-lustre-ss/2.2.73_1.6.5/usr/bin:
            /opt/xt-boot/2.2.73/bin/snos64:/opt/xt-os/2.2.73/bin/snos64:
            /opt/xt-service/2.2.73/bin/snos64:/opt/xt-prgenv/2.2.73/bin:
            /opt/cray/MySQL/5.0.64-1.0000.2342.16.1/sbin:
            /opt/cray/MySQL/5.0.64-1.0000.2342.16.1/bin:/opt/moab/default/bin:
            /opt/torque/default/bin:/sw/xt5/bin:/opt/modules/3.1.6/bin:/usr/local/bin:
            /usr/bin:/usr/X11R6/bin:/bin:/usr/games:/opt/bin:/opt/gnome/bin:/opt/kde3/bin:
            /usr/lib/jvm/jre/bin:/usr/lib/mit/bin:/usr/lib/mit/sbin:
            /opt/pathscale/bin:.:/usr/lib/qt3/bin:/opt/bin:/opt/public/bin
COLORTERM=1
JAVA_ROOT=/usr/lib/jvm/jre
LUSTRE_DIR=/opt/xt-lustre-ss/2.2.73_1.6.5
_=/usr/bin/env
```

Q4 Input Settings

The input decks did not change for OMEN in Q4, the inputs from Q2 remain valid. There are nominal changes to the inputs for NEMO5 in Q4. They are summarized here now.

In Q4, NEMO5 uses SVN revision number 5160, PETSc changeset 18779, SLEPc revision 2304 and libmesh revision 4359.

UTB (UTB.in)

The option `use_adaptive_grid1 = true` was replaced with `use_adaptive_grid1=false`.
The options `kspace_strategy = combined` and `calculate_current_flux=false` were introduced for the Q4 input deck. The `Korigin=(0,0,0.5)`, `Kradius=0.5` and `Knum=7` options were replaced with `Krange = [(0.0,0.5)]` and `Knum=15`.

Strain (qd.in)

The option `preconditioner=bjacobi` was replaced with `preconditioner=asm`.
The options `assume_all_atom_active = true` and `petsc_double_report = true` were removed.

Eigenstates

The options `assume_all_atom_active = true` and `petsc_double_report = true` were removed.

Q4 Compilation Scripts

The Q2 compilation scripts are valid for both OMEN and NEMO5 in Q4 for the enhanced codes. (does not make sense for n5 -sebastian?)

Q4 Runtime Environment -NEMO5 NEMO 5

```
LESSKEY=/etc/lesskey.bin
MODULE_VERSION_STACK=3.1.6
FFTW_POST_LINK_OPTS= -L/opt/fftw/3.2.2.1/lib -Wl,-rpath=/opt/fftw/3.2.2.1/lib -lfftw3 -lfftw3f
INFODIR=/usr/local/info:/usr/share/info:/usr/info
```

```
NNTPSERVER=news
GNU_VERSION=4.4.2
HOSTNAME=jaguarpf-login8
PBS_VERSION=TORQUE-2.4.8-snap.201004151117
XKEYSYMDB=/usr/X11R6/lib/X11/XKeysymDB
GNOME2_PATH=/usr/local:/opt/gnome:/usr
_MODULESBEGINENV_=/ccs/home/steiger/.modulesbeginenv.jaguarpf-login8
PE_ENV=GNU
SHELL=/bin/bash
TERM=xterm
HOST=jaguarpf-login8
BATCH_ALLOC_COOKIE=0
HISTSIZE=1000
PROFILEREAD=true
XTOS_VERSION=2.2.73
PBS_JOBNAME=STDIN
SSH_CLIENT=128.46.94.37 57143 22
MPT_DIR=/opt/cray/mpt/4.0.0/xt/seastar
LIBRARY_PATH=/sw/xt5/valgrind/3.4.1/sles10.1_gnu4.2.0.quadcore/lib:/sw/xt5/subversion/1.5.0/sles10.1_gnu4.2.4/lib
FFTW_INC=/opt/fftw/3.2.2.1/include
PBS_ENVIRONMENT=PBS_INTERACTIVE
BATCH_JOBID=477037
MORE=-sl
QTDIR=/usr/lib/qt3
BOOT_DIR=/opt/xt-boot/2.2.73
VALGRIND_DIR=/sw/xt5/valgrind/3.4.1/sles10.1_gnu4.2.0.quadcore
SSH_TTY=/dev/pts/2
PRGENV_DIR=/opt/xt-prgenv/2.2.73
INCLUDE_PATH=/sw/xt5/valgrind/3.4.1/sles10.1_gnu4.2.0.quadcore/include: \
/sw/xt5/subversion/1.5.0/sles10.1_gnu4.2.4/include
FFTW_VERSION=3.2.2.1
PBS_O_WORKDIR=/ccs/home/steiger/src/app-nemo/NEMO/prototype/benchmarks/scaling/Hamiltonian/eigensolver
FFTW_DIR=/opt/fftw/3.2.2.1/lib
PBS_TASKNUM=1
USER=steiger
JRE_HOME=/usr/lib/jvm/jre
GROFF_NO_SGR=yes
INTEL_PRE_COMPILE_OPTS= -msse3
BUILD_OPTS=/opt/cray/xt-asyncpe/3.7/bin/build-opts
ASYNCPE_DIR=/opt/cray/xt-asyncpe/3.7
LD_LIBRARY_PATH=/opt/xt-pe/2.2.73/lib:/opt/cray/pmi/1.0-1.0000.7628.10.2.ss/lib64: \
/opt/cray/mpt/4.0.0/xt/seastar/mpich2-gnu/lib:/opt/cray/mpt/4.0.0/xt/sma/lib:/opt/gcc/mpfr/2.3.1/lib: \
/opt/gcc/4.4.2/snos/lib64:/opt/fftw/3.2.2.1/lib:/sw/xt5/valgrind/3.4.1/sles10.1_gnu4.2.0.quadcore/lib: \
/sw/xt5/subversion/1.5.0/sles10.1_gnu4.2.4/lib:/opt/cray/projdb/1.0.0-1.0202.19483.52.1/lib64: \
/opt/cray/csa/3.0.0-1_2.0202.21426.77.6/lib64:/opt/cray/job/1.5.5-0.1_2.0202.21413.56.6/lib64: \
/opt/cray/MySQL/5.0.64-1.0000.2342.16.1/lib64/mysql:/opt/cray/MySQL/5.0.64-1.0000.2342.16.1/lib64
LC_PE_ENV=gnu
PBS_O_HOME=/ccs/home/steiger
XNLSPATH=/usr/X11R6/lib/X11/nls
TVDSVRLAUNCHCMD=ssh
WORKDIR=/tmp/work/steiger
PBS_NNODES=12
ENV=/etc/bash.bashrc
MPICH_DIR=/opt/cray/mpt/4.0.0/xt/seastar/mpich2-gnu
HOSTTYPE=x86_64
RCLOCAL_PRGENV=true
PBS_MOMPORT=15003
```

```
FROM_HEADER=
PE_PRODUCT_LIST=ASYNCPE:PE:CRAY_PMI:XTMPT:LIBSCI:GCC:GNU:LUSTRE:XTPE_ISTANBUL:FFTW
MPT_VERSION=4.0.0
GCC_VERSION=4.4.2
PAGER=less
FFTW_SYSTEM_WISDOM_DIR=/opt/xt-libsci/10.4.4
CSHEDIT=emacs
OS_DIR=/opt/xt-os/2.2.73
XTPE_INFO_MESSAGE_OFF=1
PBS_O_QUEUE=debug
XDG_CONFIG_DIRS=/usr/local/etc/xdg/:/etc/xdg/:/etc/opt/gnome/xdg/
MPICHBASEDIR=/opt/cray/mpt/4.0.0/xt
MINICOM=-c on
MAIL=/var/mail/steiger
MODULE_VERSION=3.1.6
APRUN_XFER_LIMITS=0
YOD_LOGFILE=syslog
PBS_O_LOGNAME=steiger
PATH=/opt/cray/xt-asyncpe/3.7/bin:/opt/cray/pmi/1.0-1.0000.7628.10.2.ss/bin:/opt/gcc/4.4.2/bin: \
/sw/xt5/nedit/5.5/sles10.1_binary/bin:/sw/xt5/cmake/2.8.0/sles10.1_gnu4.1.2/bin: \
/opt/fftw/3.2.2.1/bin:/lustre/widow/sw/visit/bin:/sw/xt5/valgrind/3.4.1/sles10.1_gnu4.2.0.quadcore/bin: \
/sw/xt5/subversion/1.5.0/sles10.1_gnu4.2.4/bin:/opt/cray/projdb/1.0.0-1.0202.19483.52.1/bin: \
/opt/cray/account/1.0.0-2.0202.19482.49.17/bin:/opt/cray/csa/3.0.0-1_2.0202.21426.77.6/sbin: \
/opt/cray/csa/3.0.0-1_2.0202.21426.77.6/bin:/opt/cray/job/1.5.5-0.1_2.0202.21413.56.6/bin: \
/opt/xt-lustre-ss/2.2_1.6.5/usr/sbin:/opt/xt-lustre-ss/2.2_1.6.5/usr/bin:/opt/xt-boot/2.2.73/bin/snos64: \
/opt/xt-os/2.2.73/bin/snos64:/opt/xt-service/2.2.73/bin/snos64:/opt/xt-prgenv/2.2.73/bin: \
/opt/cray/MySQL/5.0.64-1.0000.2342.16.1/sbin:/opt/cray/MySQL/5.0.64-1.0000.2342.16.1/bin: \
/opt/moab/default/bin:/opt/torque/default/bin:/sw/xt5/bin:/opt/modules/3.1.6/bin:/usr/local/bin: \
/usr/bin:/usr/X11R6/bin:/bin:/usr/games:/opt/bin:/opt/gnome/bin:/opt/kde3/bin:/usr/lib/jvm/jre/bin: \
/usr/lib/mit/bin:/usr/lib/mit/sbin:/opt/pathscale/bin:.:/usr/lib/qt3/bin:/opt/bin:/opt/public/bin: \
/opt/bin:/opt/public/bin:/opt/bin:/opt/public/bin
CPU=x86_64
PBS_O_LANG=en_US.UTF-8
PBS_JOBCOOKIE=CF76DC805573AC6C03E082DE47C3DE76
SSH_SENDS_LOCALE=yes
JAVA_BINDIR=/usr/lib/jvm/jre/bin
ASYNCPE_VERSION=3.7
PWD=/tmp/work/steiger
INPUTRC=/etc/inputrc
RCLOCAL_MYSQL=true
XTPE_COMPILE_TARGET=linux
JAVA_HOME=/usr/lib/jvm/jre
_LMFILES_=/opt/modulefiles/modules/3.1.6:/sw/xt5/modulefiles/DefApps: \
/opt/modulefiles/torque/2.4.1b1-snap.200905191614:/opt/modulefiles/moab/5.3.6: \
/opt/cray/xt-asyncpe/default/modulefiles/xtpe-istanbul:/opt/modulefiles/cray/MySQL/5.0.64-1.0000.2342.16.1: \
/opt/modulefiles/xtpe-target-cnl:/opt/modulefiles/xt-service/2.2.73:/opt/modulefiles/xt-os/2.2.73: \
/opt/modulefiles/xt-boot/2.2.73:/opt/modulefiles/xt-lustre-ss/2.2_1.6.5: \
/opt/modulefiles/cray/job/1.5.5-0.1_2.0202.21413.56.6:/opt/modulefiles/cray/csa/3.0.0-1_2.0202.21426.77.6: \
/opt/modulefiles/cray/account/1.0.0-2.0202.19482.49.17:/opt/modulefiles/cray/projdb/1.0.0-1.0202.19483.52.1: \
/opt/modulefiles/Base-opts/2.2.73:/opt/modulefiles/PrgEnv-gnu/2.2.73:/opt/modulefiles/xt-asyncpe/3.7: \
/opt/modulefiles/xt-pe/2.2.73:/opt/modulefiles/xt-mpt/4.0.0:/opt/cray/modulefiles/pmi/1.0-1.0000.7628.10.2.ss: \
/opt/modulefiles/xt-libsci/10.4.4:/opt/modulefiles/gcc/4.4.2:/sw/xt5/modulefiles/subversion/1.5.0: \
/sw/xt5/modulefiles/valgrind/3.4.1:/sw/xt5/modulefiles/visit/1.11.1:/opt/modulefiles/fftw/3.2.2.1: \
/sw/xt5/modulefiles/cmake/2.8.0:/sw/xt5/modulefiles/nedit/5.5
FFTW_INCLUDE_OPTS= -I/opt/fftw/3.2.2.1/include
PBS_NODENUM=0
SYSTEM_USERDIR=/tmp/work/steiger
```

```
LANG=en_US.UTF-8
PYTHONSTARTUP=/etc/pythonstart
MODULEPATH=/opt/cray/xt-asyncpe/3.7/modulefiles:/opt/modulefiles:/opt/cray/modulefiles:/opt/modules/3.1.6: \
/sw/xt5/modulefiles
LOADEDMODULES=modules/3.1.6:DefApps:torque/2.4.1b1-snap.200905191614:moab/5.3.6:xtpe-istanbul: \
cray/MySQL/5.0.64-1.0000.2342.16.1:xtpe-target-cnl:xt-service/2.2.73:xt-os/2.2.73:xt-boot/2.2.73: \
xt-lustre-ss/2.2_1.6.5:cray/job/1.5.5-0.1_2.0202.21413.56.6:cray/csa/3.0.0-1_2.0202.21426.77.6: \
cray/account/1.0.0-2.0202.19482.49.17:cray/projdb/1.0.0-1.0202.19483.52.1:Base-opts/2.2.73: \
PrgEnv-gnu/2.2.73:xt-asyncpe/3.7:xt-pe/2.2.73:xt-mpt/4.0.0:1.0-1.0000.7628.10.2.ss:xt-libsci/10.4.4: \
gcc/4.4.2:subversion/1.5.0:valgrind/3.4.1:visit/1.11.1:fftw/3.2.2.1:cmake/2.8.0:nedit/5.5
PATHSCALE_PRE_COMPILE_OPTS= -march=barcelona
PBS_O_SHELL=/bin/bash
PBS_SERVER=pbs
PBS_JOBID=477037.nid00004
CRAY_PMI_POST_LINK_OPTS=$(pkg-config cray-pmi --libs-only-L) -Wl,-rpath=/opt/cray/pmi/1.0-1.0000.7628.10.2.ss/lib64
HOME=/ccs/home/steiger
SHLVL=2
QT_SYSTEM_DIR=/usr/share/desktop-data
OSTYPE=linux
LESS_ADVANCED_PREPROCESSOR=no
PTL_SNOS_NAL=SS
LS_OPTIONS=--N --color=none -T 0
XTPE_ISTANBUL_ENABLED=ON
LIBLUSTRE_DEBUG_CONSOLE=0
CRAY_PMI_INCLUDE_OPTS=$(pkg-config cray-pmi --cflags)
SE_DIR=/opt/xt-service/2.2.73
PBS_O_HOST=jaguarpf-login8.ccs.ornl.gov
WINDOWMANAGER=
GCC_PATH=/opt/gcc/4.4.2
PBS_VNODENUM=0
GTK_PATH=/usr/local/lib/gtk-2.0:/opt/gnome/lib/gtk-2.0:/usr/lib/gtk-2.0
LOGNAME=steiger
MACHTYPE=x86_64-suse-linux
LESS=-M -I
G_FILENAME_ENCODING=@locale,UTF-8,ISO-8859-15,CP1252
CVS_RSH=ssh
GTK_PATH64=/usr/local/lib64/gtk-2.0:/opt/gnome/lib64/gtk-2.0:/usr/lib64/gtk-2.0
PBS_QUEUE=debug
SSH_CONNECTION=128.46.94.37 57143 160.91.205.245 22
ACLOCAL_FLAGS=-I /opt/gnome/share/aclocal
XDG_DATA_DIRS=/usr/local/share/:/usr/share/:/etc/opt/kde3/share/:/opt/kde3/share/:/opt/gnome/share/
LC_CTYPE=en_US.UTF-8
MODULESHOME=/opt/modules/3.1.6
PE_POST_LINK_OPTS=-Wl,-rpath=/opt/xt-pe/2.2.73/lib
LESSOPEN=lessopen.sh %s
PKG_CONFIG_PATH=/opt/cray/pmi/1.0-1.0000.7628.10.2.ss/lib64/pkgconfig: \
/opt/cray/projdb/1.0.0-1.0202.19483.52.1/lib64/pkgconfig: \
/opt/cray/MySQL/5.0.64-1.0000.2342.16.1/lib64/pkgconfig:/usr/local/lib/pkgconfig: \
/usr/local/share/pkgconfig:/usr/lib64/pkgconfig:/usr/share/pkgconfig:/opt/kde3/lib64/pkgconfig: \
/opt/gnome/lib64/pkgconfig:/opt/gnome/lib64/pkgconfig:/opt/gnome/share/pkgconfig
PBS_O_MAIL=/var/mail/steiger
LIBSCI_BASE_DIR=/opt/xt-libsci/10.4.4
INFOPATH=/opt/gcc/4.4.2/snos/info:/opt/fftw/3.2.2.1/info:/usr/local/info:/usr/share/info:/usr/info: \
/opt/gnome/share/info
LIBSCI_VERSION=10.4.4
DISPLAY=localhost:12.0
XAUTHLOCALHOSTNAME=jaguarpf-login8
```

```
LESSCLOSE=lessclose.sh %s %s
GNU_PRE_COMPILE_OPTS= -march=amdfam10
PE_DIR=/opt/xt-pe/2.2.73
PBS_NODEFILE=/var/spool/torque/aux//477037.nid00004
G_BROKEN_FILENAMES=1
COLORTERM=1
JAVA_ROOT=/usr/lib/jvm/jre
LUSTRE_DIR=/opt/xt-lustre-ss/2.2_1.6.5
PBS_O_PATH=/opt/cray/xt-asyncpe/3.7/bin:/opt/cray/pmi/1.0-1.0000.7628.10.2.ss/bin:/opt/gcc/4.4.2/bin: \
/sw/xt5/nedit/5.5/sles10.1_binary/bin:/sw/xt5/cmake/2.8.0/sles10.1_gnu4.1.2/bin:/opt/fftw/3.2.2.1/bin: \
/lustre/widow/sw/visit/bin:/sw/xt5/valgrind/3.4.1/sles10.1_gnu4.2.0.quadcore/bin: \
/sw/xt5/subversion/1.5.0/sles10.1_gnu4.2.4/bin:/opt/cray/projdb/1.0.0-1.0202.19483.52.1/bin: \
/opt/cray/account/1.0.0-2.0202.19482.49.17/bin:/opt/cray/csa/3.0.0-1_2.0202.21426.77.6/sbin: \
/opt/cray/csa/3.0.0-1_2.0202.21426.77.6/bin:/opt/cray/job/1.5.5-0.1_2.0202.21413.56.6/bin: \
/opt/xt-lustre-ss/2.2_1.6.5/usr/sbin:/opt/xt-lustre-ss/2.2_1.6.5/usr/bin:/opt/xt-boot/2.2.73/bin/snos64: \
/opt/xt-os/2.2.73/bin/snos64:/opt/xt-service/2.2.73/bin/snos64:/opt/xt-prgenv/2.2.73/bin: \
/opt/cray/MySQL/5.0.64-1.0000.2342.16.1/sbin:/opt/cray/MySQL/5.0.64-1.0000.2342.16.1/bin: \
/opt/moab/default/bin:/opt/torque/default/bin:/sw/xt5/bin:/opt/modules/3.1.6/bin:/usr/local/bin: \
/usr/bin:/usr/X11R6/bin:/bin:/usr/games:/opt/bin:/opt/gnome/bin:/opt/kde3/bin:/usr/lib/jvm/jre/bin: \
/usr/lib/mit/bin:/usr/lib/mit/sbin:/opt/pathscale/bin:.:/usr/lib/qt3/bin:/opt/bin:/opt/public/bin
_=/usr/bin/env
OLDPWD=/ccs/home/steiger
```

0.6.4 OSIRIS

Q2 Input Settings

warm.3d

```
!----------------------------------------
! Warm Plasma Benchmark for ASCR Joule
!
!----------------------------------------

simulation
{
  ndump_prof = 100,        ! Profiling info dump frequency
                           ! - this corresponds to ~ 15 min wall clock time
}

node_conf
{
  node_number(1:3) =  48, 48, 24,! 55296 nodes total
  if_periodic(1:3) = .true., .true., .true.,
}


!----------spatial grid----------
grid
{
  nx_p(1:3) = 6144, 6144, 1536,

  ! Load info dump frequency
  ndump_global_load = 100,  ! global simulation load (min, max, avg particles / node )
  ndump_node_load   = 250,  ! number of particles per node for all nodes
```

```
  ndump_grid_load   = 0,    ! number of particles per cell for all grid points

}


!----------time step and global data dump timestep number----------
time_step
{
  dt     =   0.05,
  ndump  =    0,
}


!----------spatial limits of the simulations----------
space
{
  xmin(1:3) =    0.0, 0.0,  0.0,
  xmax(1:3) =   614.4 ,  614.4 ,  153.6 ,

}


!----------time limits ----------
time
{
  tmin = 0.0,
  tmax  = 285.0,
}

!----------boundary conditions for em-fields ----------
emf_bound
{
  ! these aren't necessary since we're using periodic
  ! boundaries
}


!----------number of particle species----------
particles
{
  num_species = 1,                       ! use 1 particle species

}


!----------information for species 1----------

species
{
  name = "electrons" ,

  num_par_x(1:3) = 2, 2, 2,

  num_par_max = 10000000,

  rqm=-1.0,
```

```
  vth(1:3) =  0.01 , 0.01 , 0.01 ,
  vfl(1:3) =  0.0 , 0.0 , 0.0 ,

  interpolation = "quadratic",
}

spe_bound
{
  ! these aren't necessary since we're using periodic
  ! boundaries
}



! --------------------- end of osiris input file ---------------
```

lwf-r01.3d

```
!-------------------------------------------------------------
! LWFA Benchmark for ASCR Joule
! - Run 01
! - High resolution run
!-------------------------------------------------------------


simulation
{
  ndump_prof = 1000,        ! Profiling info dump frequency

  !  n0 = 1.74d21,
}

!----------the node configuration for this simulation---------
node_conf
{

  node_number(1:3) =  96, 24, 24,           ! 55296 nodes total

  ! this is another option
  ! node_number(1:3) =  384, 12, 12,   ! 55296 nodes total

  if_periodic(1:3) = .false., .false., .false.,
}

!----------spatial grid----------
grid
{
  ! 8064/96 = 84 cells/node along x1
  nx_p(1:3) = 8064, 480, 480,

  ! Load info dump frequency
  ndump_global_load = 1000, ! global simulation load (min, max, avg particles / node )
  ndump_node_load   = 10000, ! number of particles per node for all nodes
  ndump_grid_load   = 0,     ! number of particles per cell for all grid points

}
```

```
!----------time step and global data dump timestep number----------
time_step
{
  ! Courant condition gives 0.0998326502998
  ! 0.999 * courant = 0.0997328176495005

  dt     =  0.0997328176495005,

  ! Dumps at every 2000.0 1/\omega_0 (or as close as we can get them)
  ! (rounded down to insure the final dumps get written)
  ndump  =  20053,
}

!----------restart information----------
restart
{
  ! I turned off writing restarts, as per Ken's request, the idea is that this run
  ! should finish within a 14 - 16 hour wall clock time

! write restart at final iteration
  ndump_fac  = 10,
}

!----------spatial limits of the simulations----------
!(note that this includes information about
! the motion of the simulation box)
space
{
  xmin(1:3) =   0.000d0 , 0.000d0, 0.000d0,
  xmax(1:3) =  806.4, 1171.88, 1171.88,
  if_move= .true., .false., .false.,
}


!----------time limits ----------
time
{
  tmin = 0.0d0,

  tmax = 4089.0,
}

!----------boundary conditions for em-fields ----------
emf_bound
{
  type(1:2,1) =   0,  0,
  type(1:2,2) =  60, 60,
  type(1:2,3) =  60, 60,
}

!----------diagnostic for electromagnetic fields----------
diag_emf
{
  ndump_fac         = 1,  ! this is ignored, no full grid diagnostics requested
  ndump_fac_ave     = 1,  ! Spatial/time averaged diagnostics
  ndump_fac_lineout = 1,  ! slice and lineout diagnostics
```

```
  reports = "e1, savg", "e2, savg", "e3, savg", "e1, line, x1, 150, 150",
            "psi, savg",

  ndump_fac_ene_int = 1,

  n_ave(1:3)        = 8, 4, 4,
}

!----------number of particle species----------
particles
{
  num_species = 1,
}

!----------information for species 1----------

species
{
  name        = 'electrons',
  num_par_max = 3500000,
  rqm         = -1.0,
  num_par_x(1:3) = 2, 1, 1,

  vth(1:3) = 0.0d0 , 0.0d0 , 0.0d0 ,
  vfl(1:3) = 0.0d0 , 0.0d0 , 0.0d0 ,

  den_min = 0.0000001d0,

  interpolation = 'quadratic',
  add_tag = .true.,
}

!----------density profile for this species----------
profile
{
  density=0.000862,
  num_x = 6,

  fx(:,1) = 0.,  0.0000,   1.0,    1.0,    0.0000,   0.0,
   x(:,1) = 0., 806.4000,  807.400,    1.00d5,    1.00d6, 1.1d6,
  fx(:,2) = 0.,  0.0000,   1.0000,    1.0000,    0.0000,   0.0,
   x(:,2) = 0.,  20.0000,   20.1,  1151.88,   1151.98, 1.1d6,
  fx(:,3) = 0.,  0.0000,   1.0000,    1.0000,    0.0000,   0.0,
   x(:,3) = 0.,  20.0000,   20.1,  1151.88,   1151.98, 1.1d6,
}


!----------boundary conditions for this species----------
spe_bound
{
  type(1:2,1) =  0, 0,
  type(1:2,2) =  5, 5,
  type(1:2,3) =  5, 5,
}

!----------diagnostic for this species----------
```

```
diag_species
{
  ! Full grid diagnostics : charge, m, ene, q1, q2, q3, j1, j2, j3

  ndump_fac = 1,              ! this is ignored, no full grid dianostics requested
  ndump_fac_ave = 1,

  reports = "charge, savg",
  n_ave(1:3)=8,4,4,


  ! Phasespaces
  ndump_fac_pha = 1,
  ps_xmin(1:3) =  0.0,  0.0, 0.0,
  ps_xmax(1:3) = 806.4, 1171.88, 1171.88,
  ps_nx(1:3)   = 1000,  300, 300,

  ps_pmin(1:3) = -10.0, -5.0, -5.0,
  ps_pmax(1:3) =  200.0,  5.0,  5.0,
  ps_np(1:3)   =  1000,  200,  200,
  if_ps_p_auto(1:3) = .true., .true., .true.,

  phasespaces = "p1x1", "p2x1", "p2x2", "p1x2", "x1x2", "p3x3", "p1x3",
                "x2x3", "x1x3", "p3x1",

  ! Raw particle dumps
  ndump_fac_raw = 1,

  ! since we are not in a moving window using raw_gamma_limit gives the same end result and is
  ! much faster

  ! raw_math_expr = "p1 > 20.0",

  raw_gamma_limit = 20.0,
}


zpulse
{
  a0 = 2.83,
  omega0 = 1.0,
  pol_type = 0,
  pol = 0.0,
  phase=0.0,

  propagation = "forward",
  lon_type = "polynomial",
  lon_rise = 86.86,
  lon_flat = 0.0,
  lon_fall = 86.86,
  lon_start = 0.5,
  per_type = "gaussian",
  per_w0 = 153.0,
  per_focus = -1.0,

}
```

```
zpulse
{
  a0 = 2.83,
  omega0 = 1.0,
  pol_type = 0,
  pol = 90.0,
  phase=90.0,

  propagation = "forward",
  lon_type = "polynomial",
  lon_rise = 86.86,
  lon_flat = 0.0,
  lon_fall = 86.86,
  lon_start = 0.5,
  per_type = "gaussian",
  per_w0 = 153.0,
  per_focus = -1.0,

}


current{}

smooth
{
  ifsmooth(1) = .true.,
  smooth_level(1)=5,
  swfj(1:3,1,1) = 1,2,1,
  swfj(1:3,2,1) = 1,2,1,
  swfj(1:3,3,1) = 1,2,1,
  swfj(1:3,4,1) = 1,2,1,
  swfj(1:3,5,1) = -5,14,-5,
}

! --------------------- end of osiris input file ---------------
```

lwf-r02.3d

```
!-----------------------------------------------------------------
! LWFA Benchmark for ASCR Joule
! - Run 02
! - K parameter set
!-----------------------------------------------------------------


simulation
{
  ndump_prof = 1000,        ! Profiling info dump frequency

  !  n0 = 1.74d21,
}

!----------the node configuration for this simulation---------
node_conf
{
```

```
  node_number(1:3) =  96, 24, 24,           ! 55296 nodes total

  if_periodic(1:3) = .false., .false., .false.,
}

!----------spatial grid----------
grid
{
  ! 8832/96 = 92 cells/node along x1
  nx_p(1:3) = 8832, 432, 432,

  ! Load info dump frequency
  ndump_global_load = 1000, ! global simulation load (min, max, avg particles / node )
  ndump_node_load   = 10000, ! number of particles per node for all nodes
  ndump_grid_load   = 0,     ! number of particles per cell for all grid points

}

!----------time step and global data dump timestep number----------
time_step
{
  ! Courant condition gives 0.1996426659282
  ! 0.999 * courant = 0.199443023262237

  dt     =  0.199443023262237,

  ! Dumps at every 2000.0 1/\omega_0 (or as close as we can get them)
  ! (rounded down to insure the final dumps get written)
  ndump  =  10027,

}

!----------restart information----------
restart
{
  ! I turned off writing restarts, as per Ken's request, the idea is that this run
  ! should finish within a 14 - 16 hour wall clock time

! write restart at final iteration
  ndump_fac  = 11,
}

!----------spatial limits of the simulations----------
!(note that this includes information about
! the motion of the simulation box)
space
{
  xmin(1:3) =   0.000d0 , 0.000d0, 0.000d0,
  xmax(1:3) =  1766.40, 2041.31, 2041.31,
  if_move= .true., .false., .false.,
}


!----------time limits ----------
time
{
  tmin = 0.0d0,
```

```
  tmax = 9373.8,
}

!----------boundary conditions for em-fields ----------
emf_bound
{
  type(1:2,1) =   0,  0,
  type(1:2,2) =  60, 60,
  type(1:2,3) =  60, 60,
}

!----------diagnostic for electromagnetic fields----------
diag_emf
{
  ndump_fac         = 1,  ! this is ignored, no full grid diagnostics requested
  ndump_fac_ave     = 1,  ! Spatial/time averaged diagnostics
  ndump_fac_lineout = 1,  ! slice and lineout diagnostics

  reports = "e1, savg", "e2, savg", "e3, savg", "e1, line, x1, 150, 150",
            "psi, savg"

  ndump_fac_ene_int = 1,

  n_ave(1:3)        = 4, 2, 2,
}

!----------number of particle species----------
particles
{
  num_species = 1,
}

!----------information for species 1----------

species
{
  name        = 'electrons',
  num_par_max = 3500000,
  rqm         = -1.0,
  num_par_x(1:3) = 1, 2, 2,

  vth(1:3) = 0.0d0 , 0.0d0 , 0.0d0 ,
  vfl(1:3) = 0.0d0 , 0.0d0 , 0.0d0 ,

  den_min = 0.0000001d0,

  interpolation = 'quadratic',
  add_tag = .true.,
}

!----------density profile for this species----------
profile
{
  density=0.000287356,
  num_x = 6,

  fx(:,1) = 0.,  0.0000,   1.0,    1.0,    0.0000,   0.0,
```

```
   x(:,1) = 0., 1766.4000,  1767.400,    1.00d5,    1.00d6, 1.1d6,
  fx(:,2) = 0.,  0.0000,   1.0000,    1.0000,    0.0000,   0.0,
   x(:,2) = 0.,  60.0000,   60.1,  1981.31,   1981.41, 1.1d6,
  fx(:,3) = 0.,  0.0000,   1.0000,    1.0000,    0.0000,   0.0,
   x(:,3) = 0.,  60.0000,   60.1,  1981.31,   1981.41, 1.1d6,
}


!----------boundary conditions for this species----------
spe_bound
{
  type(1:2,1) =  0, 0,
  type(1:2,2) =  5, 5,
  type(1:2,3) =  5, 5,
}

!----------diagnostic for this species----------
diag_species
{
  ! Full grid diagnostics : charge, m, ene, q1, q2, q3, j1, j2, j3

  ndump_fac = 1,              ! this is ignored, no full grid dianostics requested
  ndump_fac_ave = 1,

  reports = "charge, savg",
  n_ave(1:3)=4,2,2,

  ! Phasespaces
  ndump_fac_pha = 1,
  ps_xmin(1:3) =  0.0,  0.0, 0.0,
  ps_xmax(1:3) = 1766.40, 2041.31, 2041.31,
  ps_nx(1:3)   = 1000,  216, 216,

  ps_pmin(1:3) = -10.0, -5.0, -5.0,
  ps_pmax(1:3) =  200.0,  5.0,  5.0,
  ps_np(1:3)   =  1000,  200,  200,
  if_ps_p_auto(1:3) = .true., .true., .true.,

  phasespaces = "p1x1", "p2x1", "p2x2", "p1x2", "x1x2", "p3x3", "p1x3",
                "x2x3", "x1x3", "p3x1",

  ! Raw particle dumps
  ndump_fac_raw = 1,

  ! since we are not in a moving window using raw_gamma_limit gives the same end result and is
  ! much faster

  ! raw_math_expr = "p1 > 20.0",

  raw_gamma_limit = 20.0,
}


zpulse
{
  a0 = 3.2377,
  omega0 = 1.0,
```

```
  pol_type = 0,
  pol = 0.0,
  phase=0.0,

  propagation = "forward",
  lon_type = "polynomial",
  lon_rise = 107.12,
  lon_flat = 0.0,
  lon_fall = 107.12,
  lon_start = 0.5,
  per_type = "gaussian",
  per_w0 = 264.7,
  per_focus = -1.0,

}

zpulse
{
  a0 = 3.2377,
  omega0 = 1.0,
  pol_type = 0,
  pol = 90.0,
  phase=90.0,

  propagation = "forward",
  lon_type = "polynomial",
  lon_rise = 107.12,
  lon_flat = 0.0,
  lon_fall = 107.12,
  lon_start = 0.5,
  per_type = "gaussian",
  per_w0 = 264.7,
  per_focus = -1.0,

}


current{}

smooth
{
  ifsmooth(1) = .true.,
  smooth_level(1)=5,
  swfj(1:3,1,1) = 1,2,1,
  swfj(1:3,2,1) = 1,2,1,
  swfj(1:3,3,1) = 1,2,1,
  swfj(1:3,4,1) = 1,2,1,
  swfj(1:3,5,1) = -5,14,-5,
}

! --------------------- end of osiris input file ---------------
```

lwf-r03.3d

```
!-------------------------------------------------------------
```

```
! LWFA Benchmark for ASCR Joule
! - Run 03
! - M parameter set with moving ions
! - Using 16 particles per cell per species, 32 particles per cell total
!-------------------------------------------------------------


simulation
{
  ndump_prof = 1000,        ! Profiling info dump frequency

  !  n0 = 1.74d21,
}

!----------the node configuration for this simulation---------
node_conf
{

  node_number(1:3) =  96, 24, 24,           ! 55296 nodes total

  ! this is another option
  ! node_number(1:3) =  384, 12, 12,   ! 55296 nodes total

  if_periodic(1:3) = .false., .false., .false.,
}

!----------spatial grid----------
grid
{
  ! 4032/96 = 42 cells/node along x1
  nx_p(1:3) = 4032, 312, 312,

  ! Load info dump frequency
  ndump_global_load = 1000, ! global simulation load (min, max, avg particles / node )
  ndump_node_load   = 10000, ! number of particles per node for all nodes
  ndump_grid_load   = 0,     ! number of particles per cell for all grid points

}

!----------time step and global data dump timestep number----------
time_step
{

  ! Courant condition gives 0.1994353353186
  ! 0.999 * courant = 0.199235899983302

  dt     =  0.199235899983302,

  ! Dumps at every 2000.0 1/\omega_0 (or as close as we can get them)
  ! (rounded down to insure the final dumps get written)
  ndump  =  10038,
}

!----------restart information----------
restart
{
  ! I turned off writing restarts, as per Ken's request, the idea is that this run
```

```
  ! should finish within a 14 - 16 hour wall clock time

! write restart at final iteration
  ndump_fac  = 6,
}

!----------spatial limits of the simulations----------
!(note that this includes information about
! the motion of the simulation box)
space
{
  xmin(1:3) =   0.000d0 , 0.000d0, 0.000d0,
  xmax(1:3) =  806.4, 1171.88, 1171.88,
  if_move= .true., .false., .false.,
}


!----------time limits ----------
time
{
  tmin = 0.0d0,
  tmax = 10360.2,
}

!----------boundary conditions for em-fields ----------
emf_bound
{
  type(1:2,1) =   0,  0,
  type(1:2,2) =  60, 60,
  type(1:2,3) =  60, 60,
}

!----------diagnostic for electromagnetic fields----------
diag_emf
{
  ndump_fac         = 1,  ! this is ignored, no full grid diagnostics requested
  ndump_fac_ave     = 1,  ! Spatial/time averaged diagnostics
  ndump_fac_lineout = 1,  ! slice and lineout diagnostics

  reports = "e1, savg", "e2, savg", "e3, savg", "e1, line, x1, 150, 150",
            "psi, savg", "psi, savg",

  ndump_fac_ene_int = 1,

  n_ave(1:3)        = 4, 2, 2,
}

!----------number of particle species----------
particles
{
  num_species = 2,
}

!----------information for species 1----------

species
{
```

```
  name        = 'electrons',
  num_par_max = 3500000,
  rqm         = -1.0,
  num_par_x(1:3) = 4, 2, 2,

  vth(1:3) = 0.0d0 , 0.0d0 , 0.0d0 ,
  vfl(1:3) = 0.0d0 , 0.0d0 , 0.0d0 ,

  den_min = 0.0000001d0,

  interpolation = 'quadratic',
  add_tag = .true.,
}

!----------density profile for this species----------
profile
{
  density=0.000862,
  num_x = 6,

  fx(:,1) = 0.,  0.0000,   1.0,    1.0,    0.0000,   0.0,
   x(:,1) = 0., 806.4000,  807.400,    1.00d5,    1.00d6, 1.1d6,
  fx(:,2) = 0.,  0.0000,   1.0000,    1.0000,    0.0000,   0.0,
   x(:,2) = 0.,  20.0000,   20.1,  1151.88,   1152.88, 1.1d6,
  fx(:,3) = 0.,  0.0000,   1.0000,    1.0000,    0.0000,   0.0,
   x(:,3) = 0.,  20.0000,   20.1,  1151.88,   1152.88, 1.1d6,
}


!----------boundary conditions for this species----------
spe_bound
{
  type(1:2,1) =  0, 0,
  type(1:2,2) =  5, 5,
  type(1:2,3) =  5, 5,
}

!----------diagnostic for this species----------
diag_species
{
  ! Full grid diagnostics : charge, m, ene, q1, q2, q3, j1, j2, j3

  ndump_fac = 1,              ! this is ignored, no full grid dianostics requested
  ndump_fac_ave = 1,

  reports = "charge, savg",
  n_ave(1:3)=4,2,2,


  ! Phasespaces
  ndump_fac_pha = 1,
  ps_xmin(1:3) =  0.0,  0.0, 0.0,
  ps_xmax(1:3) = 806.4, 1151.88, 1151.88,
  ps_nx(1:3)   = 1000,  312, 312,

  ps_pmin(1:3) = -10.0, -5.0, -5.0,
  ps_pmax(1:3) =  200.0,  5.0,  5.0,
```

```
  ps_np(1:3)   =  1000,  200,  200,
  if_ps_p_auto(1:3) = .true., .true., .true.,

  phasespaces = "p1x1", "p2x1", "p2x2", "p1x2", "x1x2", "p3x3", "p1x3",
                "x2x3", "x1x3", "p3x1",

  ! Raw particle dumps
  ndump_fac_raw = 1,

  ! since we are not in a moving window using raw_gamma_limit gives the same end result and is
  ! much faster

  ! raw_math_expr = "p1 > 20.0",

  raw_gamma_limit = 20.0,
}

!----------information for species 2----------

species
{
  name        = 'ions',
  num_par_max = 3500000,
  rqm         = 1836.1527,
  num_par_x(1:3) = 4, 2, 2,

  vth(1:3) = 0.0d0 , 0.0d0 , 0.0d0 ,
  vfl(1:3) = 0.0d0 , 0.0d0 , 0.0d0 ,

  den_min = 0.0000001d0,

  interpolation = 'quadratic',
  add_tag = .true.,
}

!----------density profile for this species----------
profile
{
  density=0.000862,
  num_x = 6,

  fx(:,1) = 0.,  0.0000,   1.0,    1.0,    0.0000,   0.0,
   x(:,1) = 0., 806.4000,  807.400,    1.00d5,    1.00d6, 1.1d6,
  fx(:,2) = 0.,  0.0000,   1.0000,    1.0000,    0.0000,   0.0,
   x(:,2) = 0.,  20.0000,   20.1,  1151.88,   1152.88, 1.1d6,
  fx(:,3) = 0.,  0.0000,   1.0000,    1.0000,    0.0000,   0.0,
   x(:,3) = 0.,  20.0000,   20.1,  1151.88,   1152.88, 1.1d6,
}


!----------boundary conditions for this species----------
spe_bound
{
  type(1:2,1) =  0, 0,
  type(1:2,2) =  5, 5,
  type(1:2,3) =  5, 5,
}
```

```
!----------diagnostic for this species----------
diag_species
{
  ! Full grid diagnostics : charge, m, ene, q1, q2, q3, j1, j2, j3

  ndump_fac = 1,              ! this is ignored, no full grid dianostics requested
  ndump_fac_ave = 1,

  reports = "charge, savg",
  n_ave(1:3)=4,2,2,


  ! Phasespaces
  ndump_fac_pha = 1,
  ps_xmin(1:3) =  0.0,  0.0, 0.0,
  ps_xmax(1:3) = 806.4, 1151.88, 1151.88,
  ps_nx(1:3)   = 1000,  312, 312,

  ps_pmin(1:3) = -10.0, -5.0, -5.0,
  ps_pmax(1:3) =  200.0,  5.0,  5.0,
  ps_np(1:3)   =  1000,  200,  200,
  if_ps_p_auto(1:3) = .true., .true., .true.,

  phasespaces = "p1x1", "p2x1", "p2x2", "p1x2", "x1x2", "p3x3", "p1x3",
                "x2x3", "x1x3", "p3x1",

  ! Raw particle dumps
  ndump_fac_raw = 1,

  ! since we are not in a moving window using raw_gamma_limit gives the same end result and is
  ! much faster

  ! raw_math_expr = "p1 > 20.0",

  raw_gamma_limit = 20.0,
}


zpulse
{
  a0 = 2.83,
  omega0 = 1.0,
  pol_type = 0,
  pol = 0.0,
  phase=0.0,

  propagation = "forward",
  lon_type = "polynomial",
  lon_rise = 86.86,
  lon_flat = 0.0,
  lon_fall = 86.86,
  lon_start = 0.5,
  per_type = "gaussian",
  per_w0 = 153.0,
  per_focus = -1.0,
```

```
}

zpulse
{
  a0 = 2.83,
  omega0 = 1.0,
  pol_type = 0,
  pol = 90.0,
  phase=90.0,

  propagation = "forward",
  lon_type = "polynomial",
  lon_rise = 86.86,
  lon_flat = 0.0,
  lon_fall = 86.86,
  lon_start = 0.5,
  per_type = "gaussian",
  per_w0 = 153.0,
  per_focus = -1.0,

}


current{}

smooth
{
  ifsmooth(1) = .true.,
  smooth_level(1)=5,
  swfj(1:3,1,1) = 1,2,1,
  swfj(1:3,2,1) = 1,2,1,
  swfj(1:3,3,1) = 1,2,1,
  swfj(1:3,4,1) = 1,2,1,
  swfj(1:3,5,1) = -5,14,-5,
}

! --------------------- end of osiris input file ---------------
```

Q2 Compilation Script

```
#compilers
##########
FPP = gcc -C -E -x assembler-with-cpp
F90 = ftn
LD = ftn

cc  = cc
CC  = CC

# Fortran flags for Pathscale compiler

F90FLAGS = -Ofast
CFLAGS   = -Ofast
LDFLAGS  = -Ofast
```

```
# Object files

OS_VERSION := r362

OBJSBASE = os-sys-multi-c.o os-sys-multi.o os-param.o \
           os-units.o os-math.o os-files.o

OBJSLOG = papi-profile.o os-logprof-papi.o os-logprof.o

OBJSALL =         os-stringutil.o os-util.o  os-restart.o\
  krp-prof.o \
  os-random.o os-nconf.o    os-fparser.o  \
  os-grid.o \
  os-space.o    os-tstep.o  os-dutil-hdf5.o os-dutil.o  \
  os-vdf-define.o os-vdf-comm.o os-vdf-math.o os-vdf-smooth.o \
  os-vdf-average.o os-vdf.o os-vdf-report.o os-vdf-interpolate.o \
  os-wall.o  os-time.o     os-timeavg.o   \
  os-emf-define.o os-emf-lindman.o os-emf-vpml.o os-emf-boundary.o \
  os-emf-solver.o os-emf-interpolate.o os-emf-psi.o os-emf-diagnostics.o \
  os-emf-gridval.o os-emf.o \
  os-current-diagnostics.o  os-current-boundary.o   os-current.o \
  os-spec-define.o  os-spec-push-c.o os-spec-profile.o os-spec-tracks.o os-spec-util.o \
  os-spec-comm.o  os-spec-current-c.o os-spec-current.o os-spec-charge.o \
  os-spec-piston.o  os-spec-rawdiag.o os-spec-phasespace.o os-spec-diagnostics.o os-spec-boundary.o \
  os-spec-loadbalance.o  os-species.o   \
  os-cathode.o  os-cross.o os-dneutral.o   os-neutral.o  \
  os-spec-collisions.o os-particles.o     os-zpulse.o  os-antenna.o \
  os-antenna_array.o

OBJSMAIN =  os-main.o

OBJS    = $(OBJSBASE) $(OBJSLOG) $(OBJSALL) $(OBJSMAIN)


# flags
CF   = $(CFLAGS) -D$(UNDERSCORE)
F90F = $(F90FLAGS)
FPPF = $(FPPPREFIX)-DP_X_DIM=$(OS_DIM) -DOS_REV=\"$(OS_VERSION)\" -D$(UNDERSCORE)
LDF  = $(LDFLAGS)

#c
%.o : ../source/%.c ../config/osiris_config
$(cc) $(CF) $(INCPATH) -c $<

#cpp
%.o : ../source/%.cpp ../config/osiris_config
$(CC) $(CF) $(INCPATH) -c $<

%.o :../source/%.f90 ../config/osiris_config ../source/os-param.f90
$(FPP) $(FPPF) -I../source $< > $(<F)
$(F90) $(F90F) $(INCPATH) -c $(<F)

osiris.e : $(OBJS) ../config/osiris_config
$(LD) $(LDFLAGS) -o $@ $(OBJS) $(LDF)
chmod 755 osiris.e
cp osiris.e ../bin/osiris-$(OS_VERSION)-$(OS_DIM)D.e
```

```
ln -f -s osiris-$(OS_VERSION)-$(OS_DIM)D.e ../bin/osiris-$(OS_DIM)D.e

.PHONY: clean
clean :
touch a.o && rm -f *.o
touch a.f90 && rm -f *.f90
```

Q2 Run Script

```
--- warm.3d.job ---

#PBS -V
#PBS -l walltime=20:00:00,size=55296
#PBS -N warm.3d
#PBS -A CSC074COMP
#PBS -o osiris.stdout
#PBS -e osiris.stderr

cd $PBS_O_WORKDIR
date
time aprun -n 55296 ./osiris-3D.e warm.3d

--- lwfa-r01.3d.job ---

#PBS -V
#PBS -l walltime=20:00:00,size=55296
#PBS -N lwfa-r01.3d
#PBS -A CSC074COMP
#PBS -o osiris.stdout
#PBS -e osiris.stderr

cd $PBS_O_WORKDIR
date
time aprun -n 55296 ./osiris-3D.e lwfa-r01.3d


--- lwfa-r02.3d.job ---

#PBS -V
#PBS -l walltime=20:00:00,size=55296
#PBS -N lwfa-r02.3d
#PBS -A CSC074COMP
#PBS -o osiris.stdout
#PBS -e osiris.stderr

cd $PBS_O_WORKDIR
date
time aprun -n 55296 ./osiris-3D.e lwfa-r02.3d


--- lwfa-r03.3d.job ---

#PBS -V
#PBS -l walltime=20:00:00,size=55296
#PBS -N lwfa-r03.3d
#PBS -A CSC074COMP
#PBS -o osiris.stdout
```

```
#PBS -e osiris.stderr

cd $PBS_O_WORKDIR
date
time aprun -n 55296 ./osiris-3D.e lwfa-r03.3d
```

Q2 Runtime Environment

```
Currently Loaded Modulefiles:
  1) modules/3.1.6
  2) DefApps
  3) torque/2.4.1b1-snap.200905191614
  4) moab/5.3.6
  5) /opt/cray/xt-asyncpe/default/modulefiles/xtpe-istanbul
  6) cray/MySQL/5.0.64-1.0000.2342.16.1
  7) xtpe-target-cnl
  8) xt-service/2.2.73
  9) xt-os/2.2.73
 10) xt-boot/2.2.73
 11) xt-lustre-ss/2.2_1.6.5
 12) cray/job/1.5.5-0.1_2.0202.21413.56.6
 13) cray/csa/3.0.0-1_2.0202.21426.77.6
 14) cray/account/1.0.0-2.0202.19482.49.17
 15) cray/projdb/1.0.0-1.0202.19483.52.1
 16) Base-opts/2.2.73
 17) PrgEnv-pathscale/2.2.73
 18) xt-asyncpe/3.7
 19) xt-libsci/10.4.4
 20) xt-mpt/4.0.0
 21) pmi/1.0-1.0000.7628.10.2.ss
 22) xt-pe/2.2.73
 23) pathscale/3.2.99
 24) hdf5-parallel/1.8.4.1
 25) xt-papi/3.6.2.2

MODULE_VERSION_STACK=3.1.6
LESSKEY=/etc/lesskey.bin
PAPI_POST_LINK_OPTS= -L/opt/xt-tools/papi/3.6.2.2/lib -lpapi -lpfm
NNTPSERVER=news
INFODIR=/usr/local/info:/usr/share/info:/usr/info
HOSTNAME=jaguarpf-login2
GNOME2_PATH=/usr/local:/opt/gnome:/usr
XKEYSYMDB=/usr/X11R6/lib/X11/XKeysymDB
PAPI_VERSION=3.6.2.2
PE_ENV=PATHSCALE
_MODULESBEGINENV_=/ccs/home/zamb/.modulesbeginenv.jaguarpf-login2
HOST=jaguarpf-login2
TERM=xterm-color
SHELL=/bin/bash
XTOS_VERSION=2.2.73
PROFILEREAD=true
HISTSIZE=1000
PERFMON_VERSION=v23
TMPDIR=/tmp/work/zamb
SSH_CLIENT=89.181.204.177 33936 22
MPT_DIR=/opt/cray/mpt/4.0.0/xt/seastar
MORE=-sl
```

```
BOOT_DIR=/opt/xt-boot/2.2.73
QTDIR=/usr/lib/qt3
PRGENV_DIR=/opt/xt-prgenv/2.2.73
SSH_TTY=/dev/pts/7
ASYNCPE_DIR=/opt/cray/xt-asyncpe/3.7
BUILD_OPTS=/opt/cray/xt-asyncpe/3.7/bin/build-opts
INTEL_PRE_COMPILE_OPTS= -msse3
GROFF_NO_SGR=yes
JRE_HOME=/usr/lib/jvm/jre
USER=zamb
CRAY_HDF5_PARALLEL_VERSION=1.8.4.1
LD_LIBRARY_PATH=/opt/xt-tools/papi/3.6.2.2/lib:
                /opt/cray/hdf5-parallel/1.8.4.1/hdf5-parallel-pathscale/lib:
                /opt/cray/pmi/1.0-1.0000.7628.10.2.ss/lib64:
                /opt/cray/mpt/4.0.0/xt/seastar/mpich2-pathscale/lib:
                /opt/cray/mpt/4.0.0/xt/sma/lib:
                /opt/xt-pe/2.2.73/lib:/opt/pathscale/lib/3.2.99:
                /opt/cray/projdb/1.0.0-1.0202.19483.52.1/lib64:
                /opt/cray/csa/3.0.0-1_2.0202.21426.77.6/lib64:
                /opt/cray/job/1.5.5-0.1_2.0202.21413.56.6/lib64:
                /opt/cray/MySQL/5.0.64-1.0000.2342.16.1/lib64/mysql:
                /opt/cray/MySQL/5.0.64-1.0000.2342.16.1/lib64
LS_COLORS=
LC_PE_ENV=pathscale
WORKDIR=/tmp/work/zamb
TVDSVRLAUNCHCMD=ssh
XNLSPATH=/usr/X11R6/lib/X11/nls
MPICH_DIR=/opt/cray/mpt/4.0.0/xt/seastar/mpich2-pathscale
ENV=/etc/bash.bashrc
HOSTTYPE=x86_64
RCLOCAL_PRGENV=true
MPT_VERSION=4.0.0
PE_PRODUCT_LIST=CRAY_HDF5:ASYNCPE:LIBSCI:CRAY_PMI:XTMPT:
                PE:PATHSCALE:LUSTRE:XTPE_ISTANBUL:PAPI
FROM_HEADER=
FFTW_SYSTEM_WISDOM_DIR=/opt/xt-libsci/10.4.4
PAGER=less
PAPI_INCLUDE_OPTS= -I/opt/xt-tools/papi/3.6.2.2/include
OS_DIR=/opt/xt-os/2.2.73
CSHEDIT=emacs
MPICHBASEDIR=/opt/cray/mpt/4.0.0/xt
XDG_CONFIG_DIRS=/usr/local/etc/xdg/:/etc/xdg/:/etc/opt/gnome/xdg/
PATHSCALE_VERSION=3.2.99
MINICOM=-c on
YOD_LOGFILE=syslog
APRUN_XFER_LIMITS=0
MODULE_VERSION=3.1.6
MAIL=/var/mail/zamb
PATH=/opt/xt-tools/papi/3.6.2.2/bin:/opt/cray/hdf5-parallel/1.8.4.1/hdf5-pathscale/bin:
     /opt/cray/hdf5-parallel/1.8.4.1/hdf5-parallel-pathscale/bin:
     /opt/cray/xt-asyncpe/3.7/bin:/opt/cray/pmi/1.0-1.0000.7628.10.2.ss/bin:
     /opt/cray/projdb/1.0.0-1.0202.19483.52.1/bin:
     /opt/cray/account/1.0.0-2.0202.19482.49.17/bin:
     /opt/cray/csa/3.0.0-1_2.0202.21426.77.6/sbin:
     /opt/cray/csa/3.0.0-1_2.0202.21426.77.6/bin:
     /opt/cray/job/1.5.5-0.1_2.0202.21413.56.6/bin:/opt/xt-lustre-ss/2.2_1.6.5/usr/sbin:
     /opt/xt-lustre-ss/2.2_1.6.5/usr/bin:/opt/xt-boot/2.2.73/bin/snos64:
```

```
     /opt/xt-os/2.2.73/bin/snos64:/opt/xt-service/2.2.73/bin/snos64:
     /opt/xt-prgenv/2.2.73/bin:/opt/cray/MySQL/5.0.64-1.0000.2342.16.1/sbin:
     /opt/cray/MySQL/5.0.64-1.0000.2342.16.1/bin:/opt/moab/default/bin:
     /opt/torque/default/bin:/sw/xt5/bin:/opt/modules/3.1.6/bin:/ccs/home/zamb/bin:
     /usr/local/bin:/usr/bin:/usr/X11R6/bin:/bin:/usr/games:/opt/bin:/opt/gnome/bin:
     /opt/kde3/bin:/usr/lib/jvm/jre/bin:/usr/lib/mit/bin:/usr/lib/mit/sbin:
     /opt/pathscale/bin:.:/usr/lib/qt3/bin:/opt/bin:/opt/public/bin
CPU=x86_64
CRAY_HDF5_DIR=/opt/cray/hdf5-parallel/1.8.4.1
JAVA_BINDIR=/usr/lib/jvm/jre/bin
ASYNCPE_VERSION=3.7
XTPE_COMPILE_TARGET=linux
RCLOCAL_MYSQL=true
INPUTRC=/etc/inputrc
PWD=/ccs/home/zamb
_LMFILES_=/opt/modulefiles/modules/3.1.6:/sw/xt5/modulefiles/DefApps:
          /opt/modulefiles/torque/2.4.1b1-snap.200905191614:
          /opt/modulefiles/moab/5.3.6:
          /opt/cray/xt-asyncpe/default/modulefiles/xtpe-istanbul:
          /opt/modulefiles/cray/MySQL/5.0.64-1.0000.2342.16.1:
          /opt/modulefiles/xtpe-target-cnl:/opt/modulefiles/xt-service/2.2.73:
          /opt/modulefiles/xt-os/2.2.73:/opt/modulefiles/xt-boot/2.2.73:
          /opt/modulefiles/xt-lustre-ss/2.2_1.6.5:
          /opt/modulefiles/cray/job/1.5.5-0.1_2.0202.21413.56.6:
          /opt/modulefiles/cray/csa/3.0.0-1_2.0202.21426.77.6:
          /opt/modulefiles/cray/account/1.0.0-2.0202.19482.49.17:
          /opt/modulefiles/cray/projdb/1.0.0-1.0202.19483.52.1:
          /opt/modulefiles/Base-opts/2.2.73:
          /opt/modulefiles/PrgEnv-pathscale/2.2.73:
          /opt/modulefiles/xt-asyncpe/3.7:/opt/modulefiles/xt-libsci/10.4.4:
          /opt/modulefiles/xt-mpt/4.0.0:
          /opt/cray/modulefiles/pmi/1.0-1.0000.7628.10.2.ss:
          /opt/modulefiles/xt-pe/2.2.73:/opt/modulefiles/pathscale/3.2.99:
          /opt/modulefiles/hdf5-parallel/1.8.4.1:/opt/modulefiles/xt-papi/3.6.2.2
JAVA_HOME=/usr/lib/jvm/jre
SYSTEM_USERDIR=/tmp/work/zamb
LANG=en_US.UTF-8
PATHSCALE_PATH=/opt/pathscale
MODULEPATH=/opt/cray/xt-asyncpe/3.7/modulefiles:/opt/modulefiles:
           /opt/cray/modulefiles:/opt/modules/3.1.6:/sw/xt5/modulefiles
PYTHONSTARTUP=/etc/pythonstart
PATHSCALE_PRE_COMPILE_OPTS= -march=barcelona
LOADEDMODULES=modules/3.1.6:DefApps:torque/2.4.1b1-snap.200905191614:
              moab/5.3.6:xtpe-istanbul:cray/MySQL/5.0.64-1.0000.2342.16.1:
              xtpe-target-cnl:xt-service/2.2.73:xt-os/2.2.73:xt-boot/2.2.73:
              xt-lustre-ss/2.2_1.6.5:cray/job/1.5.5-0.1_2.0202.21413.56.6:
              cray/csa/3.0.0-1_2.0202.21426.77.6:
              cray/account/1.0.0-2.0202.19482.49.17:
              cray/projdb/1.0.0-1.0202.19483.52.1:Base-opts/2.2.73:
              PrgEnv-pathscale/2.2.73:xt-asyncpe/3.7:xt-libsci/10.4.4:
              xt-mpt/4.0.0:1.0-1.0000.7628.10.2.ss:xt-pe/2.2.73:
              pathscale/3.2.99:hdf5-parallel/1.8.4.1:xt-papi/3.6.2.2
CRAY_PMI_POST_LINK_OPTS=$(pkg-config cray-pmi --libs-only-L)
                       -Wl,-rpath=/opt/cray/pmi/1.0-1.0000.7628.10.2.ss/lib64
TEXINPUTS=:/ccs/home/zamb/.TeX:/usr/share/doc/.TeX:/usr/doc/.TeX
QT_SYSTEM_DIR=/usr/share/desktop-data
SHLVL=1
```

```
HOME=/ccs/home/zamb
PTL_SNOS_NAL=SS
LESS_ADVANCED_PREPROCESSOR=no
OSTYPE=linux
SE_DIR=/opt/xt-service/2.2.73
CRAY_PMI_INCLUDE_OPTS=$(pkg-config cray-pmi --cflags)
LIBLUSTRE_DEBUG_CONSOLE=0
XTPE_ISTANBUL_ENABLED=ON
LS_OPTIONS=-N --color=none -T 0
WINDOWMANAGER=
GTK_PATH=/usr/local/lib/gtk-2.0:/opt/gnome/lib/gtk-2.0:/usr/lib/gtk-2.0
G_FILENAME_ENCODING=@locale,UTF-8,ISO-8859-15,CP1252
LESS=-M -I
MACHTYPE=x86_64-suse-linux
LOGNAME=zamb
GTK_PATH64=/usr/local/lib64/gtk-2.0:/opt/gnome/lib64/gtk-2.0:/usr/lib64/gtk-2.0
CVS_RSH=ssh
XDG_DATA_DIRS=/usr/local/share/:/usr/share/:/etc/opt/kde3/share/:
              /opt/kde3/share/:/opt/gnome/share/
ACLOCAL_FLAGS=-I /opt/gnome/share/aclocal
SSH_CONNECTION=89.181.204.177 33936 160.91.205.195 22
PE_POST_LINK_OPTS=-Wl,-rpath=/opt/xt-pe/2.2.73/lib
MODULESHOME=/opt/modules/3.1.6
PKG_CONFIG_PATH=/opt/cray/pmi/1.0-1.0000.7628.10.2.ss/lib64/pkgconfig:
                /opt/cray/projdb/1.0.0-1.0202.19483.52.1/lib64/pkgconfig:
                /opt/cray/MySQL/5.0.64-1.0000.2342.16.1/lib64/pkgconfig:
                /usr/local/lib/pkgconfig:/usr/local/share/pkgconfig:
                /usr/lib64/pkgconfig:/usr/share/pkgconfig:
                /opt/kde3/lib64/pkgconfig:/opt/gnome/lib64/pkgconfig:
                /opt/gnome/lib64/pkgconfig:/opt/gnome/share/pkgconfig
LESSOPEN=lessopen.sh %s
LIBSCI_BASE_DIR=/opt/xt-libsci/10.4.4
LIBSCI_VERSION=10.4.4
INFOPATH=/usr/local/info:/usr/share/info:/usr/info:/opt/gnome/share/info
DISPLAY=localhost:13.0
XAUTHLOCALHOSTNAME=jaguarpf-login2
PE_DIR=/opt/xt-pe/2.2.73
GNU_PRE_COMPILE_OPTS= -march=amdfam10
LESSCLOSE=lessclose.sh %s %s
G_BROKEN_FILENAMES=1
LUSTRE_DIR=/opt/xt-lustre-ss/2.2_1.6.5
JAVA_ROOT=/usr/lib/jvm/jre
COLORTERM=1
_=/usr/bin/env
zamb@jaguarpf-login2:~>
```

Q4 Input Settings

Note that the `frozen` benchmarks described in the text are printed here first. These runs were added in Q3 and Q4 and are not referenced in the Q2 appendix data for OSIRIS.

frozen.s1.3d

```
!-----------------------------------------------------------------------------------------
! Frozen Plasma Benchmark for ASCR Joule
```

```
! -  Linear Interpolation
!-----------------------------------------------------------------------------------------

simulation
{
  ndump_prof = 100,        ! Profiling info dump frequency

  wall_clock_limit = "00:50:00",
  wall_clock_check = 100,

}

node_conf
{
  node_number(1:3) =  96, 96, 24,! 221184 nodes total
  if_periodic(1:3) = .true., .true., .true.,
}


!----------spatial grid----------
grid
{
  nx_p(1:3) = 12288, 12288, 1536,

  ! Load info dump frequency
  ndump_global_load = 100,! global simulation load (min, max, avg particles / node )
  ndump_node_load   = 250,  ! number of particles per node for all nodes
  ndump_grid_load   = 0,    ! number of particles per cell for all grid points

}


!----------time step and global data dump timestep number----------
time_step
{
  dt     =   0.05,
  ndump  =    0,
}


!----------spatial limits of the simulations----------
space
{
  xmin(1:3) =    0.0, 0.0,  0.0,
  xmax(1:3) =   1228.8 ,  1228.8 ,  153.6 ,

}


!----------time limits ----------
time
{
  tmin = 0.0,
  tmax = 1000.0,
}

!----------boundary conditions for em-fields ----------
```

```
emf_bound
{
  ! these aren't necessary since we're using periodic
  ! boundaries
}


!----------number of particle species----------
particles
{
  num_species = 1,                        ! use 1 particle species
  interpolation = "linear",

}


!----------information for species 1----------

species
{
  name = "electrons" ,

  num_par_x(1:3) = 2, 2, 2,

  num_par_max = 10000000,

  rqm=-1.0,

  vth(1:3) =  0.0 , 0.0 , 0.0 ,
  vfl(1:3) =  0.0 , 0.0 , 0.0 ,

}

spe_bound
{
  ! these aren't necessary since we're using periodic
  ! boundaries
}



! --------------------- end of osiris input file ---------------
```

frozen.s2.3d

```
!-----------------------------------------------------------------------------------------
! Frozen Plasma Benchmark for ASCR Joule
! -  Quadratic Interpolation
!-----------------------------------------------------------------------------------------

simulation
{
  ndump_prof = 100,        ! Profiling info dump frequency

  wall_clock_limit = "00:50:00",
  wall_clock_check = 100,
```

```
}

node_conf
{
  node_number(1:3) =  96, 96, 24,! 221184 nodes total
  if_periodic(1:3) = .true., .true., .true.,
}


!----------spatial grid----------
grid
{
  nx_p(1:3) = 12288, 12288, 1536,

  ! Load info dump frequency
  ndump_global_load = 100,! global simulation load (min, max, avg particles / node )
  ndump_node_load   = 250,  ! number of particles per node for all nodes
  ndump_grid_load   = 0,    ! number of particles per cell for all grid points

}


!----------time step and global data dump timestep number----------
time_step
{
  dt     =   0.05,
  ndump  =    0,
}


!----------spatial limits of the simulations----------
space
{
  xmin(1:3) =    0.0, 0.0,  0.0,
  xmax(1:3) =   1228.8 ,  1228.8 ,  153.6 ,

}


!----------time limits ----------
time
{
  tmin = 0.0,
  tmax = 1000.0,
}

!----------boundary conditions for em-fields ----------
emf_bound
{
  ! these aren't necessary since we're using periodic
  ! boundaries
}


!----------number of particle species----------
particles
```

```
{
  num_species = 1,                        ! use 1 particle species
  interpolation = "quadratic",

}


!----------information for species 1----------

species
{
  name = "electrons" ,

  num_par_x(1:3) = 2, 2, 2,

  num_par_max = 10000000,

  rqm=-1.0,

  vth(1:3) =  0.0 , 0.0 , 0.0 ,
  vfl(1:3) =  0.0 , 0.0 , 0.0 ,

}

spe_bound
{
  ! these aren't necessary since we're using periodic
  ! boundaries
}



! --------------------- end of osiris input file ---------------
```

warm.3d

```
!-----------------------------------------------------------------------------------------
! Warm Plasma Benchmark for ASCR Joule
!-----------------------------------------------------------------------------------------

simulation
{
  ndump_prof = 100,        ! Profiling info dump frequency

  wall_clock_limit = "00:50:00",
  wall_clock_check = 100,

}

node_conf
{
  node_number(1:3) =  48, 48, 24,! 55296 nodes total
  if_periodic(1:3) = .true., .true., .true.,
}
```

```
!----------spatial grid----------
grid
{
  nx_p(1:3) = 6144, 6144, 1536,

  ! Load info dump frequency
  ndump_global_load = 100,! global simulation load (min, max, avg particles / node )
  ndump_node_load   = 250,  ! number of particles per node for all nodes
  ndump_grid_load   = 0,    ! number of particles per cell for all grid points

}


!----------time step and global data dump timestep number----------
time_step
{
  dt     =   0.05,
  ndump  =    0,
}


!----------spatial limits of the simulations----------
space
{
  xmin(1:3) =    0.0, 0.0,  0.0,
  xmax(1:3) =   614.4 ,  614.4 ,  153.6 ,

}


!----------time limits ----------
time
{
  tmin = 0.0,
  tmax = 1000.0,
}

!----------boundary conditions for em-fields ----------
emf_bound
{
  ! these aren't necessary since we're using periodic
  ! boundaries
}


!----------number of particle species----------
particles
{
  num_species = 1,                       ! use 1 particle species
  interpolation = "quadratic",

}


!----------information for species 1----------

species
```

```
{
  name = "electrons" ,

  num_par_x(1:3) = 2, 2, 2,

  num_par_max = 10000000,

  rqm=-1.0,

  vth(1:3) =  0.01 , 0.01 , 0.01 ,
  vfl(1:3) =  0.0 , 0.0 , 0.0 ,

}

spe_bound
{
  ! these aren't necessary since we're using periodic
  ! boundaries
}



! --------------------- end of osiris input file ---------------
```

warm.strong.3d

```
!-----------------------------------------------------------------------------------------
! Warm Plasma Benchmark for ASCR Joule
! -  Strong scaling
!-----------------------------------------------------------------------------------------

simulation
{
  ndump_prof = 100,        ! Profiling info dump frequency

  wall_clock_limit = "00:50:00",
  wall_clock_check = 100,

}

node_conf
{
  node_number(1:3) =  96, 96, 24,! 221184 nodes total
  if_periodic(1:3) = .true., .true., .true.,
}


!----------spatial grid----------
grid
{
  nx_p(1:3) = 6144, 6144, 1536,

  ! Load info dump frequency
  ndump_global_load = 100,! global simulation load (min, max, avg particles / node )
  ndump_node_load   = 250,  ! number of particles per node for all nodes
  ndump_grid_load   = 0,    ! number of particles per cell for all grid points
```

```
}


!----------time step and global data dump timestep number----------
time_step
{
  dt     =   0.05,
  ndump  =    0,
}


!----------spatial limits of the simulations----------
space
{
  xmin(1:3) =    0.0, 0.0,  0.0,
  xmax(1:3) =   614.4 ,  614.4 ,  153.6 ,

}


!----------time limits ----------
time
{
  tmin = 0.0,
  tmax = 1000.0,
}

!----------boundary conditions for em-fields ----------
emf_bound
{
  ! these aren't necessary since we're using periodic
  ! boundaries
}


!----------number of particle species----------
particles
{
  num_species = 1,                       ! use 1 particle species
  interpolation = "quadratic",

}


!----------information for species 1----------

species
{
  name = "electrons" ,

  num_par_x(1:3) = 2, 2, 2,

  num_par_max = 10000000,

  rqm=-1.0,
```

```
  vth(1:3) =  0.01 , 0.01 , 0.01 ,
  vfl(1:3) =  0.0 , 0.0 , 0.0 ,

}

spe_bound
{
  ! these aren't necessary since we're using periodic
  ! boundaries
}



! --------------------- end of osiris input file ---------------
```

warm.weak.3d

```
!-----------------------------------------------------------------------------------------
! Warm Plasma Benchmark for ASCR Joule
! -  Weak scaling
!-----------------------------------------------------------------------------------------

simulation
{
  ndump_prof = 100,        ! Profiling info dump frequency

  wall_clock_limit = "00:50:00",
  wall_clock_check = 100,

}

node_conf
{
  node_number(1:3) =  96, 96, 24,! 221184 nodes total
  if_periodic(1:3) = .true., .true., .true.,
}


!----------spatial grid----------
grid
{
  nx_p(1:3) = 12288, 12288, 1536,

  ! Load info dump frequency
  ndump_global_load = 100,! global simulation load (min, max, avg particles / node )
  ndump_node_load   = 250,  ! number of particles per node for all nodes
  ndump_grid_load   = 0,    ! number of particles per cell for all grid points

}


!----------time step and global data dump timestep number----------
time_step
{
  dt     =   0.05,
  ndump  =    0,
```

```
}


!----------spatial limits of the simulations----------
space
{
  xmin(1:3) =    0.0, 0.0,  0.0,
  xmax(1:3) =   1228.8 ,  1228.8 ,  153.6 ,

}


!----------time limits ----------
time
{
  tmin = 0.0,
  tmax = 1000.0,
}

!----------boundary conditions for em-fields ----------
emf_bound
{
  ! these aren't necessary since we're using periodic
  ! boundaries
}


!----------number of particle species----------
particles
{
  num_species = 1,                        ! use 1 particle species
  interpolation = "quadratic",

}


!----------information for species 1----------

species
{
  name = "electrons" ,

  num_par_x(1:3) = 2, 2, 2,

  num_par_max = 10000000,

  rqm=-1.0,

  vth(1:3) =  0.01 , 0.01 , 0.01 ,
  vfl(1:3) =  0.0 , 0.0 , 0.0 ,

}

spe_bound
{
  ! these aren't necessary since we're using periodic
  ! boundaries
```

```
}


! -------------------- end of osiris input file ---------------
```

lwfa-r01.3d

```
!-----------------------------------------------------------------------------------------
! LWFA Benchmark for ASCR Joule
! - Strong scaling of Run 01
!-----------------------------------------------------------------------------------------


simulation
{
  ndump_prof = 1000,        ! Profiling info dump frequency

  wall_clock_limit = "16:00:00",
  wall_clock_check = 100,

  !  n0 = 1.74d21,
}

!----------the node configuration for this simulation---------
node_conf
{

   node_number(1:3) =  144, 8, 8,   ! 55296 nodes total

  if_periodic(1:3) = .false., .true., .true.,

  n_threads = 6,
}

!----------spatial grid----------
grid
{
  ! 8064/96 = 84 cells/node along x1
  nx_p(1:3) = 8064, 480, 480,

  ! Load info dump frequency
  ndump_global_load = 100, ! global simulation load (min, max, avg particles / node )
  ndump_node_load   = 10000, ! number of particles per node for all nodes
  ndump_grid_load   = 0,     ! number of particles per cell for all grid points


}

!----------time step and global data dump timestep number----------
time_step
{
  ! Courant condition gives 0.0998326502998
  ! 0.999 * courant = 0.0997328176495005

  dt     =  0.0997328176495005,
```

```
  ! Dumps at every 2000.0 1/\omega_0 (or as close as we can get them)
  ! (rounded down to insure the final dumps get written)
  ndump  =  20053,
}

!----------restart information----------
restart
{
  ! Normal restart writing is off, the code will write checkpoint information
  ! if it goes over the wall clock limit
}

!----------spatial limits of the simulations----------
!(note that this includes information about
! the motion of the simulation box)
space
{
  xmin(1:3) =   0.000d0 , 0.000d0, 0.000d0,
  xmax(1:3) =  806.4, 1171.88, 1171.88,
  if_move= .true., .false., .false.,
}


!----------time limits ----------
time
{
  tmin = 0.0d0,
  tmax  = 40000.0,
}

!----------boundary conditions for em-fields ----------
emf_bound
{
}

!----------diagnostic for electromagnetic fields----------
diag_emf
{
  ndump_fac         = 1,  ! this is ignored, no full grid diagnostics requested
  ndump_fac_ave     = 1,  ! Spatial/time averaged diagnostics
  ndump_fac_lineout = 1,  ! slice and lineout diagnostics

  reports = "e1, savg", "e2, savg", "e3, savg", "e1, line, x1, 150, 150",
            "e1, slice, x3, 240", "psi, savg",

  ndump_fac_ene_int = 1,

  n_ave(1:3)        = 8, 4, 4,
}

!----------number of particle species----------
particles
{
  num_species = 1,

  interpolation = 'quadratic',
```

```
}

!----------information for species 1----------

species
{
  name        = 'electrons',
  num_par_max = 3500000,
  rqm         = -1.0,
  num_par_x(1:3) = 2, 1, 1,

  vth(1:3) = 0.0d0 , 0.0d0 , 0.0d0 ,
  vfl(1:3) = 0.0d0 , 0.0d0 , 0.0d0 ,

  den_min = 0.0000001d0,


  ! Disabled particle tags for the benchmark (less communication)
  ! add_tag = .true.,
}

!----------density profile for this species----------
profile
{
  density=0.000862,
  num_x = 6,

  fx(:,1) = 0.,  0.0000,   1.0,    1.0,    0.0000,   0.0,
   x(:,1) = 0., 806.4000,  807.400,    1.00d5,    1.00d6, 1.1d6,
  fx(:,2) = 0.,  0.0000,   1.0000,    1.0000,    0.0000,   0.0,
   x(:,2) = 0.,  20.0000,   20.1,  1151.88,   1151.98, 1.1d6,
  fx(:,3) = 0.,  0.0000,   1.0000,    1.0000,    0.0000,   0.0,
   x(:,3) = 0.,  20.0000,   20.1,  1151.88,   1151.98, 1.1d6,
}


!----------boundary conditions for this species----------
spe_bound
{
  type(1:2,1) =  0, 0,
  type(1:2,2) =  5, 5,
  type(1:2,3) =  5, 5,
}

!----------diagnostic for this species----------
diag_species
{
  ! Full grid diagnostics : charge, m, ene, q1, q2, q3, j1, j2, j3

  ndump_fac = 1,              ! this is ignored, no full grid dianostics requested
  ndump_fac_ave = 1,
  ndump_fac_lineout = 1,  ! slice and lineout diagnostics

  reports = "charge, savg", "charge, slice, x3, 240",
  n_ave(1:3)=8,4,4,
```

```
  ! Phasespaces
  ndump_fac_pha = 1,
  ps_xmin(1:3) =  0.0,  0.0, 0.0,
  ps_xmax(1:3) = 806.4, 1171.88, 1171.88,
  ps_nx(1:3)   = 1000,  300, 300,

  ps_pmin(1:3) = -10.0, -5.0, -5.0,
  ps_pmax(1:3) =  200.0,  5.0,  5.0,
  ps_np(1:3)   =  1000,  200,  200,
  if_ps_p_auto(1:3) = .true., .true., .true.,

  phasespaces = "p1x1", "p2x1", "p2x2", "p1x2", "x1x2", "p3x3", "p1x3",
                "x2x3", "x1x3", "p3x1",

  ! Raw particle dumps
  ndump_fac_raw = 1,

  ! since we are not in a moving window using raw_gamma_limit gives the same end result and is
  ! much faster

  ! raw_math_expr = "p1 > 20.0",

  raw_gamma_limit = 20.0,
}


zpulse
{
  a0 = 2.83,
  omega0 = 1.0,
  pol_type = 0,
  pol = 0.0,
  phase=0.0,

  propagation = "forward",
  lon_type = "polynomial",
  lon_rise = 86.86,
  lon_flat = 0.0,
  lon_fall = 86.86,
  lon_start = 0.5,
  per_type = "gaussian",
  per_w0 = 153.0,
  per_focus = -1.0,

}

zpulse
{
  a0 = 2.83,
  omega0 = 1.0,
  pol_type = 0,
  pol = 90.0,
  phase=90.0,

  propagation = "forward",
  lon_type = "polynomial",
```

```
  lon_rise = 86.86,
  lon_flat = 0.0,
  lon_fall = 86.86,
  lon_start = 0.5,
  per_type = "gaussian",
  per_w0 = 153.0,
  per_focus = -1.0,

}


current{}

smooth
{
  type(1) = "custom",
  order(1) = 5,
  swfj(1:3,1,1) = 1,2,1,
  swfj(1:3,2,1) = 1,2,1,
  swfj(1:3,3,1) = 1,2,1,
  swfj(1:3,4,1) = 1,2,1,
  swfj(1:3,5,1) = -5,14,-5,
}

! -------------------- end of osiris input file ---------------
```

lwfa-r01.strong.3d

```
cat: lwf-r01.3d: No such file or directory
!-----------------------------------------------------------------------------------------
! LWFA Benchmark for ASCR Joule
! - Strong scaling of Run 01
!-----------------------------------------------------------------------------------------


simulation
{
  ndump_prof = 1000,        ! Profiling info dump frequency

  wall_clock_limit = "2:20:00",
  wall_clock_check = 100,

  !  n0 = 1.74d21,
}

!----------the node configuration for this simulation---------
node_conf
{

   node_number(1:3) =  576, 8, 8,   ! 221184 nodes total

  if_periodic(1:3) = .false., .true., .true.,

  n_threads = 6,
}
```

```
!----------spatial grid----------
grid
{
  ! 8064/96 = 84 cells/node along x1
  nx_p(1:3) = 8064, 480, 480,

  ! Load info dump frequency
  ndump_global_load = 100, ! global simulation load (min, max, avg particles / node )
  ndump_node_load   = 10000, ! number of particles per node for all nodes
  ndump_grid_load   = 0,     ! number of particles per cell for all grid points

  ! lb_type = "dynamic",
  ! load_balance(1:3) = .true., .false., .false.,
  ! n_dynamic = 1000,
  ! lb_gather = "sum",

}

!----------time step and global data dump timestep number----------
time_step
{
  ! Courant condition gives 0.0998326502998
  ! 0.999 * courant = 0.0997328176495005

  dt     =  0.0997328176495005,

  ! Dumps at every 2000.0 1/\omega_0 (or as close as we can get them)
  ! (rounded down to insure the final dumps get written)
  ndump  =  20053,
}

!----------restart information----------
restart
{
  ! Normal restart writing is off, the code will write checkpoint information
  ! if it goes over the wall clock limit
}

!----------spatial limits of the simulations----------
!(note that this includes information about
! the motion of the simulation box)
space
{
  xmin(1:3) =   0.000d0 , 0.000d0, 0.000d0,
  xmax(1:3) =  806.4, 1171.88, 1171.88,
  if_move= .true., .false., .false.,
}


!----------time limits ----------
time
{
  tmin = 0.0d0,
  tmax  = 40000.0,
}

!----------boundary conditions for em-fields ----------
```

```
emf_bound
{
}

!----------diagnostic for electromagnetic fields----------
diag_emf
{
  ndump_fac         = 1,  ! this is ignored, no full grid diagnostics requested
  ndump_fac_ave     = 1,  ! Spatial/time averaged diagnostics
  ndump_fac_lineout = 1,  ! slice and lineout diagnostics

  reports = "e1, savg", "e2, savg", "e3, savg", "e1, line, x1, 150, 150",
            "e1, slice, x3, 240", "psi, savg",

  ndump_fac_ene_int = 1,

  n_ave(1:3)        = 8, 4, 4,
}

!----------number of particle species----------
particles
{
  num_species = 1,

  interpolation = 'quadratic',

}

!----------information for species 1----------

species
{
  name        = 'electrons',
  num_par_max = 3500000,
  rqm         = -1.0,
  num_par_x(1:3) = 2, 1, 1,

  vth(1:3) = 0.0d0 , 0.0d0 , 0.0d0 ,
  vfl(1:3) = 0.0d0 , 0.0d0 , 0.0d0 ,

  den_min = 0.0000001d0,


  ! Disabled particle tags for the benchmark (less communication)
  ! add_tag = .true.,
}

!----------density profile for this species----------
profile
{
  density=0.000862,
  num_x = 6,

  fx(:,1) = 0.,  0.0000,   1.0,    1.0,    0.0000,   0.0,
   x(:,1) = 0., 806.4000,  807.400,    1.00d5,    1.00d6, 1.1d6,
  fx(:,2) = 0.,  0.0000,   1.0000,    1.0000,    0.0000,   0.0,
   x(:,2) = 0.,  20.0000,   20.1,  1151.88,   1151.98, 1.1d6,
```

```
  fx(:,3) = 0.,  0.0000,   1.0000,    1.0000,    0.0000,   0.0,
   x(:,3) = 0.,  20.0000,   20.1,  1151.88,   1151.98, 1.1d6,
}


!----------boundary conditions for this species----------
spe_bound
{
  type(1:2,1) =  0, 0,
  type(1:2,2) =  5, 5,
  type(1:2,3) =  5, 5,
}

!----------diagnostic for this species----------
diag_species
{
  ! Full grid diagnostics : charge, m, ene, q1, q2, q3, j1, j2, j3

  ndump_fac = 1,              ! this is ignored, no full grid dianostics requested
  ndump_fac_ave = 1,
  ndump_fac_lineout = 1,  ! slice and lineout diagnostics

  reports = "charge, savg", "charge, slice, x3, 240",
  n_ave(1:3)=8,4,4,


  ! Phasespaces
  ndump_fac_pha = 1,
  ps_xmin(1:3) =  0.0,  0.0, 0.0,
  ps_xmax(1:3) = 806.4, 1171.88, 1171.88,
  ps_nx(1:3)   = 1000,  300, 300,

  ps_pmin(1:3) = -10.0, -5.0, -5.0,
  ps_pmax(1:3) =  200.0,  5.0,  5.0,
  ps_np(1:3)   =  1000,  200,  200,
  if_ps_p_auto(1:3) = .true., .true., .true.,

  phasespaces = "p1x1", "p2x1", "p2x2", "p1x2", "x1x2", "p3x3", "p1x3",
                "x2x3", "x1x3", "p3x1",

  ! Raw particle dumps
  ndump_fac_raw = 1,

  ! since we are not in a moving window using raw_gamma_limit gives the same end result and is
  ! much faster

  ! raw_math_expr = "p1 > 20.0",

  raw_gamma_limit = 20.0,
}


zpulse
{
  a0 = 2.83,
  omega0 = 1.0,
  pol_type = 0,
```

```
  pol = 0.0,
  phase=0.0,

  propagation = "forward",
  lon_type = "polynomial",
  lon_rise = 86.86,
  lon_flat = 0.0,
  lon_fall = 86.86,
  lon_start = 0.5,
  per_type = "gaussian",
  per_w0 = 153.0,
  per_focus = -1.0,

}

zpulse
{
  a0 = 2.83,
  omega0 = 1.0,
  pol_type = 0,
  pol = 90.0,
  phase=90.0,

  propagation = "forward",
  lon_type = "polynomial",
  lon_rise = 86.86,
  lon_flat = 0.0,
  lon_fall = 86.86,
  lon_start = 0.5,
  per_type = "gaussian",
  per_w0 = 153.0,
  per_focus = -1.0,

}


current{}

smooth
{
  type(1) = "custom",
  order(1) = 5,
  swfj(1:3,1,1) = 1,2,1,
  swfj(1:3,2,1) = 1,2,1,
  swfj(1:3,3,1) = 1,2,1,
  swfj(1:3,4,1) = 1,2,1,
  swfj(1:3,5,1) = -5,14,-5,
}

! --------------------- end of osiris input file ---------------
```

lwfa-r02.3d

```
!-----------------------------------------------------------------------------------------
! LWFA Benchmark for ASCR Joule
! - Run 02
```

```
! - K parameter set
!-----------------------------------------------------------------------------------------


simulation
{
  ndump_prof = 1000,        ! Profiling info dump frequency

  wall_clock_limit = "3:50:00",
  wall_clock_check = 100,

  !  n0 = 1.74d21,
}

!----------the node configuration for this simulation---------
node_conf
{

  node_number(1:3) =  144, 8, 8,                ! 55296 nodes total

  if_periodic(1:3) = .false., .true., .true.,

  n_threads = 6,
}

!----------spatial grid----------
grid
{
  ! 8832/96 = 92 cells/node along x1
  nx_p(1:3) = 8832, 432, 432,

  ! Load info dump frequency
  ndump_global_load = 1000, ! global simulation load (min, max, avg particles / node )
  ndump_node_load   = 10000, ! number of particles per node for all nodes
  ndump_grid_load   = 0,     ! number of particles per cell for all grid points

}

!----------time step and global data dump timestep number----------
time_step
{
  ! Courant condition gives 0.1996426659282
  ! 0.999 * courant = 0.199443023262237

  dt     =  0.199443023262237,

  ! Dumps at every 2000.0 1/\omega_0 (or as close as we can get them)
  ! (rounded down to insure the final dumps get written)
  ndump  =  10027,

}

!----------restart information----------
restart
{
 ! Normal restart writing is off, the code will write checkpoint information
  ! if it goes over the wall clock limit
```

```
}

!----------spatial limits of the simulations----------
!(note that this includes information about
! the motion of the simulation box)
space
{
  xmin(1:3) =   0.000d0 , 0.000d0, 0.000d0,
  xmax(1:3) =  1766.40, 2041.31, 2041.31,
  if_move= .true., .false., .false.,
}


!----------time limits ----------
time
{
  tmin = 0.0d0,

! tmax  = 589000.0, ! this is the dephasing length

  tmax = 44000.0,  ! Estimate (2us) -> 14:35
}

!----------boundary conditions for em-fields ----------
emf_bound
{
}

!----------diagnostic for electromagnetic fields----------
diag_emf
{
  ndump_fac         = 1,  ! this is ignored, no full grid diagnostics requested
  ndump_fac_ave     = 1,  ! Spatial/time averaged diagnostics
  ndump_fac_lineout = 1,  ! slice and lineout diagnostics

  reports = "e1, savg", "e2, savg", "e3, savg", "e1, line, x1, 150, 150",
            "psi, savg"

  ndump_fac_ene_int = 1,

  n_ave(1:3)        = 4, 2, 2,
}

!----------number of particle species----------
particles
{
  num_species = 1,
  interpolation = 'quadratic',
}

!----------information for species 1----------

species
{
  name        = 'electrons',
  num_par_max = 3500000,
  rqm         = -1.0,
```

```
  num_par_x(1:3) = 1, 2, 2,

  vth(1:3) = 0.0d0 , 0.0d0 , 0.0d0 ,
  vfl(1:3) = 0.0d0 , 0.0d0 , 0.0d0 ,

  den_min = 0.0000001d0,

}

!----------density profile for this species----------
profile
{
  density=0.000287356,
  num_x = 6,

  fx(:,1) = 0.,  0.0000,   1.0,    1.0,    0.0000,   0.0,
   x(:,1) = 0., 1766.4000,  1767.400,    1.00d5,    1.00d6, 1.1d6,
  fx(:,2) = 0.,  0.0000,   1.0000,    1.0000,    0.0000,   0.0,
   x(:,2) = 0.,  60.0000,   60.1,  1981.31,   1981.41, 1.1d6,
  fx(:,3) = 0.,  0.0000,   1.0000,    1.0000,    0.0000,   0.0,
   x(:,3) = 0.,  60.0000,   60.1,  1981.31,   1981.41, 1.1d6,
}


!----------boundary conditions for this species----------
spe_bound
{
}

!----------diagnostic for this species----------
diag_species
{
  ! Full grid diagnostics : charge, m, ene, q1, q2, q3, j1, j2, j3

  ndump_fac = 1,              ! this is ignored, no full grid dianostics requested
  ndump_fac_ave = 1,

  reports = "charge, savg",
  n_ave(1:3)=4,2,2,

  ! Phasespaces
  ndump_fac_pha = 1,
  ps_xmin(1:3) =  0.0,  0.0, 0.0,
  ps_xmax(1:3) = 1766.40, 2041.31, 2041.31,
  ps_nx(1:3)   = 1000,  216, 216,

  ps_pmin(1:3) = -10.0, -5.0, -5.0,
  ps_pmax(1:3) =  200.0,  5.0,  5.0,
  ps_np(1:3)   =  1000,  200,  200,
  if_ps_p_auto(1:3) = .true., .true., .true.,

  phasespaces = "p1x1", "p2x1", "p2x2", "p1x2", "x1x2", "p3x3", "p1x3",
                "x2x3", "x1x3", "p3x1",

  ! Raw particle dumps
  ndump_fac_raw = 1,
```

```
  ! since we are not in a moving window using raw_gamma_limit gives the same end result and is
  ! much faster

  ! raw_math_expr = "p1 > 20.0",

  raw_gamma_limit = 20.0,
}


zpulse
{
  a0 = 3.2377,
  omega0 = 1.0,
  pol_type = 0,
  pol = 0.0,
  phase=0.0,

  propagation = "forward",
  lon_type = "polynomial",
  lon_rise = 107.12,
  lon_flat = 0.0,
  lon_fall = 107.12,
  lon_start = 0.5,
  per_type = "gaussian",
  per_w0 = 264.7,
  per_focus = -1.0,

}

zpulse
{
  a0 = 3.2377,
  omega0 = 1.0,
  pol_type = 0,
  pol = 90.0,
  phase=90.0,

  propagation = "forward",
  lon_type = "polynomial",
  lon_rise = 107.12,
  lon_flat = 0.0,
  lon_fall = 107.12,
  lon_start = 0.5,
  per_type = "gaussian",
  per_w0 = 264.7,
  per_focus = -1.0,

}


current{}

smooth
{
  type(1) = "custom",
  order(1) = 5,
  swfj(1:3,1,1) = 1,2,1,
```

```
  swfj(1:3,2,1) = 1,2,1,
  swfj(1:3,3,1) = 1,2,1,
  swfj(1:3,4,1) = 1,2,1,
  swfj(1:3,5,1) = -5,14,-5,
}

! -------------------- end of osiris input file ---------------
```

lwfa-r03.3d

```
!-----------------------------------------------------------------------------------------
! LWFA Benchmark for ASCR Joule
! - Run 03
! - M parameter set with moving ions
! - Using 16 particles per cell per species, 32 particles per cell total
!-----------------------------------------------------------------------------------------


simulation
{
  ndump_prof = 1000,        ! Profiling info dump frequency

  wall_clock_limit = "3:50:00",
  wall_clock_check = 100,

  !  n0 = 1.74d21,
}

!----------the node configuration for this simulation---------
node_conf
{

   node_number(1:3) =  144, 8, 8,   ! 55296 nodes total

  if_periodic(1:3) = .false., .true., .true.,

  n_threads = 6,

}

!----------spatial grid----------
grid
{
  ! 4032/96 = 42 cells/node along x1
  nx_p(1:3) = 4032, 312, 312,

  ! Load info dump frequency
  ndump_global_load = 1000, ! global simulation load (min, max, avg particles / node )
  ndump_node_load   = 10000, ! number of particles per node for all nodes
  ndump_grid_load   = 0,     ! number of particles per cell for all grid points

}

!----------time step and global data dump timestep number----------
time_step
{
```

```
  ! Courant condition gives 0.1994353353186
  ! 0.999 * courant = 0.199235899983302

  dt     =  0.199235899983302,

  ! Dumps at every 2000.0 1/\omega_0 (or as close as we can get them)
  ! (rounded down to insure the final dumps get written)
  ndump  =  10038,
}

!----------restart information----------
restart
{
  ! I turned off writing restarts, as per Ken's request, the idea is that this run
  ! should finish within a 14 - 16 hour wall clock time

! write restart at final iteration
  ndump_fac  = 6,
}

!----------spatial limits of the simulations----------
!(note that this includes information about
! the motion of the simulation box)
space
{
  xmin(1:3) =   0.000d0 , 0.000d0, 0.000d0,
  xmax(1:3) =  806.4, 1171.88, 1171.88,
  if_move= .true., .false., .false.,
}


!----------time limits ----------
time
{
  tmin = 0.0d0,

  tmax  = 40000.0,
}

!----------boundary conditions for em-fields ----------
emf_bound
{
  type(1:2,1) =   0,  0,
  type(1:2,2) =  60, 60,
  type(1:2,3) =  60, 60,
}

!----------diagnostic for electromagnetic fields----------
diag_emf
{
  ndump_fac         = 1,  ! this is ignored, no full grid diagnostics requested
  ndump_fac_ave     = 1,  ! Spatial/time averaged diagnostics
  ndump_fac_lineout = 1,  ! slice and lineout diagnostics

  reports = "e1, savg", "e2, savg", "e3, savg", "e1, line, x1, 150, 150",
            "psi, savg", "psi, savg",
```

```
  ndump_fac_ene_int = 1,

  n_ave(1:3)        = 4, 2, 2,
}

!----------number of particle species----------
particles
{
  num_species = 2,
  interpolation = 'quadratic',
}

!----------information for species 1----------

species
{
  name        = 'electrons',
  num_par_max = 5000000,
  rqm         = -1.0,
  num_par_x(1:3) = 4, 2, 2,

  vth(1:3) = 0.0d0 , 0.0d0 , 0.0d0 ,
  vfl(1:3) = 0.0d0 , 0.0d0 , 0.0d0 ,

  den_min = 0.0000001d0,

}

!----------density profile for this species----------
profile
{
  density=0.000862,
  num_x = 6,

  fx(:,1) = 0.,  0.0000,   1.0,    1.0,    0.0000,   0.0,
   x(:,1) = 0., 806.4000,  807.400,    1.00d5,    1.00d6, 1.1d6,
  fx(:,2) = 0.,  0.0000,   1.0000,    1.0000,    0.0000,   0.0,
   x(:,2) = 0.,  20.0000,   20.1,  1151.88,   1152.88, 1.1d6,
  fx(:,3) = 0.,  0.0000,   1.0000,    1.0000,    0.0000,   0.0,
   x(:,3) = 0.,  20.0000,   20.1,  1151.88,   1152.88, 1.1d6,
}


!----------boundary conditions for this species----------
spe_bound
{
  type(1:2,1) =  0, 0,
  type(1:2,2) =  5, 5,
  type(1:2,3) =  5, 5,
}

!----------diagnostic for this species----------
diag_species
{
  ! Full grid diagnostics : charge, m, ene, q1, q2, q3, j1, j2, j3
```

```
  ndump_fac = 1,              ! this is ignored, no full grid dianostics requested
  ndump_fac_ave = 1,

  reports = "charge, savg",
  n_ave(1:3)=4,2,2,


  ! Phasespaces
  ndump_fac_pha = 1,
  ps_xmin(1:3) =  0.0,  0.0, 0.0,
  ps_xmax(1:3) = 806.4, 1151.88, 1151.88,
  ps_nx(1:3)   = 1000,  312, 312,

  ps_pmin(1:3) = -10.0, -5.0, -5.0,
  ps_pmax(1:3) =  200.0,  5.0,  5.0,
  ps_np(1:3)   =  1000,  200,  200,
  if_ps_p_auto(1:3) = .true., .true., .true.,

  phasespaces = "p1x1", "p2x1", "p2x2", "p1x2", "x1x2", "p3x3", "p1x3",
                "x2x3", "x1x3", "p3x1",

  ! Raw particle dumps
  ndump_fac_raw = 1,

  ! since we are not in a moving window using raw_gamma_limit gives the same end result and is
  ! much faster

  ! raw_math_expr = "p1 > 20.0",

  raw_gamma_limit = 20.0,
}

!----------information for species 2----------

species
{
  name        = 'ions',
  num_par_max = 5000000,
  rqm         = 1836.1527,
  num_par_x(1:3) = 4, 2, 2,

  vth(1:3) = 0.0d0 , 0.0d0 , 0.0d0 ,
  vfl(1:3) = 0.0d0 , 0.0d0 , 0.0d0 ,

  den_min = 0.0000001d0,

}

!----------density profile for this species----------
profile
{
  density=0.000862,
  num_x = 6,

  fx(:,1) = 0.,  0.0000,   1.0,    1.0,    0.0000,   0.0,
   x(:,1) = 0., 806.4000,  807.400,    1.00d5,    1.00d6, 1.1d6,
  fx(:,2) = 0.,  0.0000,   1.0000,    1.0000,    0.0000,   0.0,
```

```
   x(:,2) = 0.,  20.0000,   20.1,  1151.88,   1152.88, 1.1d6,
  fx(:,3) = 0.,  0.0000,   1.0000,    1.0000,    0.0000,   0.0,
   x(:,3) = 0.,  20.0000,   20.1,  1151.88,   1152.88, 1.1d6,
}


!----------boundary conditions for this species----------
spe_bound
{
  type(1:2,1) =  0, 0,
  type(1:2,2) =  5, 5,
  type(1:2,3) =  5, 5,
}

!----------diagnostic for this species----------
diag_species
{
  ! Full grid diagnostics : charge, m, ene, q1, q2, q3, j1, j2, j3

  ndump_fac = 1,              ! this is ignored, no full grid dianostics requested
  ndump_fac_ave = 1,

  reports = "charge, savg",
  n_ave(1:3)=4,2,2,


  ! Phasespaces
  ndump_fac_pha = 1,
  ps_xmin(1:3) =  0.0,  0.0, 0.0,
  ps_xmax(1:3) = 806.4, 1151.88, 1151.88,
  ps_nx(1:3)   = 1000,  312, 312,

  ps_pmin(1:3) = -10.0, -5.0, -5.0,
  ps_pmax(1:3) =  200.0,  5.0,  5.0,
  ps_np(1:3)   =  1000,  200,  200,
  if_ps_p_auto(1:3) = .true., .true., .true.,

  phasespaces = "p1x1", "p2x1", "p2x2", "p1x2", "x1x2", "p3x3", "p1x3",
                "x2x3", "x1x3", "p3x1",

  ! Raw particle dumps
  ndump_fac_raw = 1,

  ! since we are not in a moving window using raw_gamma_limit gives the same end result and is
  ! much faster

  ! raw_math_expr = "p1 > 20.0",

  raw_gamma_limit = 20.0,
}


zpulse
{
  a0 = 2.83,
  omega0 = 1.0,
  pol_type = 0,
```

```
  pol = 0.0,
  phase=0.0,

  propagation = "forward",
  lon_type = "polynomial",
  lon_rise = 86.86,
  lon_flat = 0.0,
  lon_fall = 86.86,
  lon_start = 0.5,
  per_type = "gaussian",
  per_w0 = 153.0,
  per_focus = -1.0,

}

zpulse
{
  a0 = 2.83,
  omega0 = 1.0,
  pol_type = 0,
  pol = 90.0,
  phase=90.0,

  propagation = "forward",
  lon_type = "polynomial",
  lon_rise = 86.86,
  lon_flat = 0.0,
  lon_fall = 86.86,
  lon_start = 0.5,
  per_type = "gaussian",
  per_w0 = 153.0,
  per_focus = -1.0,

}


current{}

smooth
{
  type(1) = "custom",
  order(1) = 5,
  swfj(1:3,1,1) = 1,2,1,
  swfj(1:3,2,1) = 1,2,1,
  swfj(1:3,3,1) = 1,2,1,
  swfj(1:3,4,1) = 1,2,1,
  swfj(1:3,5,1) = -5,14,-5,
}

! --------------------- end of osiris input file ---------------
```

Q4 Compilation Script

Compilation was done using the GNU gfortran compiler for the Fortran code and the Intel icc compiler for the C (SSE) code. The reason for this was that the Intel compilers gave the best performance for both the Fortran and the SSE code; however, due to some incompatibility between the Intel implementa-

tion of OpenMP and the Cray system, using the Intel compiler with OpenMP would cause a factor of 2 performance drop. A good performance balance was found by using the GNU compiler for the Fortran / OpenMP code and the Intel compiler for the C code only.

Makefile

```
# Compilation was done using the GNU gfortran compiler for the Fortran code and the Intel icc
# compiler for the C (SSE) code. The reason for this was that the Intel compilers gave the best
# performance for both the Fortran and the SSE code; however, due to some incompatibility between
# the Intel implementation of OpenMP and the Cray system, using the Intel compiler with OpenMP
# would cause a factor of 2 performance drop. A good performance balance was found by using the
# GNU compiler for the Fortran / OpenMP code and the Intel compiler for the C code only.


# Preprocessor
FPP = gcc -C -E -x assembler-with-cpp

# Fortran compiler - Using gfortran 4.5.1 for compiling and linking
F90 = ftn
LD = ftn

# C compiler - Using Intel 11.1 for compiling C / SSE code
CC  = icc

# Fortran flags for GNU compiler
F90FLAGS = -O3 -fno-range-check -fopenmp -ffast-math

# C flags for Intel compiler
CFLAGS   = -O3 -no-prec-div -static -xHost -msse3 \
           -DFORTRANSINGLEUNDERSCORE -DSIMD_SSE -D__POSIX_TIMER__ \
           $(PAPI_INCLUDE_OPTS)

# Preprocessor Flags
FPPF = -D__USE_PAPI__ -D_OPENMP -DP_X_DIM=3 -DOS_REV=\"develop-2.2.415M\" \
       -DFORTRANSINGLEUNDERSCORE  -DSIMD_SSE -D__NO_FILE_FLUSH__

# Linker Flags - Link Intel compiler runtime libraries
LDF = -L$(INTEL_PATH)/$(INTEL_MAJOR_VERSION)/$(INTEL_MINOR_VERSION)/lib/intel64 -lirc \
      $(PAPI_POST_LINK_OPTS)

# Object files

OBJSBASE = os-sys-multi-c.o os-sys-multi.o memory.o os-param.o \
           os-units.o os-math.o os-files.o

OBJSLOG = os-logprof.o

OBJSALL =  os-stringutil.o os-util.o  os-restart.o\
  os-random.o os-nconf.o    os-fparser.o  \
  os-grid-define.o os-grid-memory.o os-grid-parallel.o os-grid-report.o os-grid.o \
  os-space.o    os-tstep.o  os-dutil-hdf5.o os-dutil.o  \
  os-vdf-define.o os-vdf-memory.o os-vdf-comm.o os-vdf-math.o os-vdf-smooth.o \
  os-vdf-average.o os-vdf.o os-vdf-report.o os-vdf-interpolate.o \
  os-wall-define.o os-wall-comm.o os-wall.o  os-time.o     os-timeavg.o   \
  os-emf-define.o os-emf-lindman.o os-emf-vpml.o os-emf-boundary.o \
  os-emf-solver.o os-emf-interpolate.o os-emf-psi.o os-emf-diagnostics.o \
```

```
  os-emf-gridval.o os-emf.o \
  os-current-diagnostics.o  os-current.o \
  os-spec-define.o os-spec-memory.o os-spec-push-c.o os-spec-profile.o os-spec-tracks.o os-spec-util.o \
  os-spec-comm.o  os-spec-current-c.o os-spec-current.o os-spec-charge.o \
  os-spec-piston.o  os-spec-rawdiag.o os-spec-phasespace.o os-spec-diagnostics.o os-spec-boundary.o \
  os-spec-loadbalance.o  os-species.o   \
  os-cathode.o  os-cross.o os-dneutral.o   os-neutral.o  \
  os-spec-collisions.o os-particles.o     os-zpulse.o  os-antenna.o \
  os-antenna_array.o

OBJSMAIN =  os-main.o


OBJS    = $(OBJSBASE) $(OBJSLOG) $(OBJSALL) $(OBJSMAIN)


# Rules

#c
%.o : ../source/%.c
$(cc) $(CFLAGS) -c $<


%.o :../source/%.f90
$(FPP) $(FPPF) -I../source $< > $(<F)
$(F90) $(F90FLAGS) -c $(<F)

osiris.e : $(OBJS)
$(LD) $(F90FLAGS) $(LDFLAGS) -o $@ $(OBJS)
chmod 755 osiris.e
cp osiris.e ../bin/osiris-3D.e

.PHONY: clean
clean :
touch a.o && rm -f *.o
touch a.mod && rm -f *.mod
touch a.f90 && rm -f *.f90
```

Q4 Run Script

There are two run scripts utilized for the Q4 studies intelligently labelled according to a `small` or `large` resource requests.

jobscripts-large.txt

```
--- frozen.s1.3d.job ---

#PBS -l walltime=01:00:00,size=221184
#PBS -N frozen.s1.3d
#PBS -A CSC074COMP
#PBS -o osiris.stdout
#PBS -e osiris.stderr
#PBS -V

cd $PBS_O_WORKDIR
```

```
date

time aprun -n 221184 ./osiris-3D.e frozen.s1.3d

date

--- frozen.s2.3d.job ---

#PBS -l walltime=01:00:00,size=221184
#PBS -N frozen.s2.3d
#PBS -A CSC074COMP
#PBS -o osiris.stdout
#PBS -e osiris.stderr
#PBS -V

cd $PBS_O_WORKDIR

date

time aprun -n 221184 ./osiris-3D.e frozen.s2.3d

date

--- warm.strong.3d.job ---

#PBS -l walltime=01:00:00,size=221184
#PBS -N warm.strong.3d
#PBS -A CSC074COMP
#PBS -o osiris.stdout
#PBS -e osiris.stderr
#PBS -V

cd $PBS_O_WORKDIR

date

time aprun -n 221184 ./osiris-3D.e warm.strong.3d

date


--- warm.weak.3d.job ---

#PBS -l walltime=01:00:00,size=221184
#PBS -N warm.weak.3d
#PBS -A CSC074COMP
#PBS -o osiris.stdout
#PBS -e osiris.stderr
#PBS -V

cd $PBS_O_WORKDIR

date

time aprun -n 221184 ./osiris-3D.e warm.weak.3d
```

```
date

--- lwfa-r01.strong.3d ---

#PBS -l walltime=02:30:00,size=221184
#PBS -N lwfa-r01.strong.3d
#PBS -A CSC074COMP
#PBS -o osiris.stdout
#PBS -e osiris.stderr
#PBS -V

cd $PBS_O_WORKDIR

date

export OMP_NUM_THREADS=6
time aprun -n 36864 -d 6 ./osiris-3D.e lwfa-r01.strong.3d
```

jobscripts-small.txt

```
--- warm.3d.job ---

#PBS -l walltime=01:00:00,size=55296
#PBS -N warm.3d
#PBS -A CSC074COMP
#PBS -o osiris.stdout
#PBS -e osiris.stderr
#PBS -V

cd $PBS_O_WORKDIR

date

time aprun -n 55296 ./osiris-3D.e warm.3d

date

--- lwfa-r01.3d.job ---

#PBS -l walltime=04:00:00,size=55296
#PBS -N lwfa-r01.3d
#PBS -A CSC074COMP
#PBS -o osiris.stdout
#PBS -e osiris.stderr
#PBS -V

cd $PBS_O_WORKDIR

date

export OMP_NUM_THREADS=6
time aprun -n 9216 -d 6 ./osiris-3D.e lwfa-r01.3d

date

--- lwfa-r02.3d.job ---
```

```
#PBS -l walltime=04:00:00,size=55296
#PBS -N lwfa-r02.3d
#PBS -A CSC074COMP
#PBS -o osiris.stdout
#PBS -e osiris.stderr
#PBS -V

cd $PBS_O_WORKDIR

date

export OMP_NUM_THREADS=6
time aprun -n 9216 -d 6 ./osiris-3D.e lwfa-r02.3d

date

--- lwfa-r03.3d.job ---

#PBS -l walltime=04:00:00,size=55296
#PBS -N lwfa-r03.3d
#PBS -A CSC074COMP
#PBS -o osiris.stdout
#PBS -e osiris.stderr
#PBS -V

cd $PBS_O_WORKDIR

date

export OMP_NUM_THREADS=6
time aprun -n 9216 -d 6 ./osiris-3D.e lwfa-r03.3d

date
```

Q4 Runtime Environment

The modules:

```
Currently Loaded Modulefiles:
  1) modules/3.1.6                                          15) cray/projdb/1.0.0-1.0202.19483.52.1
  2) DefApps                                                16) Base-opts/2.2.73
  3) torque/2.4.1b1-snap.200905191614                       17) PrgEnv-gnu/2.2.73
  4) moab/5.3.6                                             18) xt-asyncpe/4.9
  5) /opt/cray/xt-asyncpe/default/modulefiles/xtpe-istanbul 19) xt-pe/2.2.73
  6) cray/MySQL/5.0.64-1.0000.2342.16.1                     20) xt-mpt/5.1.3
  7) xtpe-target-cnl                                        21) pmi/1.0-1.0000.8160.39.1.ss
  8) xt-service/2.2.73                                      22) xt-libsci/10.5.0
  9) xt-os/2.2.73                                           23) gcc/4.5.1
 10) xt-boot/2.2.73                                         24) subversion/1.5.0
 11) xt-lustre-ss/2.2_1.6.5                                 25) hdf5-parallel/1.8.5.0
 12) cray/job/1.5.5-0.1_2.0202.21413.56.6                   26) xt-papi/3.6.2.2
 13) cray/csa/3.0.0-1_2.0202.21426.77.6                     27) intel/11.1.046
 14) cray/account/1.0.0-2.0202.19482.49.17
```

The environment:

```
MODULE_VERSION_STACK=3.1.6
LESSKEY=/etc/lesskey.bin
GNU_VERSION=4.5.1
PAPI_POST_LINK_OPTS= -L/opt/xt-tools/papi/3.6.2.2/lib -lpapi -lpfm
NNTPSERVER=news
INFODIR=/usr/local/info:/usr/share/info:/usr/info
HOSTNAME=jaguarpf-login6
GNOME2_PATH=/usr/local:/opt/gnome:/usr
XKEYSYMDB=/usr/X11R6/lib/X11/XKeysymDB
PAPI_VERSION=3.6.2.2
PE_ENV=GNU
_MODULESBEGINENV_=/ccs/home/zamb/.modulesbeginenv.jaguarpf-login6
HOST=jaguarpf-login6
TERM=xterm-256color
SHELL=/bin/bash
INTEL_MINOR_VERSION=046
XTOS_VERSION=2.2.73
PROFILEREAD=true
HISTSIZE=1000
PERFMON_VERSION=v23
TMPDIR=/tmp/work/zamb
SSH_CLIENT=193.136.137.102 51133 22
LIBRARY_PATH=/sw/xt5/subversion/1.5.0/sles10.1_gnu4.2.4/lib
MPT_DIR=/opt/cray/mpt/5.1.3/xt/seastar
INTEL_PATH=/opt/intel/Compiler
MORE=-sl
BOOT_DIR=/opt/xt-boot/2.2.73
QTDIR=/usr/lib/qt3
INCLUDE_PATH=/sw/xt5/subversion/1.5.0/sles10.1_gnu4.2.4/include
PRGENV_DIR=/opt/xt-prgenv/2.2.73
SSH_TTY=/dev/pts/17
LC_ALL=C
SVN_EDITOR=vim
ASYNCPE_DIR=/opt/cray/xt-asyncpe/4.9
BUILD_OPTS=/opt/cray/xt-asyncpe/4.9/bin/build-opts
INTEL_PRE_COMPILE_OPTS= -msse3
GROFF_NO_SGR=yes
JRE_HOME=/usr/lib/jvm/jre
USER=zamb
CRAY_HDF5_PARALLEL_VERSION=1.8.5.0
LD_LIBRARY_PATH=/opt/intel/Compiler/11.1/046/lib/intel64:/opt/xt-pe/2.2.73/lib: \
/opt/cray/pmi/1.0-1.0000.8160.39.1.ss/lib64:/opt/cray/mpt/5.1.3/xt/seastar/mpich2-gnu/lib: \
/opt/cray/mpt/5.1.3/xt/seastar/sma/lib64:/opt/gcc/mpc/0.8.1/lib:/opt/gcc/mpfr/2.4.2/lib: \
/opt/gcc/gmp/4.3.2/lib:/opt/gcc/4.5.1/snos/lib64:/opt/xt-tools/papi/3.6.2.2/lib: \
/opt/cray/hdf5-parallel/1.8.5.0/hdf5-parallel-pgi/lib: \
/sw/xt5/subversion/1.5.0/sles10.1_gnu4.2.4/lib:/opt/cray/projdb/1.0.0-1.0202.19483.52.1/lib64: \
/opt/cray/csa/3.0.0-1_2.0202.21426.77.6/lib64:/opt/cray/job/1.5.5-0.1_2.0202.21413.56.6/lib64: \
/opt/cray/MySQL/5.0.64-1.0000.2342.16.1/lib64/mysql:/opt/cray/MySQL/5.0.64-1.0000.2342.16.1/lib64
LS_COLORS=
LC_PE_ENV=gnu
WORKDIR=/tmp/work/zamb
TVDSVRLAUNCHCMD=ssh
XNLSPATH=/usr/X11R6/lib/X11/nls
INTEL_VERSION=11.1.046
MPICH_DIR=/opt/cray/mpt/5.1.3/xt/seastar/mpich2-gnu
```

```
ENV=/etc/bash.bashrc
HOSTTYPE=x86_64
RCLOCAL_PRGENV=true
GCC_VERSION=4.5.1
MPT_VERSION=5.1.3
PE_PRODUCT_LIST=ASYNCPE:PE:CRAY_PMI:XTMPT:LIBSCI:GCC:GNU:CRAY_HDF5:LUSTRE:XTPE_ISTANBUL:PAPI
FROM_HEADER=
FFTW_SYSTEM_WISDOM_DIR=/opt/xt-libsci/10.5.0
PAGER=less
PAPI_INCLUDE_OPTS= -I/opt/xt-tools/papi/3.6.2.2/include
OS_DIR=/opt/xt-os/2.2.73
CSHEDIT=emacs
INTEL_MAJOR_VERSION=11.1
MPICHBASEDIR=/opt/cray/mpt/5.1.3/xt
XDG_CONFIG_DIRS=/usr/local/etc/xdg/:/etc/xdg/:/etc/opt/gnome/xdg/
MINICOM=-c on
YOD_LOGFILE=syslog
APRUN_XFER_LIMITS=0
MODULE_VERSION=3.1.6
MAIL=/var/mail/zamb
PATH=/opt/intel/Compiler/11.1/046/bin/intel64:/opt/cray/xt-asyncpe/4.9/bin: \
/opt/cray/pmi/1.0-1.0000.8160.39.1.ss/bin:/opt/gcc/4.5.1/bin:/opt/xt-tools/papi/3.6.2.2/bin: \
/opt/cray/hdf5-parallel/1.8.5.0/hdf5-pgi/bin: \
/opt/cray/hdf5-parallel/1.8.5.0/hdf5-parallel-pgi/bin: \
/sw/xt5/subversion/1.5.0/sles10.1_gnu4.2.4/bin:/opt/cray/projdb/1.0.0-1.0202.19483.52.1/bin: \
/opt/cray/account/1.0.0-2.0202.19482.49.17/bin:/opt/cray/csa/3.0.0-1_2.0202.21426.77.6/sbin: \
/opt/cray/csa/3.0.0-1_2.0202.21426.77.6/bin:/opt/cray/job/1.5.5-0.1_2.0202.21413.56.6/bin: \
/opt/xt-lustre-ss/2.2_1.6.5/usr/sbin:/opt/xt-lustre-ss/2.2_1.6.5/usr/bin: \
/opt/xt-boot/2.2.73/bin/snos64:/opt/xt-os/2.2.73/bin/snos64:/opt/xt-service/2.2.73/bin/snos64: \
/opt/xt-prgenv/2.2.73/bin:/opt/cray/MySQL/5.0.64-1.0000.2342.16.1/sbin: \
/opt/cray/MySQL/5.0.64-1.0000.2342.16.1/bin:/opt/moab/default/bin:/opt/torque/default/bin: \
/sw/xt5/bin:/opt/modules/3.1.6/bin:/ccs/home/zamb/bin:/usr/local/bin:/usr/bin:/usr/X11R6/bin: \
/bin:/usr/games:/opt/bin:/opt/gnome/bin:/opt/kde3/bin:/usr/lib/jvm/jre/bin:/usr/lib/mit/bin: \
/usr/lib/mit/sbin:/opt/pathscale/bin:.:/usr/lib/qt3/bin:/opt/bin:/opt/public/bin
CRAY_LIBSCI_FFTW_PATH=/opt/fftw/3.2.2.1/lib
CPU=x86_64
CRAY_HDF5_DIR=/opt/cray/hdf5-parallel/1.8.5.0
JAVA_BINDIR=/usr/lib/jvm/jre/bin
SSH_SENDS_LOCALE=yes
ASYNCPE_VERSION=4.9
XTPE_COMPILE_TARGET=linux
RCLOCAL_MYSQL=true
INPUTRC=/etc/inputrc
PWD=/ccs/home/zamb
_LMFILES_=/opt/modulefiles/modules/3.1.6:/sw/xt5/modulefiles/DefApps: \
/opt/modulefiles/torque/2.4.1b1-snap.200905191614:/opt/modulefiles/moab/5.3.6: \
/opt/cray/xt-asyncpe/default/modulefiles/xtpe-istanbul: \
/opt/modulefiles/cray/MySQL/5.0.64-1.0000.2342.16.1:/opt/modulefiles/xtpe-target-cnl: \
/opt/modulefiles/xt-service/2.2.73:/opt/modulefiles/xt-os/2.2.73: \
/opt/modulefiles/xt-boot/2.2.73:/opt/modulefiles/xt-lustre-ss/2.2_1.6.5: \
/opt/modulefiles/cray/job/1.5.5-0.1_2.0202.21413.56.6: \
/opt/modulefiles/cray/csa/3.0.0-1_2.0202.21426.77.6: \
/opt/modulefiles/cray/account/1.0.0-2.0202.19482.49.17: \
/opt/modulefiles/cray/projdb/1.0.0-1.0202.19483.52.1:/opt/modulefiles/Base-opts/2.2.73: \
/opt/modulefiles/PrgEnv-gnu/2.2.73:/opt/modulefiles/xt-asyncpe/4.9: \
/opt/modulefiles/xt-pe/2.2.73:/opt/modulefiles/xt-mpt/5.1.3: \
/opt/cray/modulefiles/pmi/1.0-1.0000.8160.39.1.ss:/opt/modulefiles/xt-libsci/10.5.0: \
```

```
/opt/modulefiles/gcc/4.5.1:/sw/xt5/modulefiles/subversion/1.5.0: \
/opt/modulefiles/hdf5-parallel/1.8.5.0:/opt/modulefiles/xt-papi/3.6.2.2: \
/opt/modulefiles/intel/11.1.046
JAVA_HOME=/usr/lib/jvm/jre
SYSTEM_USERDIR=/tmp/work/zamb
MODULEPATH=/opt/cray/xt-asyncpe/4.9/modulefiles:/opt/modulefiles:/opt/cray/modulefiles: \
/opt/modules/3.1.6:/sw/xt5/modulefiles
PYTHONSTARTUP=/etc/pythonstart
PATHSCALE_PRE_COMPILE_OPTS= -march=barcelona
LOADEDMODULES=modules/3.1.6:DefApps:torque/2.4.1b1-snap.200905191614:moab/5.3.6: \
xtpe-istanbul:cray/MySQL/5.0.64-1.0000.2342.16.1:xtpe-target-cnl:xt-service/2.2.73: \
xt-os/2.2.73:xt-boot/2.2.73:xt-lustre-ss/2.2_1.6.5:cray/job/1.5.5-0.1_2.0202.21413.56.6: \
cray/csa/3.0.0-1_2.0202.21426.77.6:cray/account/1.0.0-2.0202.19482.49.17: \
cray/projdb/1.0.0-1.0202.19483.52.1:Base-opts/2.2.73:PrgEnv-gnu/2.2.73:xt-asyncpe/4.9: \
xt-pe/2.2.73:xt-mpt/5.1.3:pmi/1.0-1.0000.8160.39.1.ss:xt-libsci/10.5.0:gcc/4.5.1: \
subversion/1.5.0:hdf5-parallel/1.8.5.0:xt-papi/3.6.2.2:intel/11.1.046
CRAY_PMI_POST_LINK_OPTS=$(pkg-config cray-pmi --libs-only-L) -Wl,-rpath=/opt/cray/pmi/1.0-1.0000.8160.39.1.ss/lib64
TEXINPUTS=:/ccs/home/zamb/.TeX:/usr/share/doc/.TeX:/usr/doc/.TeX
QT_SYSTEM_DIR=/usr/share/desktop-data
SHLVL=1
HOME=/ccs/home/zamb
PTL_SNOS_NAL=SS
LESS_ADVANCED_PREPROCESSOR=no
OSTYPE=linux
SE_DIR=/opt/xt-service/2.2.73
CRAY_PMI_INCLUDE_OPTS=$(pkg-config cray-pmi --cflags)
LIBLUSTRE_DEBUG_CONSOLE=0
XTPE_ISTANBUL_ENABLED=ON
LS_OPTIONS=-N --color=none -T 0
GCC_PATH=/opt/gcc/4.5.1
WINDOWMANAGER=
GTK_PATH=/usr/local/lib/gtk-2.0:/opt/gnome/lib/gtk-2.0:/usr/lib/gtk-2.0
BASH_ENV=/etc/bash.bashrc
LESS=-M -I
MACHTYPE=x86_64-suse-linux
LOGNAME=zamb
GTK_PATH64=/usr/local/lib64/gtk-2.0:/opt/gnome/lib64/gtk-2.0:/usr/lib64/gtk-2.0
CVS_RSH=ssh
XDG_DATA_DIRS=/usr/local/share/:/usr/share/:/etc/opt/kde3/share/:/opt/kde3/share/:/opt/gnome/share/
ACLOCAL_FLAGS=-I /opt/gnome/share/aclocal
SSH_CONNECTION=193.136.137.102 51133 160.91.205.246 22
LC_CTYPE=UTF-8
PE_POST_LINK_OPTS=-Wl,-rpath=/opt/xt-pe/2.2.73/lib
MODULESHOME=/opt/modules/3.1.6
PKG_CONFIG_PATH=/opt/cray/pmi/1.0-1.0000.8160.39.1.ss/lib64/pkgconfig: \
/opt/cray/projdb/1.0.0-1.0202.19483.52.1/lib64/pkgconfig: \
/opt/cray/MySQL/5.0.64-1.0000.2342.16.1/lib64/pkgconfig:/usr/local/lib/pkgconfig: \
/usr/local/share/pkgconfig:/usr/lib64/pkgconfig:/usr/share/pkgconfig: \
/opt/kde3/lib64/pkgconfig:/opt/gnome/lib64/pkgconfig:/opt/gnome/lib64/pkgconfig: \
/opt/gnome/share/pkgconfig
LESSOPEN=lessopen.sh %s
LIBSCI_BASE_DIR=/opt/xt-libsci/10.5.0
LIBSCI_VERSION=10.5.0
INFOPATH=/opt/gcc/4.5.1/snos/info:/usr/local/info:/usr/share/info:/usr/info:/opt/gnome/share/info
CRAY_CPU_TARGET=istanbul
DISPLAY=localhost:17.0
PE_DIR=/opt/xt-pe/2.2.73
```

```
GNU_PRE_COMPILE_OPTS= -march=amdfam10
LESSCLOSE=lessclose.sh %s %s
G_BROKEN_FILENAMES=1
LUSTRE_DIR=/opt/xt-lustre-ss/2.2_1.6.5
JAVA_ROOT=/usr/lib/jvm/jre
COLORTERM=1
_=/usr/bin/env
```