OAK RIDGE
NATIONAL LABORATORY
MANAGED BY UT-BATTELLE
FOR THE DEPARTMENT OF ENERGY

ORNL/TM-2009/322

FY 2009 Annual Report of Joule Software Metric SC GG 3.1/2.5.2, Improve Computational Science Capabilities

December 2009

UT-BATTELLE
ORNL-27 (4-00)

DOCUMENT AVAILABILITY

Reports produced after January 1, 1996, are generally available free via the U.S. Department of Energy (DOE) Information Bridge.

Web site http://www.osti.gov/bridge

Reports produced before January 1, 1996, may be purchased by members of the public from the following source.

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
Telephone 703-605-6000 (1-800-553-6847)
TDD 703-487-4639
Fax 703-605-6900
E-mail info@ntis.gov
Web site http://www.ntis.gov/support/ordernowabout.htm

Reports are available to DOE employees, DOE contractors, Energy Technology Data Exchange (ETDE) representatives, and International Nuclear Information System (INIS) representatives from the following source.

Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831
Telephone 865-576-8401
Fax 865-576-5728
E-mail reports@osti.gov
Web site http://www.osti.gov/contact.html

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-2009/xxx

Center for Computational Sciences

FY 2009 Annual Report of Joule Software Metric SC GG 3.1/2.5.2, Improve Computational Science Capabilities

Date Published: December 2009

Prepared for
U.S. Department of Energy
Office of Science
Advanced Scientific Computing Research Program

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831-6283
managed by
UT-Battelle, LLC
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CREDITS

Application Credits

VisIt

Sean Ahern (Oak Ridge National Laboratory)
URL: http://www.llnl.gov/visit/

CAM

James Hack (Oak Ridge National Laboratory)
URL: http://www.ccsm.ucar.edu/models/atm-cam/

XGC1

Choong-Seock Chang (Courant Institute of Mathematical Sciences, New York University)
URL: http://w3.physics.lehigh.edu/~xgc/, www.cims.nyu.edu/cpes/

RAPTOR

Joseph C. Oefelein (Sandia National Laboratories)
URL: http://public.ca.sandia.gov/crf/research/index.php

Technical Team

VisIt

Dave Pugmire, Sean Ahern, Tom Evans (Oak Ridge National Laboratory); Hank Childs (Lawrence Berkeley National Laboratory)

CAM

Jim Rosinski, Pat Worley, Kate Evans (Oak Ridge National Laboratory)

XGC1

Scott Klasky, Pat Worley, Ed D'Azevedo (Oak Ridge National Laboratory); Seung-Hoe Ku (Courant Institute of Mathematical Sciences, New York University); Mark Adams (Columbia University)

RAPTOR

Ramanan Sankaran (Oak Ridge National Laboratory)

Additional Credits

Kenneth Roche (University of Washington)
Ricky Kendall, Doug Kothe (Oak Ridge National Laboratory)

DOE Program Contacts

Christine Chalk (christine.chalk@science.doe.gov)
Barbara Helland (helland@ascr.doe.gov)
Daniel Hitchcock (daniel.hitchcock@science.doe.gov)
Michael Strayer (michael.strayer@science.doe.gov)

Additional Contacts

Douglas B. Kothe (kothe@ornl.gov)
Kenneth Roche (k8r@u.washington.edu)

CONTENTS

CREDITS iii
LIST OF TABLES vii
LIST OF FIGURES ix
ABBREVIATED TERMS xi

1. METRIC STATEMENT FOR COMPUTATIONAL EFFECTIVENESS 1
   1.1 JOULE METRICS 1
   1.2 FY09 JOULE GOALS FOR THE DOE ASCR PROGRAM 1
   1.3 QUARTERLY TASKS RELATED TO SC GG 3.1/2.5.2 2

2. METRIC RESULTS FOR COMPUTATIONAL EFFECTIVENESS 3
   2.1 TARGET HPC SYSTEM: JAGUARPF.CCS.ORNL.GOV 3
   2.2 RESULTS SUMMARY 3
      2.2.1 VisIt 4
      2.2.2 CAM 4
      2.2.3 XGC1 5
   2.3 CONCLUSIONS 6

3. OVERVIEW OF COMPUTATIONAL SCIENCE CAPABILITIES AND ANALYSIS OF METRIC RESULTS 8
   3.1 VISIT 8
      3.1.1 Introduction 8
      3.1.2 Background and Motivation 8
      3.1.3 Capability Overview 9
      3.1.4 Science Driver for Metric Problem 9
      3.1.5 The Model and Algorithm 10
      3.1.6 Q2 Baseline Problem Results 11
      3.1.7 Computational Performance Gains 14
      3.1.8 Q4 Metric Problem Results 15
      3.1.9 Interpretation of Results 17
      3.1.10 Summary and Conclusions 17
   3.2 CAM 19
      3.2.1 Introduction 19
      3.2.2 Background and Motivation 19
      3.2.3 Capability Overview 19
      3.2.4 Science Driver for Metric Problem 21
      3.2.5 The Model and Algorithm 21
      3.2.6 Q2 Baseline Problem Results 22
      3.2.7 Computational Performance Gains 23
      3.2.8 Q4 Metric Problem Results 25
      3.2.9 Interpretation of Results 26
      3.2.10 Summary and Conclusions 27
   3.3 XGC1 28
      3.3.1 Introduction 28
      3.3.2 Background and Motivation 29
      3.3.3 Capability Overview 29
      3.3.4 Science Driver for Metric Problem 31
      3.3.5 Q2 Baseline Problem Results 33

3.3.6 Computational Performance Gains 36
3.3.7 Q4 Metric Problem Results 37
3.3.8 Interpretation of Results 41
3.3.9 Summary and Conclusions 42
3.4 RAPTOR 43
3.4.1 Introduction 43
3.4.2 Background and Motivation 43
3.4.3 Capability Overview 46
3.4.4 Science Driver for Metric Problem 47
3.4.5 Q2 Baseline Problem Results 48
3.4.6 Computational Performance Gains 51
3.4.7 Q4 Metric Problem Results 54
3.4.8 Interpretation of Results 57
3.4.9 Summary and Conclusions 58

REFERENCES 60

APPENDIXES: BENCHMARK PROBLEM ENVIRONMENTS

APPENDIX A. OVERVIEW A-1
A.1 Parallel Matrix Multiply Example A-1
A.2 Modules Available on the Target Architecture A-9
A.3 Compilation for Instrumentation and Execution A-14

APPENDIX B. VisIt B-1
B.1 Input Settings B-1
B.2 Compilation B-5
B.3 Batch Script B-5
B.4 Runtime Environment B-5

APPENDIX C. CAM C-1
A.1 Input Settings C-1
A.2 Compilation C-1
A.3 Batch Script C-2
C.4 Runtime Environment C-2

APPENDIX D. XGC1 ..... D-1
D.1 Input Settings D-1
D.2 Compilation D-4
D.3 Batch Script D-11
D.4 Runtime Environment D-12

APPENDIX E. RAPTOR E-1
E.1 Input Settings E-1
E.2 Compilation E-3
E.3 Batch Script E-11
E.4 Runtime Environment E-11
E.5 Comparison of Total Run Time vs Initialization Time E-20

LIST OF TABLES

Table Page

1 FY09 Joule software summary of Q2 baseline and Q4 metric performance simulations and data 7
2 Per core timings for the Q2 isosurfacing benchmark 12
3 PAPI hardware counter data for the Q2 isosurfacing benchmark 12
4 Per core timings for the Q2 volume rendering baseline 13
5 PAPI hardware counter data collected for the Q2 volume rendering baseline 13
6 Per core timings for the Q4 isosurfacing benchmark 15
7 PAPI hardware counter data for the Q4 isosurfacing benchmark 15
8 Weak scaling results of the Q4 benchmark 15
9 Per core timing results for the Q4 volume rendering 16
10 PAPI hardware counter data collected for the Q4 volume rendering benchmark 16
11 Weak scaling results of the volume rendering benchmark timings 16
12 Q2 and Q4 simulation sizes 17
13 CAM performance data for the Q2 benchmark run* 23
14 CAM performance data for the Q4 modified run* 26
15 XGC1 performance data collected on the Q2 benchmark with PAPI hardware counters 35
16 XGC1 performance data collected on the Q4 benchmark with PAPI hardware counters 37
17 Baseline grid sizes for Joule benchmark runs* 50
18 Counter data acquired from CrayPAT 4.2 for Q2 benchmark run using RAPTOR 51
19 Summary of measured timings (in seconds) after each set of code revisions using the 10.3 million cell Q2 test problem and 200 time steps 54
20 Counter data acquired from CrayPAT 4.2 for the Q2 benchmark run using RAPTOR 55
21 Summary of results from the Q4 run compared to the Q2 baseline 56
22 Counter data acquired from CrayPAT 4.2 for the Q4 run using RAPTOR 56
E.1 Counter data acquired from CrayPAT 4.2 for Q2 benchmark run using RAPTOR E-20
E.2 Counter data acquired from CrayPAT 4.2 for Q2 benchmark run using RAPTOR but with the integration loop bypassed E-22

LIST OF FIGURES

Figure Page

1 *(a)* An isosurface of a Raleigh-Taylor instability problem. *(b)* A volume rendering of a turbulence problem. *(c)* Volume rendering and streamlines of a core collapse supernova collapse simulation 8

2 Data flow network-processing model. Data flows from the network source to the network sink 9

3 Nuclear power plant, a PWR facility, set up for the Denovo simulation. 10

4 *(a)* Extraction of radiation dose contours and *(b)* a volume rendering from the nuclear power plant simulation from the Denovo code. Q2 18

5 *(a)* Extraction of radiation dose contours and *(b)* a volume rendering from the nuclear power plant simulation from the Denovo code. Q4 18

6 Global average surface temperature in observations, modeled with and without anthropogenic forcing 21

7 Current CAM strong scaling performance for the T341 mesh on the Jaguar/XT5 platform 27

8 Schematic of the ITER tokamak, where the first wall of the innermost structure of the device is shown, with the divertor chamber at the bottom 31

9 *(left)* Nonlocal nature of the ion temperature (T_i) profile. *(right)* The cross section of DIII-D magnetic surface inside the first wall 32

10 Early-time plasma density and temperature profiles. 34

11 Initial profile of $R_0/L_T = R_0|\partial \log T_i/\partial r|$ 34

12 *(a)* Relationship between the normalized poloidal flux ψN and real distance in meters from the magnetic axis (Raxis = Ro) to the flux surface (R) along the midplane. *(b)* Radial profile of the safety factor q 34

13 Turbulent eddies on the whole poloidal cross-sectional plane at *(left)* an earlier time and *(right) a* later time 35

14 Inward propagation of the square root of turbulence intensity Sqrt(I) during the bursty nonlinear period, where $I = <(\delta\varphi)2>$ 36

15 An enlarged image of the turbulence intensity $<|\delta\varphi|^2>$ contour in the radius-time space in the pedestal area 38

16 The same simulation as in Fig. 15 in the localized radial domain, with the usual particle simulation boundary at r = 0.5 38

17 Heat flux contour in the global space-time space, exhibiting the out-to-in propagation of the turbulence front 39

18 Self-organizing modification of the background temperature profile by the incoming turbulence 39

19 Time behavior of effective ion thermal conductivity (thermal flux divided by local Ti gradient) from the start of the simulation across $\psi N = 0.64$, which corresponds to r = 42 cm on the outside midplane 40

20 Phase relation between the temperature gradient, heat flux, and E×B shearing dynamics at a radial location 40

21 Energy accounting within $0.3 \leq \psi N \leq 0.7$ between the total influx across the inner boundary (black curve) and the sum of the consumed energy (blue curve) to the particles, the electric field, and across the outer boundary 41

22 Key experiments currently being studied using RAPTOR 44
23 Photograph and corresponding LES of the DLR-A flame 46
24 Baseline flame used for problem scaling 48
25 Cross section of the computational domain showing key features of the grid topology 49
26 Comparison of experimentally measured (symbols) and modeled (lines) results showing acceptable agreement 50
27 Performance profile of the original code on 5,952 XT5 cores 52
28 Performance profile of RAPTOR on 5,952 cores after reducing global MPI operations 53
29 Performance profile of RAPTOR after software revisions 54

ABBREVIATED TERMS

2D, 3D, …	two dimensional, three dimensional, …
ADIOS	Adaptable I/O System
AGCM	atmospheric general circulation model
AMR	adaptive mesh refinement
AMWG	Atmospheric Model Working Group
ASCAC	Advanced Scientific Computing Advisory Committee
ASCR	Advanced Scientific Computing Research (DOE program)
BES	Basic Energy Sciences (DOE SC program)
CAM	Community Atmosphere Model
CCM	Community Climate Model
CCSM	Community Climate System Model
CLM	Common Land Model
CPES	Center for Plasma Edge Simulation
CPU	central processing unit
CRF	Combustion Research Facility
DDR2	double data-rate two
DEMO	Demonstration Power Plant (proposed)
DIMM	dual inline memory module
DNS	direct numerical simulation
DOE SC	DOE Office of Science
DOE	U.S. Department of Energy
DoF	degree of freedom
ECMWF	European Centre for Medium-Range Weather Forecasts
EERE	Office of Energy Efficiency and Renewable Energy (of DOE)
EFFIS	End-to-End Framework for Fusion Integrated Simulation
flop	floating point operation
FSP	Fusion Simulation Project
GAM	geodesic acoustic mode
GPRA	Government Performance and Results Act of 1993
GPTL	General Purpose Timing Library
HCCI	homogenous charge compression ignition
HPC	high-performance computing
I/O	input/output
IEEE	Institute of Electrical and Electronics Engineers
ISCCP	International Satellite Cloud Climatology Project
ITER	international fusion reactor project
ITG	ion temperature gradient
KBA	Koch-Baker-Alcouffe
LES	large-eddy simulation
MPI	Message Passing Interface Standard
NCAR	National Center for Atmospheric Research
NCCS	National Center for Computational Sciences
NERSC	National Energy Research Scientific Computing Center
OMB	U.S. Office of Management and Budget
ORNL	Oak Ridge National Laboratory
OVT	Office of Vehicle Technologies (of DOE EERE)
PAPI	Performance Application Programming Interface
PART	Performance Assessment Rating Tool

PE	processing element
PGI	Portland Group
PIC	particle in cell
PWR	pressurized water reactor
Q1, Q2, …	Quarter 1, Quarter 2, …
R&D	research and development
RANS	Reynolds-Averaged Navier-Stokes
Re	Reynolds
SciDAC	Scientific Discovery through Advanced Computing (DOE program)
SGS	subgrid scale
SMP	symmetric multiprocessor
SOC	self-organized critical
SPMD	single program–multiple data
SST	sea surface temperature
TNF	turbulent nonpremixed flame
VTK	Visualization ToolKit

1. METRIC STATEMENT FOR COMPUTATIONAL EFFECTIVENESS

1.1 JOULE METRICS

The Joule Software Metric for Computational Effectiveness is established by 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)* oversees the preparation and administration of the President's 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?
- 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. Joule enables the chief financial officer and senior DOE management to track annual performance on a quarterly basis. Joule 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). Joule links the DOE strategic plan† to the underlying base program targets.

1.2 FY09 JOULE GOALS FOR THE DOE ASCR PROGRAM

The DOE Advanced Scientific Computing Research (ASCR)‡ program 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. FY09 performance metric: capability usage is at least 40%.

---

* http://www.whitehouse.gov/omb
† http://www.er.doe.gov/about/MissionStrategic.htm
‡ http://www.sc.doe.gov/ascr/About/about.html

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. FY09 performance metric: efficiency measure is ≥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).

1.3 QUARTERLY TASKS RELATED TO SC GG 3.1/2.5.2

The Joule 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 must 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—namely the machine operation count, execution time, and machine instance—is collected and explained. If an application is striving to achieve a new science result in addition to demonstrating improved performancing, then providing a detailed discussion of its current (prior to Q2) capability, a discussion of why the capability is insufficient, and a description of why the new capability being developed 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 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).

2. METRIC RESULTS FOR COMPUTATIONAL EFFECTIVENESS

Each application is discussed and its baseline and metric problem described in the respective application sections. A brief description of the machine used for the application problems is given. A summary of measured results for each application is provided.

2.1 TARGET HPC SYSTEM: JAGUARPF.CCS.ORNL.GOV

The Cray XT5 high-performance computing (HPC) system, Jaguar/XT5, at the Oak Ridge National Laboratory (ORNL) National Center for Computational Sciences (NCCS) is used to exercise the DOE ASCR FY09 Joule software metric.

Jaguar/XT5 has a total of 18,688 XT5 compute nodes. The compute node operating system is a variant of Linux (CNL2.0 during the Q2 baseline, CLE2.1 thereafter). The dual-socket compute nodes are Quad-Core AMD Opteron™ Processor 23 (B3) chips operating at 2.3 GHz with 16 gigabytes (GB) of unbuffered memory per node, 2 megabytes (MB) of shared L3 cache per chip, 512 kilobytes (KB) of L2 cache per core, and 64 KB instruction and 64 KB data 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.

Jaguar/XT5 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 4 petabytes (PB) of disk space are available in the scratch file systems that support massive I/O parallelism through the Lustre file system software.* HyperTransport links all nodes to Cray's proprietary SeaStar2+chips, which are used to construct a 3D 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† 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.‡ For chip-specific information on the single socket 1000 series, see the AMD website.§

2.2 RESULTS SUMMARY

The FY09 studies demonstrate both 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, and weak scaling, where the goal is to compute in the same wall-clock time a more complex problem on an increased hardware allocation (e.g., maintaining fixed work per processing element).

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, or flops) 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

---

* http://www.lustre.org
† http://www.nccs.gov/computing-resources/jaguar/
‡ http://www.cray.com/Assets/PDF/products/xt/CrayXT5Blade.pdf
§ http://www.amd.com/us-en/Processors/ProductInformation/0,,3_118_8796_15226,00.html

system observables such as number of processing elements (PEs) dedicated to executing the program, execution time, total number of instructions executed,* the magnitude of the memory demand, etc.

2.2.1 VisIt

In Q2, a 103,716,288 cell, 4,096 domain, and 27 energy group Denovo nuclear power plant energy deposition study was executed on 4,096 cores of the Jaguar/XT5 target machine. The resulting run generated 4,096 HDF5 formatted files totaling 83.457 GB of storage. During execution of the isosurface benchmark, six different isosurfaces were computed at 1,024 × 1,024 pixel resolution. The rate that VisIt computed isocontours was 0.01778 second per isocontour on 4,096 cores. The isosurface benchmark required 173,459,136,793 floating point operations to complete. During execution of the volume rendering benchmark on 4,096 PEs, 2,000 samples are computed per ray at 1,024 × 1,024 pixel resolution. The average compute time (measured per process) to render was 28.7293 seconds. The volume rendering exercise required 178,848,487,657 floating point operations to complete.

In Q4, a 321,117,696 cell, 12,720 domain, 27 energy group Denovo nuclear power plant energy deposition study was executed on 12,720 PEs (cores) of the Jaguar/XT5 target machine. The resulting run generated 12,720 HDF5 formatted files totaling 258.391 GB of storage. Again, in the isosurface benchmark, six different isosurfaces were computed at 1,024 × 1,024 pixel resolution. The rate that VisIt computed isocontours was 0.01686 second per isocontour on 12,720 cores. The isosurface benchmark required 522,908,518,594 floating point operations to complete. During execution of the volume rendering benchmark on 12,720 PEs, 2,000 samples are computed per ray at the same resolution as in Q2. The average compute time (measured per process) to render was 6.37796 seconds. The volume rendering required 502,828,537,797 floating point operations to complete.

In summary, the total wall-clock times spent in the processing pipeline, discussed in the detailed description of VisIt, include overheads not reported in these software benchmarks (such as I/O times in Q2 and Q4). The benchmarks performed in FY09 demonstrate the significant capabilities to volume render and isosurface large spatial data sets employing parallel computing techniques and resources. The isosurface benchmark revealed better than linear weak scaling of the rate to compute isocontours with the software. Indeed, a problem composed of 3.1054 more physical domains over a factor of 3.0961 more cells requiring a factor of 3.0145 floating point computations was computed at a rate that was 1.0545 times faster in Q4 than in Q2 on 3.1054 times more PEs of the same target machine. The volume rendering benchmark demonstrates substantially much better than linear weak scaling performance. Between Q2 and Q4 a performance and scaling bottleneck was identified and fixed in VisIt's volume rendering capability (discussed in the detailed write-up of VisIt). The time to volume render the problem composed of a factor of 3.1054 more physical domains over a factor of 3.0961 times more cells requiring a factor of 2.8114 floating point computations was computed 4.5044 times faster in Q4 than in Q2. The weak scaling results for VisIt are outstanding; hence *VisIt met its target Joule metric on weak scaling problems in both isosurfacing and volume rendering.*

2.2.2 CAM

In both Q2 and Q4, CAM Version 3.5—configured with the spectral Eulerian dynamical core—was executed in uncoupled mode for a T341 grid (approximately 0.35° in latitude and longitude) with 26 vertical levels on 8,192 processor cores of the target machine for a one-month simulation and constant time step of 150 seconds (17,856 simulation time steps). The uncoupled mode includes a fully active land model, and sea surface temperatures and sea ice concentrations are provided by external forcing datasets. The difference between the Q2 and Q4 execution models of CAM is characterized by changes made after

* The instruction set is not to be confused with basic operations that are defined in the language of the instruction set of the chip. For instance, in a single cycle, a single cup-core (1 PE) on Jaguar/XT5 can compute four double-precision mathematical operations (fused multiply and add).

the Q2 benchmark that enabled an improved use of the multicore target architecture for the Q4 benchmark. The details are discussed in Sect. 3.2.

The Q2 benchmark completed execution on 8,192 cores of Jaguar/XT5 in 6,481.724 seconds. The measured execution time of the main computation phases includes 5,916.475 seconds for the atmosphere model (with ~4,247 seconds being the dynamical core) and 112.048 seconds for the land model. Writing the history files (I/O) took 115.024 seconds.

The Q4 benchmark completed execution on 8,192 cores of Jaguar/XT5 in 3,241.144 seconds. In the main phases of the computation, the atmosphere model took 2,823.365 seconds, the land model took 101.310 seconds, and the I/O phase was 41.302 seconds.

The strong scaling result for CAM is outstanding. The same amount of Jaguar/XT5 resource (same number of cores) was utilized to compute the same physical problem in both Q2 and Q4, yet the Q4 software executed 2 times faster than the Q2 version. *CAM therefore met its target Joule metric by virtue of its factor of two reduction in execution time on a fixed size problem.* Considering only the execution time of the atmosphere model with the spectral Eulerian dynamical core (which is the portion of the model where the algorithmic improvements were made), the Q4 variant executed 2.0955 times faster than the Q2 version. The improvements enable better throughput for climate scientists.

2.2.3 XGC1

The XGC1 magnetic fusion application yields solutions to the gyrokinetic Maxwell's equations with a full plasma distribution function. This solution includes both the heat source in the core and particle loss on the edge (at the wall) for the entire volume of various tokamak geometries. XGC1 simulations include open and closed magnetic field regions, and the separatrix surface between these regions believed to be of vital significance to the construction of an ITER-scale tokamak. The science goal behind the XGC1 benchmarks was to study non-local H-mode turbulent coupling driven by free energy in the ion temperature gradient in a simulation day or less utilizing as much hardware from the target architecture as possible. In an effort to enable a one-to-one comparison, the DIII-D tokamak geometry with a fixed number of plasma particles (13.5 billion) was used for both the Q2 benchmark and Q4 baseline simulations. The existing target architecture did not possess enough hardware (compute cores and associated memory) to allow ITER geometry computations in Q2 due to XGC1 scaling issues that existed at that time.

In Q2, XGC1 was executed on 29,952 PEs (cores) of the target machine. The simulation was executed for 24 hours and terminated after 4,000 physical time steps. While the simulation was unable to evolve to the desired quasi-steady self-organized state in this time period, the simulation did reach a nonlinear phase where the turbulence intensity was seen to propagate from the edge to the core, indicating a nonlocal coupling between the edge and core regions.

In Q4, 119,808 PEs (cores) were utilized to execute 16,000 physical time steps in 21 hours of simulation time. The Q4 simulation evolved beyond the initial, bursty nonlinear turbulence phase observed in the results of the Q2 simulation to the quasi-steady self-organized phase characteristic of experiments. Valuable insights into the nonlocal turbulence propagation and the evolution of the turbulence and the plasma profile to the quasi-steady self-organized-critical (SOC) state were obtained in Q4 *for the first time* through simulation. These results provide invaluable insight into numerous experiments significant to the design of the ITER.

The performance result for XGC1 is outstanding. In Q4, the software computed 4 times as many physical time steps with 4 times the number of processes in less time than the Q2 simulation. Indeed the execution time in Q4 was 0.875 times the execution time in Q2. The enhancements made between Q2 and Q4 focused on improving several of the particle computations by employing light weight processes and eliminating essentially one-fourth of the communications in these phases. Precomputing spline coefficients and performing a table lookup (instead of repeatedly recomputing the coefficients), for example, were major contributors in the optimization of the particle interpolation scheme. Also, better use

was made of partial derivatives required in the interpolation scheme. *XGC1 therefore met its target Joule metric as measured by both "grind time" (simulation time per step) and particle push rate.*

RAPTOR. The purpose of the RAPTOR benchmarks is to study the effects of large-eddy simulation (LES) grid resolution on scalar mixing processes, to try and understand the relationship between the grid spacing and the measured turbulence length scales using a companion set of experimental data, and to study the effects of increasing jet Reynolds (Re) number on the dynamics of turbulent scalar mixing. The benchmark is performed using the experimental DLR-A configuration for validation (see details in the following sections), which is one of a series of internationally recognized datasets used by the combustion community. The computational domain includes the entire burner geometry (inside the jet nozzle and the outer co-flow) plus the downstream space around burner. The inner nozzle has a diameter of 8 mm with the outer nozzle surface tapered to a sharp edge at the burner exit. There are 110 inner jet diameters in the axial direction (88 cm length) and 40 jet diameters in the radial direction (32 cm length). In both Q2 and Q4 simulations, exactly the same physical apparatus and flame were modeled for 50 physical time steps but at different grid resolutions and Re numbers (starting at 15,200).

In Q2, 10,285,056 cells were used to partition the computational domain. The simulation was executed in 1,425.761 seconds on 47,616 PEs (cores) of the target Jaguar/XT5 architecture. The time integration routines (which integrate the Navier-Stokes equations) dominated the cost of RAPTOR computations. In Q2, time integration required 1,034 seconds. In total, the computation retired $2.059456803 \times 10^{17}$ total instructions, while executing $3.780249864 \times 10^{14}$ floating point operations.

In Q4, 24,261,120 cells were used to partition the domain. The simulation completed execution in 1,972.397426 seconds on 112,320 PEs (cores) of Jaguar/XT5. Time integration required 444 seconds. The Q4 computation retired $6.633655878 \times 10^{17}$ instructions, while executing $8.928138372 \times 10^{14}$ floating point operations.

The performance results obtained in Q2 and Q4 for RAPTOR can be interpreted as follows. First, the principal phase of the computation that requires scalability is the cost of an integation time step. The cost of the initialization phase (problem input and setup) is amortized over the time evolution phase; hence the initialization time is not included in exercising the Joule metric. Second, the cost to integrate multiple (50) time steps reveals a remarkable result, namely that the RAPTOR performance metric increases by a factor of 2.34 beyond the required 1.0 necessary for meeting its Joule metric. The Q4 metric problem requires 2.3617 times more floating point operations on a domain having a factor of 2.3588 more cells on 2.3588 times more PEs than the Q2 benchmark. In particular, the compute time per number of grid cells per number of time steps (the "grind time") is a generic measure of performance for RAPTOR. In Q2, the grind time is

$$(1,034 \text{ seconds} \times 47,616 \text{ cores}) / 10,285,056 \text{ cells} / 50 \text{ time steps} = 0.096 .$$

In Q4, the grind time is

$$(444 \text{ s} \times 112,320 \text{ cores}) / 24,261,120 \text{ cells} / 50 \text{ time steps} = 0.041 .$$

The ratio between Q4 and Q2 is 2.3414—a truly outstanding weak scaling result. *RAPTOR therefore met its target Joule metric by virtue of a remarkable 2.34 factor reduction in grind time on a weak scaling problem.* The enhancements after the Q2 benchmark that led to this remarkable result are discussed in the detailed description of RAPTOR.

2.3 CONCLUSIONS

The aggregated machine event information collected while executing the Q2 baseline and Q4 metric problems is presented in Table 1. This approximates the total computational complexity executed for the FY09 Joule computational science capabilities measure (all on Jaguar/XT5).

Some of the applications were also improved for efficiency or simply performed better from the machine perspective when executing a larger problem.

Table 1. FY09 Joule software summary of Q2 baseline and Q4 metric performance simulations and data

Application	VisIt		CAM	XGC1	RAPTOR
Metric	Simulation time	Simulation time	Simulation time	Grind time and particle rate Time per time step Particles pushed per second	Grind time Time per cell per time step
Problem	Isosurface • 1,024 × 1,024 pixels • Iso @ 0.001, 0.01, 0.1, 1.0, 10.0, 100.0 • Q2 dataset: 103.7M cells, 4,096 cores, 27 groups • Q4 dataset: 321.1M cells, 12,720 cores, 27 groups	Volume render • 1,024 × 1,024 pixels • 2,000 samples per ray • Q2 dataset: 103.7M cells, 4,096 cores, 27 groups • Q4 dataset: 321.1M cells, 12,720 cores, 27 groups	1 simulated month • T341 mesh • 150 sec time step • 26 vertical levels • Spectral Eulerian core	DIII-D experimental tokamak • 13.5B particles • Q2: 4000 time steps • Q4: 16,000 time steps	DLR-A configuration • 50 time steps • 110 × 40 jet diam in axial and radial directions • Q2: 10,285,056 cells • Q4: 24,261,120 cells
Hardware (cores) Q2 Q4	 4,096 12,720	 4,096 12,720	 8,192 8,192	 29,952 119,808	 47,616 112,320
Time (seconds) Q2 Q4	 0.01778 per contour 0.01686 per contour	 28.729 6.378	 6,481.724 3,241.144	 86,400 75,600	 1,034.0 444.0
Metric target	Q2:Q4 contour time ≥ 1.0	Q2:Q4 time ≥ 3.10	Q2:Q4 time ≥ 2.0	Q2:Q4 grind time ≥ 1.0 Q2:Q4 particle rate ≥ 4.0	Q2:Q4 grind time ≥ 1.0
Metric result	1.05	4.50	2.10	1.14 4.57	2.34

3. OVERVIEW OF COMPUTATIONAL SCIENCE CAPABILITIES AND ANALYSIS OF METRIC RESULTS

3.1 VISIT

3.1.1 Introduction

VisIt is an open source interactive parallel analysis and visualization tool for scientific data. It can be used to visualize scalar, vector, and tensor fields defined on 2D and 3D structured and unstructured meshes. VisIt was designed to handle very large data set sizes in the petascale range and yet can also handle small data sets in the kilobyte range. It is widely used throughout the scientific community, including government laboratories, universities, and industry. VisIt won an R&D 100 award in 2005 and has been downloaded over 100,000 times. Computer scientists at a number of DOE laboratories and universities have invested approximately 50 person-years of development in VisIt. VisIt is intended for more than just visualization and is built around five primary use cases: data exploration, quantitative analysis, comparative analysis, visual debugging, and communication of results. VisIt has a client-server design for remote visualization, with the server operating in a fully data parallel manner and in a distributed memory setting. VisIt has been deployed on a variety of computing platforms, including Linux, Mac OS, Windows, and on a diverse set of high performance computing platforms, including Cray, Sun, and IBM.

VisIt is built on top of a number of well-established third-party libraries and applications. These include the Qt widget library for user interface, the Python programming language for a command line interpreter and scripting capability, and the Visualization ToolKit (VTK) for the data model and many of the analysis algorithms.

3.1.2 Background and Motivation

As supercomputers become increasingly powerful, the size, scope, and complexity of the simulations continue to increase. This results in increasingly larger quantities of output data that need to be analyzed and understood. Effectively and efficiently understanding the results has been a long-standing challenge. To meet this challenge, postprocessing analysis and visualization tools have been developed which read in the simulation data, perform various operations, and present the results using visual or quantitative techniques. A great diversity of quantitative and visual techniques has been developed to give insight about the data, including isosurface extraction (Fig. 1*(a)*), volume rendering (Fig. 1*(b,c)*) and streamline generation (Figure 1*(c)*).

Fig. 1. *(a)* An isosurface of a Raleigh-Taylor instability problem. *(b)* A volume rendering of a turbulence problem. *(c)* Volume rendering and streamlines of a core collapse supernova collapse simulation.

As the size, scope, and complexity of the simulations increase, the capability of the postprocessing analysis and visualization tools must be able to similarly scale. The failure of these tools to scale will place unacceptable restrictions on their use, such as operating only on portions or downsampled versions of the data or taking an inordinate amount of time to complete. These restrictions have enormous negative impacts on a scientist's ability to understand and reason about a simulation.

3.1.3 Capability Overview

VisIt's basic execution model is to employ "data flow networks," a standard model used by visualization systems for approximately two decades. Data flow networks consist of relatively independent filters, with each filter corresponding to an analysis algorithm. Filters can have one or more inputs and one or more outputs. Connecting filters together, attaching sources (file readers), and connecting the network's final outputs to a rendering algorithm creates data flow networks. An example of a dataflow network that renders isocontours of a dataset is shown in Fig. 2.

Fig. 2. Data flow network-processing model.
Data flows from the network source to the network sink.

VisIt's large data strategy is to use distributed memory data parallelism. To create a given rendering, each processor creates an identical data flow network, and the networks are differentiated by the input data they process, very much in the multiple instruction, multiple data stream (MIMD) model. The input data set is partitioned across the processors, with each processor owning a different portion. Sometimes "ghost data" is replicated along the boundaries of the partitions to prevent interpolation artifacts and other problems. Once each processor is assigned its portion of the larger data set, it reads its portion and executes its data flow network. The reading and the data flow network execution can take place with communication between the processors or entirely independently, depending on the specifics of the input data and the algorithms being executed. Once the data flow network is executed, an image is rendered. This rendering is typically parallelized for large data, which consists of every processor rendering the geometry for its portion independently, followed by a large communication phase where the individual images produced by each processor are composited into a final image. More complex rendering strategies are employed for transparent rendering, shadows, etc.

3.1.4 Science Driver for Metric Problem

In Q2, VisIt was used to perform two important and very common visualization tasks; namely, isosurface extraction and volume rendering of the output of an important radiation transport code, Denovo. In Q4, the same visualization tasks were performed on a radiation transport simulation 3 times the size of the Q2 simulation.

In the baseline and metric problems, VisIt processes datasets from the Denovo simulation code. Denovo is a new, state-of-the-art, 3D radiation transport code being developed at ORNL. It is currently being used to study and analyze radiation dose levels in a variety of engineering environments. The particular problem used for this benchmark involves the reactor core, containment vessel, turbines, and surrounding buildings in a nuclear power generating plant (a pressurized water reactor [PWR] facility). The code is currently being used to study the radiation dose levels under normal operating conditions, with plans to study doses following terrorist attack scenarios in the future. Accurate, high-fidelity understanding of the dose contours around the core and in the surrounding buildings is critical for health and safety assessment and cleanup, as well as new plant design and remodeling.

Scalability of this code is critical due to the enormous memory requirements of this type of transport calculation. High-fidelity 3D calculations using 1,000 energy groups results, for example, in 288,000

degrees of freedom (DoF) per cell. With a limited amount of memory available per node, massive parallelism is mandatory for this type of calculation. As codes such as Denovo scale to higher degrees of parallelism and larger numbers of computational domains, the resulting output files will similarly increase. It is therefore critical that the capability of the analysis and visualization tool similarly scale to handle increasingly larger numbers of computational domains.

The example used in the baseline and metric problem is a steady-state transport calculation of radiation dose in a PWR facility, as shown in Fig. 3. The model consists of material components consisting of concrete, reactor fuel, steel, reduced density steel, and air decomposed over many spatial domains.

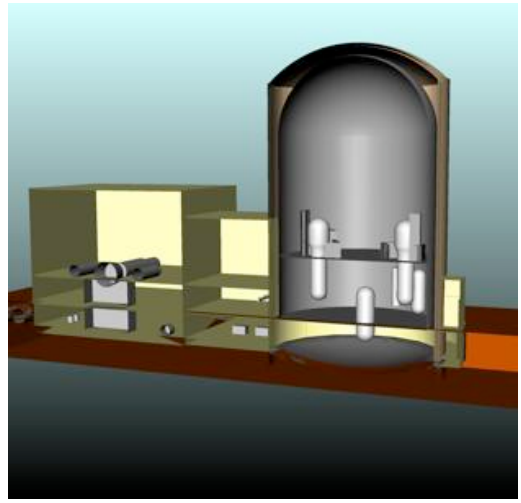


Fig. 3. Nuclear power plant, a PWR facility, set up for the Denovo simulation.

Denovo is a 3D, discrete ordinates (S_N) transport code that utilizes state-of-the-art solution methods to obtain accurate solutions to the Boltzmann transport equation. Denovo uses the Koch-Baker-Alcouffe (KBA) parallel sweep algorithm to obtain high parallel efficiency on hundreds of processors on block-structured Cartensian (orthogonal) meshes. As opposed to traditional S_N codes that employ source iteration, Denovo uses nonstationary Krylov methods to solve the within-group equations. Krylov methods are far more efficient than stationary schemes. Additionally, classic acceleration schemes (Diffusion Synthetic Acceleration) do not suffer from stability problems when used as a preconditioner to a Krylov solver. Denovo's generic programming framework allows multiple spatial discretization schemes and solution methodologies. Denovo currently provides diamond-difference, theta-weighted diamond difference, linear-discontinuous finite element, trilinear-discontinuous finite element, and step characteristics spatial differencing schemes. Also, users have the option of running traditional source iteration instead of Krylov iteration. Multigroup upscatter problems can be solved using Gauss-Seidel iteration with transport, two-grid acceleration. A parallel first-collision source is also available. Denovo has been verified against a number of problems, including several from the Kobayashi benchmark set. Initial parallel performance tests exhibit excellent strong scaling up to 100 processors and good scaling to 1,000 processors for high-fidelity problems.

3.1.5 The Model and Algorithm

For the analysis and visualization scaling study, two common algorithms will be used, isosurfacing and volume rendering. An isosurface is the 3D analog of a level set (or "contour"). Given a scalar-valued function, the level set is defined as the set where a function has a specific value, as follows:

$$f(x_1, \cdots, x_n) = c$$

In the visualization community, a number of different techniques have been developed for computing the solution to the level set equation, the most common being the Marching Cubes algorithm [1], which computes a polygonal approximation to the level set. Isosurface extraction is a very useful visualization technique as it clearly illustrates interface boundaries of a scalar variable. This is particularly useful in the numerical quantification of radiation dose level (obtained from Denovo solutions) because it clearly delineates areas within the computational domain that experience a given dose.

Volume rendering is a technique that produces an image directly from a scalar field in a 3D data set without producing intermediate geometry. Each value in the scalar field is assigned a color and opacity, as defined by a user-specified transfer function. After the transfer function has been applied to the scalar values in the mesh, the resulting color and opacity values are composited in front-to-back order (as defined by the viewing direction) to form the final image.

There are a number of techniques that have been used to composite color and opacity from scalar fields, including splatting, texture mapping, and ray casting. Generally, ray-casting techniques are the most accurate and produce the highest quality images. For the problem set chosen, the ray-casting volume render algorithm in VisIt is used.

In volume rendering, the amount of work is determined by three factors: the size of the computational domain (mesh), the size of the final image, and the number of sample points extracted along each ray. Volume rendering is a powerful tool for analyzing the distribution of scalar values in a volume and particularly useful to the radiation transport dose solution as a way to see the entirety of the dose levels.

The ray-casting volume renderer in VisIt consists of two fundamental stages. In the first stage, the algorithm takes as input a large data set that has been partitioned over its processors. The second input to the algorithm is the definition of the rays, which are determined by the view frustum and image size, with one ray per image pixel. Then, in parallel, each processor calculates intersections of each of the rays with its portion of the larger data set. At this point, no processor has enough data to composite the values along the rays into the final color for the pixel, since the intersections for a given ray will be spread over many processors. VisIt solves this problem by creating a second parallel partition, which is over all pixels. It then redistributes the partical ray data from the intersections so that they honor this new partition. This redistribution phase sends data points using numerous parallel point-to-point communications. Once the data from the intersections are repartitioned, compositing is trivial, because all of the data for a given ray is on a single processor. Each processor composites the set of rays, then the resulting image is collected to processor 0 where it can be displayed to the user.

3.1.6 Q2 Baseline Problem Results

For the Q2 baseline problem, we have selected two common analysis and visualization techniques, isosurface extraction and volume rendering. These two algorithms are exercised on the output of a Denovo solution of the radiation dose concentrations around a reactor core in a nuclear power generating plant. The intent is to demonstrate weak scaling in the analysis and visualization of the radiation dose transport using two different algorithms, isosurface extraction and ray-casted volume rendering.

The Q2 benchmark consists of a Denovo simulation of 4,096 spatial domains run on 4,096 processor cores of Jaguar/XT5. The computational mesh contains 103,716,288 cells with scalar flux values for 27 energy groups computed within each zone. The simulation outputs each computational domain to a separate file (hence 4,096 files) in the binary Silo/HDF5 format using double-precision floating point values. The cumulative size of all output files is 83.457 GB.

Isosurface Baseline. In this benchmark, contours at dose isovalues of 0.001, 0.01, 0.1, 1.0, 10.0, and 100.0 are computed, extracted, and rendered at a resolution of 1,024 × 1,024 pixels using VisIt running on 4,096 cores. The radiation dose is computed as part of the data flow network using the expression engine inside VisIt. VisIt's expression engine reads in the 27 energy level flux values from the simulation output files and, using a set of user-defined weights, linearly combines them to create the dose variable that is ultimately presented to the analyst.

In postproduction analysis and visualization tools such as VisIt, the most important productivity (hence benchmarking) metric is the time it takes to render a frame to a user display. Under normal use cases, the user loads the simulation data from disk into memory and then repeatedly interacts with the data, successively modifying isovalues and observing the results. The metric focus is therefore on the scalability of the isosurface extraction algorithms and not on the scalability of the one-time expense of reading simulation data from disk. The data in Tables 2 and 3 summarize the results for the Q2 baseline run. *Pipeline* is the execution time for the entire isosurfacing data flow network, excluding I/O. The timing value for *Pipeline* includes the following major components: *Isosurface,* the time to extract the isosurface geometry; *Render,* the time to render the geometry; *Comp*, the time required to composite and display the resulting image; and *Expr,* the time required to create the radiation dose variable using the expression engine. The minimum, maximum, and average times of the 4,096 core timings are reported.

Since the result image is not displayed until all cores are finished, the maximum and average pipeline times determine how quickly results are displayed to the user.

Table 2. Per core timings for the Q2 isosurfacing benchmark

	Minimum	Maximum	Average
Pipeline			
Isosurface	0.0140	0.0270	0.01768
Render	0.0200	0.0650	0.02245
Comp	0.0480	0.0870	0.05193
Expr	0.1810	0.2450	0.21097

Table 3. PAPI hardware counter data for the Q2 isosurfacing benchmark

PAPI hardware counters	Counter value
Total instructions	9.48E+14
FP instructions	1.73E+11
L2 Cache misses	33.83E+10
Real cycles	5.69E+11
Real (μs)	2.47E+08
User cycles	2.63E+11
User (μs)	1.14E+08

Volume Rendering Baseline. In this benchmark, a 1,024 × 1,024 pixel volume-rendering image of the radiation dose variable is computed. As in the isosurface benchmark, this variable is computed using VisIt's expression engine from the 27 energy group flux values stored in the simulation output files. In ray-casted volume rendering, the viewpoint has a direct impact on the amount of computational work to be performed prior to rendering the image. For this reason, the viewpoint is set so that the data is centered and fills the entire image. This maximizes the amount of work required during pipeline execution.

As in isosurface extraction, the time to render frames once the data is loaded from disk is the most relevant metric to the end user. The focus is therefore on the scalability of the volume rendering algorithms and not on the scalability of the one-time expense of reading simulation data from disk.

Table 4 summarizes the timing data for the Q2 baseline run. *Pipeline* is the execution time for all stages in the volume rendering data flow network, excluding I/O. *Vol Render* is the execution time for the volume rendering filter, which consists of three major components: *S Extract,* sample point extraction; *S Comm,* sample point communication; and *Expr,* which is the time required to create the radiation dose variable using the expression engine. The minimum, maximum, and average times of the 4,096 core timings are reported. Since the resultant image is not displayed until all cores are finished, the maximum and average pipeline times determine how quickly results are displayed to the user. The per core timings and hardware counter results for 500, 1,000, 2,000, and 4,000 samples per ray are given in Tables 4 and 5.

Table 4. Per core timings for the Q2 volume rendering baseline

	500			1000			2000			4000		
	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
Pipeline	38.9050	38.98100	38.91322	34.51800	34.74800	34.67289	28.91100	29.01800	28.92484	31.47800	31.62200	31.49796
Vol Render	38.7050	38.73800	38.71340	34.31200	34.36000	34.32562	28.71600	28.78000	28.72930	31.27700	31.41100	31.29756
S Extract	0.05900	0.14200	0.11783	0.09900	0.23600	0.19002	0.10000	0.33200	0.25329	0.13400	0.51000	0.40080
S Comm	38.5090	38.56600	38.54388	34.03100	34.11200	34.07852	28.33500	28.46500	28.41073	30.69500	30.90200	30.81717
I Comm	0.00000	0.08800	0.00301	0.00000	0.13000	0.00444	0.00000	0.21500	0.00741	0.00000	0.36800	0.01221
Expr	0.156	0.205	0.16285	0.156	0.199	0.16283	0.156	0.199	0.16275	0.155	0.199	0.16281

Table 5. PAPI hardware counter data collected for the Q2 volume rendering baseline

	500	1000	2000	4000
Total instructions	1.10E+15	1.18E+15	1.038E+15	1.068E+15
FP instructions	1.33E+11	1.50E+11	1.79E+11	2.01E+11
L2 cache misses	7.11E+10	7.11E+10	6.70E+10	7.20E+10
Real cycles	3.78E+11	4.08E+11	4.47E+11	4.57E+11
Real (μs)	1.64E+08	1.78E+08	1.95E+08	1.99E+08
User cycles	2.31E+11	1.81E+11	2.10E+11	2.16E+11
User (μs)	1.01E+08	7.86E+07	9.14E+07	9.38E+07

3.1.7 Computational Performance Gains

The isosurfacing pipeline exhibited excellent weak scaling as implemented and no modifications were required. However, while performing this work, the volume rendering pipeline was run on more processors than had previously been attempted. In experimental volume rendering studies, it was discovered that good scalability was observed up to 1,024 processors, but performance thereafter dramatically dropped off at 2,048 processors and beyond. Investigation led to the discovery of two bottlenecks to scalability, one major and one minor.

The major bottleneck was discovered in an $O(n^2)$ algorithm (n is the number of processors) that performed an optimization step to minimize communication between processors. Ray-casted volume rendering is an image space-rendering technique. At each pixel in the output image, a ray located at a pixel and parallel to the viewing direction is created. This ray is intersected with the data, and a specified number of samples are extracted along the ray. Each sample along the ray is assigned a color and transparency according to the user-specified transfer function. To produce a final color at each pixel, these samples must be combined in a back-to-front compositing step. Performing this operation efficiently in a parallel, distributed memory setting is very complicated, as described in [2].

The VisIt volume-rendering algorithm executes in two major phases, the first parallelizing across the mesh and the second parallelizing across the pixels in the final image. In the first phase, the cells in the mesh are evenly distributed across the processor set, and then each processor generates samples from the cells it owns from each ray. At the end of this phase, all of the samples along a given ray can be located on many different processors, so it is not yet possible to calculate the final pixel color. To solve this, the algorithm enters a second phase, where the data is partitioned such that all of the samples along a given ray are located on a single processor. Redistributing the sample points requires substantial point-to-point (arbitrary processor to arbitrary processor) communication. It is in this part of the algorithm where a barrier to scalability was discovered. There is the potential for a tremendous amount of communication when the sample points are redistributed. To try and minimize this communication, this phase of the algorithm examines the distribution of the sample points to processors and then attempts to create an assignment of pixels to processors in a manner that "minimizes" redistribution communication. That is, if a processor p already has many of the sample points for pixel q, the algorithm attempts to assign pixel q to processor p. This is implemented with an all-to-all communication primitive that requires an $O(n^2)$ amount of memory. This optimization is effective for small processor counts, but it was found that the coordination overhead does not scale and, at large enough processor counts, causes VisIt to run out of memory and fail. The solution is to skip the optimization and simply assign pixels to processors without concern for the distribution of the sample points. This enables the avoidance of $O(n^2)$ and even $O(n)$ buffers. It is possible to revisit this algorithm in an effort to minimize overall communication, but this is likely to cause the presence of, at a minimum, $O(n)$ buffers and hence would not be cost effective. Finally, because the data is being partitioned to a finer and finer degree with such a large processor count, the amount of communication saved becomes progressively smaller as the number of processors rises.

The minor problem encountered is in the VisIt tiling algorithm. When volume rendering a $1{,}024 \times 1{,}024$ pixel image with 1,000 samples per pixel, the algorithm must manage over one billion ($1{,}024 \times 1{,}024 \times 1{,}000$) samples. In a serial setting, or with a small number of processors, these samples are more than can be contained in primary memory. VisIt solves this problem by taking advantage of the fact that ray-casted volume rendering is an image space-rendering technique. Specifically, the image can be tiled into a sequence of smaller images. Each tiled image can be treated as an independent volume rendering, and the resulting tiles can be reassembled to form the final image. In doing this, the memory requirements are reduced. However, in a parallel setting, this tiling strategy has a very negative side effect. The cells owned by a given processor are fixed before the volume rendering begins, so it is quite possible for a processor to have no work to perform on a given tile. And so the processor will wait until the next tile is ready to volume render, decreasing the parallel efficiency of the algorithm. In effect, the tiling strategy serializes the volume rendering over tiles, with the benefit of ensuring a lower memory footprint.

However, this adaptation is not necessary when running in parallel on many processors. Each processor will own a portion of the samples, and as the number of cores grows large, the number of samples each core owns (which is essentially fixed) decreases. After a certain number of processors, the memory requirement of the sample points is small enough to fit into main memory. At this point, the tiling strategy can be removed and the volume rendering can be effectively *unserialized*. The solution was to simply disable the tiling strategy when running in parallel.

3.1.8 Q4 Metric Problem Results

For the Q4 metric problem, the same isosurface and volume rendering algorithms were run on an identical Denovo simulation 3 times the size of the Q2 simulation. Specifically, the Q4 benchmark simulation contained 12,720 spatial domains run on 12,720 processor cores of Jaguar/XT5. The computational mesh contains 321,117,360 cells with scalar flux values for 27 energy groups computed at each cell. The simulation outputs each computational domain to a separate file, making 12,720 files, output in the binary Silo format using double-precision floating points values. The cumulative size of all output files is 258.391 GB.

The timing results of the isosurface extraction on the Q4 metric problem are shown in Table 6 and the PAPI hardware counting data in Table 7. Weak scaling results for isosurfacing timings are shown in Table 8.

The timing results of the ray-casted volume rendering on the Q4 metric problem are shown in Table 9 and the PAPI hardware counting data in Table 10. Weak scaling results for volume rendering timings are shown in Table 11.

Table 6. Per core timings for the Q4 isosurfacing benchmark

	Minimum	Maximum	Average
Pipeline			
Isosurface	0.01411	0.02654	0.01686
Render	0.02079	0.06907	0.02319
SR	0.050	0.0912	0.05274
Expr	0.1933	0.2501	0.22390

Table 7. PAPI hardware counter data for the Q4 isosurfacing benchmark

PAPI hardware counters	Counter value
Total instructions	9.12E+14
FP instructions	5.23E+11
L2 cache misses	2.31E+11
Real cycles	1.08E+11
Real (μs)	4.70E+07
User cycles	7.42E+10
User (μs)	3.23E+07

Table 8. Weak scaling results of the Q4 benchmark

	Minimum	Maximum	Average
Pipeline			
Isosurface	0.992	1.0550	1.05456
Render	0.9620	0.95555	0.97187
SR	0.95865	0.95394	0.98483
Expr	0.93636	0.97960	0.94225

Table 9. Per core timing results for the Q4 volume rendering

	500			1000			2000			4000		
	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
Pipeline	7.47642	7.57774	7.4826	6.51973	6.63473	6.52798	6.5838	6.73839	6.60026	6.8265	7.04683	6.84922
Vol Render	7.24796	7.30064	7.25474	6.29565	6.36192	6.30441	6.36193	6.46614	6.37796	6.60335	6.77199	6.62592
S Extract	0.01855	0.0453	0.03042	0.02316	0.06854	0.04253	0.0282	0.1014	0.06169	0.03697	0.14928	0.0858
S Comm	7.1393	7.16601	7.15538	6.16343	6.20998	6.19064	6.19944	6.28523	6.24175	6.39171	6.53656	6.46006
Expr	0.1494	0.18839	0.16432	0.14966	0.18698	0.16435	0.14966	0.18803	0.16432	0.14956	0.18697	0.16440

Table 10. PAPI hardware counter data collected for the Q4 volume rendering benchmark

	PAPI Hardware Counters			
	500	1000	2000	4000
Total instructions	1.10E+15	1.18E+15	1.038E+15	1.06E+15
FP instructions	1.33E+11	1.50E+11	1.79E+11	2.01E+11
L2 cache Misses	7.11E+10	7.11E+10	6.70E+10	7.20E+10
Real cycles	3.78E+11	4.08E+11	4.47E+11	4.57E+11
Real (μs)	1.64E+08	1.78E+08	1.95E+08	1.99E+08
User cycles	2.31E+11	1.81E+11	2.10E+11	2.16E+11
User (μs)	1.01E+08	7.86E+07	9.14E+07	9.38E+07

Table 11. Weak scaling results of the volume rendering benchmark timings

	500			1000			2000			4000		
	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
Pipeline	5.20	5.14	5.20	5.29	5.24	5.31	4.39	4.31	4.38	4.61	4.49	4.60
Vol Render	5.34	5.31	5.34	5.45	5.40	5.44	4.51	4.45	4.50	4.74	4.64	4.72
S Extract	3.18	3.13	3.87	4.27	3.44	4.47	3.55	3.27	4.11	3.62	3.42	4.67
S Comm	5.39	5.38	5.39	5.52	5.49	5.50	4.57	4.53	4.55	4.80	4.73	4.77
Expr	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

3.1.9 Interpretation of Results

The Q4 metric problem was run on a Denovo simulation using the identical setup but using a computational mesh that was roughly 3 times larger than the baseline problem. The simulation was run with a total of 321,117,696 computational zones across 12,720 spatial domains with scalar flux values for 27 energy groups in each zone. Each spatial domain was output to a separate file, resulting in a total input file size of 258.391 GB. A comparison of the Q2 and Q4 problems sizes is shown in Table 12.

For the Q4 metric problem, the same isosurface extraction and volume rendering operations were performed using VisIt running on 12,720 cores.

Table 12. Q2 and Q4 simulation sizes

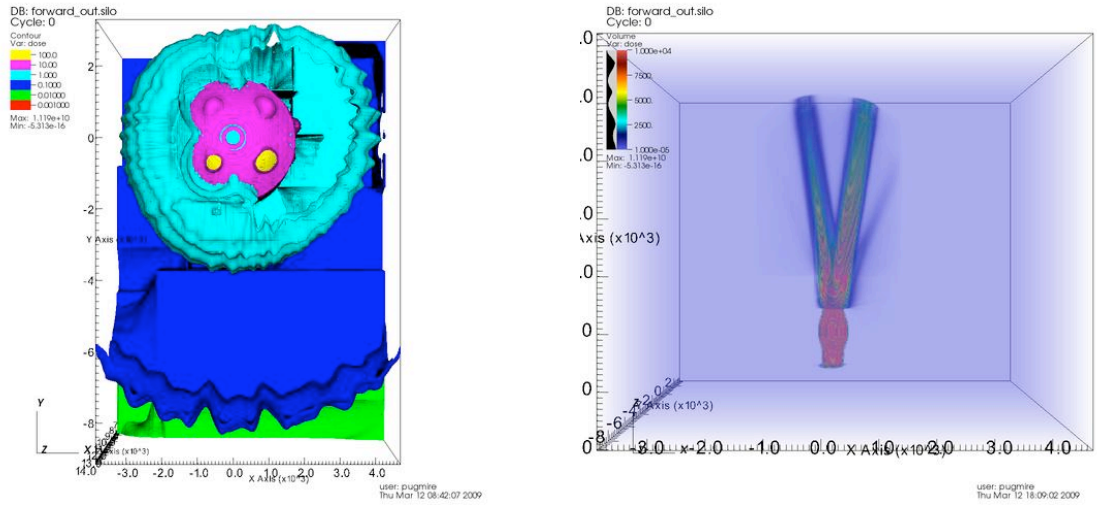
Problem metric	Q2 Problem	Q4 Problem	Q2–Q4 Comparison
Number of zones	103,716,288	321,117,696	3.096
Number of domains	4,096	12,720	3.105
Total file size	83.457 GB	258.391 GB	3.105

The isosurface extraction algorithms are a fairly well understood, and scalability was expected. It was therefore gratifying to find that the framework in VisIt that manages and executes the isosurface extraction and rendering was able to exhibit very good weak scaling, satisfying the Joule criterion.

3.1.10 Summary and Conclusions

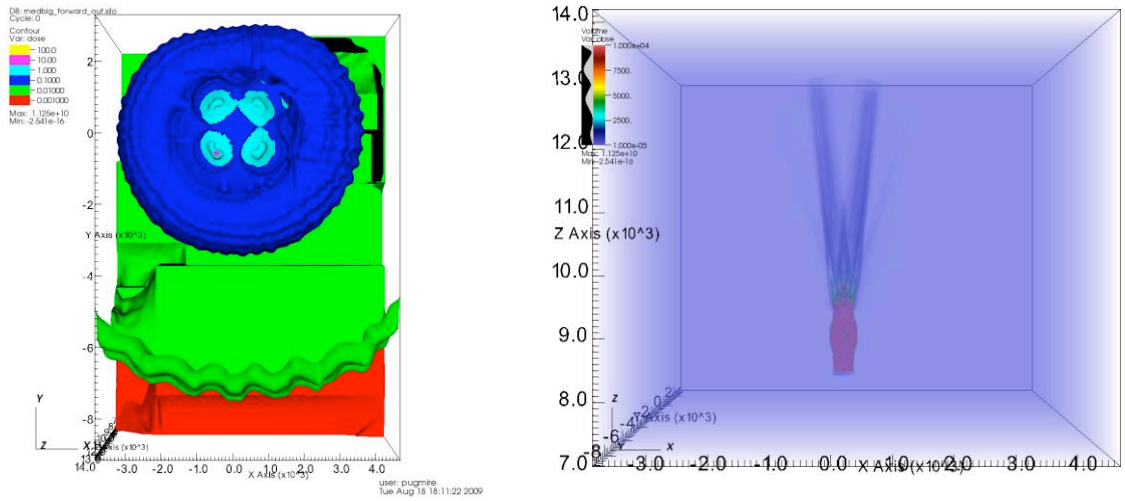
We have run two benchmarks using VisIt, isosurfacing and volume rendering, on two radiation dose transport simulations from Denovo. The resulting images from the Q2 baseline and the Q4 metric problem are shown in Figs. 4 and 5, respectively. The isosurfacing metric was shown to exhibit ideal weak scaling. The Joule metric proved to be particularly useful to the volume rendering as two barriers to scalability were identified and addressed. The modifications to the volume-rendering algorithm resulting in significant performance improvements will benefit the entire analysis and visualization community as simulations continue to grow in size and scope.

The output produced from the VisIt analysis and simulation runs produced expected results that have been verified with the code developer and are deemed acceptable. We have therefore accepted the results for the Q2 benchmark runs.



(a) *(b)*

Fig. 4. *(a)* Extraction of radiation dose contours and *(b)* a volume rendering from the nuclear power plant simulation from the Denovo code. Q2



(a) *(b)*

Fig. 5. *(a)* Extraction of radiation dose contours and *(b)* a volume rendering from the nuclear power plant simulation from the Denovo code. Q4

3.2 CAM

3.2.1 Introduction

The Community Atmosphere Model (CAM) [3–5] is the latest in a series of global atmosphere models developed at the National Center for Atmospheric Research (NCAR) for the weather and climate research communities. CAM also serves as the atmospheric component of the Community Climate System Model (CCSM). The latest version of CAM (in its fifth generation) has been designed through a collaborative effort that includes NCAR, university, and laboratory users and developers, and its contents are defined by the CCSM Atmospheric Model Working Group (AMWG). Some of the key features in CAM include updated parameterizations for prognostic cloud water, cloud ice, precipitation, and cloud fraction; the radiative treatment of atmospheric aerosols (sulfate, dust, sea salt, carbon, and volcanic), the optional prognostic treatment of sulfate aerosols; improved energy conservation; improvements to the long-wave radiation interaction with water vapor; updates to the shortwave radiative transfer scheme to more accurately model trace gas absorption; and an atmosphere-land interface that now supports rain and snow phases. CAM also includes an optional slab ocean model and incorporates an International Satellite Cloud Climatology Project (ISCCP) cloud simulator to emulate ISCCP statistical cloud diagnostics.

3.2.2 Background and Motivation

Over the last two decades, NCAR has provided a comprehensive, 3D global atmospheric model to scientists all over the world for use in the analysis and understanding of global climate. Because of its widespread use, the model was designated a community tool and given the name Community Climate Model (CCM). The original versions of the NCAR CCM, CCM0A [6] and CCM0B [7], were based on the Australian spectral model [8] and an adiabatic, inviscid version of the European Centre for Medium-Range Weather Forecasts (ECMWF) spectral model [9]. The CCM0B implementation matched the earlier CCM0A model to within natural variability, but in addition provided a more flexible infrastructure for conducting medium- and long-range global forecast studies. All aspects of the model in the CCM0B effort were described in a series of technical notes [10] and a detailed code and algorithm description [11]. The most recent version of CAM (CAM 3.0) incorporates significant improvements to the physics package (e.g., generalized cloud overlap for radiation calculations), new capabilities such as the incorporation of thermodynamic sea ice, and a number of enhancements to the implementation (e.g., clean separation between physics and dynamics).

3.2.3 Capability Overview

Physical Model. The model implementation is characterized by two computational phases: the resolved dynamics, which advances the evolution equations for atmospheric flow, and the physics, which treats subgrid-scale phenomena such as precipitation processes, clouds, long-wave and short-wave radiation transfer, and turbulent mixing. Control moves between the dynamics and the physics twice during each model simulation time step. A dynamics–physics coupler moves information between data structures representing the dynamics state and the physics state.

Numerical Model. CAM includes multiple options for the dynamics, referred to as dynamical cores: a spectral Eulerian, a spectral semi-Lagrangian, finite volume, and cubed sphere. The spectral and semi-Lagrangian dynamical cores use the same computational grids. Finite volume and cubed-sphere grids both differ from these grids due to differences in the mathematical formulations. An explicit interface exists between the dynamics and the physics, and the physics data structures and parallelization strategies are independent from those in the dynamics.

Software Implementation. The software design of the CAM model includes a hybrid (OpenMP/MPI) approach in order to efficiently map to the multiple symmetric multiprocessor (SMP) node nature of many modern supercomputers, including the Cray XT-series machines at ORNL. The strategy for parallel decomposition is different in the physical parameterizations vs. the dynamical cores,

so a data transpose is necessary twice each model time step: once from the physics grid to the dynamics grid, and once from the dynamics grid to the physics grid.

Data dependence in the physical parameterizations is only in the z (vertical) dimension. Physical parameterization data in this dimension is always on-processor in parallel OpenMP threads as well as Message Passing Interface Standard (MPI) processes, so the parameterizations themselves do no communication. Data are arranged in sets of (x,y) points called *chunks*. Each chunk can be thought of as a set of independent vertical "pencils," where the computations on each pencil are independent of each other. A given MPI process is assigned some number of chunks, with per-task parallelism achieved by an OpenMP loop over the number of chunks. The size of a chunk is a compile-time setting. Most of the physical parameterizations have an inner loop over the chunk size, which provides opportunities for vectorization on machines that provide it.

Since no communication is required in the parameterizations, their performance scales well with increasing thread and process count. The scaling is not perfect, however. There are two key reasons for this. First, as additional threads are added, the requirements on the memory subsystem increase. CAM is a memory-intensive code, so memory performance degrades when the nodes are fully populated (8 threads per node on the current XT5 system at ORNL).

The second reason that the parameterizations do not scale linearly is that there is an inherent load imbalance imposed by the physics being modeled. As an example of load imbalance, consider that the sun is above the horizon on only half of the (global) model grid points at any particular time. There is a shortwave radiation calculation required at sunlit points, which is not done for points that fall below the terminator. This can cause a substantial load imbalance since the shortwave radiation calculation is relatively expensive, and the set of points that require it constantly changes as the simulated time of day changes. An attempt is made to statically load-balance the distribution of points to processors. The approach involves including points near the North Pole with points near the South Pole in the same chunk. Therefore, in northern hemisphere winter, for example, a given polar chunk should contain roughly half southern hemisphere points which are sunlit, and half northern hemisphere points which are not.

Data decomposition for parallelization of the spectral Eulerian dynamical core is across latitude bands. Each MPI task is assigned some number of latitudes in the part of the spectral transform that begins in grid-point space and applies a Fourier transform. If more that a single latitude band is assigned to a particular MPI task, OpenMP parallelism is applied via a loop over the number of assigned latitudes. Once in spectral (wave number) space, parallelization is across Fourier wave numbers. To transform back to grid-point space, a data transpose is applied within the dynamical core to rearrange the data such that all Fourier wave numbers are contiguous in order to apply a reverse Fourier transform.

The data decomposition in the spectral Eulerian dynamical core is only one dimensional (y direction). Prior to the advent of massively parallel computational platforms, this did not pose any bottleneck. Also, at the lower spatial resolutions of earlier simulations, the cost of the spectral dynamics relative to the physical parameterizations is much less (see next section). Our intent is to address overall performance and scaling issues mainly related to the spectral Eulerian dynamical core at high resolution.

3.2.4 Science Driver for Metric Problem

A current focus area of climate change research is predictability on decadal timescales. These studies require a numerical model with very high spatial resolution (e.g., on the order of 30 km resolution in longitude and latitude). Until recently such simulations were not feasible due to the concomitant computational requirements. Only now with petascale-level platforms possessing adequate per-processor performance do such studies become tractable. Specifically, an ongoing atmospheric science research focus at ORNL involves the use of a high-resolution global atmospheric general circulation model (AGCM) to hindcast the climatic impact of volcanic eruptions (Fig. 6).

An important design aspect of this work calls for the configuration of a global atmospheric model and associated land surface model with forcing datasets that enable us to address specific science questions about the response of the climate system to natural and anthropogenic aerosol forcing. These forcing datasets employ best estimates of observed solar variability and greenhouse gas mixing ratios during the experimental period. The atmospheric model is configured at a resolution of approximately 30 km to ensure adequate representation of regional features such as the orographic signal of precipitation. Such high resolution is also essential for the land model to develop a realistic soil moisture pattern. Another benefit from this high-resolution configuration is a more realistic representation of both extra-tropical and tropical storms.

The atmospheric model and land model chosen for this study, along with the spatial resolution, form the basis of the climate component of the FY09 Joule exercise. A description of the models, parallel decomposition, boundary datasets, and initial Q2 results are described in the following sections.

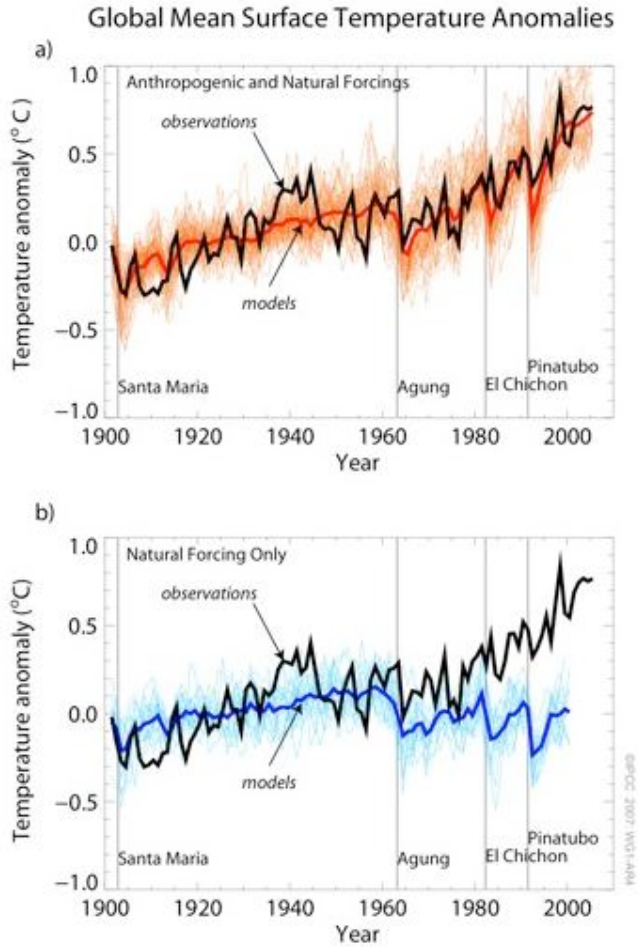


Fig. 6. Global average surface temperature in observations, modeled with and without anthropogenic forcing. Volcanic eruptions are clearly visible in their rapid cooling effect. Source: *Fourth Assessment Report of the United Nations Intergovernmental Panel on Climate Change.*

3.2.5 The Model and Algorithm

The Community Atmosphere Model (CAM) version 3.0 is the fifth generation of the National Center for Atmospheric Research (NCAR) AGCM used in climate studies. The name of the model series was changed from Community Climate Model to Community Atmosphere Model (CAM) to reflect the role of CAM 3.0 in the fully coupled Community Climate System Model (CCSM). The CCSM couples CAM and active land, ocean, and sea ice components together to form a fully interactive climate system model. CAM is designed through a collaborative process with users and developers in the Atmospheric Model Working Group (AMWG). The AMWG includes scientists from NCAR, the university community, and government laboratories and agencies such as ORNL.

The CAM configuration for this study runs in uncoupled (stand-alone) mode. This configuration includes a fully active land model (Common Land Model or CLM), with sea surface temperatures (SSTs) and sea ice concentrations provided by external forcing datasets. The term "uncoupled" refers to the fact that there is only one executable image, with communication between component models via a subroutine interface. In "coupled" mode, boundary flux information is passed between individual models and a coupler using separate executables that communicate with one another via MPI. CAM can be configured to use any of four dynamical cores: spectral Eulerian, finite volume, semi-Lagrangian, or cubed sphere. The spectral Eulerian dynamical core is used for this study because its characteristics and behavior are well understood. The horizontal model resolution is T341 (see http://vets.ucar.edu/vg/T341), which results in a transform grid (latitude/longitude) of 1,024 × 512 points. This represents approximately 0.35° of latitude and longitude. There are 26 vertical levels.

The CAM model version used in this study is 3.5. CAM 3.5 contains some modifications to the physical parameterizations beyond CAM 3.0. Details of the mathematical formulations are available in ref. [5]. Further details on the Eulerian dynamical core and physical parameterizations can also be found in this section.

A key element in the design and implementation of the numerical methods for the CAM model is the coupling between the physical parameterizations and the dynamical core (this physics–dynamics coupling is not the same as the inter-model coupling just described). There are important performance ramifications of this coupling process as implemented on a parallel architecture. The mathematical formulation of the coupling is described in Chapter 2 of ref. [5].

3.2.6 Q2 Baseline Problem Results

Since the AGCM used in this study is a global model, traditional application of a weak-scaling approach to increasing the problem size is not possible given the science needs. This is because increasing the number of model grid points for a constant physical size domain necessarily shortens the distance between them. Numerical stability considerations (e.g., CFL constraint) dictate a shorter dynamics time step as grid points become more tightly spaced. This results in a substantial increase in the amount of floating-point work performed per core for a given model time integration. The computational cost of the dynamics part of the calculation as a result of doubling the resolution in both x and y dimensions goes as the cube of the resolution increase rather than the square. Computational cost of the physics part of the calculation resulting from this same increase in resolution only goes as the square of the resolution increase. As a consequence, the cost of the dynamics part of the calculation begins to dominate the cost of running the entire model as the horizontal resolution is increased.

Scientific demands from climate models do not at this time justify moving to a much finer mesh than outlined above. Therefore, the CAM Joule metric applied for this exercise is model run time for a T341 configuration at high process count. The model was run for one simulated month. This is the minimal run time required to accurately represent the relative time taken by various model components for an arbitrarily long run, including the impact of model I/O.

A total of 8,192 processor cores were used for the model run. On the Jaguar/XT5 system this is distributed as 1,024 MPI tasks, with 8 OpenMP threads per task. This means one MPI task per eight-processor node on the system. For the final (Q4) simulation, improvements to the computational algorithms were devised and implemented to enhance the performance of the model. The algorithms themselves were not changed in a wholesale manner, but allowance was given for potential numerical differences at the round-off level in the improved implementation. This approach simplifies the process of assuring that the model is generating the "right" answers. "Correct" answers are defined by the answers generated in the baseline (Q2) simulation. The same initial and boundary condition files were used in both the Q2 and the Q4 runs.

The CAM model can be configured to periodically write a snapshot file of prognostic data that can be used as an initial conditions file for subsequent runs. This is useful because the model requires a number of years of simulation time before it settles to a balanced (quasi-equilibrium) state. Fortunately, a run

performed with an earlier version of the model produced a number of these files. We chose one of these as an initial conditions file for this exercise.

SST and sea ice concentration data are prescribed by monthly boundary datasets. These data are read in each simulated month and then interpolated in time. Monthly ozone and aerosol concentration bounding data are also prescribed, input to the model and interpolated to current model time as the model runs.

The CLM requires specification of surface type and characteristics. These are provided by boundary datasets. Like the atmospheric model, CLM can start from an initial dataset. Although such a dataset is currently not available at the resolution of this study, the model instead provides the facility to start from an internally specified initial state. Spinning up to an equilibrium state requires a number of years of simulation. However, the computational characteristics between a spun-up state and this so-called "arbitrary" initialization are nearly identical.

The benchmark results reported here are for a one-month simulation on 8,192 processor cores of the Jaguar/XT5 platform.* A constant time step of 150 seconds was used, with the total integration being 17,856 time steps. The cost of the dynamical core itself was 4,247 seconds, which represents approximately two-thirds of the total simulation time of 6,482 seconds. The CLM cost was only 112 seconds. The cost of writing CAM history files was similarly small (115 seconds). This is because in default mode, the model only writes history files once per simulated month. In the Q2 runs a substantial performance gain was not realized by increasing the processor count from 4,096 to 8,192. This is primarily due to unexploited opportunities for parallelism in the dynamical core, and the fact that the relative computational cost of the dynamical core is quite high at a horizontal resolution of T341. In contrast to the dynamical core, the physical parameterizations in CAM take full advantage of the 2D (x–y) opportunities for parallelism. Performance data of the benchmark computation are shown in Table 13. The General Purpose Timing Library (GPTL) [12] was used in conjunction with the Performance Application Programming Interface (PAPI) to extract the performance data.

Table 13. CAM performance data for the Q2 benchmark run*

	Atmosphere	CLM	I/O	Total
Time (seconds)	5,916.475	112.048	115.024	6,481.724
FP instructions	2.13×10^{15}	3.89×10^{13}		2.17×10^{15}

*The 8192-core simulation on Jaguar/XT5 consisted of a 17,856 time step integration (one month at 150 seconds per step) on a T341 grid (1024 latitude × 512 longitude × 26 vertical).

A few items are worth noting for the CAM benchmark problem and associated collection of performance data. First, timing for the run is the same across all OpenMP threads and MPI tasks. This is because of barriers and synchronization necessary for the algorithms. Second, floating-point instruction data is collected from an equivalent MPI-only simulation (one MPI task per core) in order to avoid having to aggregate over all OpenMP regions. The MPI-only and hybrid OpenMP/MPI runs yield the exact same numerical results, so at present the floating-point instruction mix (number and order) for the two runs is the same although this is not necessarily true in general. Total instruction count data must be interpreted carefully because extraneous integer instructions executed by a given task can signify a load imbalance (e.g., spinning in user space waiting for any kind of barrier) rather than actual computational work.

3.2.7 Computational Performance Gains

Modifications implemented to achieve a greater than 2 times speedup for the CAM portion of a one-month CAM+CLM T341 model run fall in four categories: source code modifications, compiler flags and improvements, run time configuration flags, and modifications to the I/O subsystem.

* http://www.nccs.gov/jaguar/

We devised numerous source code modifications to the spectral Eulerian dynamical core in CAM. The impact was to obtain substantial speedups when run at particularly high resolution. The main theme of these modifications was to exploit embedded opportunities for parallelism at the OpenMP (threading) level that did not exist in the standard code base. At T341 resolution, the granularity of the new regions parallelized was sufficient to overcome the overhead of the threading itself and produce substantial performance gains. Good performance gains were also the result of the fact that existing MPI parallelism in the spectral Eulerian dynamics peaks at 512 processes, so additional opportunities (in this case OpenMP parallelism in the dynamics) needed to be pursued to push the parallelism to 8,192 cores. In addition, the fact that the cost of the spectral dynamics scales roughly as the cube of the horizontal resolution, while almost all of the physics scales as the square, meant that the modifications to the spectral dynamics would have a particularly significant impact at high resolution. Finally, since the XT5 system at ORNL was upgraded in the past year from four cores per SMP node to eight, this provided an additional performance boost since these OpenMP code modifications all apply at the node level.

In the category of compiler flags and improvements, we note that the PGI compiler used for all runs went through two major revision upgrades between the Q2 and Q4 runs, from 7.2.3 to 9.0.1. This change alone had a beneficial impact of approximately 10% on overall model run time. PGI is the default compiler on the ORNL Jaguar machines.

One change to default compiler flag settings from that used in the Q2 benchmark runs was to remove the flag "-Kieee" (the Q2 runs used exactly the same settings as the Jaguar-based Makefile maintained by NCAR). The impact of this flag was to force strict conformance with the IEEE 754 standard. By removing it, the compiler was able to utilize certain optimizations to its internal numerics that it would not otherwise have been able to do. The impact of turning off this flag sped up the CAM execution somewhat, without a significant impact on the generated solution. The original need for the flag was historical and it is no longer required.

The flag -Mvect=nosse was changed to -Mvect=sse. The default CAM model Makefile specifies -Mvect=nosse, which disables vector instructions. Enabling vector instructions increases the theoretical peak speed of the AMD processor by a factor of 2. The speedup realized by the CAM model when vector instructions were enabled was more like 20%, which is still a significant number. The reason -Mvect=nosse was in the original Makefile was for numerical reproducibility across all thread counts used in OpenMP or hybrid OpenMP/MPI runs. The effect on model answers by enabling this flag was at roundoff.

Flags -fast and -fastsse were also added to the Makefile. These are general optimization flags that had a minor impact on performance (though -fast implies an optimization level of at least -O2).

Various compile-time and run-time Fortran name-list settings were tested as modified from their default values between the Q2 and Q4 runs. These had no impact on model answers but did affect model performance. The run-time and name-list settings exist to address issues such as cache blocking in the physical parameterizations, and to optimize the use of MPI primitives as used mostly in the dynamics. The settings had never been optimized for a T341 resolution, or specifically T341 on the XT5 architecture, so some experimentation was necessary to achieve optimal results. Since there is some variability in model run time from one execution to the next simply due to issues such as operating system noise and overall load on the system, multiple runs had to be done to determine the best settings. This was particularly true of settings that had only a small impact on performance.

Compile-time C-preprocessor variable PCOLS is essentially a cache-blocking parameter. It is the number columns (or vertical "pencils" using the nomenclature from earlier in this document) in a “chunk.” Recall that the columns defining a chunk need not necessarily be contiguous in physical space, though many inner loops in the physics do index over this variable. Generally, the best setting is independent of horizontal resolution. Since cache line sizes generally involve a power of 2, a PCOLS setting that is likewise a power of 2 is normally the best choice. And indeed it turned out that the default setting of PCOLS=16 gave the best results. PCOLS=8 was nearly as efficient.

The value of name-list parameter phys_loadbalance was changed from its default value of 0 to 3 between Q2 and Q4 runs. “0” says not to do any load balancing within the physical parameterizations. “3”

says to find one process to exchange data with when balancing the physics load. For the usual latitude decomposition, this translates to finding the process with the "mirrored" latitude in the other hemisphere, and the load balance is usually very good. “2” says to do the optimal remapping, which is essentially an all-to-all communication. Physics load imbalances typically are not very large, so it is difficult to amortize the communication cost of option 2. “3” gave the best performance for the Q4 runs.

Name-list parameter dyn_alltoall has possible settings of 0 or 1. If set to “0,” MPI_Alltoall is used for the transposes within certain routines in the dynamical core. Otherwise a point-to-point implementation is used. Run-time variability swamped any signal from varying this setting. Therefore, the default setting of 0 was used in the final runs.

Time taken to write the CAM history file was only a small fraction of the total model integration time in both the Q2 and Q4 runs (less than 2% of the total). However, we found it remarkable that some combination of current system load, upgrades to the underlying Lustre software, and migration to a new center-wide external Lustre file server resulted in more than a 2 times improvement in time to perform this physical I/O. We checked to be certain that exactly the same amount of data were written in both the Q2 and Q4 runs. Restart (checkpoint) files were not written in order to obtain an accurate measurement of the relative fraction of time taken to do I/O in both the Q2 and Q4 runs.

3.2.8 Q4 Metric Problem Results

A wide variety of performance analysis tools are available on the Cray XT5 architecture. Our original intent was to use CrayPat to gather timings and underlying performance statistics such as total floating point operations. Unfortunately, we could not get believable numbers from this tool when applying it to hybrid OpenMP/MPI codes such as CAM. So instead we manually instrumented the code utilizing the GPTL timing library [12]. This library gives consistent, reliable performance data for hybrid OpenMP/MPI codes and also provides an optional interface to the PAPI library. PAPI provides detailed low-level hardware performance counter data such as floating point operation count. We double-checked correct behavior of GPTL+PAPI by constructing a test OpenMP/MPI code with a known floating point operation count. The floating-point operations (FP_OPS) measured by GPTL/PAPI were extremely accurate. This GPTL-based approach was very useful for diagnosing fine-grained performance improvements between Q2 and Q4. As a convenience in generating total floating point operation and instruction counts across all cores for the full model, we performed an additional simulation in MPI-only (unthreaded) mode. This allowed us to easily instrument a single code region across all MPI tasks, then gather the results with a simple post-processing script. Otherwise we would have had to manually instrument all OpenMP threaded regions (CAM and CLM contain many of these), then sum the results across all processes and threads. The numerical results in hybrid OpenMP/MPI mode identically match those of the MPI-only run, so we are confident that the PAPI results reported here are accurate.

Wall-clock times reported in results here were for thread 0 of MPI task 0. Since there are numerous synchronization points as the model integrates, the total wall-clock time of all MPI tasks will always be nearly identical. Floating-point operations and instruction counts are summed across all processes in an MPI-only run (as described above), with a single grand total reported.

For the Q4 run, the wall-clock time for the atmospheric component of the simulation on 8,192 cores using 1,024 MPI tasks and 8-way threading was 2,823.365 seconds (Table 14). This excludes initialization time. To aggregate statistics for instruction count and floating point operations, a Perl script was used to sum these statistics across the output timing data for all threads and tasks. Q4 comparison results are for the same region specified in the timing output files (“DRIVER_ATM_RUN”).

The simulation done for Q4 matches that done for Q2 both in problem size (T341) and processor count (8,192). The distribution of MPI tasks (1,024) and OpenMP threads per task (8) was also the same. As such, the CAM configuration chosen for this Joule exercise was strictly a strong-scaling problem. The performance results shown in Table 14 below demonstrate a greater than 2 times speedup in the Q4 CAM run as compared with the Q2 benchmark run (see Table 13). Comparing the time taken for the column labeled “Atmosphere” defines a speedup factor of approximately 2.1.

Table 14. CAM performance data for the Q4 modified run*

	Atmosphere	CLM	I/O	Total
Time (s)	2,823.365	101.310	41.302	3,241.144
FP instructions	2.31×10^{15}	6.09×10^{13}		2.38×10^{15}

*The 8,192-core simulation on Jaguar/XT5 consisted of a 17,856 time step integration (one month at 150 seconds per step) on a T341 grid (1,024 latitude × 512 longitude × 26 vertical).

The count of floating point instructions executed by CAM differs slightly between the Q2 and Q4 runs (2.13 × 1,015 vs. 2.31 × 1,015). This is to be expected and is due to a number of factors, including compiler upgrades and additional round-off differences introduced as a result of code modifications. The CLM portion of the simulation was unchanged vs. the Q2 simulation. Thus the 12% speedup observed in that part of the calculation can be attributed to compiler upgrades and whatever system noise may have been present. The floating point operation count for CLM is more than an order of magnitude smaller that that for CAM. As such it does not represent a significant fraction of the total, but it is curious that the value increased by nearly 50% from the Q2 run to the Q4 run, all without any code changes.

Time spent doing history file I/O decreased dramatically between the Q2 run and the Q4 run. This result is unrelated to code changes (there were no modifications to the I/O portion of the calculation) but rather the installation of a new system-wide Luster file system at ORNL. While the time spent writing history files was not a significant portion of total model time even in the Q2 run, the speedup of more than 250% observed in writing the same amount of history data is impressive.

The reasons behind these observed results are described in the next section.

3.2.9 Interpretation of Results

The CAM model at T341 scales reasonably well to about 4,096 processor cores. Attempting to execute this problem across additional cores rapidly reaches a point of diminishing returns. This is depicted in Fig. 7, where total CAM performance is broken down into dynamics and physics components. The physics scales well to 8,192 cores, but the performance of the dynamics flattens beyond about 2,000 cores. Since the current climate science goals that are driving the higher resolution CAM runs at ORNL demand reasonable turnaround (at least 2 simulated years per wall-clock day), there is a clear benefit realized by improving the strong-scale performance of the model. Figure 7 clearly shows that the performance of the dynamical core is the primary culprit in limiting scalability.

The definition of metric success with the modified CAM software and run-time environment was to achieve at least a 2 times speedup between Q2 and Q4. We succeeded in meeting this goal. Comparing the time taken by the CAM model in Q2 vs. Q4 (5,916.475 seconds vs. 2,823.365 seconds), results in a speedup factor of 2.095. Thus it is now possible to simulate more than 2 years per wall-clock day with the high resolution Eulerian spectral version of CAM, but in Q2 it was only possible to simulate 1 year per wall-clock day.

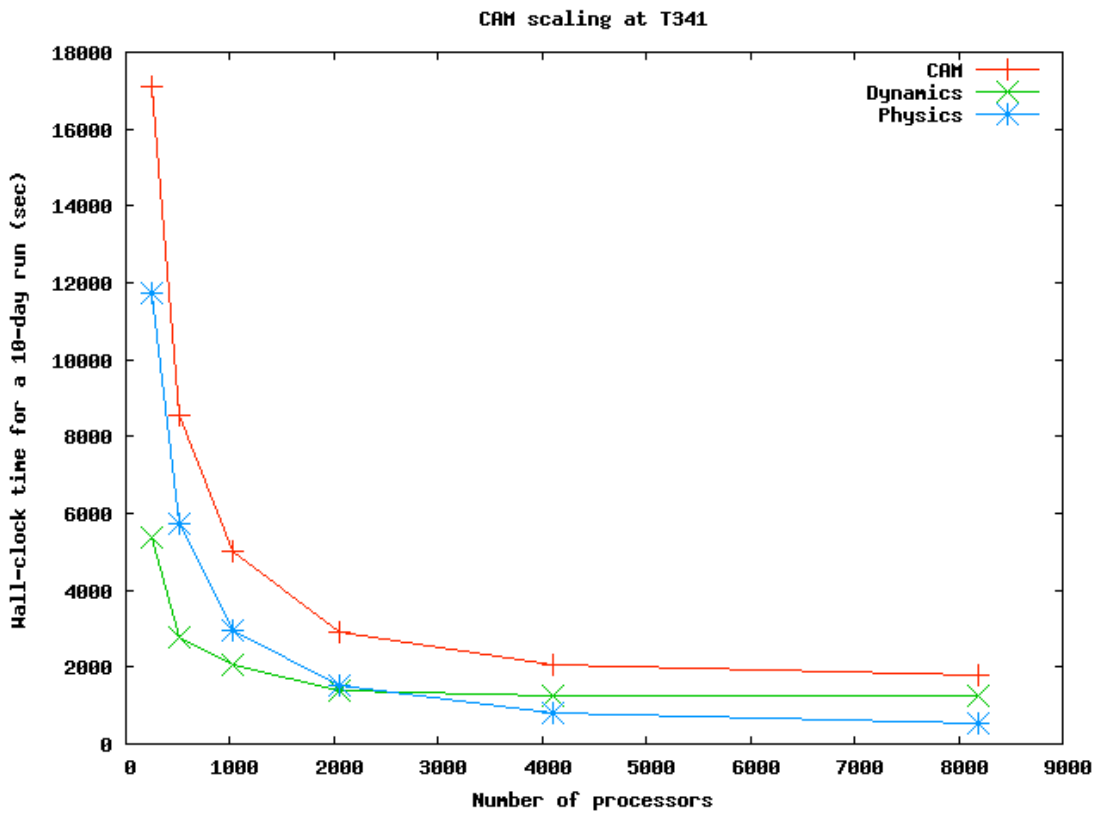


Fig. 7. Current CAM strong scaling performance for the T341 mesh on the Jaguar/XT5 platform.

3.2.10 Summary and Conclusions

The Q2 benchmark results translate to an ability to simulate a bit more than one year of model time in one day of wall-clock time on the XT5 machine at ORNL. Including all the modifications to code, name lists, and compiler flags described above, as well as upgrades to the compiler and I/O subsystem, the amount of simulation time increased by more than a factor of 2 for the Q4 runs (or the ability to simulate more than 2 model years per wall-clock day). The many weeks of computer time required to complete multi-decadal high-resolution simulations with the spectral Eulerian dynamical core in CAM has now been reduced to just a few weeks. This increased turnaround efficiency can have a dramatic effect on scientific productivity.

3.3 XGC1

3.3.1 Introduction

Prediction of the plasma transport property in ITER is one of the most urgent research topics in the thermonuclear fusion program. Hot and dense plasma fuel in the central core must be adequately confined to produce much higher fusion energy output than the energy input required to operate the device, and at the same time the plasma in the edge must be cold enough to prevent costly damage to the material wall.

By utilizing the fundamental property that the charged particles mostly flow along the magnetic field lines, magnetic field lines in the vicinity of the material wall are designed to intersect special target plates (diverter plates) in order to prevent the main material wall from damage. At the boundary between the closed and open magnetic field line regions, magnetic field separatrix surface exists (see Fig. 9 in Sect. 3.3.4). Since the plasma on the open field lines is unconfined, it is naturally cold. The plasma temperature difference between the hot burning central core radii and the cold open field radii is not a free parameter for external control but self-determined by radial profile of thermal transport profile, which is mostly controlled by turbulence phenomena abundantly driven and supported by the temperature difference. If more heat is injected to the central plasma in an effort to increase the central temperature, greater temperature difference with the edge plasma drives stronger turbulent transport, hence opposing the rise of central temperature against the edge temperature. Rise of the central temperature will need to be accompanied by the rise of the edge temperature, which is, however, not allowed by the heat tolerance limit of the material wall. One of the formidable early efforts of the fusion plasma physicists was in the reduction of turbulence transport by an external mean, thus to raise the difference between the edge and core temperatures. Without such a mean, an economical production of fusion energy was expected to be difficult.

It was then discovered by experimentalists over a quarter of a century ago that adequately heated tokamak plasmas can form a thermal barrier in the plasma edge just inside the magnetic separatrix surface [13], which separates the hot plasma inside the magnetic separatrix surface from the cold plasma outside it within a thin radial shell (a few centimeters in a DIII-D tokamak). This transport barrier makes the plasma form a steep pressure pedestal just inside the separatrix surface. Turbulence level within this high confinement layer (H-mode layer) is reduced to an almost undetectable level. When this happens, the transport level of the core plasma is improved simultaneously. Thus, the plasma temperature of the central core is now allowed to rise regardless of the temperature in the open field line region. As a matter of fact, H-mode plasma allows the central temperature to increase by as much as the incremental ampunt of the edge pedestal temperature. This H-mode phenomenon occurs spontaneously by self-organization of plasma in the whole torus, in response to sufficient heat input in the core. Possibility of successful fusion reactor and ITER were escalated by this self-organization capability of toroidal plasma.

The central core temperature of ITER needs to be predicted for the efficient design of the device and the systematic planning of the experiments. However, after a quarter century of endeavor, we still do not have a community understanding of the spontaneous H-mode transition phenomenon in the edge and its relation to the plasma transport in the core. To get to the bottom of the physics understanding with predictive capability, a first principles kinetic simulation of the edge plasma has been requested to the Center for Plasma Edge Simulation (CPES), a SciDac Fusion Simulation Prototype Center. The edge simulation then needs to be coupled or extended to the core to understand the core temperature behavior in relation to the edge plasma behavior. The key new capability here is the nonlocal self-organization of the whole toroidal plasma. Once we obtain such a capability in the near future, the first principles simulation tool can be used to optimize the fusion yield, engineering requirement, and economy of future fusion reactors. Such a simulation capability can also help guide the development of the reduced-model transport codes in the proper direction, which can be used for experimental timescale simulation.

3.3.2 Background and Motivation

A preferable simulation method for such a first principles code is the 5D full-function gyrokinetic scheme without using the delta-f perturbation approximation. This method has been difficult to develop due to the embedded multiscale interaction between the small-scale turbulence dynamics and the large-scale background relaxation and evolution, and the necessity of high performance computing. The large-scale equilibrium drives small-scale turbulence. In return, the small-scale turbulence evolves the large-scale equilibrium, closing the loop. The experimentally observed plasma is the end result of such self-organization. Reduced model codes, such as gyrofluid or fluid, are computationally less demanding but lack the fundamental closure information required to describe such a multiscale self-organization. Delta-f simulation is efficient when the background is assumed fixed without participating into the multiscale self-organization dynamics and is virtually impossible in the open magnetic field line region in the edge.

With the availability of petascale computing, it is now possible to attack the most robust and baseline ion-temperature-gradient turbulence together with the background neoclassical dynamics in the whole-volume tokamak plasma in full-function gyrokinetic formalism. As the HPC capability grows, the simulation can be extended to include many other relevant turbulence and heating physics calculation for first principles prediction of ITER plasma performance.

3.3.3 Capability Overview

XGC1 is a new 5D gyrokinetic particle-in-cell (PIC) code designed to model the whole plasma dynamics in experimentally realistic device geometry [14]. The main new features in XGC1 are the full-function (full-f) description of the marker particles, as opposed to the previous perturbative delta-f description; the inclusion of the magnetic separatrix, magnetic X-point and the conducting material wall; and the particle/momentum/energy conserving Coulomb collisions. XGC1 allows the background profile to evolve to a self-organized state. To model more realistic plasma, XGC1 uses a heat source in the core plasma. The heat then flows to the material wall by a plasma transport process in the code. XGC1 is presently used to study the electrostatic turbulence, transport, and background plasma profile with full-f ions and adiabatic electrons. XGC1 will soon be upgraded to simulate electromagnetic turbulence.

XGC1, together with a simplified model version XGC0 [15], is the principal code for the existing SciDAC CPES project. The main purpose of the XGC1 code development has been to understand and predict the plasma transport and profile in the edge pedestal around the magnetic separatrix. Edge pedestal formation is an essential required feature for the success of ITER.* The code also computes scrape-off and wall loss physics. Due to the unknown nonlocal nature of the plasma turbulence and transport, XGC1 runs preferably in the full-f mode on the whole-volume toroidal plasma, ranging from the magnetic axis to the material wall. At the present time, there is no other code in the world fusion community with this advanced capability.

Marker particles are initiated in the entire toroidal volume in accordance with the initial density and temperature profiles. A random number generator has been used in the Maxwellian envelope. The plasma density profile is adjusted by the marker particle weights, while the marker particle density is spatially uniform. This technique improves the particle noise problem at the low plasma density region. For PIC executions, fixed grid cells are predesigned for a fixed experimental magnetic equilibrium. Due to the complexity of geometry in a diverted magnetic field, XGC1 uses an unstructured rectangular grid. To take advantage of the highly elongated neoclassical and turbulent electric potential structures along the magnetic field lines, the grid nodes follow the equilibrium magnetic field lines approximately.

* www.iter.org

Marker particles are time advanced according to the following Lagrangian equation of motion:

$$\begin{aligned}\dot{\mathbf{X}} &= (1/D)[v_\parallel \hat{b} + (v_\parallel^2/B)\nabla B \times \hat{b} + \{\mathbf{B} \times (\mu \nabla B - \mathbf{E})\}/B^2] \\ \dot{v_\parallel} &= -(1/D)(\mathbf{B} + v_\parallel \nabla B \times \hat{b}) \cdot (\mu \nabla B - \mathbf{E}) \\ D &= 1 + (v_\parallel/B)\, \hat{b} \cdot (\nabla \times \hat{b}),\end{aligned}$$

These equations of motion are solved using a 4th or 5th order Predictor-Corrector scheme in a weakly collisional case, and a mixed 2nd–4th order Runge-Kutta scheme in a strongly collisional case. In the Runge-Kutta scheme, the new turbulent field is solved in 2nd order and the particle position is solved in 4th order.

At each time advance step, charges are interpolated to the grid node points. The following gyrokinetic Poisson equation is solved on the grid nodes.

$$-\nabla_\perp \frac{\rho_i^2}{\lambda_{Di}^2} \nabla_\perp \Phi = \frac{e}{\varepsilon_0}\left(1 - \nabla_\perp \rho_i^2 \nabla_\perp\right)\left(\overline{n}_i - n_e\right)$$

$$\overline{n}_i = \frac{1}{2\pi}\int n(\mathrm{R} + \boldsymbol{\rho}, t)\, d\varphi$$

$$n_e = n_0 \exp\left(e\delta\Phi / T_e\right) \cong n_0\left(1 + e\delta\Phi / T_e\right)$$

The electric field information is interpolated back to the particle positions to execute another time advance of the particles in accordance with the above Lagrangian equation of motion. The equilibrium magnetic field data from the experimental g-eqdsk file is stored on a rectangular mesh. At each particle position, the magnetic field is evaluated by spatial spline while conserving Div B = 0 [16]. In some specified time intervals, a linear Monte Carlo collision operation routine is called to execute the Coulomb collisions in the velocity space. After each collision process, total momentum and energy of the particles are adjusted to ensure conservation within the colliding particles.

The gyrokinetic Poisson equation is solved by PETSc [17]. The conjugate gradient method is used with various preconditioners available in PETSc, such as (1) an algebraic mutligrid preconditioner (HYPRE) for the (elliptic) equilibrium solver and (2) a diagonal preconditioner for the (parabolic) turbulence solve.Even though the solver is a global operation (hence there are limits to the amount of parallelism available in the preconditioners), the amount of work by equation solver in XGC1 is fairly small relatively. In a typical XGC1 run, about 10% of the time is spent in the solver, which includes global reductions and scatters to assemble the charge vectors and to communicate the potential solution. One alternative to consider in getting ready for the future extreme-scale parallel machines is to form an explicit inverse of the matrix.

The ion density at the node point is determined by summation of particle weights located on the triangles that contain the node point with linear interpolation. A triangle search operation is required to determine the triangle that contains a given particle and the interpolation coefficient. At the initialization phase, XGC1 prepares a table of rectangular grid which stores an index of every triangle overlapped. The triangle search routine uses the particle coordinates to perform geometric hashing into the rectangular grid to locate the target triangle. An optional preprocessing phase reorders the triangle and vertex labels using a Hilbert space-filling curve to improve spatial locality and cache performance.

To properly manage the simulation data used for the Joule report, we are using the CPES EFFIS framework (End-to-End Framework for Fusion Integrated Simulation). This framework consists of the ADIOS componentized I/O system, the Kepler workflow for monitoring the simulation, and the eSimmon dashboard system. Each piece of the software stack that we have used is highly flexible and allows us a framework that makes running XGC1 simulations similar to the concept of running in an end-station. Researchers from the MIT System Design and Management Program, CPES, Georgia Tech, and ORNL

carried out this work. Adaptable I/O System, or ADIOS, is a componentization of the I/O layer. It provides an easy-to-use programming interface, which can be as simple as Fortran file I/O statements. ADIOS has been shown to get over 50% of the maximal I/O performance on the Crays when run properly. ADIOS will be tuned for XGC1 code to the optimal capacity for Q4 petascale computing. The scientific and performance results will be analyzed using the CPES dashboard for efficient collaboration among scientists, and with the applied mathematics and computer science collaborators.

3.3.4 Science Driver for Metric Problem

To meet the fusion energy yield goal of Q = 10, the plasma fuel ions in the central core of the ITER tokamak (Fig. 8) must maintain a sufficiently high temperature (≥15 keV). At the same time, plasma temperature near the wall must remain sufficiently cold (<<1 keV) to avoid premature plasma damage to the material wall. Since the global slope of the plasma temperature is upper bounded by the turbulent transport in a normal situation, the only way to increase the core temperature to a burning level seems to be by making the plasma size large enough. However, this leads to utilization of only a small fraction of plasma volume for fusion and, thus, to an uneconomical fusion device which cannot meet the ITER goal. One way to remedy this problem is to find or invent plasma facing material that can withstand plasma bombardment at temperature much above 1 keV, which has not been a viable option to date.

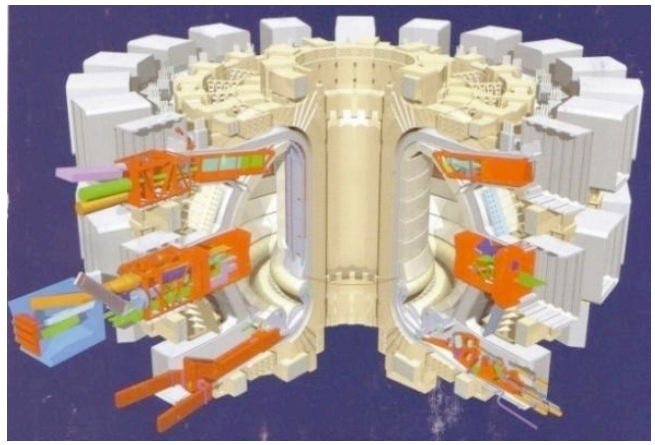


Fig. 8. Schematic of the ITER tokamak, where the first wall of the innermost structure of the device is shown, with the divertor chamber at the bottom. For more information, see www.iter.org.

Fortunately, experiments in the present-day tokamak devices consistently find that, with sufficient core heating above a threshold power, plasma bifurcates into a state in which the edge temperature (and density) abruptly rises from ~100 eV in the open magnetic field region in front of the material wall to ~ keV just inside the last closed magnetic surface (separatrix) region. Thus, in such a bifurcated state, the core plasma temperature can rise on an "edge pedestal" without the high temperature plasma contacting the material wall. In this pedestal layer, which occurs just inside the magnetic separatrix, experiments find that the turbulence level is dramatically lower than the ambient level and that a deep well structure forms in the radial electric field profile. The density pedestal width is observed to be even narrower than the ion temperature pedestal width. Mysteriously, the reduction in the turbulence amplitude and the increase in the ion temperature in the core plasma appear to respond to the edge bifurcation in a much faster timescale than the radial plasma energy transport timescale. The core ion temperature increases in proportion to the edge pedestal temperature with its radial slope being "stiff," independent of the core heating power and, thus, the edge pedestal temperature (Fig. 9). This type of operation mode is called "H-mode," meaning a high confinement mode. As observations from other areas of nonlinear science have shown (e.g., oceanography, climate, economy, sociology, planetary science), the experiments indicate that there is a strong nonlocal component in the tokamak plasma turbulence dynamics.

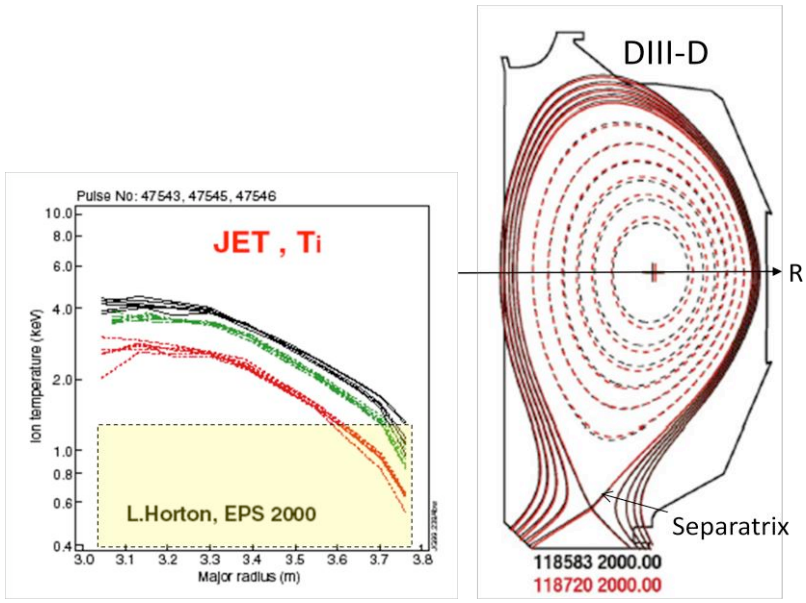


Fig. 9. *(left)* Nonlocal nature of the ion temperature (T_i) profile. As the edge T_i increases, the core T_i increases together in a stiff shape. *(right)* The cross section of DIII-D magnetic surface inside the first wall. Open and closed magnetic surfaces are shown, with the magnetic separatrix surface in between. For more information on the DIII-D device see http://web.gat.com/global/DIII-D.

ITER's performance goal to achieve the fusion yield ratio Q = 10 is based upon the assumption that a good H-mode operation is to be sustained. However, the reasons why the edge pedestal forms in such a shape, why a strong core heating is necessary, why there is an instantaneous central T_i and turbulence improvement after the H-mode bifurcation, why the radial T_i profile is stiff, etc. are yet unknown after over 25 years of H-mode research. Due to the nonlocal, nonlinear, and multiscale nature of the H-mode physics, a large-scale first-principles gyrokinetic simulation of the turbulence and background plasma dynamics in the whole device volume has been a necessary component of the H-mode research. However, such HPC power has not been available so far and is just beginning to be realized in the United States. With the aggressive planning of HPC development in the United States, Japan, and possibly elsewhere, the future of the large-scale H-mode simulation looks brighter.

The whole-volume gyrokinetic simulation must be performed using the full distribution function (full-f) method, before simplifying it to the popular perturbed distribution function method (delta-f). It needs to be done in a realistic tokamak geometry since the geometry effect appears to be important in the experiments, including the open and closed magnetic field regions with the magnetic separatrix surface in between. It must deal with the heat source in the core and the particle loss to the material wall.

We hope that the Joule metric exercise performed here not only improves our numerical code capability to scale but also sheds light on the H-mode physics at the first principles level so that it can help predict the performance of ITER and DEMO reactors.

The science metric of this simulation exercise is to use the full-f gyrokinetic code XGC1 on the almost full capacity Jaguar/XT5 to study the most robust and large physical scale turbulence, which is driven by the free energy in the ion temperature gradient (ITG), self-consistently with the neoclassical equilibrium dynamics in a realistic DIII-D tokamak geometry (Fig. 9*(right)*). This simulation represents the first attempt in fusion research to study the nonlocal H-mode coupling physics between the edge and core turbulences in a realistic tokamak geometry. Smaller physical scale turbulences will be added later as HPC capability grows in the near future.

The goal is to obtain the physics results in 24 hours or less of Jaguarpf wall-clock time. The Q2 version was not optimized to scale well on much more than 30,000 Jaguarpf XT5 cores. We thus used

29,952 XT5 cores (which is 1/5 of the Jaguar capacity). The Q2 XGC1 could take 4,000 time steps in 24 hours. The science we observed in Q2 is the development of the global full-f ITG turbulence to the nonlinear stage in the whole volume of realistic DIII-D geometry. From the Q2 simulation, we only observe turbulence intensity propagation from edge to core, which is a sure sign of nonlocal interaction between edge and core. Initial turbulence intensity is strong and bursty. Interaction of turbulence intensity bursts with the local E×B shearing rate and temperature gradient is clearly demonstrated. The number of cores is insufficient to reach the quasi-steady self-organized state, which is more relevant to the experimental observations. In Q4, we used the improved XGC1 to scale up to the maximal number of XT5 cores and ran the same simulation on 4 × 29,952 = 119,808 cores, which is about 4/5 of the maximal available Jaguarpf capacity. In 16,000 time steps (which took about 20 wall-clock hours), the peta-scale Q4 simulation reached a quasi-steady self-organized state, after a long bursty nonlinear turbulent transport stage. The Q4 simulation results shed light on the key unexplained experimental H-mode phenomena, including the reasons why strong core heating is necessary, why there is an instantaneous central T_i and turbulence improvement with the H-mode bifurcation, and why the radial T_i profile is stiff. The Joule metric provided a significant scientific advance in XGC1.

3.3.5 Q2 Baseline Problem Results

The metric baseline here is the particle processing counts per second. We choose an actual experimental device for the Q2 benchmark exercise so as to contribute to the progress of a real scientific program. The experimental device size and the physics grid size (= ion gyro radius ρ_i) determine the total marker particle number used in the simulation. Marker particle number per grid node is set by particle noise level in the physical observables. For the strong scaling metric between Q2 and Q4, we chose an ITG turbulence transport study within a day of wall-clock time in the whole-volume DIII-D tokamak at General Atomics, in realistic physical size and diverted geometry including material wall. The total number of marker particles thus determined for this problem size is 13.5 billion.

Figure 10 shows our model for the initial plasma density and ion temperature profiles, with the electron temperature assumed to be equal to the ion temperature. Notice here that the ion temperature pedestal knee is located at a somewhat smaller minor radius than the density pedestal knee, making the relative temperature slope high between the two knees. We assume that this is a common feature in H-mode operation. To date, all the machines, which have adequate ion temperature profile diagnostics in H-mode, reported this feature. Figure 11 shows the relative ion temperature gradient $\eta_i \equiv R_0/L_T = R_0/\partial \log T_i/\partial r/$, where R_0 is the major radius of the torus. Shown together in the plot is the nonlinear stability criterion of ITG mode evaluated in the core plasma [18]. It can be seen that the plasma is supposed to be stable to ITG turbulence at $\psi_N \leq 0.5$ if the turbulence is a local phenomenon. As can be seen from the relation between the real minor radius in meters and ψ_N in Fig. 12*(left)*, $\psi_N \leq 0.5$ constitutes a significant portion of the core plasma since the half minor radius corresponds to about $0.37\psi_N$. The temperature pedestal top is at about $0.8\psi_N$, and the density pedestal top is at about $0.85\psi_N$. The right-hand side of Fig. 12 shows the radial profile of the magnetic safety factor q, which represents the toroidal windings of the equilibrium magnetic field relative to a poloidal winding. It is shown here because q is an important indicator of plasma stability. These model initial profiles are common to Q2 and Q4 benchmark runs.

A total of 4.5 MW of heat is added to the ions around the magnetic axis ($\psi_N \leq 0.04 \approx$0–10 cm) to force a heat flux into the turbulence region. The heating is achieved by raising the particle energy uniformly in the heating region by a small fraction of kinetic energy while keeping the pitch angle invariant.

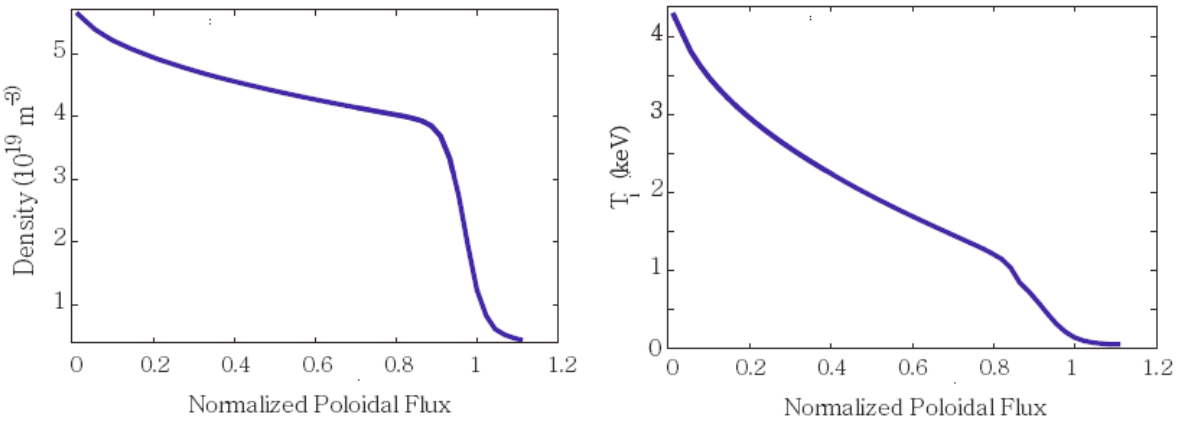


Fig. 10. Early-time plasma density and temperature profiles. Electron temperature is assumed to be equal to ion temperature. Notice that the temperature pedestal knee is located at a somewhat smaller minor radius than the density pedestal knee, making the relative temperature slope high between the two knees.

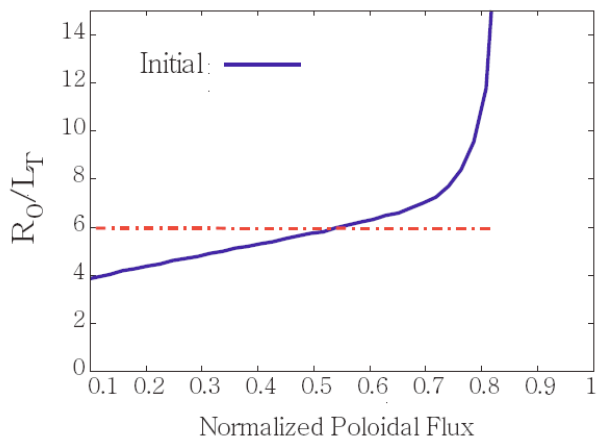


Fig. 11. Initial profile of $R_0/L_T = R_0/\partial \log T_i/\partial r/$. The horizontal dashed line is the nonlinear stability criterion of a core plasma.

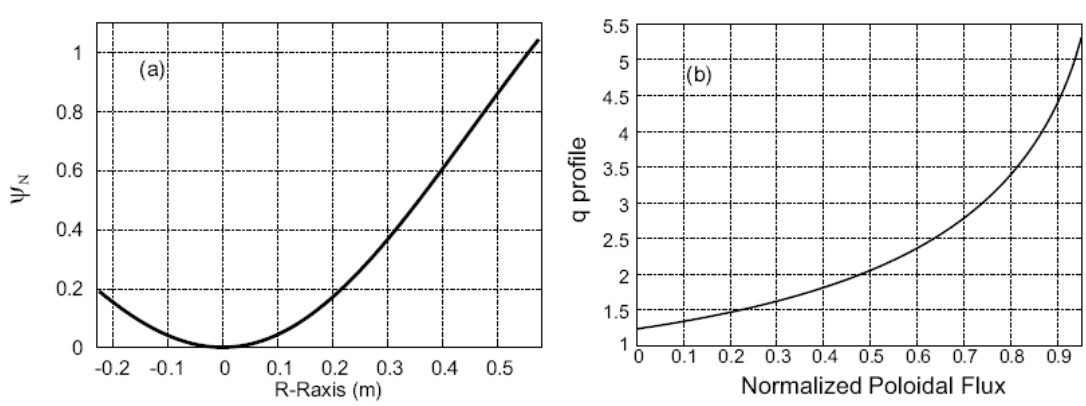


Fig. 12. *(a)* Relationship between the normalized poloidal flux ψN and real distance in meters from the magnetic axis (Raxis = Ro) to the flux surface (R) along the midplane. *(b)* Radial profile of the safety factor q.

The Q2 baseline benchmark was run on 29,952 XT5 cores on Jaguarpf, which is about 1/5 of the maximal available Jaguarpf configuration at the present time. Much above this number of cores, the XGC1 version at the time of Q2 execution does not scale well. Eight MPI processes were used per node. The Q2 run was performed in two parts. An initial run was made for 2,000 time steps without saving the performance data to separate out the initialization cost. The simulation was then continued using a checkpoint file for another 2,000 time steps, and the performance data was recorded. The second run gets into the nonlinear turbulence stage in the DIII-D benchmark plasma. The total wall-clock time spent was about 24 hours. Table 15 shows operation counts during the second run from the hardware performance counters, obtained using the PAPI performance data collection interface.

Table 15. XGC1 performance data collected on the Q2 benchmark with PAPI hardware counters

Number of processing elements	29,952
Cycles per second per processor	$2{,}255.35 \times 10^6$
Instructions executed per second per processor	$2{,}293.69 \times 10^6$
Instructions per cycle	1.02
Floating point operations executed per second per processor	222.65×10^6
Particles pushed per second	0.628×10^9

Figure 13 shows the nonlinear turbulent eddies of the electrostatic potential over the whole poloidal cross section. The image at left is at an earlier time, showing turbulence generation in the edge. The image at right is at the end of the Q2 run, showing that the edge turbulence has propagated to core. At the central core, the heat-source enhanced ITG turbulence can be seen, while the rest of the plasma is occupied with turbulent activities propagated from the edge. Stronger turbulence eddies are observed at the weaker magnetic field side due to the toroidal ballooning effect. Figure 14 shows the inward propagation of turbulence intensity in the initial nonlinear period Q2 simulation. We note again here that the nonlinear turbulence and plasma in Q2 are not in a quasi-steady state yet. Our Q4 goal was to obtain the quasi-steady self-organized nonlinear stage in 4 times more time steps as in Q2 by increasing the number of processor cores by factor of 4.

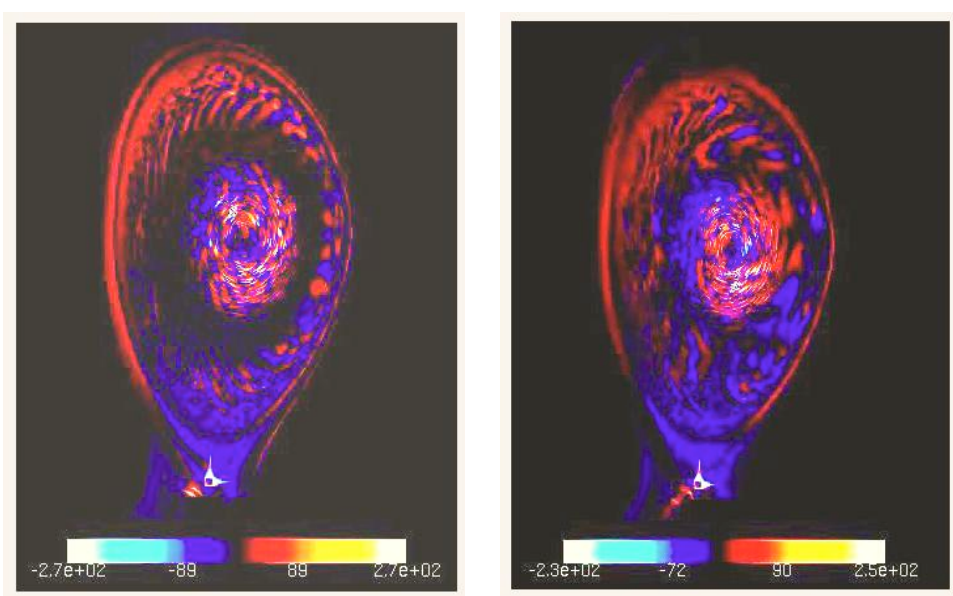


Fig. 13. Turbulent eddies on the whole poloidal cross-sectional plane at *(left)* an earlier time and *(right) a* later time. The nonliear turbulence and the plasma are not in a quasi-steady state, yet.

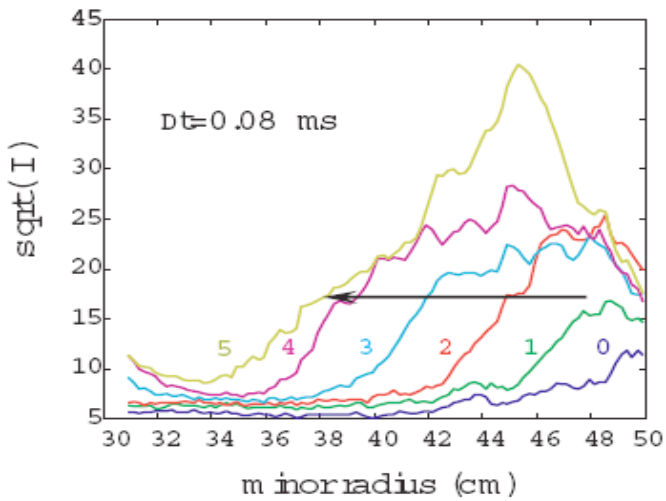


Fig. 14. Inward propagation of the square root of turbulence intensity Sqrt(I) during the bursty nonlinear period, where $I = <(\delta\varphi)2>$. Each plot is drawn in 0.08 millisecond interval, and the total propagation time is only 0.4 millisecond.

The simulation result is rejected when the particle noise dominates the simulation, which can be detected by comparing the effective turbulent ion thermal conductivity with a noise-driven thermal conductivity level (0.05 m^2/s). We have also investigated the convergence of the solution in particle numbers. The simulation is also rejected when a numerical oscillation dominates the turbulent fluctuation. We have observed that this could happen if the ion temperature in the open magnetic field region (scrape-off plasma) is much higher than that observed experimentally.

3.3.6 Computational Performance Gains

There is sufficient work in processing the particles in this experiment to use hundreds of thousands of processors, even when holding the problem size fixed while increasing the number of processors (strong scaling). The primary inhibitor to scaling is the MPI communication overhead arising in the solution of the Poisson problem and in the reassignment of particles to processes as the result of the time advance.

Studies in the fall of 2008 indicated that OpenMP parallelization might improve the scalability by allowing us to decrease the number of MPI processes used for a fixed number of processors, thereby decreasing some of the MPI communication overhead. In particular, on the Cray XT5, up to four OpenMP threads could be used efficiently to parallelize the processing of the particles. Using more than four is not useful currently due to the nonuniform memory access characteristics in the XT5 compute node. Beginning in 2009, OpenMP parallelism was implemented in XGC1, allowing the use of 1/4 as many MPI processes as would otherwise be required in the Q4 simulations. This not only contributed to achieving the Joule performance metric but is also a critical capability for scaling to even larger processor counts.

The other major performance enhancement that occurred between the Q2 and Q4 experiments was the optimization of the interpolation scheme used in the evaluation of the magnetic field (and elsewhere). By precomputing many of the spline coefficients and by taking advantage of common partial results in the computation of derivatives, the number of required floating operations is decreased, resulting in significant reductions in run time. As described in the next section, this has the seemingly anomalous effect of decreasing the achieved computation rate, but it decreased the amount of computation even more, improving throughput by approximately 30% in addition to the improvement achieved through OpenMP parallelism.

There are also other improvements made in XGC1, which include higher parallelization of particle operations, increased cache efficiency, and I/O speed improvement in ADIOS.

3.3.7 Q4 Metric Problem Results

In Q4 the number of XT5 processor cores was increased by 4 times to 119,808, which is about 4/5 of the maximal allowed number of cores in Jaguarpf. The total number of time steps was also increased to 4 times longer (16,000 steps), in proportion to the number of processors. As in Q2, the Q4 runs were performed in two consecutive runs using restart file, and the performance data were obtained during the second run to avoid the initialization counts.

Two performance enhancements described in the previous section (OpenMP parallelism and interpolation scheme optimization) enabled the performance to improve by a factor of 4.6 between the Q2 and Q4 experiments, reducing the execution time per model time step from 21.6 seconds to 4.7 seconds. The Q4 experiment used 4 times as many processors, so this reflects superlinear speedup compared to the Q2 experiment. To reiterate, the OpenMP parallelism enhanced the ability to use efficiently 4 times as many processors, while the interpolation scheme optimization decreased the amount of work required in the Q4 experiment. It is this latter performance enhancement that led to the more than ideal linear speedup for this fixed-size problem. Due to the superlinear speedup, the total wall-clock time has decreased from 24 hours to about 21 hours.

Table 16 shows Q4 operation counts during the second run from the hardware performance counters, obtained using the PAPI performance data collection interface. Q2 operation counts are shown together for a direct comparison.

Table 16. XGC1 performance data collected on the Q4 benchmark with PAPI hardware counters

	Q2	Q4
Number of processing elements	29,952	119,808
Cycles per second per processor	$2{,}255.35 \times 10^6$	$1{,}622.65 \times 10^6$
Instructions executed per second per processor	$2{,}293.69 \times 10^6$	$1{,}399.46 \times 10^6$
Instructions per cycle	1.02	0.86
Floating point operations executed per second per processor	222.65×10^6	151.23×10^6
Particles pushed per second	0.628×10^9	2.87×10^9

Increasing the number of processor cores by 4 times to 119,808 enabled the improved XGC1 to execute the simulation to 4 times longer physical time steps within a day from the 29,952 core Q2 simulation. As a result, while the Q2 simulation went only into the initial bursty nonlinear turbulence phase, the Q4 simulation was carried to the self-organized quasi-steady phase where the real experimental plasmas stay. Valuable information on the overall picture of the nonlocal turbulence propagation and the settling down of the turbulence and the plasma profile to the quasi-steady SOC state was thus obtained.

Figure 15 is an enlarged image of the turbulence intensity $<|\delta\Phi|^2>$ contour in the radius-time space in the pedestal area. It can be clearly seen that the turbulence starts around the temperature pedestal knee ($\psi_N \approx 0.83$) and propagates inward (outward propagation is much weaker). A localized simulation in the small radial domain can distort the propagating turbulence dynamics due to the artificial inner boundary condition. A global simulation is needed to study the nonlocal turbulence dynamics. Figure 16 is the result of the localized simulation with a simulation boundary at r = 0.5 m and indeed shows a highly different result from the global simulation result of Fig. 15. A minor radius of 0.52 m in Fig. 16 corresponds to about $\psi_N = 0.89$ in Fig. 15.

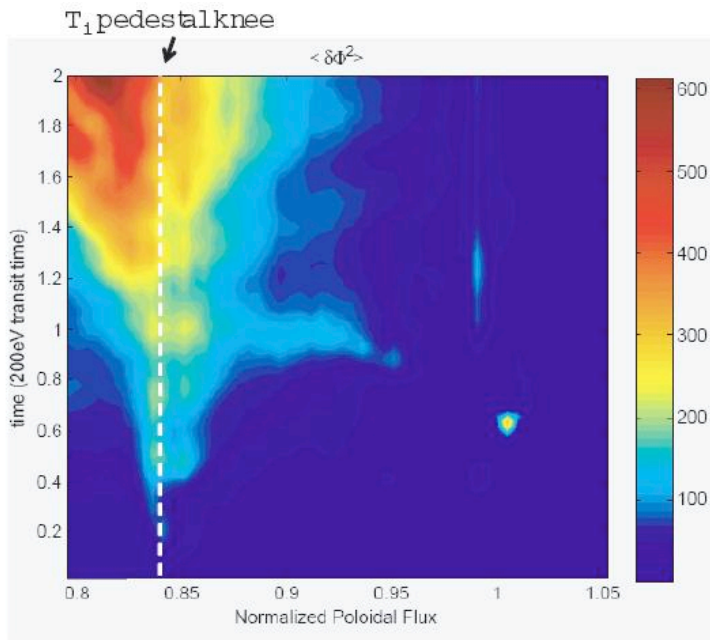


Fig. 15. An enlarged image of the turbulence intensity $<|\delta\varphi|^2>$ contour in the radius-time space in the pedestal area.

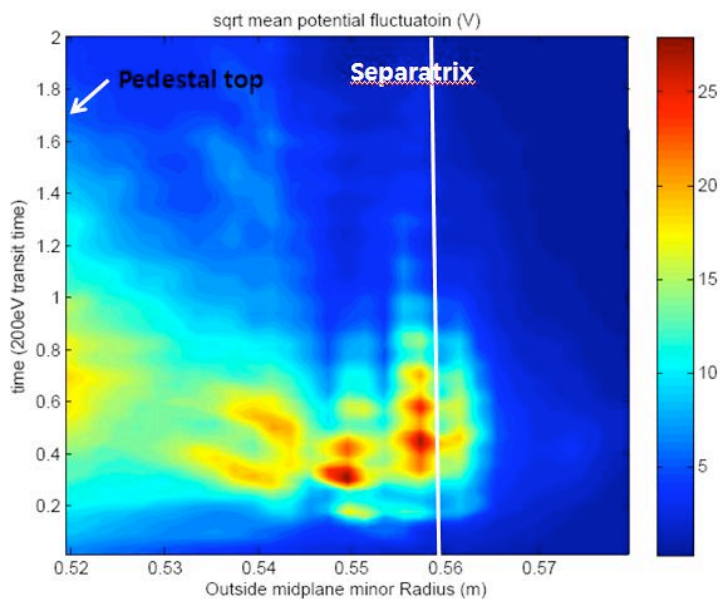


Fig. 16. The same simulation as in Fig. 15 in the localized radial domain, with the usual particle simulation boundary at r = 0.5.

Figure 17 is the heat flux contour in the global space-time space, which indeed shows that the out-to-in propagation of the turbulence front is all the way to the plasma core. As the turbulence front arrives (solid arrow), heat bursts appear radially outward (dotted arrows). The inward propagation stops when the edge-originated turbulence meets the strongly sheared central turbulence at $t \approx 150\ R/v_i$ ($\approx$0.6 millisecond). In the pedestal region, both the turbulence intensity and the heat flux remain small, which is characteristic of H-mode plasma. Turbulence in the edge pedestal may be better described by electromagnetic effect. However, the weak turbulence intensity and heat flux of the electrostatic ITG turbulence in the pedestal area suffice the present purpose of investigating the nonlocal edge-core relation.

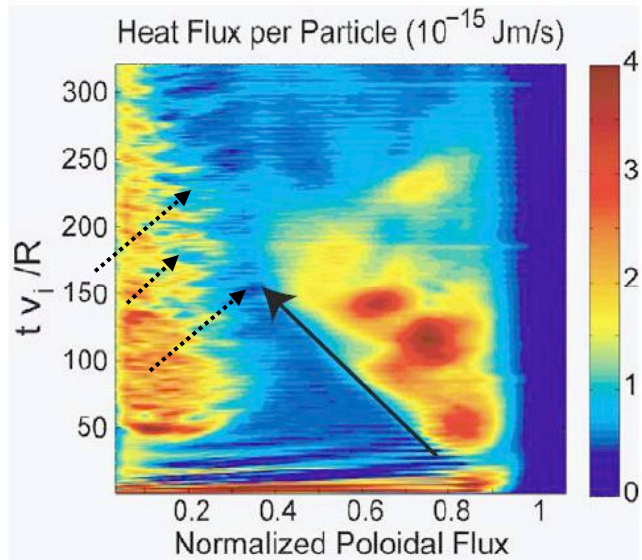


Fig. 17. Heat flux contour in the global space-time space, exhibiting the out-to-in propagation of the turbulence front.

Another remarkable observation made from the simulation is the self-organizing modification of the background temperature profile by the incoming turbulence, as can be seen in Fig. 18. Before the arrival of the edge originated turbulence, the ion temperature gradient was below the nonlinear ITG criticality (dotted horizontal line) [18]. However, arrival of the edge-generated turbulence raises the local temperature gradient above the nonlinear criticality. The ion thermal conductivity is then self-regulated to a new criticality by the self-generated E×B shearing. In other words, the turbulence criticality is nonlocally self-organized by the edge turbulence source. This state is maintained by the out-flowing heat flux. Combination of the low heat thermal conductivity near the magnetic separatrix surface and the large heat flux from the core keeps the η_i value at the T_i knee well above the nonlinear criticality, continuously supplying the ITG turbulence energy to maintain the new self-organized criticality. Without the strong heat flux from the core, the η_i value at the T_i knee would collapse and the driver for the new SOC state would be lost. In other words, the heat flux from the core is a "fuel" to the edge turbulence energy source. The present simulation reveals that this is how the global ITG turbulence maintains an H-mode profile, if ITG is the strongest global turbulence transport mechanism. We have examined a few different heating power levels and have found that the η_i profile shown here is "stiff" with respect to the change of heating power, which is consistent with the experimental findings.

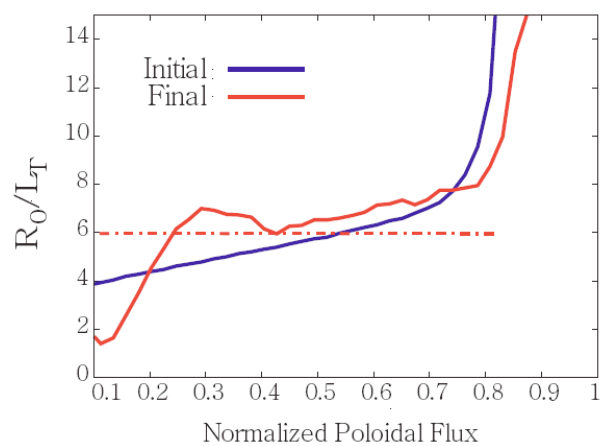


Fig. 18. Self-organizing modification of the background temperature profile by the incoming turbulence. The horizontal line represents the nonlinear criticality in the core plasma.

Figure 19 is the time behavior of effective ion thermal conductivity (thermal flux divided by local T_i gradient) from the start of the simulation across $\psi_N = 0.64$, which corresponds to r = 42 cm on the outside midplane. The short initial jittering of high frequency is the large amplitude GAM oscillations during the self-organization of the toroidal plasma in the initial local Maxwellian loading and is subdued at about 30 ~ v_i/R. At about 60 v_i/R, ITG modes start to grow. It is well known that unless the jittering from the initial GAM activities is subdued, ITG turbulence does not grow in a full-f simulation [19]. The total simulation time is about twice the ion 90° collision time. The Q2 run corresponds to 1/4 of the time length.

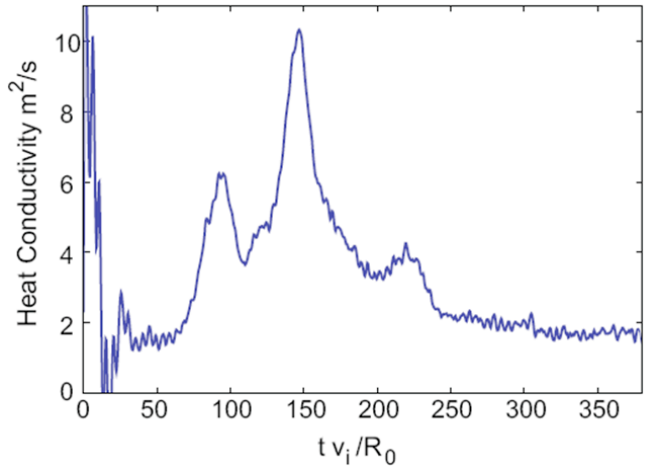


Fig. 19. Time behavior of effective ion thermal conductivity (thermal flux divided by local *Ti* gradient) from the start of the simulation across $\psi N = 0.64$, which corresponds to r = 42 cm on the outside midplane. The self-organizing process is bursty.

In Fig. 19, after the arrival of the turbulence front at the radial location, there is a distinctive bursty type of heat flux behavior in the initial stage of nonlinear turbulent transport until about t = 240 R/v_i. This behavior corresponds to the heat bursts in Fig. 17. The inter-burst period is much greater than the initial GAM period. The radial speed of the ballistic motion of heat burst is about $V_r \approx (1/5)\, \rho_i\, v_i/R \approx (1/30)\, \rho_i\, v_i/L_T$, which is similar to the analytic intensity burst estimates reported in refs. [20] through [22].

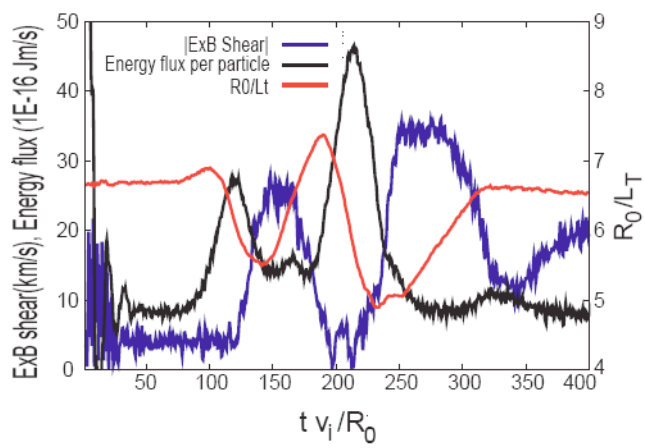


Fig. 20. Phase relation between the temperature gradient, heat flux, and E×B shearing dynamics at a radial location.

Figure 20 shows interplay between the temperature gradient, heat flux, and the E×B shearing dynamics during the bursty heat flux at a radial location r = 42 cm. Arrival of the turbulence front is first noticed by the steepening of local temperature gradient and increase in heat flux (∝ turbulence intensity), followed by time delayed increase in the local E×B shearing rate. Increase of the local E×B shearing rate then suppresses turbulence until the turbulence-driven sheared flow is reduced. The burst cycle continues until a steady turbulence is reached at the end, where the turbulence shows a 1/f avalanche type of power law. This is a textbook demonstration on the interplay between E×B shearing, turbulence intensity, and local temperature gradient, which is possible only in a full-f simulation.

Energy conservation has been investigated within the volume $0.3 \leq \psi_N \leq 0.7$. Between the total energy flowing into the volume across the inner surface $\psi_N = 0.3$, the sum of the particle energy change, the field energy created, and the out-flowing energy across the outer surface $\psi_N = 0.7$ shows about 2% error in the total energy conservation (Fig. 21). The energy conservation error in a full-f code does not grow unless numerical errors grow.

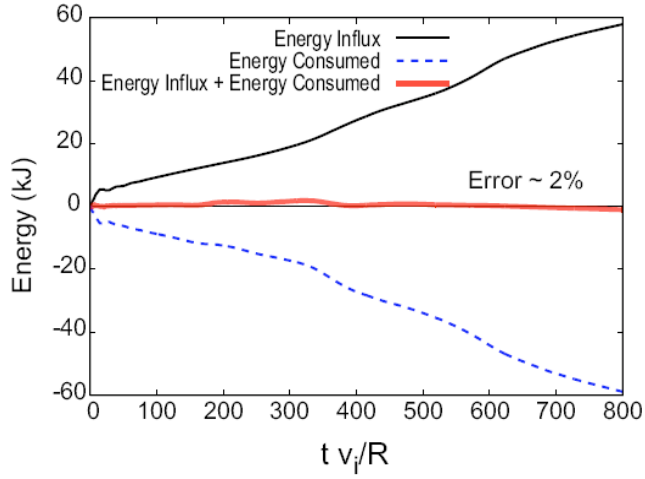


Fig. 21. Energy accounting within $0.3 \leq \psi N \leq 0.7$ between the total influx across the inner boundary (black curve) and the sum of the consumed energy (blue curve) to the particles, the electric field, and across the outer boundary. Red line shows sum of black and blue curves. About 2% error is noticed.

3.3.8 Interpretation of Results

A fusion experiment measures its performance in quasi-steady-state operation. Study of the transient behavior is important for physics understanding. However, a simulation will have to reach a quasi-steady state for an eventual understanding and prediction of the experimental performance. The goal of this metric is designed to obtain the initial transient nonlinear turbulence behavior in the realistic whole-volume DIII-D geometry in Q2, and to achieve the quasi-steady self-organized turbulent state in Q4 within a day of wall-clock time. The original XGC1 did not scale too well much above 30,000 processor cores. We thus chose 29,952 Jaguar/XT5 cores for the Q2 metric base. The improved XGC1 (OpenMP parallelism and interpolation scheme optimization) scales super-linearly to the Q4 metric base of 199,808 cores, which is 4 times the Q2 number of cores, and enabled the performance of XGC1 to improve by a factor of 4.5 between the Q2 and Q4 experiments, reducing the execution time per model time step from 21 seconds to 4.7 seconds.

Improvement of the physics capability as a result of the above Joule metric exercise is significant. In Q2 the propagation of the nonlinear edge turbulence into the core was observed within a day of wall-clock time, which provides an exciting evidence for nonlocal edge effect on the core turbulence and confinement in realistic DIII-D geometry. However, in order to produce an experimentally relevant result, the nonlinear simulation has to be carried through the quasi-steady self-organized stage, while keeping the multiscale dynamics self-consistently. The improved XGC1 performance in Q4 is good enough to reach to the quasi-steady self-organized stage within a day of wall-clock time. As a result, many new physics results have been obtained to shed light on the over 25 year old H-mode plasma physics mysteries, which ITER is heavily relying upon for its success.

As the computing power increases, we will be able to include more physics into XGC1 code, en route to the whole-physics modeling in first principles.

OpenMP parallelism was implemented in XGC1, allowing the use of 1/4 as many MPI processes as would otherwise be required in the Q4 simulations. This not only contributed to achieving the Joule performance metric but is also a critical capability for scaling to even larger processor counts.

The other major performance enhancement that occurred between the Q2 and Q4 experiments was the optimization of the interpolation scheme used in the evaluation of the magnetic field (and elsewhere). By precomputing many of the spline coefficients and by taking advantage of common partial results in the computation of derivatives, the number of required floating operations is decreased, resulting in significant reductions in run time. This has the seemingly anomalous effect of decreasing the achieved computation rate, but it decreased the amount of computation even more, improving throughput by approximately 30% in addition to the improvement achieved through OpenMP parallelism.

Other improvements have also contributed, to lesser degrees, to Q4 enhancements of the performance, which include higher parallelization of particle operations, increased cache efficiency, and I/O speed improvement in ADIOS.

3.3.9 Summary and Conclusions

A fusion experiment measures its performance in quasi-steady state operation. Study of the transient behavior is important for physics understanding. However, a simulation will have to reach a quasi-steady state for an eventual understanding and prediction of the experimental performance. The goal of this metric is designed to obtain the initial transient nonlinear turbulence behavior in the realistic whole-volume DIII-D geometry in Q2, and to achieve the quasi-steady self-organized turbulent state in Q4 within a day of wall-clock time. The original XGC1 did not scale too well much above 30,000 processor cores. We thus chose 29,952 Jaguar/XT5 cores for the Q2 metric base. The improved XGC1 (OpenMP parallelism and interpolation scheme optimization) scales super-linearly to the Q4 metric base of 199,808 cores, which is 4 times the Q2 number of cores, and enabled the performance of XGC1 to improve by a factor of 4.5 between the Q2 and Q4 experiments, reducing the execution time per model time step from 21 seconds to 4.7 seconds.

Improvement of the physics capability as a result of the above Joule metric exercise is significant. In Q2 the propagation of the nonlinear edge turbulence into the core was observed within a day of wall-clock time, which provides an exciting evidence for nonlocal edge effect on the core turbulence and confinement in realistic DIII-D geometry. However, in order to produce an experimentally relevant result, the nonlinear simulation has to be carried through the quasi-steady self-organized stage, while keeping the multiscale dynamics self-consistently. The improved XGC1 performance in Q4 was good enough to reach to the quasi-steady self-organized stage within a day of wall-clock time. As a result, many new physics results have been obtained to shed light on the over 25 year old H-mode plasma physics mysteries, which ITER is heavily relying upon for its success.

As the computing power increases, we will be able to include more physics into XGC1 code, en route to the whole-physics modeling in first principles.

3.4 RAPTOR

3.4.1 Introduction

Turbulent combustion processes are prevalent in a wide variety of propulsion and power systems, including internal combustion engines, gas turbines, and liquid rockets. As such, development and rigorous validation of science-based predictive models for turbulent combustion have long been recognized as important priorities in research, and there are a variety of challenges. Turbulent flows involving heterogeneous chemically reacting and/or multiphase mixtures (as is the case for all propulsion and power systems) have a variety of complicating factors, including highly nonlinear chemical kinetics, small-scale velocity and scalar mixing, turbulence–chemistry interactions, compressibility effects (volumetric changes induced by changes in pressure), and variable inertia effects (volumetric changes induced by variable composition or heat addition). Coupling between processes occurs over a wide range of time and length scales, many being smaller than can be resolved in a numerically feasible manner. Further complications arise when multiple phases are present due to the introduction of dynamically evolving interface boundaries and the complex exchange processes that occur as a consequence. At the device level, high performance, dynamic stability, low pollutant emissions, and low soot formation must be achieved simultaneously in highly confined geometries that generate extremely complex flow and acoustic patterns. Flow and combustion processes are highly turbulent (i.e., integral-scale Reynolds numbers of $O(10^5)$ or greater), and geometry or various operating transients inherently dominate the turbulence dynamics. In many cases operating pressures approach or exceed the thermodynamic critical pressure of the fuel or oxidizer. Operation at elevated pressures significantly increases the system Reynolds number(s) and inherently broaden the range of spatial and temporal turbulence scales over which interactions occur.

No one experimental or numerical technique is capable of providing a complete description of the processes described above. The highest quality experimental diagnostics provide only partial information from highly idealized flows relative to a given application. Modeling and simulation of these processes has historically been limited by computational power. Even with peta-scale computing (and beyond), Direct Numerical Simulation (DNS) of the fully coupled equations of fluid motion, transport, and chemical reaction can only be applied over a limited range of turbulence scales, in the high wave number, low Reynolds number, diffusive regime of turbulence. Thus, simulating these phenomena almost always begins with some form of formal filtering of the governing conservation equations. The Reynolds-Averaged Navier-Stokes (RANS) approximation, for example, employs filtering in time to derive the governing conservation equations for the mean state. For this approach all dynamic degrees of freedom smaller than the largest energy-containing eddies in a flow are averaged, and no information exists to describe interactions between the small scales. The Large Eddy Simulation (LES) technique, on the other hand, has historically employed spatial filtering to split the field variables into time-dependent resolved-scale and subgrid-scale (SGS) components. For this approach the large energetic scales are resolved and SGS quantities are modeled to provide a complete, time-accurate representation of dynamic processes over the full range of multidimensional scales in a turbulent reacting flow. RAPTOR is a massively parallel flow solver that has been optimized for application of LES to turbulent, chemically reacting and/or multiphase flows in complex geometries, with emphasis placed on propulsion and power systems.

3.4.2 Background and Motivation

The limitations and challenges associated with turbulent combustion research require that a hierarchy of approaches be taken to fully understand key processes and work toward predictive models. The primary challenge is to bridge the gap between basic research and the conditions of interest in typical applications. As part of the Reacting Flow Research and Advanced Engine Combustion programs at Sandia National Laboratories' Combustion Research Facility (CRF), two complementary projects have been established to achieve this goal. The first is funded under the DOE SC Basic Energy Sciences (BES) program and focuses on the LES of turbulence–chemistry interactions in reacting multiphase flows. The

second is funded under the DOE Office of Energy Efficiency and Renewable Energy (EERE), Office of Vehicle Technologies (OVT) program and focuses on the application of LES to combustion research on high-pressure, low-temperature internal combustion engines. Figure 22 shows the key experiments currently being studied under these two projects using RAPTOR. A subset of experiments associated with the Reacting Flow Research Program is shown on the left. A subset of experiments associated with the Advanced Engine Combustion Program is shown on the right. Objectives and milestones for both projects are aimed at establishing high-fidelity computational benchmarks that identically match the geometry and operating conditions of key target experiments using a single unified theoretical-numerical framework (i.e., RAPTOR). The projects are complementary in that the DOE SC BES activity provides the basic science foundation for detailed model development and that the EERE-OVT activity provides the applied component for advanced engine research.

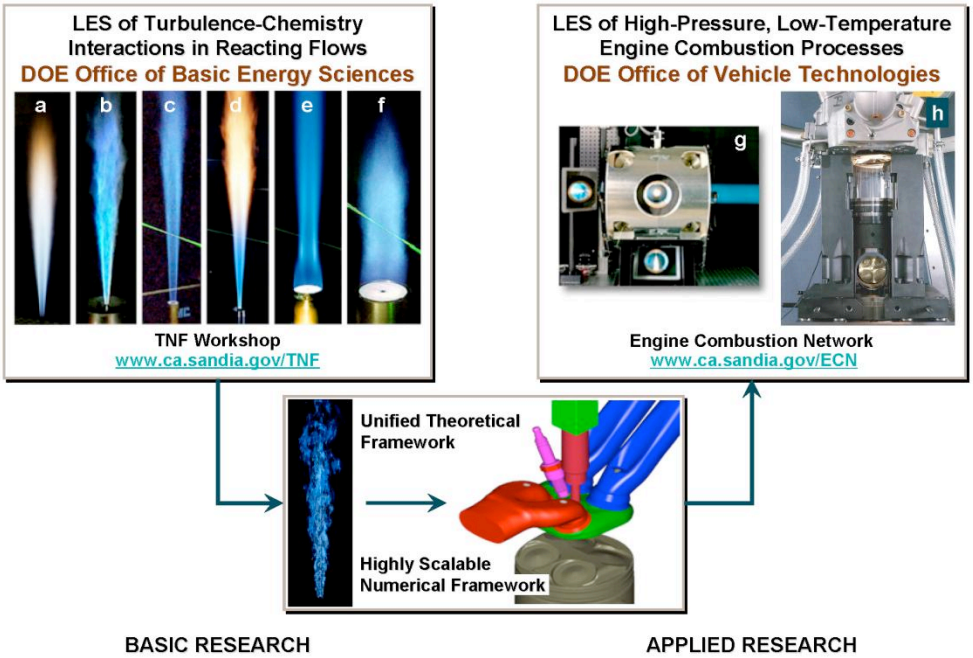


Fig. 22. Key experiments currently being studied using RAPTOR. *(left)* A subset of experiments associated with the Reacting Flow Research program (a,b: simple jet flames; c,d: piloted jet flames; e: bluff-body; f: bluff-body with swirl). *(right)* A subset of experiments associated with the Advanced Engine Combustion program (g: Constant-Volume Diesel combustion facility; h: typical single-cylinder optically accessible internal combustion engine).

Flames studied under Reacting Flow Research (see a–f in Fig. 22, for example) are internationally recognized benchmarks that provide some of the most detailed experimental data available for model validation. Using these data, significant collaborations with key modeling groups worldwide have been established as part of the International Workshop on Measurement and Computation of Turbulent Nonpremixed Flames (see Barlow et al. [23] for details). The TNF Workshop is an ongoing collaboration among experimental and computational researchers. A central theme of the series has been to use detailed comparisons of results from experiments and multiple modeling approaches to quantify state-of-the-art modeling capabilities and identify future research needs toward a predictive capability. As part of this activity, RAPTOR has been used to provide benchmark simulations that reach beyond the capabilities and resources of most universities and industry in a manner consistent with a national laboratory's role of using high-performance computing. We have two primary objectives. The first is to establish a set of high-fidelity computational benchmarks that identically match the geometry and operating conditions of selected experimental target flames. The second is to establish a scientific foundation for advanced model

development. The benchmark simulations provide a direct one-to-one correspondence between measured and modeled results at conditions unattainable using DNS by performing simulations that represent the fully coupled dynamic behavior of a reacting flow with detailed chemistry and realistic levels of turbulence. After achieving an adequate level of validation, results from these simulations provide fundamental information not measurable directly that is imperative for model development and provides a strong link between theory, canonical studies, experiments, and critical applications.

In contrast to the turbulent nonpremixed flames (TNFs), research activities related to Advanced Engine Combustion are focused on internal combustion engines. Needs and milestones related to RAPTOR have been established in three critical areas: (1) perform a progression of LES studies focused on the CRF optically accessible hydrogen-fueled internal combustion engine (see h in Fig. 22), (2) establish a parallel task focused on homogeneous charge compression ignition (HCCI) engines, and (3) perform a series of supporting studies focused on the development and validation of multiphase injection and combustion models with emphasis placed on direct-injection processes in IC-engines (see g in Fig. 22). The integrated set of research includes an optimal combination of in-cylinder and canonical (out-of-engine) studies to validate and understand key phenomenological processes that are present in internal combustion engine flow environments. These milestones are being facilitated in collaboration with ORNL as part of the 2009 INCITE project entitled "High-Fidelity Simulations for Clean and Efficient Combustion for Alternative Fuels." RAPTOR is being used to provide benchmark simulations in a manner identical to that described above for the TNFs. However, there are two key distinctions that must be made. Compared to the TNFs, the phenomenological and geometric complexities of device scale systems (such as internal combustion engines) reduce the level and fidelity of the experimental diagnostic techniques that can be applied. They also preclude the use of canonical DNS studies since appropriate initial and boundary conditions for such studies are largely unknown and unverified. Operating pressures are much greater, system Reynolds numbers are orders of magnitude higher, the flow fields associated with these devices are extremely complex, and a much broader range of dynamically evolving time and length scales need to be considered. To maximize the benefits of our fundamental and research efforts under these types of conditions, there is a clear need to understand what changes phenomenologically in various systems when one scales from laboratory conditions at atmospheric pressure (or equivalently lower Reynolds numbers) to application-relevant conditions at high pressures and Reynolds numbers.

Given the importance of Reynolds number scaling and its relation to combustion modeling and the INCITE calculations, our focal point for the Joule metric using RAPTOR will be the flames studied under the Reacting Flow Research program. A related set of experiments focused on passive scalar mixing will also be considered. Figure 23 shows a photograph of the baseline flame (known as DLR-A) along with an instantaneous image from LES. This flame corresponds to that shown in Fig. 22*(left, b)*. The photograph was taken in the Turbulent Combustion Laboratory at the CRF. The corresponding LES was performed using RAPTOR. In general, the integral-scale Reynolds numbers for the TNFs (which correspond to the jet Reynolds number here) are of $O(10^4)$, whereas those associated with internal combustion engines and related injection processes are of $O(10^5)$ or greater. The jet Reynolds number for this case is 15,200. Quantifying the effects of increasing Reynolds number on turbulent flame dynamics and the related scalar-mixing processes requires significant increases in CPU resources and is directly aligned with the need for weak scaling. Here, we will simultaneously study the related issues of Reynolds number scaling and resolution requirements for LES by successively increasing the problem size. A range of jet Reynolds numbers, starting from 15,200, will be considered. We will perform a series of weak scaling studies to demonstrate the effects of increasing Reynolds number on the dynamics of scalar mixing. Initial benchmark runs will be performed using 47,616 cores. Subsequent runs will be performed by systematically increasing the total CPU time required (i.e., total number of floating point operations per case) by factors of 2 as a function of increasing jet Reynolds number. The Joule metric will be accomplished by demonstrating we can simulate successively larger problems in the same amount of time.

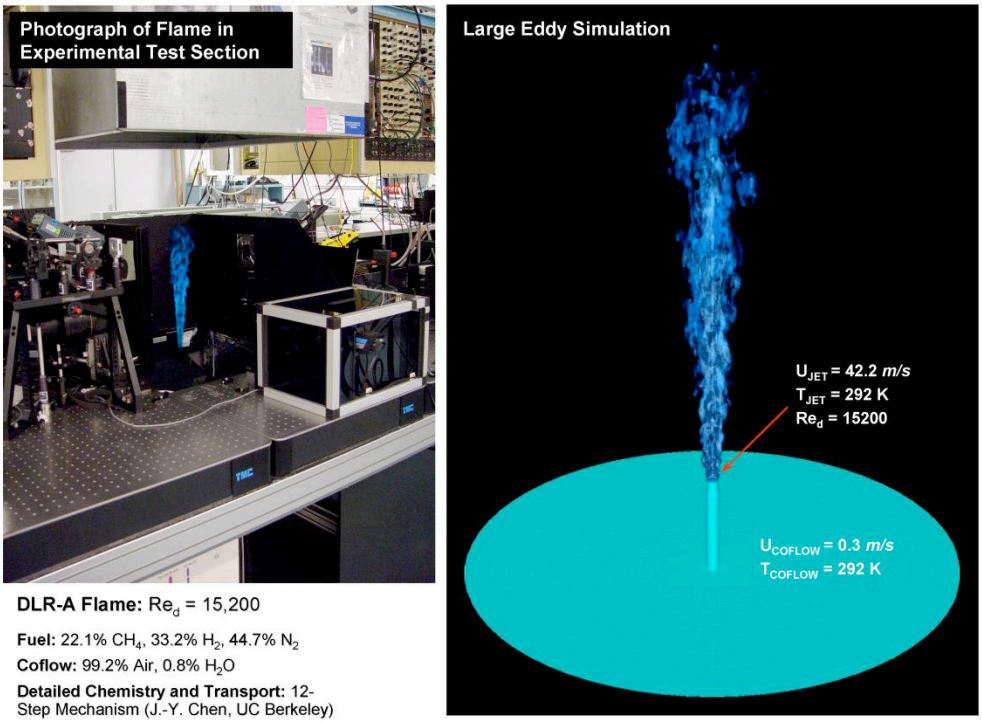


Fig. 23. Photograph and corresponding LES of the DLR-A flame (corresponds to b in Fig. 22).

3.4.3 Capability Overview

Physical Model. RAPTOR is a massively parallel flow solver designed specifically for application of the LES technique to turbulent, chemically reacting, multiphase flows. It solves the fully coupled conservation equations of mass, momentum, total energy, and species for a chemically reacting flow system (gas or liquid) in complex geometries. It also accounts for detailed chemistry, thermodynamics, and transport processes at the molecular level and uses detailed chemical mechanisms. The code is sophisticated in its ability to handle complex geometries and a generalized subgrid-scale model framework. It is capable of treating spray combustion processes and multiphase flows using a Lagrangian-Eulerian formulation. The numerical formulation treats the compressible form of the conservation equations but can be evaluated in the incompressible limit. The theoretical framework handles both multicomponent and mixture-averaged systems. The baseline formulation also employs a general treatment of the equation of state, thermodynamics, and transport properties that accommodates real gas or liquids with detailed chemistry (i.e., not constrained to ideal gas applications). Details are given by Oefelein [24].

Numerical Method. The temporal integration scheme employs an all Mach number formulation using the dual time stepping technique with generalized preconditioning. The approach is fourth-order accurate in time and provides a fully implicit solution using a fully explicit (highly scalable) multistage scheme in “pseudo time.” Preconditioning is applied in the inner pseudo time loop and coupled to local time-stepping techniques to minimize convective, diffusive, geometric, and source term anomalies (i.e., stiffness) in an optimal manner. This maximizes convergence rates as the system is advanced in time. The formulation is A-stable, which allows one to set the physical time step based solely on accuracy considerations. This attribute typically provides a 2 to 3 order-of-magnitude increase in the allowable integration time step compared to compressible flow solvers in the incompressible, low Mach number limit.

The spatial scheme is designed using nondissipative, discretely conservative, staggered, finite volume differencing stencils. The discretization is formulated in generalized curvilinear (i.e., body-fitted)

coordinates and employs a general R-refinement adaptive mesh (AMR) capability. This allows us to account for the inherent effects of geometry on turbulence over the full range of relevant scales while significantly reducing the total number of grid cells required in the computational domain. Treating the full range of scales is a critical requirement since turbulence–chemistry interactions are inherently coupled through a cascade of nonlinear interactions between the largest and smallest scales of the flow.

The differencing methodology has been specifically designed for LES. In particular, the second-order accurate staggered grid formulation, where we store scalar values at cell centers and velocity components at respective cell faces, fulfills two key accuracy requirements. First, it is spatially nondissipative, which eliminates numerical contamination of the subgrid-scale models due to artificial dissipation. Second, the stencils provide discrete conservation of mass, momentum, total energy, and species, which is an imperative requirement for LES. This eliminates the artificial buildup of velocity and scalar energy at the high wave numbers, which causes both accuracy problems and numerical instabilities in turbulent flow calculations. The algorithm includes appropriate switches to handle shocks, detonations, flame fronts, and contact discontinuities. It has also been designed using a generalized treatment for boundary conditions based on the method of characteristics.

Software Implementation. The RAPTOR code framework is massively parallel and has been optimized to provide excellent parallel scalability attributes using a distributed multiblock domain decomposition with a generalized connectivity scheme. Distributed memory message passing is performed using MPI and the Single Program–Multiple Data (SPMD) model. It accommodates complex geometric features and time varying meshes with generalized hexahedral cells while maintaining the high accuracy attributes of structured spatial stencils. The numerical framework has been ported to all major platforms and provides highly efficient coarse- and fine-grain (i.e., weak and strong) scalability attributes. The code is fully vectorized and has been optimized for both vector and commodity architectures. Further optimization is currently in progress to account for new issues associated with state-of-the-art multicore technology. The complete package is fully modular, self-contained, and written in ANSI standard Fortran 90. The complete theoretical–numerical framework (i.e., governing equations, physical submodels, numerics, and parallel efficiency) has been extensively validated over the course of the last 16 years. Representative results can be found in refs. [25] through [30].

3.4.4 Science Driver for Metric Problem

Given the importance of Reynolds number scaling and its relation to combustion modeling, our focal point for the Joule metric using RAPTOR are the flames studied under the Reacting Flow Research program at Sandia National Laboratories. A related set of experiments focused on passive scalar mixing will also be used. Figure 24 shows a photograph of the baseline flame (known as DLR-A) along with an instantaneous image from LES. The photograph was taken in the Turbulent Combustion Laboratory at the CRF. The corresponding LES was performed using RAPTOR. In general, the integral-scale Reynolds numbers for the TNFs (which correspond to the jet Reynolds number here) are of $O(10^4)$, whereas those associated with internal combustion engines and related injection processes are of $O(10^5)$ or greater. The jet Reynolds number for this case is 15,200. Quantifying the effects of increasing Reynolds number on turbulent flame dynamics and the related scalar-mixing processes requires significant increases in CPU resources. Here, we will study the related issues of Reynolds number scaling and resolution requirements for LES by successively increasing the problem size. A range of jet Reynolds numbers, starting from 15,200, will be considered. We will perform a series of weak scaling studies to demonstrate the effects of increasing Reynolds number on the dynamics of scalar mixing. Initial benchmark runs will be performed using 47,616 cores. Subsequent runs will be performed by systematically increasing the total CPU time required (i.e., total number of floating point operations per case) by factors of approximately 2 as a function of increasing jet Reynolds number. The Joule metric will be accomplished by demonstrating we can simulate successively larger problems in the same amount of time.

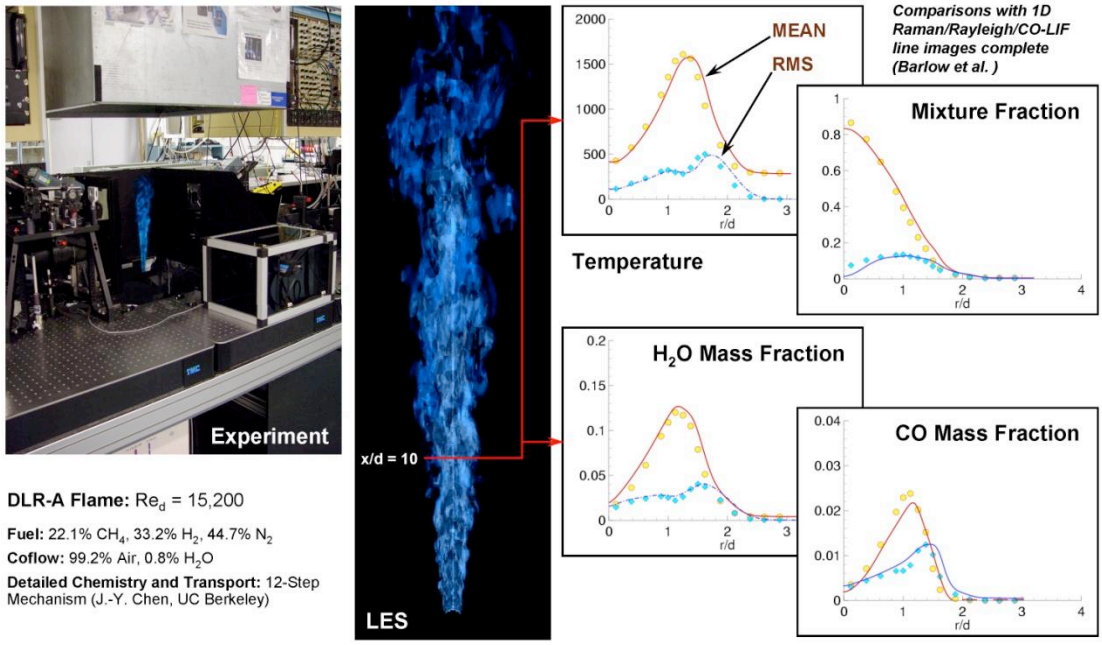


Fig. 24. Baseline flame used for problem scaling. *(left)* Photograph and *(center)* corresponding LES of the DLR-A flame. *(right)* Representative comparisons between experimentally measured (symbols) and modeled (lines) results showing acceptable agreement.

3.4.5 Q2 Baseline Problem Results

Our Q2 benchmark established the initial baseline for a series of weak scaling studies that demonstrate the combined computational effectiveness of the ORNL NCCS Jaguar/XT5 platform and RAPTOR. We simultaneously studied the related issues of Reynolds number scaling and resolution requirements for LES by successively refining the grid and temporal resolution of the DLR-A configuration shown in Fig. 23. A range of jet Reynolds numbers, starting from 15,200, were considered. The three primary objectives were to (1) study the effects of LES grid resolution on scalar-mixing processes, (2) understand the relationship between the grid spacing and the measured turbulence length scales from a companion set of experimental data, and (3) study the effects of increasing jet Reynolds number on the dynamics of turbulent scalar mixing. The initial benchmark was run using 47,616 cores. Subsequent runs were performed by systematically increasing the total CPU time required (i.e., total number of floating point operations per case) by factors of approximately 2 as a function of increasing jet Reynolds number.

Figure 25 shows a cross section of the computational domain that highlights key features of the optimized curvilinear grid topology. To eliminate ambiguities associated with boundary conditions, the computational domain includes the entire burner geometry (inside the jet nozzle and the outer co-flow) and extends downstream over a span that covers the same dimensions as the experimental test section. The nozzle geometry corresponds to that shown in Fig. 23. The inner nozzle diameter is 8.0 mm. The outer nozzle surface is tapered to a sharp edge at the burner exit. The overall dimensions of the computational domain are 110 inner jet diameters in the axial direction and 40 jet diameters in the radial direction (88 cm by 32 cm, respectively). Flow inside the jet nozzle is simulated by assuming that the turbulent flow dynamics far upstream are fully developed. Using this assumption we impose a time-dependent inflow condition 10 jet diameters upstream of the nozzle exit (i.e., at the base of the image shown in Fig. 23) and allow it to evolve in a time accurate manner to the nozzle exit. The outer co-flow is imposed in a similar manner. A far field force-free pressure condition is applied at the downstream and transverse boundaries.

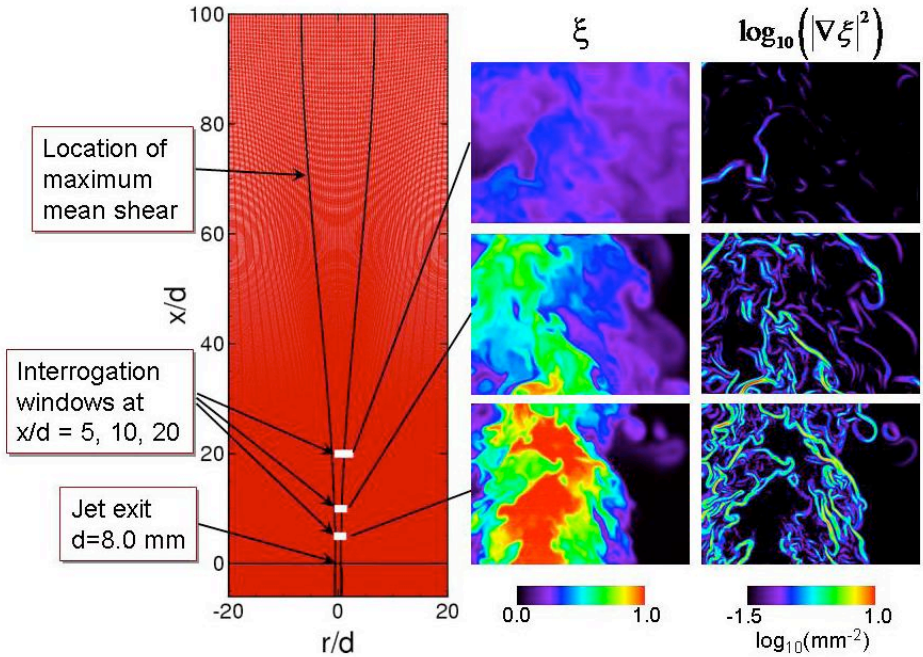


Fig. 25. Cross section of the computational domain showing key features of the grid topology. The domain includes the entire burner geometry, as shown in Fig. 23. To the right are single-shot measurements of mixture fraction and scalar dissipation. The jet Reynolds number is 15,200.

A novel feature of our approach is to design respective grids using the dissipation spectrum cutoff length scales measured from the companion experiments. These scales represent the average thickness of the scalar mixing layers (i.e., the structural dimensions of the turbulent scalar eddies). Example images of both mixture fraction and the scalar dissipation layers are shown in Fig. 25. The white boxes on the grid indicate the experimental interrogation windows at x/d = 5, 10, and 20. Elongated filaments of high dissipation reveal the convoluted inhomogeneous structure of the fine-scale scalar mixing processes. Data similar to that shown in Fig. 25 were used to design a set of optimally stretched curvilinear grids that provided a consistent level of resolution in all three coordinate directions relative to the local physical mixing layers in the flow. A representative set of grid sizes are listed in Table 17. Grid 3 was established as an initial arbitrary baseline by sizing cells such that the local spacing throughout the domain was nominally the same size as the cutoff length scales. Grids 2 and 1 were obtained by successively coarsening Grid 3 by a factor of 2 in each coordinate direction while maintaining the curvilinear topology shown in Fig. 25. Using these three grids, we have performed an initial series of calculations that identically match the experimental flow conditions. Calculations were carried out on Grid 1 first to determine what the appropriate time step was. For the case considered, a time step of 1 μs was found to give an appropriate level of time accuracy. The Q2 benchmark case was performed using Grid 2 with a corresponding time step of 0.5 μs. Other relevant run parameters are listed in Figs. 24 and 25.

Table 17. Baseline grid sizes for Joule benchmark runs*

Grid Number	Total Cells	Δt (Re_d = 15,200)
1	1,285,632	1.00 μs
2	10,285,056	0.50 μs
3	82,280,448	0.25 μs

*Respective grids are successively refined by a factor of 2 in each coordinate direction while maintaining the curvilinear topology shown in Fig. 23. The corresponding integration time steps (Δt) are incremented by factors of 2 in a manner consistent with the spatial refinement.

To acquire the appropriate performance statistics, the DLR-A configuration described above was run for 50 physical time steps. The physical results were validated using the experimental data provided by Barlow et al. [23]. A representative set of results are given in Fig. 26, which shows comparisons between numerical results via RAPTOR (lines) and measured Raman/Rayleigh/CO-LIF line image data (symbols). Here we show mean and RMS profiles. These results, coupled with similar comparisons performed throughout the domain, provide a validated level of confidence in the accuracy of the solution. The computational performance was simultaneously evaluated by using the CrayPAT instrumented executable in place of the original executable. The program was instrumented to provide hardware performance counter information from start to finish. The simulation was performed using 47,616 processor cores on the Cray XT5 system. The CrayPAT output was postprocessed using pat report, as shown in Table 18. On average, each processor core performed 7.94 billion floating point operations, leading to an aggregate of 378 trillion floating-point operations being performed by the 47,616 cores. We measure the computational performance of RAPTOR for a given problem using the metric

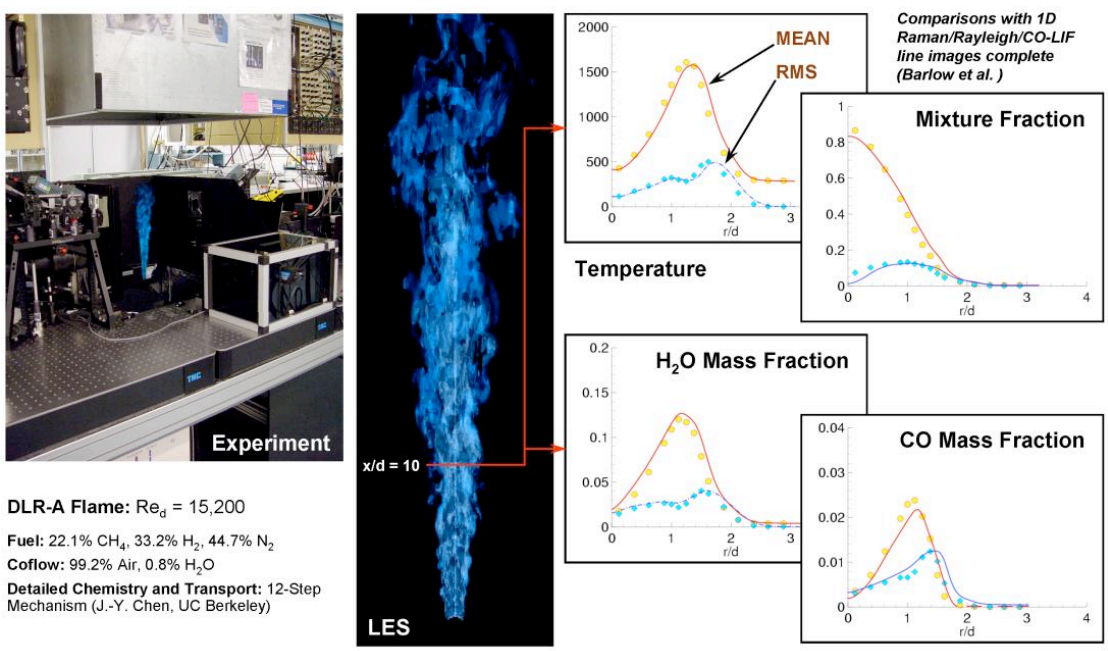


Fig. 26. Comparison of experimentally measured (symbols) and modeled (lines) results showing acceptable agreement.

Performance = CPU time / number of grid cells / number of time steps ,

where CPU time is the product of two quantities: (1) wall-clock time taken from start to finish of the time integrator portion of the solver and (2) the number of processor cores occupied by the job while the program was executing.

For the benchmark run, the code's internal timers reported that the time integration through 50 time steps took 1,034 seconds. The remaining time (approximately 300 seconds) was consumed by the initialization step when the computational mesh and initial condition information were read from the disk and the software prepared itself for the simulation. Therefore, the performance of RAPTOR during the benchmark simulation was

(1, 034 seconds × 47, 616 cores) / 10, 285, 056 cells / 50 time steps = 0.096 .

This indicates that it cost 96 milliseconds of processor time per cell per time step to simulate the problem on 47,616 cores. Subsequent runs will be performed by systematically increasing the total CPU time required (i.e., total number of floating point operations per case) by approximate factors of 2 as a function of increasing jet Reynolds number. The Joule metric will be accomplished by demonstrating we can simulate successively larger problems in the same amount of time.

Table 18. Counter data acquired from CrayPAT 4.2 for Q2 benchmark run using RAPTOR

```
Time% 100.0%
Time 1425.761880 secs
Imb.Time -- secs
Imb.Time% --
Calls 0.0 /sec 4.0 calls
PAPI_L1_DCM 20.674M/sec 26457314029 misses
PAPI_TOT_INS 3379.668M/sec 4325136094614 instr
PAPI_L1_DCA 1348.943M/sec 1726311709236 refs
PAPI_FP_OPS 6.204M/sec 7939032813 ops
User time (approx) 1279.752 secs 2943428628772 cycles 89.8%Time
Average Time per Call 356.440470 sec
CrayPat Overhead : Time 0.0%
HW FP Ops / User time 6.204M/sec 7939032813 ops 0.1%peak(DP)
HW FP Ops / WCT 5.568M/sec
HW FP Ops / Inst 0.2%
Computational intensity 0.00 ops/cycle 0.00 ops/ref
Instr per cycle 1.47 inst/cycle
MIPS 160926295.28M/sec
MFLOPS (aggregate) 295389.35M/sec
Instructions per LD & ST 39.9% refs 2.51 inst/ref
D1 cache hit,miss ratios 98.5% hits 1.5% misses
D1 cache utilization (M) 65.25 refs/miss 8.156 avg uses
```

3.4.6 Computational Performance Gains

During Q3 the performance of RAPTOR on the model problem was studied and the software was revised to obtain better computational performance. To obtain a quick turn-around time in the queues and for easier postprocessing of CrayPAT output, the Q2 model problem was resized to run on 5,952 cores. The performance profiles obtained on 5,952 cores were then used to guide the tuning. The results of code changes were tested by measuring the execution time on 5,952 and 47,616 cores.

CrayPAT was used to obtain a performance profile of the 10.3 million cell Q2 model problem on 5,952 cores. Figure 27 shows a budget of the time spent in the code. Given the inherent fine-grain nature

of the problem, a significant amount of time was being spent in MPI calls for this case and only 20% of the time was being spent in the Fortran routines.

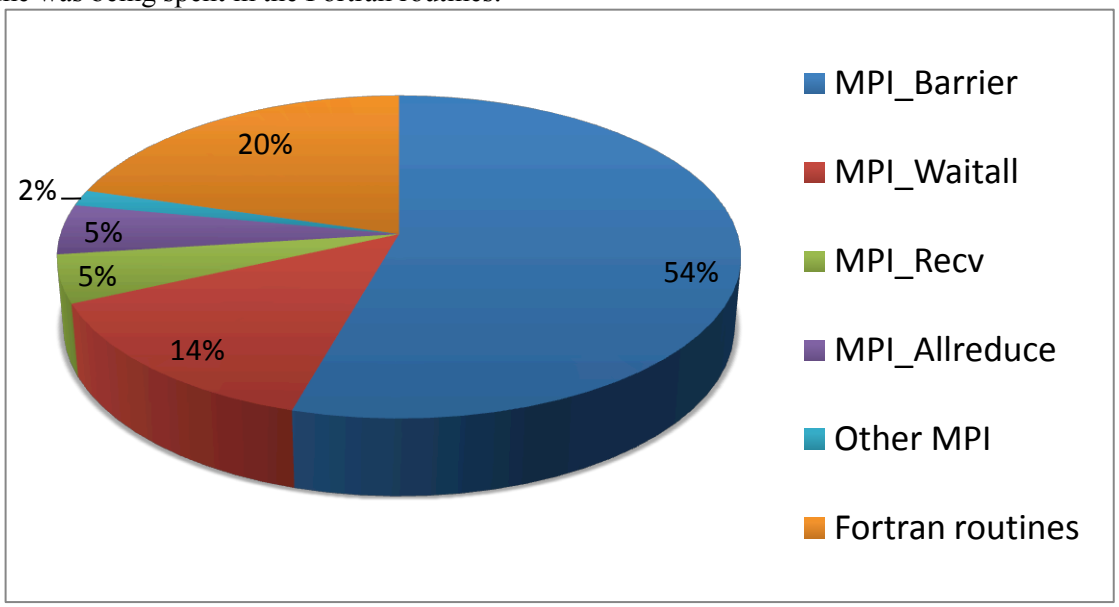


Fig. 27. Performance profile of the original code on 5,952 XT5 cores.

It was found that the largest amount of time was being spent in MPI_Barrier and MPI_Allreduce. MPI_Barrier was being called from the user routine dbsh, which provides the time-dependent inflow boundary condition for the case. The inflow boundary condition is read as a function of time (i.e., at the beginning of each time step in the integrator) from a file, and when the end of file is reached, the file is rewound and the boundary condition was recycled at the inlet. One of the tasks performed by subroutine dbsh was to check the end of the inlet boundary condition and rewind the file accordingly. This portion of the routine was rewritten such that it calls a MPI_barrier only when a file rewind was necessary.

MPI_Allreduce was being called from the subroutine norm, which is used to monitor convergence rates of respective residuals in the dual-time integrator. This routine provides the convergence of the solution vector using either a L_2 or L_infinite (max) norm. The computed norm was then compared against the error criterion to determine when to terminate the inner pseudo time step iteration and proceed to the next physical time step. A global MPI_allreduce was necessary for computing the error norm. Since MPI_allreduce affects the scalability of the software, the convergence test was modified taking into account the fact that the number of pseudo time iterations necessary to obtain convergence will not vary drastically between consecutive time steps. In the revised routine, the last pseudo time step in which convergence was achieved in the previous physical time step is saved in a static variable, say N_c. In the next physical time step, the convergence check is deferred until $N_c - 1$ pseudo time steps. This way the solver would perform a few extra iterations occasionally while avoiding the expensive convergence check after each iteration.

A profile of the revised code after the above-mentioned changes to global MPI operations is shown in Fig. 28. It is seen that the MPI barrier and allreduce costs have decreased. However, a significant amount of time is still being spent in the MPI communication routines, especially point-to-point send/receives and related waits.

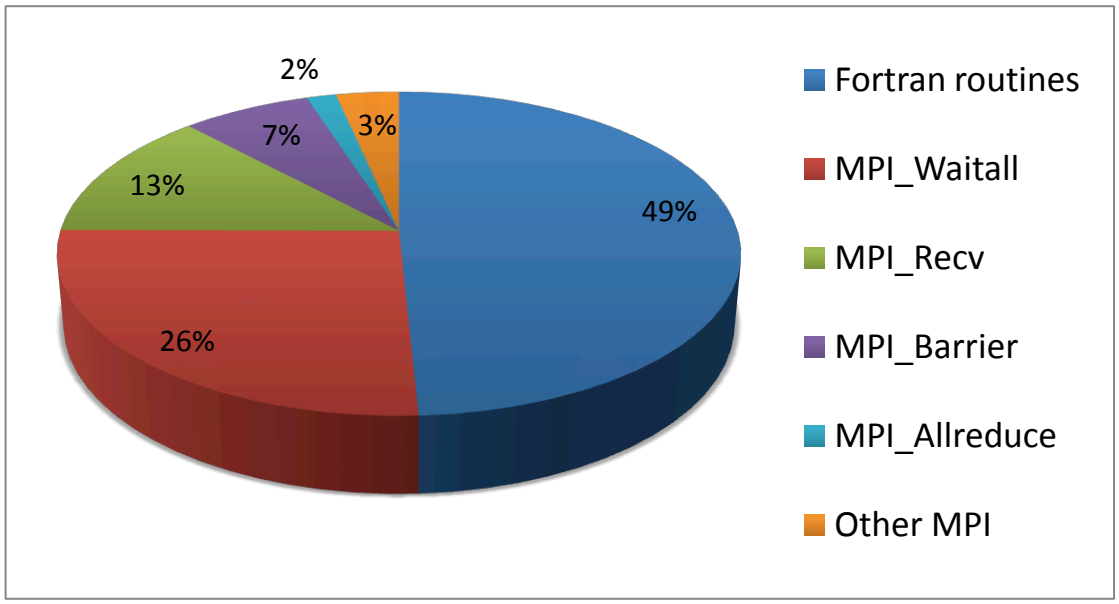


Fig. 28. Performance profile of RAPTOR on 5,952 cores after reducing global MPI operations.

RAPTOR uses halo communications to build a ghost zone around the problem domain in each MPI rank. The halo communications are performed between the nearest neighbors within the 3D grid topology. The MPI cost in performing the nearest neighbor communication was reduced through a rewrite of three main halo exchange routines—halo, halo_dqv, and halo_flx. The MPI communications in these routines were rearranged with the following principles: (1) prepost all receives as the first operation in the routine, (2) post the sends as soon as the data is available, and (3) postpone the waits on send operations until the end of the routine. Nonblocking sends and receives are used throughout, both before and after these modifications.

A last set of modifications was aimed at reducing wait times due to load imbalances induced at the boundaries and grid centerline. It was also noticed that any load imbalance in routines prior to halo exchanges led to increased MPI wait times. The main source of this imbalance was due to treatment of the boundaries and centerline where only the MPI ranks at these respective locations were assigned work and the remaining ranks did not perform any computation. A prime candidate for tuning in this respect was subroutine pole, which handles the singularity associated with the swept grid design. This routine was tuned by creating separate subcommunicators consisting of the centerline ranks at various axial planes. Then the required velocity averaging was implemented using MPI_allreduce on the subcommunicator instead of send/receive operations. This was found to reduce the time taken by this routine and thereby lead to better load balance and lower MPI wait times.

The current performance profile of the code with all revisions made to date is shown in Fig. 29. Table 19 shows the net reductions in time to solution on 5,952 and 47,616 cores as a result of the code changes.

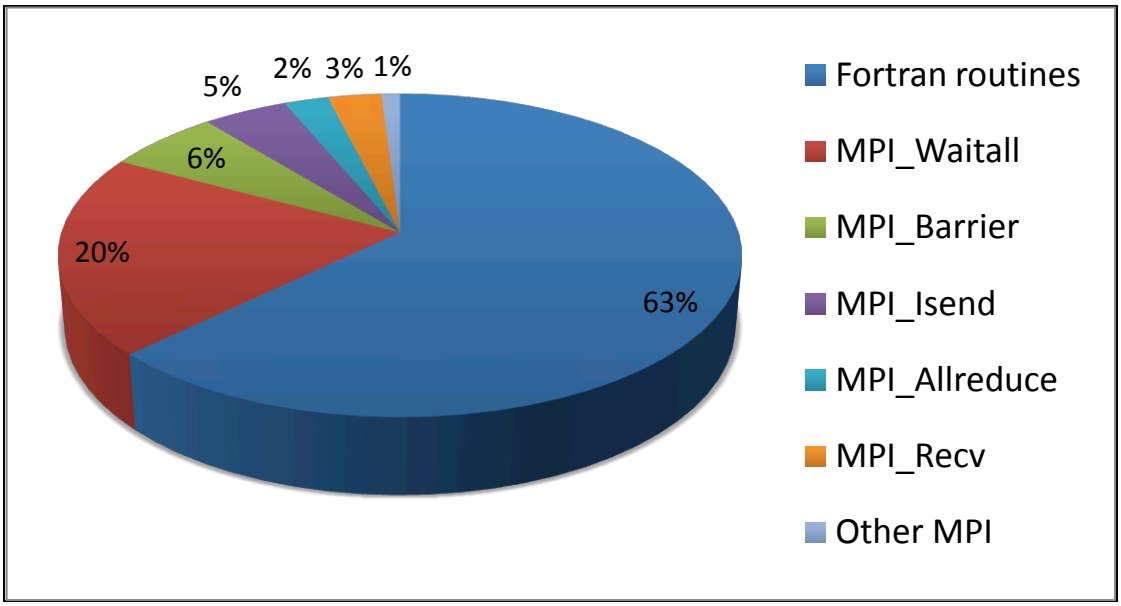


Fig. 29. Performance profile of RAPTOR after software revisions.

Table 19. Summary of measured timings (in seconds) after each set of code revisions using the 10.3 million cell Q2 test problem and 200 time steps

	5,952 cores	47,616 cores
Original Q2 software	1,414	4,136
After revising dbsh (less barrier)	555	315
After revising norm (less allreduce)	510	242
Revised halo and pole (better point to point)	450	192

3.4.7 Q4 Metric Problem Results

RAPTOR was compiled using the default PGI Fortran compiler. Output from the compilation is included in the Appendix and is identical for both the Q2 benchmark and Q4 metric. Here we show only the skeletal output, which includes the options used for optimization of the code. Note that the complete output, which includes all information related to the optimization, is also available but spans 21,807 lines and has thus been omitted in the interest of space. In all cases the code was profiled using CrayPAT 4.2 using the following recipe to build the executable

```
module load xt-craypat
make
pat_build -w -Drtenv=PAT_RT_HWPC=0 DTMS.out DTMS_pat.out .
```

The instrumented executable (DTMS pat.out) was run using the batch script listed in the Appendix. Here we show the script used for Q2. The corresponding run time environment is also listed and essentially identical for both the Q2 and Q4 cases except for the different number of cores used for each. Performance data was generated by issuing the commands

```
module load xt-craypat
pat_report DTMS_pat.out+xxxyyy > report.out ,
```

where DTMS pat.out+xxxyyy is the name of the directory created by CrayPAT after the run completed. Table 20 lists the resultant set of data.

To acquire the appropriate performance statistics, the DLR-A configuration described above was run for 50 physical time steps. The physical results were validated using the experimental data provided by Barlow et al. [23], where a representative set of results is given in Fig. 4, which shows comparisons between numerical results via RAPTOR (lines) and measured Raman/Rayleigh/CO-LIF line image data (symbols). Here we show mean and RMS profiles. These results, coupled with similar comparisons performed throughout the domain, provide a validated level of confidence in the accuracy of the solution. The computational performance was simultaneously evaluated by using the CrayPAT instrumented executable in place of the original executable. The program was instrumented to provide hardware performance counter information from start to finish. The simulation was performed using 47,616 processor cores on the Cray XT5 system. The CrayPAT output was post processed using pat report, as shown in Table 20. On average, each processor core performed 7.94 billion floating point operations, leading to an aggregate of 378 trillion floating point operations being performed by the 47,616 cores. We measure the computational performance of RAPTOR for a given problem using the metric

$$\text{Performance} = \text{CPU time} / \text{number of grid cells} / \text{number of time steps} ,$$

where CPU time is the product of two quantities: (1) wall-clock time taken from start to finish of the time integrator portion of the solver, and (2) the number of processor cores occupied by the job while the program was executing.

For the Q2 benchmark run, the code's internal timers reported that the time integration through 50 time steps took 1,034 seconds. The remaining time (approximately 300 seconds) was consumed by the initialization step for CrayPAT and when the computational mesh and initial condition information were read from the disk. Therefore, the performance of RAPTOR during the benchmark simulation was

$$(1,034 \text{ seconds} \times 47,616 \text{ cores}) / 10,285,056 \text{ cells} / 50 \text{ time steps} = 0.096 .$$

It cost 96 milliseconds of processor time per cell per time step to simulate the problem on 47,616 cores. Using this benchmark, our Q4 metric was performed by increasing the total CPU time required (i.e., total number of floating point operations per case) by factors of approximately 2 as a function of increasing jet Reynolds number. The Joule goal metric was accomplished by demonstrating we can simulate successively larger problems in the same amount of time.

Table 20. Counter data acquired from CrayPAT 4.2 for the Q2 benchmark run using RAPTOR

```
Time% 100.0%
Time 1425.761880 secs
Imb.Time -- secs
Imb.Time% --
Calls 0.0 /sec 4.0 calls
PAPI_L1_DCM 20.674M/sec 26457314029 misses
PAPI_TOT_INS 3379.668M/sec 4325136094614 instr
PAPI_L1_DCA 1348.943M/sec 1726311709236 refs
PAPI_FP_OPS 6.204M/sec 7939032813 ops
User time (approx) 1279.752 secs 2943428628772 cycles 89.8%Time
Average Time per Call 356.440470 sec
CrayPat Overhead : Time 0.0%
HW FP Ops / User time 6.204M/sec 7939032813 ops 0.1%peak(DP)
HW FP Ops / WCT 5.568M/sec
HW FP Ops / Inst 0.2%
Computational intensity 0.00 ops/cycle 0.00 ops/ref
Instr per cycle 1.47 inst/cycle
MIPS 160926295.28M/sec
```

```
MFLOPS (aggregate) 295389.35M/sec
Instructions per LD & ST 39.9% refs 2.51 inst/ref
D1 cache hit,miss ratios 98.5% hits 1.5% misses
D1 cache utilization (M) 65.25 refs/miss 8.156 avg uses
```

For the Q4 benchmark, we modified the DLR-A model problem run in Q2 by systematically increasing the jet Reynolds number. Increasing the Reynolds number induces a reduction in the smallest turbulence scales in the flow and thus increases the range of scales that must be considered in the calculation. To account for this increase, both the grid resolution and physical time step must be refined appropriately (i.e., linear scaling in the weak sense must be achieved to keep the total time required for the calculation the same as Q2). To demonstrate the performance of RAPTOR, we increased the total CPU time required (i.e., the total number of floating point operations per case) and total number of cores used by a factor of 2.359. The final Q4 case used 24,261,120 grid cells and 112,320 cores (compared to 10,285,056 cells and 47,616 cores for Q2). The resultant run was analyzed in a manner identical to the Q2 run. The final results are shown in Table 21. Weak scaling from Q2 to Q4 was near linear. In addition to scaling linearly, we were able to achieve an additional net improvement in the overall code performance of 2.329 beyond the linear metric due to the above-mentioned code improvements.

Table 21. Summary of results from the Q4 run compared to the Q2 baseline

	Q2	Q4
Grid size	10.3 million	24.3 million
Number of XT5 cores	47,616	112,320
Time taken for integrating 50 time steps	1,034 seconds	444 seconds
Number of floating point operations	378×10^{12}	893×10^{12}
Flop rate sustained by the unsteady solver	0.36 TF/second	2.0 TF/second
Cost per grid point per time step	0.096 seconds	0.041 seconds

Performance statistics summarized in Table 21 were acquired in a manner identical to the Q2 run. The DLR-A configuration was run for 50 physical time steps at a higher Reynolds number to study issues related to scalar mixing and the related structural dynamics of the flow. The computational performance was evaluated by using a CrayPAT instrumented executable in place of the original executable and configured to give hardware performance counter information from start to finish. The CrayPAT output was postprocessed using pat report and is given in Table 22.

Table 22. Counter data acquired from CrayPAT 4.2 for the Q4 run using RAPTOR

```
Time% 100.0%
Time  1972.397426 secs
Imb.Time secs
Imb.Time% --
Calls 0.0 /sec 4.0 calls
PAPI_L1_DCM 17.939M/sec    30225838071 misses
PAPI_TOT_INS 3505.170M/sec  5906032655453 instr
PAPI_L1_DCA  1400.128M/sec  2359144379197 refs
PAPI_FP_OPS  4.718M/sec     7948841143 ops
User time (approx) 1684.949 secs  3875382374683 cycles  85.4%Time
Average Time per Call 493.099356 sec
CrayPat Overhead : Time 0.0%
HW FP Ops / User time 4.718M/sec     7948841143 ops  0.1%peak(DP)
HW FP Ops / WCT 4.030M/sec
```

```
HW FP Ops / Inst 0.1%
Computational intensity 0.00 ops/cycle      0.00 ops/ref
Instr per cycle 1.52 inst/cycle
MIPS 393700725.39M/sec
MFLOPS (aggregate) 529875.93M/sec
Instructions per LD & ST  39.9% refs 2.50 inst/ref
D1 cache hit,miss ratios  98.7% hits 1.3% misses
D1 cache utilization (M)  78.05 refs/miss 9.756 avg uses
```

In running the Q4 case, we observed an anomaly associated with the time required for the initialization stage of the calculation (which is not compute intensive) compared to the integration stage (which is compute intensive). This anomaly was traced to CrayPAT. In all cases, our executables that were instrumented with CrayPAT exhibited a wide range of initialization times compared to those that were not. In the results for Q2, for example, the total run time reported by CrayPAT was 1,423 seconds. However, the time spent in the integration part of the calculation was only 1,034 seconds. Similarly, the Q4 calculation took a total of 1,972 seconds for both initialization and integration; however, only 444 seconds were spent in the integration part. To verify this we performed several additional tests. First, we reran the Q2 case with the integration loop bypassed to isolate the time associated with initialization. Results from this run are provided in Appendix E, Sect. E.5 and compared to the original Q2 counter data shown in Fig. 27. These data verify that a negligible amount of floating point operations occur during initialization, and also that the internal clock used to measure the amount of time spent in the integrator was accurate, as reported in Table 21 above. As a second test, we ran both cases without CrayPAT installed and verified that the initialization times for both became negligible (i.e., less than 10 percent of the total integration time). The combined set of tests confirms that the integration times and estimated floating point operation rates reported are accurate.

For the Q4 run, the code's internal timers reported that the time integration through 50 time steps took 444 seconds. Here the code performed 7.94 billion floating point operations on each core (as in Q2), leading to an aggregate of 893 trillion floating point operations being performed by the 112,320 cores. The remaining time was consumed by the initialization step required for CrayPAT and when the computational mesh and initial condition information were read from the disk. Thus, the performance of RAPTOR for Q4, which includes the performance enhancements described in the last section, was

$$(444 \text{ seconds} \times 112{,}320 \text{ cores}) / 24{,}261{,}120 \text{ cells} / 50 \text{ time steps} = 0.041 \ ,$$

which is a factor of 2.3 improvement in speed beyond the linear weak-scaling metric specified as our target.

3.4.8 Interpretation of Results

The selected Joule goal metric described here has established an initial baseline for a series of weak scaling studies aimed at demonstrating the combined computational effectiveness of the ORNL NCCS Jaguar/XT5 platform and RAPTOR. Given the importance of Reynolds number scaling and its relation to combustion modeling, our focal point for the Joule metric is the experimental flames studied under the Reacting Flow Research program at Sandia National Laboratories. These flames are internationally recognized as important benchmarks for model validation and provide a significant amount of high quality data for model development. A key issue, however, is that the integral-scale Reynolds number for these flames (which corresponds to the jet Reynolds number here) is of $O(10^4)$, whereas those associated with several important applications are of $O(10^5)$ or greater. Thus, it is necessary to understand the phenomenological changes that occur as a function of Reynolds number when one scales to device-level conditions.

Quantifying the effects of increasing Reynolds number on turbulent flame dynamics and the related scalar mixing processes requires significant increases in CPU resources and is directly aligned with the need for highly efficient weak scaling attributes. Here our goal is to study the related issues of Reynolds

number scaling and resolution requirements for LES by successively increasing the problem size. A range of jet Reynolds numbers, starting from 15,200, was considered. We will perform a series of weak scaling studies to demonstrate the effects of increasing Reynolds number on the dynamics of scalar mixing. Our initial Q2 benchmark runs were performed using 47,616 cores. Subsequent runs were performed by systematically increasing the total CPU time required (i.e., total number of floating point operations per case) by factors of approximately 2 as a function of increasing jet Reynolds number. The Joule metric was accomplished by demonstrating we can simulate successively larger problems in the same amount of time.

Our initial Q2 benchmark run was performed using 47,616 cores on the Cray XT5 system. On average, each processor core performed 7.94 billion floating point operations, leading to an aggregate of 378 trillion floating point operations. We measured the computational performance of RAPTOR using the metric

Performance = CPU time / number of grid cells / number of time steps ,

where CPU time is the product of two quantities: (1) wall-clock time taken from start to finish of the time integrator portion of the solver and (2) the number of processor cores occupied by the job while the program was executing (i.e., the "grind time"). For the benchmark run, the code's internal timers reported that the time integration through 50 time steps took 1,034 seconds. Therefore, the benchmark performance of RAPTOR was calculated as 0.096 (i.e., it cost 96 milliseconds of processor time per cell per time step to simulate the problem on 47,616 cores).

During Q3 the performance of RAPTOR on the model problem was studied and the software was revised to obtain better computational performance. To obtain a quick turn-around time in the queues and for easier postprocessing of CrayPAT output, the Q2 model problem was resized to run on 5,952 cores. The performance profiles obtained on 5,952 cores were then used to guide the tuning. The results of code changes were tested by measuring the execution time on 5,952 and 47,616 cores. The collective efforts led to a net increase in the time spent by the Fortran routines from 20 percent to 63 percent (i.e., latency due to fine-grain communication overhead was reduced from 80 percent to 37 percent for the selected model problem), which provided a significant net speedup in the performance of RAPTOR.

For the Q4 benchmark, we modified the DLR-A configuration run in Q2 by systematically increasing the jet Reynolds number. Increasing the Reynolds number induces a reduction in the smallest turbulence scales in the flow and thus increases the range of scales that must be considered in the calculation. To account for this increase, both the grid resolution and physical time step must be refined appropriately (i.e., linear scaling in the weak sense must be achieved to keep the total time required for the calculation the same as Q2). To demonstrate the performance of RAPTOR, we increased the total CPU time required (i.e., the total number of floating point operations per case) and total number of cores used by a factor of 2.359. The final Q4 case used 24,261,120 grid cell and 112,320 cores. The resultant run was analyzed in a manner identical to the Q2 run. The code's internal timers reported that the time integration through 50 time steps took 444 seconds. The remaining time was consumed by both the initialization step when the computational mesh and initial condition information were read from the disk and the CrayPAT instrumentation in the software. Thus, the performance of RAPTOR for Q4 (which includes the performance enhancements performed as part of Q3 activities) was calculated to be 0.041 (compared to 0.096 for Q2), which is a factor of 2.3 improvement in speed beyond the linear weak-scaling metric specified as our target.

3.4.9 Summary and Conclusions

The Joule metric selected here was designed to establish a baseline for a series of weak scaling studies aimed at demonstrating the combined computational effectiveness of the ORNL NCCS Jaguar/XT5 platform and RAPTOR. Given the importance of Reynolds number scaling and its relation to combustion modeling, our focal point for the Joule metric is the experimental flames studied under the Reacting Flow Research program at Sandia National Laboratories. These flames are internationally recognized as important benchmarks for model validation and provide a significant amount of high

quality data for model development. A key issue, however, is that the integral-scale Reynolds number for these flames (which corresponds to the jet Reynolds number here) is of $O(10^4)$, whereas those associated with several important applications are of $O(10^5)$ or greater. Thus, it is necessary to understand the phenomenological changes that occur as a function of Reynolds number when one scales to device-level conditions.

Quantifying the effects of increasing Reynolds number on turbulent flame dynamics and the related scalar mixing processes requires significant increases in CPU resources and is directly aligned with the need for highly efficient weak scaling attributes. Here our goal is to study the related issues of Reynolds number scaling and resolution requirements for LES by successively increasing the problem size. A range of jet Reynolds numbers, starting from 15,200, was considered. We performed a series of weak scaling studies to demonstrate the effects of increasing Reynolds number on the dynamics of scalar mixing. Our initial Q2 benchmark was performed using 47,616 cores on the Cray XT5 system. On average, each processor core performed 7.94 billion floating point operations, leading to an aggregate of 378 trillion floating point operations. We measured the computational performance of RAPTOR using the metric

Performance = CPU time / number of grid cells / number of time steps ,

where CPU time is the product of two quantities: (1) wall-clock time taken from start to finish of the time integrator portion of the solver and (2) the number of processor cores occupied by the job while the program was executing (i.e., the "grind time"). For the benchmark run, the code's internal clock reported that the time integration through 50 time steps took 1,034 seconds. Therefore, the benchmark performance of RAPTOR was calculated as 0.096 (i.e., it costs 96 milliseconds of processor time per cell per time step to simulate the problem on 47,616 cores).

During Q3 the performance of RAPTOR on the DLR-A model problem was studied and the software was revised to obtain better computational performance. To obtain a quick turn-around time in the queues and for easier postprocessing of CrayPAT output, the Q2 model problem was resized to run on 5,952 cores. The performance profiles were then used to guide the tuning. The changes were tested by measuring the execution time on 5,952 and 47,616 cores. The collective efforts led to a net increase in the time spent by the Fortran routines from 20 percent to 63 percent (i.e., latency due to fine-grain communication overhead was reduced from 80 percent to 37 percent for the selected model problem), which provided a significant net speedup in the performance of RAPTOR.

For the Q4 benchmark, we modified the DLR-A case considered in Q2 by systematically increasing the jet Reynolds number. Increasing the Reynolds number induces a reduction in the smallest turbulence scales in the flow and thus increases the range of scales that must be considered in the calculation. To account for this increase, both the grid resolution and physical time step must be refined appropriately (i.e., linear scaling in the weak sense must be achieved to keep the total time required for the calculation the same as Q2). To demonstrate the performance of RAPTOR, we increased the total CPU time required (i.e., the total number of floating point operations per case) and total number of cores used by a factor of 2.359. The final Q4 case used 24,261,120 grid cells and 112,320 cores.

Results from the Q4 run were analyzed in a manner identical to the Q2 run. The code's internal clocks reported that the time integration through 50 time steps took 444 seconds. The remaining time was consumed by the initialization routines for CrayPAT, reading the computational mesh from the disk and setting the initial conditions. Thus, the performance of RAPTOR for the Q4 benchmark (which includes the performance enhancements made as part of the Q3 activities) was calculated to be 0.041 (compared to 0.096 for Q2). This represents a factor of 2.3 improvement in speed beyond the linear weak scaling metric specified as our target.

REFERENCES

1. W. Lorensen, H. Cline: *Marching Cubes: A High Resolution 3D Surface Construction Algorithm,* Computer Graphics, Vol. 21, Nr. 4, July 1987
2. H. Childs, E. S. Brugger, K. S. Bonnell, J. S. Meredith, M. Miller, B. J. Whitlock, and N. Max, A contract-based system for large data visualization, in Proceedings of IEEE Visualization, pp. 190–198, 2005.
3. The Intergovernmental Panel on Climate Change Working Group I (WGI) Report, *Climate Change 2007: The Physical Basis* (http://ipcc-wg1.ucar.edu/wg1/wg1-report.html).
4. *The Community Atmosphere Model (CAM)* (http://www.ccsm.ucar.edu/models/atm-cam).
5. W. D. Collins et al., *A Description of the NCAR Community Atmosphere Model (CAM 3.0)*, Technical Report NCAR/TN-464+STR, National Center for Atmospheric Research, Boulder, Colorado, 2004.
6. W. M. Washington, *Documentation for the Community Climate Model (CCM), Version 0*, National Center for Atmospheric Research, Boulder, Colorado, NTIS No. PB82 194192, 1982.
7. D. L. Williamson, *Description of NCAR Community Climate Model (CCM0B)*, Technical Report NCAR/TN-210+STR, National Center for Atmospheric Research, Boulder, Colorado, NTIS No. PB83 23106888, 1983.
8. W. Bourke, B. McAvaney, K. Puri, and R. Thurling, "Global modeling of atmospheric flow by spectral methods," in *Methods in Computational Physics,* Vol. 17, 267–324, Academic Press, New York, 1977.
9. A. P. M. Baede, M. Jarraud, and U. Cubasch, *Adiabatic Formulation and Organization of ECMWF's Model*, Technical Report 15, ECMWF, Reading, U.K., 1979.
10. R. K. Sato, L. M. Bath, D. L. Williamson, and G. S. Williamson, *User's Guide to NCAR CCM0B*, Technical Report NCAR/TN-211+IA, National Center for Atmospheric Research, Boulder, Colorado, 1983.
11. D. L. Williamson, L. M. Bath, R. K. Sato, T. A. Mayer, and M. L. Kuhn, *Documentation of NCAR CCM0B Program Modules*, Technical Report NCAR/TN-212+IA, National Center for Atmospheric Research, Boulder, Colorado, NTIS No. PB83 263996, 1983.
12. J. Rosinski, *The General Purpose Timing Library* (www.burningserver.net/rosinski/gptl).
13. F. Wagner et al., *Phys. Rev. Lett.* 49, 1408 (1982).
14. C.S. Chang, S. Ku, P. Diamond, et al., *Phys. Plasmas,* accepted for publication (2009).
15. C. S. Chang and S. Ku, *Phys. Plasmas* 11, 2649 (2004); *Contrib. Plasma Phys.* 46, 496 (2006).
16. D. McCune, *PSPLINE: Princeton Spline and Hermite Cubic Interpolation Routines*, Princeton Plasma Physics Laboratory (http://w3.pppl.gov/ntcc/PSPLINE/).
17. B. Smith et al., *PETSc: Portable, Extensible Toolkit for Scientific Computation*, Argonne National Laboratory (http://www.mcs.anl.gov/petsc/petsc-as/).
18. A. M. Dimits, G. Bateman, M. A. Beer, B. I. Cohen, W. Dorland, G. W. Hammett, C. Kim, J. E. Kinsey, M. Kotschenreuther, A. H. Kritz, L. L. Lao, J. Mandrekas, W. M. Nevins, S. E. Parker, A. J. Redd, D. E.Shumaker, R. Sydora, and J. Weiland, *Phys. Plasmas* 7, 969 (2000).
19. G. Dif-pradalier, V. Grandgirard, Y. Sarazin, X. Garbet, Ph. Ghendrih, and P. Angelino, *Phys. Plasma* 15, 042314 (2008).
20. X. Garbet, Y. Sarazin, F. Imbeaux, P. Ghendrih, O. D. Gurcan C. Bourdelle, and P. H. Diamond, *Phys. Plasmas* 14, 122305 (2007).
21. D. Gurcan, P. H. Diamond, and T. S. Hahm, *Phys. Plasmas* 14, 055902 (2007).
22. T. S. Hahm, P. H. Diamond, and Z. Lin, *Phys. Plasmas* 12, 090903 (2005).
23. R. S. Barlow, *International Workshop on Measurement and Computation of Turbulent Nonpremixed Flames,* Combustion Research Facility, Sandia National Laboratories, www.ca.sandia.gov/TNF, 1996–2009.
24. J. C. Oefelein, "Large eddy simulation of turbulent combustion processes in propulsion and power systems," *Progress in Aerospace Sciences* 42(1), 2–37 (2006).

25. J. C. Oefelein, “Thermophysical characteristics of LOX–H2 flames at supercritical pressure,” *Proceedings of the Combustion Institute* 30, 2929–2937 (2005).
26. J. C. Oefelein, “Mixing and combustion of cryogenic oxygen-hydrogen shear-coaxial jet flames at supercritical pressure,” *Combustion Science and Technology* 178(1–3), 229–252 (2006).
27. J. C. Oefelein, R. W. Schefer, and R. W. Barlow, “Toward validation of LES for turbulent combustion,” *AIAA Journal* 44(3), 418–433 (2006).
28. J. C. Oefelein, V. Sankaran, and T. G. Drozda, “Large eddy simulation of swirling particle-laden flow in a model axisymmetric combustor,” *Proceedings of the Combustion Institute* 31, 2291–2299 (2007).
29. T. G. Drozda and J. C. Oefelein, “Large eddy simulation of direct injection processes for hydrogen and LTC engine applications,” Paper 2008-01-0939, *SAE World Congress,* Detroit, Michigan, April 14–17, 2008.
30. H. Childs, M. Duchanieau, and K-L. Ma, “A Scalable, Hybrid Scheme for Volume Rendering Massive Data Sets,” pp. 153–162 in *Proceedings of Eurographics Symposium on Parallel Graphics and Visualization*, May 2006.

APPENDIXES: BENCHMARK PROBLEM ENVIRONMENTS

APPENDIX A. OVERVIEW

We present in this appendix detailed information about the build and run time environments for the various benchmarks executed in Q2 on the Cray XT5 system at ORNL's NCCS. An example follows where the source code is presented as well as the build and execution process invoked to execute instrumented (direct or automated) code on the target machine.

A.1 PARALLEL MATRIX MULTIPLY EXAMPLE

The acceptability of computed results is defined by the problem. In ASCR's Joule software exercises, the complexity of executing a problem is directly deduced according to machine events measured with supported system software on the target platform. The number of total instructions retired, the number of floating point instructions executed, the total number of processes (assuming a one-to-one relationship between processes and processor cores—not the case for various thread models), and the total execution time are the events we typically return in this report.

The target architecture has a well-developed set of tools designed for tracing and sampling analysis of a variety of machine events of interest. The vendor tools have the capability to instrument a compiled binary via recompiling, and to postanalyze the performance data captured during execution (CrayPAT, Apprentice2 tools). The degree of granularity can be controlled by the user and ranges from exhaustive fine-grain tracing (which can introduce large wall-clock time overhead) to a small set of hardware events that introduce only noise in the execution time. Alternatively, the PAPI tool can be used to instrument the application source code (C, Fortran API exists) and enables the user to declare the events of interest and pinpoint specified regions of their codes. The consistency of the approaches has been checked for a handful of scenarios on the benchmarked platform with exceptional agreement on common test problems prior to the Q2 benchmarks.

A detailed example may help here. Suppose our application problem is to have a computer program that executes (on Jaguar/XT5) the common math operation $C \leftarrow \alpha AB + \beta C$ where A,B,C are all rank two arrays of double precision, complex numbers with dimensions $A \in [m,n]$, $B \in [n,p]$, $C \in [m,p]$, and α,β are double precision, complex numbers. The problem, $P(m,n\ p)$, has complexity that is well described by the storage demands, $mn + np + mp + 2$ complex numbers, and floating point operation count, $P(m,n,p) \sim 8mpn + 13mp$. This problem's complexity (like all program instances) can be calibrated with machine capabilities (even if we did not have a theoretical estimate) by counting the instructions and specifically floating point instructions completed to execute an instance on Jaguar/XT5.

To further simplify, let $m = n = p$. In this case the theoretical complexity of $P(n)$ is $\sim 3n^2 + 2$ complex numbers and $\sim 8n^3 + 13n^2$ floating point operations. (In the real number case, the problem floating point complexity for $m = n = p$ is $P(n) \sim 2n^3 + 2n^2$ and the storage becomes $3n^2 + 2$ real numbers.)

For now, let's check the quality of the counts returned by the approaches on the target hardware. First, consider $P(n = 16{,}384)$. The theoretical complexity of this instance (within the significant digits offered by a handheld calculator) is $P(16{,}384) = 35{,}187{,}861{,}750{,}000$ floating point operations, the PAPI tool measured 36,560,640,672,864 floating point instructions as accumulated over 56 processes given the parallel implementation of the kernel. The relative difference is 3.9%. For the exact same problem parameters, the CrayPAT tool was used to automatically instrument the binary. Two different compilations were used for instrumentation with increased granularity. As an example of a low overhead glimpse into what happened, a floating point instruction count per process was sampled for a fraction of the processes from which we deduced the total floating point instruction count to be 36,563,861,900,000—the relative difference computed against theory is here 3.91%. Since we do not investigate how the chipset actually computes the various complex algebraic terms in the implementation, the agreement to theory is very good.

Let us also report one other consistency check. Here we wish to understand if increasing the number of processes we throw at a fixed problem instance introduces a large error into the machine event data collection process. To this end, the kernel $P(24{,}576)$ is executed first on 120 PEs and next on 256 PEs of the target system. The theoretical complexity for the complex representation is $P(24{,}576)$ = 113,555,757,096,871 floating point operations. The measured (PAPI in this example) count on 120 PEs was 123,390,048,340,380 floating point operations in 165.22 seconds, yielding a relative difference of 8.6% from the complexity model. The measured count on 256 PEs was 123,390,048,343,296 floating point operations in 81.33 seconds, yielding a relative difference of 8.6% from the complexity model. The results are essentially identical. As a last note, the speedup between runs was 2.03; ideally this number would be 2.133. The following is the source code used in the example.

```
/* rochekj@ornl.gov */
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <complex.h>
#include <mpi.h>
#ifdef KRP
#include <papi.h>

/*
PAPI
  _TOT_IIS
  _TOT_INS
  _INT_INS
  _FP_INS
  _FMA_INS
  _VEC_INS
  _L2_DCM
*/

#define NUM_PAPI_EVENTS 4
#endif

/* BLACS , ScaLAPACK */
/*
void Cblacs_pinfo( int * , int * ) ;
void Cblacs_setup( int * , int * ) ;
void Cblacs_get( int , int , int * ) ;
void Cblacs_gridinit( int * , char * , int , int ) ;
void Cblacs_gridinfo( int , int * , int * , int * , int * ) ;
void Cblacs_exit( int ) ;
void Cdgesd2d( int , int , int , double * , int , int , int ) ;
void Cdgerv2d( int , int , int , double * , int , int , int ) ;
void Cigebs2d( int ictxt , char * scope , char * top , int m , int n , int
   * A , int lda ) ;
void Cigebr2d( int ictxt , char * scope , char * top , int m , int n , int
   * A , int lda , int rsrc , int csrc ) ;
void pzgemm_( char * , char * , int * , int * , int * , double complex * ,
   double complex * , int * , int * , int * , double complex *
, int * , int * , int * , double complex * , double complex * , int * ,
   int * , int * ) ;
*/
```

```
/* home spun support routines */
void get_num_rows( int iamprow , int nprows , int ma , int mblk , int *
   nrow )
{
  int mydist , nrows , np ;
  int srcproc , extrarows ;
  srcproc = 0 ; /* assume that the process(0,0) owns the first element(s)
   */
  mydist  = ( nprows + iamprow - srcproc ) % nprows ;
  nrows = ma / mblk ;
  np = ( nrows / nprows ) * mblk ;
  extrarows = nrows % nprows ;
  if ( mydist < extrarows ) np += mblk ;
  else if ( mydist == extrarows ) np += ma % mblk ;
  *nrow = np ;
}

void get_num_columns( int iampcol , int npcols , int na , int nblk , int *
   ncol )
{
  int mydist , ncols , np ;
  int srcproc , extracols ;
  srcproc = 0 ; /* assume that the process(0,0) owns the first element(s)
   */

  mydist  = ( npcols + iampcol - srcproc ) % npcols ;
  ncols = na / nblk ;
  np = ( ncols / npcols ) * nblk ;
  extracols = ncols % npcols ;
  if ( mydist < extracols ) np += nblk ;
  else if ( mydist == extracols ) np += na % nblk ;
  *ncol = np ;
}

void get_mem_req_blk_cyc ( int ip , int iq , int np , int nq , int ma ,
   int na , int mblk , int nblk , int * nip , int * niq )
{
  get_num_rows( ip , np , ma , mblk , nip ) ;
  get_num_columns( iq , nq , na , nblk , niq ) ;
}

void get_mem_req_blk_cyc ( int ip , int iq , int np , int nq , int ma ,
   int na , int mblk , int nblk , int * nip , int * niq ) ;
int main( int argc , char ** argv )
{
#ifdef KRP
  /* PAPI */
  int hw_counters ;
  const PAPI_hw_info_t *hwinfo = NULL ;
  int papi_events[ NUM_PAPI_EVENTS ] ;
  long long int papi_values[ NUM_PAPI_EVENTS + 4 ] ;
  long long int papi_real_cyc_0 , papi_virt_cyc_0 , papi_real_usec_0 ,
   papi_virt_usec_0 ;
```

```
  char * papi_event_name[] = { "PAPI_TOT_INS" , "PAPI_FP_INS" ,
   "PAPI_FP_OPS" , "PAPI_L2_DCM" } ;
  long long int * llbuf , llval ;
#endif

  /* for the kernel */
  int i , j ;
  double complex * a , * b , * c ;
  double complex zone = 1. + I * 1. ; double complex zmone = -1. + I * 1.
   ;
  int np , p , q ; /* np ~ p q , np := number of processes , p := number
   of process rows , q := number of process columns */
  int ip , iq , nip , niq ; /* id (ip,iq) in (p,q) rectangular , virtual
   process grid owns (nip,niq) elements */
  int ma , na , nb , nblk ; /* matrix dimensions [ma,na][na,nb]+[ma,nb] ,
   block size */
  int iam , npmpi ;
  int DESCA[ 9 ] , DESCB[ 9 ] , DESCC[ 9 ]; /* array descriptors */
  int info , doneflag ;
  char *b_order, *scope ;
  int b_val ;
  int ione = 1 , mone = -1 , zero = 0 ;
  int iam_blacs , ictxt , nprocs_blacs ;
#ifdef VERBOSE
  int namelen ;
  char myname[ MPI_MAX_PROCESSOR_NAME ] ;
#endif

  /* blacs */
  b_val = zero ;
  b_order = "R" ;
  scope = "All" ;

  /* initialize the MPI communicator MPI_COMM_WORLD */
  MPI_Init( &argc , &argv ) ;
  MPI_Comm_size( MPI_COMM_WORLD , &npmpi ) ;
  MPI_Comm_rank( MPI_COMM_WORLD , &iam ) ;

  /* parse the command line */
  if ( argc != 7 ) {
    printf( "usage: %s ma na nb nblk p q\n" , argv[ 0 ] ) ;
    MPI_Finalize();
    return ( EXIT_SUCCESS ) ;
  }

  ma = atoi( argv[ 1 ] ) ; /* problem size data */
  na = atoi( argv[ 2 ] ) ; /* problem size data */
  nb = atoi( argv[ 3 ] ) ; /* problem size data */
  nblk = atoi( argv[ 4 ] ) ; /* block buffer data */
  p = atoi( argv[ 5 ] ) ;
  q = atoi( argv[ 6 ] ) ;
  np = p * q ;

  /* initialize the BLACS grid */
```

```
#ifdef VERBOSE
  printf( "[%d] prior to blacs_pinfo\n" , iam ) ;
#endif

  Cblacs_pinfo( &iam_blacs , &nprocs_blacs ) ;
  if ( nprocs_blacs < 1 ) Cblacs_setup( &iam_blacs , &nprocs_blacs ) ;
  Cblacs_get( mone , zero , &ictxt ) ;
  Cblacs_gridinit( &ictxt , b_order , p , q ) ;  /* 'Row-Major' */
  Cblacs_gridinfo( ictxt , &p , &q , &ip , &iq ) ; /* ip,iq: the process
   row,column id */

  /* determine memory demands for the matrix A[ma,na] */
  get_mem_req_blk_cyc ( ip , iq , p , q , ma , na , nblk , nblk , &nip ,
   &niq ) ;

#ifdef VERBOSE
  printf( "A\t[ip=%d , iq=%d] := elements := %d\t BYTES := %llu\n" , ip ,
   iq , nip * niq , ( unsigned long long ) sizeof( double comple
x ) * nip * niq ) ;
#endif

  if ( ( a = malloc( sizeof( double complex ) * nip * niq ) ) == NULL )
    {

      fprintf( stderr , "[%d,%d]error: cannot malloc() a\n...exiting\n",
   ip , iq ) ;
      MPI_Finalize() ;
      return( EXIT_SUCCESS ) ;
    }

  /* generate the matrix elements randomly */
  srand( iam + 1 ) ; /* seed the prng the for initial use */
  for ( i = 0 ; i < nip * niq ; i++ )
      a[ i ] = 0.5 - ( double ) rand() / ( double ) RAND_MAX + I * ( 0.5 -
   ( double ) rand() / ( double ) RAND_MAX ) ;

  /* the array descriptor for local_A */
  DESCA[ 0 ] = 1 ; /* descriptor type (1=global) */
  DESCA[ 1 ] = ictxt ; /* blacs process grid used for distribution */
  DESCA[ 2 ] = ma ; /* rows in global A */
  DESCA[ 3 ] = na ; /* columns in global A */
  DESCA[ 4 ] = nblk ; /* row block factor */
  DESCA[ 5 ] = nblk ; /* column block factor */
  DESCA[ 6 ] = 0 ; /* row source in the pgrid */
  DESCA[ 7 ] = 0 ; /* column source in the pgrid */
  DESCA[ 8 ] = nip ; /* local leading dimension of A */

  /* determine memory demands for the matrix B[na,nb] */
  get_mem_req_blk_cyc ( ip , iq , p , q , na , nb , nblk , nblk , &nip ,
   &niq ) ;

#ifdef VERBOSE
  printf( "B\t[ip=%d , iq=%d] := elements := %d\t BYTES := %llu\n" , ip ,
   iq , nip * niq , ( unsigned long long ) sizeof( double comple
```

```
x ) * nip * niq ) ;
#endif

  if ( ( b = malloc( sizeof( double complex ) * nip * niq ) ) == NULL )
    {
      fprintf( stderr , "[%d,%d]error: cannot malloc() b\n...exiting\n",
   ip , iq ) ;
      MPI_Finalize() ;
      return( EXIT_SUCCESS ) ;
    }

  for ( i = 0 ; i < nip * niq ; i++ ) b[ i ] = 0.5 - ( double ) rand() / (
   double ) RAND_MAX + I * ( 0.5 - ( double ) rand() / ( double
 ) RAND_MAX ) ;

  /* the array descriptor for local_B */
  DESCB[ 0 ] = 1 ; /* descriptor type (1=global) */
  DESCB[ 1 ] = ictxt ; /* blacs process grid used for distribution */
  DESCB[ 2 ] = na ; /* rows in global B */
  DESCB[ 3 ] = nb ; /* columns in global B */
  DESCB[ 4 ] = nblk ; /* row block factor */
  DESCB[ 5 ] = nblk ; /* column block factor */
  DESCB[ 6 ] = 0 ; /* row source in the pgrid */
  DESCB[ 7 ] = 0 ; /* column source in the pgrid */
  DESCB[ 8 ] = nip ; /* local leading dimension of B */

  /* determine memory demands for the matrix C[ma,nb] */
  get_mem_req_blk_cyc ( ip , iq , p , q , ma , nb , nblk , nblk , &nip ,
   &niq ) ;

#ifdef VERBOSE
  printf( "C\t[ip=%d , iq=%d] := elements := %d\t BYTES := %llu\n" , ip ,
   iq , nip * niq , ( unsigned long long ) sizeof( double comple
x ) * nip * niq ) ;
#endif

  if ( ( c = malloc( sizeof( double complex ) * nip * niq ) ) == NULL ) {
    fprintf( stderr , "[%d,%d]error: cannot malloc() c\n...exiting\n", ip
   , iq ) ;
    MPI_Finalize() ;
    return( EXIT_SUCCESS ) ;
  }

  for ( i = 0 ; i < nip * niq ; i++ ) c[ i ] = 0.5 - ( double ) rand() / (
   double ) RAND_MAX + I * ( 0.5 - ( double ) rand() / ( double
 ) RAND_MAX ) ;
  /* the array descriptor for local_C */
  DESCC[ 0 ] = 1 ; /* descriptor type (1=global) */
  DESCC[ 1 ] = ictxt ; /* blacs process grid used for distribution */
  DESCC[ 2 ] = ma ; /* rows in global C */
  DESCC[ 3 ] = nb ; /* columns in global C */
  DESCC[ 4 ] = nblk ; /* row block factor */
  DESCC[ 5 ] = nblk ; /* column block factor */
  DESCC[ 6 ] = 0 ; /* row source in the pgrid */
```

```
  DESCC[ 7 ] = 0 ; /* column source in the pgrid */
  DESCC[ 8 ] = nip ; /* local leading dimension of C */

#ifdef VERBOSE
  MPI_Get_processor_name( myname , &namelen ) ;
  printf( "[ %d , %d ][%s] prior to pgemm \n" , ip , iq , myname ) ;
#endif

#ifdef VERBOSE
  if ( ip == 0 ) {
    printf( "a[1,1]= ( %f , %f )\n" , creal( a[ 0 ] ) , cimag( a[ 0 ] ) )
   ;
    printf( "b[1,1]= ( %f , %f )\n" , creal( b[ 0 ] ) , cimag( b[ 0 ] ) )
   ;
    printf( "c[1,1]= ( %f , %f )\n" , creal( c[ 0 ] ) , cimag( c[ 0 ] ) )
   ;
  }

#endif
  MPI_Barrier(MPI_COMM_WORLD);
#ifdef KRP

  /* learn something about the system here */
  if ( PAPI_library_init( PAPI_VER_CURRENT ) != PAPI_VER_CURRENT )
    exit( 1 ) ;
  if ( ( hwinfo = PAPI_get_hardware_info() ) == NULL )
    exit( 1 ) ;
  if ( iam == 0 )
    {
      printf( "\t\tTotPEs(jagpf)[%d]\n" , hwinfo->totalcpus ) ;
      printf( "\t\tMhz[%g]\n" , hwinfo->mhz ) ;
      printf( "\t\tnCPU-SMPnode(jagpf)[%d]\n" , hwinfo->ncpu ); /* Number
   of CPU's in SMP Node */
      printf( "\t\tnSMPnodes(jagpf)[%d]\n" , hwinfo->nnodes );
      printf( "\t\t\tvendor string cpu[%s}\n" , hwinfo->vendor_string );
      printf( "\t\t\tmodel string cpu[%s}\n" , hwinfo->model_string );
      printf( "\t\t\tmodel number[%d]\n\n" , hwinfo->model );
    }

  char * eventname[] = { PAPI_FP_OPS , PAPI_FP_INS } ;
  int eventcode ;
  PAPI_event_info_t pinfo ;
  if ( ip == 0 )
    {
      for ( i = 0 ; i < 2 ; i++ )
        {
          PAPI_event_name_to_code( eventname[ i ] , &eventcode ) ;
          if ( PAPI_get_event_info( eventcode , &pinfo ) != PAPI_OK )
            {
              fprintf( stderr , "error: papi event[%s]\n" , eventcode[ i ]
   ) ;
            }
          printf( "papi event[%s]\n" , papi_event_name[ i ] ) ;
        }
```

```
    }
#endif

  /* begin PAPI profiling here */
  hw_counters = PAPI_num_counters() ;
  for ( i = 0 ; i < ( int ) NUM_PAPI_EVENTS ; i++ )
    {
      if ( PAPI_event_name_to_code( papi_event_name[ i ] , &papi_events[ i
   ] ) != PAPI_OK )
        {
          fprintf( stderr , "papi error[%s]\n" , papi_event_name[ i ] ) ;
          if ( hw_counters > i ) hw_counters = i ;
        }
    }
  if( hw_counters > NUM_PAPI_EVENTS ) hw_counters = NUM_PAPI_EVENTS ;
  papi_real_cyc_0 = PAPI_get_real_cyc() ;
  papi_real_usec_0 = PAPI_get_real_usec() ;
  papi_virt_cyc_0 = PAPI_get_virt_cyc() ;
  papi_virt_usec_0 = PAPI_get_virt_usec() ;
  PAPI_start_counters( papi_events , hw_counters ) ;
#endif

  /*
    extern void p*gemm_( char *TRANSA, char *TRANSB, int * M, int * N, int
   * K, double * ALPHA,
    double * A, int * IA, int * JA, int * DESCA, double * B, int * IB, int
   * JB, int * DESCB,
    double * BETA, double * C, int * IC, int * JC, int * DESCC );
  */
  pzgemm_( "N" , "N" , &ma , &nb , &na , &zone , a , &ione , &ione , DESCA
   , b , &ione , &ione , DESCB , &zmone , c , &ione , &ione , D
ESCC ) ;
#ifdef VERBOSE
  printf( "[ %d , %d ] return from pgemm \n" , ip , iq ) ;
  if ( ip == 0 ) {
    printf( "a[1,1]= ( %f , %f )\n" , creal( a[ 0 ] ) , cimag( a[ 0 ] ) )
   ;
    printf( "b[1,1]= ( %f , %f )\n" , creal( b[ 0 ] ) , cimag( b[ 0 ] ) )
   ;
    printf( "c[1,1]= ( %f , %f )\n" , creal( c[ 0 ] ) , cimag( c[ 0 ] ) )
   ;
  }
#endif
  free ( a ) ; free( b ) ; free( c ) ;
#ifdef KRP

  /* PAPI exit results */
  PAPI_stop_counters( papi_values , hw_counters ) ;
  papi_values[ hw_counters ] = PAPI_get_real_cyc() - papi_real_cyc_0 ;
  papi_values[ hw_counters + 1 ] = PAPI_get_real_usec() - papi_real_usec_0
   ;
  papi_values[ hw_counters + 2 ] = PAPI_get_virt_cyc() - papi_virt_cyc_0 ;
  papi_values[ hw_counters + 3 ] = PAPI_get_virt_usec() - papi_virt_usec_0
   ;
```

```
  if ( iam == 0 )
    {
      if ( ( llbuf = malloc( sizeof( long long int ) * npmpi ) ) == NULL )
        {
          fprintf( stderr , "[%d,%d]error: cannot malloc()
   llbuf\n...exiting\n", ip , iq ) ;
          MPI_Finalize() ;
          return( EXIT_SUCCESS ) ;
        }
    }
  for ( i = 0 ; i < hw_counters ; i++ )
    {
      llval = 0LL ;
      MPI_Gather( &papi_values[ i ] , 1 , MPI_LONG_LONG , llbuf , 1 ,
   MPI_LONG_LONG , 0 , MPI_COMM_WORLD ) ;
      if ( iam == 0 )
        { /* report some profile information */
          for ( j = 0 ; j < npmpi ; j++ )
            llval += llbuf[ j ] ;
          printf("%s :\tTot[ %lld ]\tRt[ %lld ]\n", papi_event_name[ i ] ,
   llval , papi_values[ i ] ) ;
        }
      MPI_Barrier( MPI_COMM_WORLD ) ;
    }
  if ( iam == 0 )
    {
      printf( "PAPI_real_cyc = %lld\n" , papi_values[ hw_counters ] ) ;
      printf( "PAPI_real_usec = %lld\n" , papi_values[ hw_counters + 1 ] )
   ;
      printf( "PAPI_user_cyc = %lld\n" , papi_values[ hw_counters + 2 ] )
   ;
      printf( "PAPI_user_usec = %lld\n" , papi_values[ hw_counters + 3 ] )
   ;
      free ( llbuf ) ;
    }
#endif
  MPI_Barrier( MPI_COMM_WORLD ) ;
#ifdef VERBOSE
  printf( "...[%d][%d]clean exit\n" , ip , iq ) ;
#endif

  MPI_Finalize( ) ;
  return ( EXIT_SUCCESS ) ;
}
```

A.2 MODULES AVAILABLE ON THE TARGET ARCHITECTURE

```
--------------------- /opt/cray/xt-asyncpe/2.0/modulefiles --------------
xtpe-quadcore      xtpe-target-native
------------------------------- /opt/modulefiles ------------------------
Base-opts/2.1.27HD
Base-opts/2.1.27HD.lusrelsave
```

```
Base-opts/2.1.29HD
Base-opts/2.1.29HD.lusrelsave
Base-opts/2.1.41HD
Base-opts/2.1.41HD.lusrelsave
Base-opts/2.1.50HD(default)
Base-opts/2.1.50HD.lusrelsave
MySQL/5.0.45
PrgEnv-cray/1.0.0(default)
PrgEnv-gnu/2.1.27HD
PrgEnv-gnu/2.1.29HD
PrgEnv-gnu/2.1.41HD
PrgEnv-gnu/2.1.50HD(default)
PrgEnv-pathscale/2.1.27HD
PrgEnv-pathscale/2.1.29HD
PrgEnv-pathscale/2.1.41HD
PrgEnv-pathscale/2.1.50HD(default)
PrgEnv-pgi/2.1.27HD
PrgEnv-pgi/2.1.29HD
PrgEnv-pgi/2.1.41HD
PrgEnv-pgi/2.1.50HD(default)
acml/4.0.1a
acml/4.1.0(default)
acml/4.2.0
apprentice2/4.3.0
apprentice2/4.4.0(default)
apprentice2/4.4.0.1
blcr/0.7.3
cce/7.0.0(default)
cce/7.0.1
cce/7.0.2
cray/audit/1.0.0-1.0000.15784.0
dwarf/8.2.0
dwarf/8.4.0
dwarf/8.6.0
dwarf/8.8.0(default)
elf/0.8.10(default)
fftw/2.1.5
fftw/3.1.1(default)
fftw/3.2.0
gcc/4.1.2
gcc/4.2.0.quadcore(default)
gcc/4.2.3
gcc/4.2.4
gcc-catamount/3.3
gnet/2.0.5
iobuf/1.0.6(default)
java/jdk1.6.0_05(default)
java/jdk1.6.0_11
libfast/1.0(default)
libfast/1.0.2
libscifft-pgi/1.0.0(default)
moab/5.2.3
moab/5.2.4(default)
moab/5.3.0
```

```
modules/3.1.6(default)
pathscale/3.2(default)
petsc/2.3.3a(default)
petsc/3.0.0
petsc-complex/2.3.3a(default)
petsc-complex/3.0.0
pgi/6.2.5
pgi/7.0.7
pgi/7.1.6
pgi/7.2.3
pgi/7.2.4
pgi/7.2.5(default)
pgi/8.0.1
pgi/8.0.2
pgi/8.0.3
pkgconfig/0.15.0(default)
torque/2.3.2-snap.200807092141(default)
xt-asyncpe/1.0c
xt-asyncpe/1.1
xt-asyncpe/1.2
xt-asyncpe/2.0(default)
xt-asyncpe/2.0.34
xt-asyncpe/2.1
xt-boot/2.1.27HD
xt-boot/2.1.29HD
xt-boot/2.1.41HD
xt-boot/2.1.50HD
xt-catamount/2.1.27HD
xt-catamount/2.1.29HD
xt-catamount/2.1.41HD
xt-catamount/2.1.50HD
xt-craypat/4.3.1
xt-craypat/4.3.3
xt-craypat/4.4.0
xt-craypat/4.4.0.2
xt-craypat/4.4.0.4(default)
xt-craypat/4.4.1
xt-libc/2.1.27HD
xt-libc/2.1.29HD
xt-libc/2.1.41HD
xt-libc/2.1.50HD
xt-libsci/10.2.1
xt-libsci/10.3.0
xt-libsci/10.3.1(default)
xt-libsci/10.3.2
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_1.6.5
xt-lustre-ss/2.1.50HD_PS04_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-mpt/2.1.27HD
xt-mpt/2.1.29HD
xt-mpt/2.1.41HD
xt-mpt/2.1.50HD
xt-mpt/3.0.1
xt-mpt/3.0.2
xt-mpt/3.0.4
xt-mpt/3.1.0(default)
xt-mpt/3.1.0.4
xt-mpt/3.1.0.6
xt-mpt/3.1.0.7
xt-mpt/3.1.1
xt-os/2.1.27HD
xt-os/2.1.29HD
xt-os/2.1.41HD
xt-os/2.1.50HD
xt-papi/3.5.99c
xt-papi/3.6
xt-papi/3.6.1a
xt-papi/3.6.2(default)
xt-pe/2.1.27HD
xt-pe/2.1.29HD
xt-pe/2.1.41HD
xt-pe/2.1.50HD
xt-service/2.1.27HD
xt-service/2.1.29HD
xt-service/2.1.41HD
xt-service/2.1.50HD
xtgdb/1.0.0(default)
xtpe-target-catamount
xtpe-target-cnl

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

----------------------------- /sw/xt5/modulefiles --------------------
DefApps                     lapack/3.1.1-dualcore
MiscApps                    lapack/3.1.1-fPIC
adios/0.9.8(default)        liblut/0.9.6
arpack/2008.03.11           m4/1.4.11
atlas/3.8.2                 matlab/7.5
```

```
atlas/3.8.2-fPIC-dualcore  mercurial/1.0.2
autoconf/2.63              metis/4.0
automake/1.10.1            mpe2/1.0.6
aztec/2.1                  mpip/3.1.2
blas/ref(default)          mumps/4.7.3_par
blas/ref-dualcore          namd/2.6
bugget/2.0                 ncl/5.0.0
cmake/2.6.1(default)       nco/3.9.4
cmake/2.6.2                ncview/1.93c
cpmd/3.13.1                nedit/5.5
cpmd/3.13.2                netcdf/3.6.2(default)
doxygen/1.5.6              netcdf/4.0.0
doxygen/1.5.8              netcdf/4.0.0_par
ferret/6.1                 ompi/ADTR65
fftpack/5-r4i4             ompi/ADTR77
fftpack/5-r8i4             ompi/ADTR78
fftpack/5-r8i8             ompi/DTR56
fftw/3.1.2                 ompi/DTR59
fftw/3.1.2-dualcore        ompi/routing-pgi
fftw/3.2                   p-netcdf/1.0.2(default)
fftw/3.2-dualcore          p-netcdf/1.0.3
fpmpi/1.0                  parmetis/3.1
fpmpi/1.1                  petsc/2.3.3-debug
fpmpi_papi/1.0             petsc-complex/2.3.3-debug
fpmpi_papi/1.1             pgplot/5.2
gamess/2008Mar04           pspline/1.0
git/1.6.0                  python/2.5.2
git/1.6.0.4                python/2.5.2-netcdf
globalarrays/4.0.8         qt/4.3.4
gnuplot/4.2.3              ruby/1.8.7
gnuplot/4.2.4(default)     ruby/1.9.1
gptl/3.4.1                 spdcp/0.3.6
gptl/3.4.3                 sprng/2.0b
gptl/3.4.7(default)        stagesub/1.0.2
grace/5.1.21               stagesub/1.0.3(default)
gromacs/3.3.3              subversion/1.4.6
gsl/1.11                   subversion/1.5.0(default)
gsl/1.11-dualcore          sundials/2.3.0
hdf5/1.6.7(default)        superlu/3.0
hdf5/1.6.7_par             superlu_dist/2.2
hdf5/1.6.8                 swig/1.3.36
hdf5/1.6.8_par             szip/2.1
hdf5/1.8.1                 tau/2.17.2
hdf5/1.8.1_par             tau/2.17.3
hdf5/1.8.2                 tkdiff/4.1.4
hdf5/1.8.2_par             totalview/8.6.0-1(default)
hypre/2.0.0                trilinos/8.0.3
idl/6.4                    udunits/1.12.4
imagemagick/6.4.2(default) udunits/1.12.9
java-jdk/1.5.0.06          umfpack/5.1.1
java-jdk/1.6.0.06          valgrind/3.3.1
java-jre/1.5.0.06          vim/7.1
lammps/4Mar08              vim/7.2
lammps/May08               visit/1.11.1
```

```
lapack/3.1.1(default)
```

A.3 COMPILATION FOR INSTRUMENTATION AND EXECUTION

PAPI instrumented case (example)

- Prepare the environment

```
module load xt-papi .
jaguarpf-login2 roche/chk-perf> env | grep PAPI
PE_PRODUCT_LIST=ASYNCPE:XTMPT:LIBSCI:PGI:LUSTRE:XTPE_QUADCORE:FFTW:PAPI
PAPI_POST_LINK_OPTS= -L/opt/xt-tools/papi/3.6.2/v23/linux/lib -lpapi -lpfm
PAPI_INCLUDE_OPTS=-I/opt/xt-
   tools/papi/3.6.2/v23/$XTPE_COMPILE_TARGET/include
PAPI_VERSION=3.6.2
```

- Compile the code

```
cc -c -DKRP ${PAPI_INCLUDE_OPTS} kr-cpblas-tst.c ; cc -o xcpbls kr-
   cpblastst.o ${PAPI_POST_LINK_OPTS} –lsci -lm
/opt/cray/xt-asyncpe/2.0/bin/cc: INFO: linux target is being used
/opt/cray/xt-asyncpe/2.0/bin/cc: INFO: linux target is being used
```

- The run script

```
#PBS -V
#PBS -l walltime=00:30:00,size=256
#PBS -A csc053
#PBS -N cpbls
#PBS -j oe

cd ${PBS_O_WORKDIR}
aprun -n 256 ./xcpbls 24576 24576 24576 80 16 16
```

- The output of the example

```
TotPEs(jagpf)[8]
Mhz[2300]
nCPU-SMPnode(jagpf)[8]
nSMPnodes(jagpf)[1]
vendor string cpu[AuthenticAMD}
model string cpu[Quad-Core AMD Opteron(tm) Processor 23 (B3)}
model number[16]

PAPI_TOT_INS :  Tot[ 111540195796692 ]  Rt[ 449702700723 ]
PAPI_FP_INS :   Tot[ 123390048343296 ]  Rt[ 522995200021 ]
PAPI_FP_OPS :   Tot[ 123390048343296 ]  Rt[ 522995200021 ]
PAPI_L2_DCM :   Tot[ 84401033569 ]      Rt[ 248334396 ]
PAPI_real_cyc = 187063030672
PAPI_real_usec = 81331753
PAPI_user_cyc = 187036000000
PAPI_user_usec = 81320000
Application 107259 resources: utime 16225, stime 129
```

Automatically instrumented case (example)

- Prepare the environment

```
module load xt-craypat .
```

- Check the modules

```
module list
Currently Loaded Modulefiles:
1) modules/3.1.6                        4) moab/5.2.4
7) xt-service/2.1.50HD                 10) xt-boot/2.1.50HD
13) Base-opts/2.1.50HD                  16) xt-libsci/10.3.1
19) xt-asyncpe/2.0
2) DefApps                              5) xtpe-quadcore
8) xt-libc/2.1.50HD                    11) xt-lustre-ss/2.1.50HD_PS04_1.6.5
14) pgi/7.2.5                           17) xt-mpt/3.1.0
20) PrgEnv-pgi/2.1.50HD
3) torque/2.3.2-snap.200807092141       6) MySQL/5.0.45
9) xt-os/2.1.50HD                      12) xtpe-target-cnl
15) fftw/3.1.1                          18) xt-pe/2.1.50HD
21) xt-craypat/4.4.0.4
```

- Compile the code

```
cc -c kr-cpblas-tst.c ; cc -o xcpbls-cp kr-cpblas-tst.o -lsci -lm
/opt/cray/xt-asyncpe/2.0/bin/cc: INFO: linux target is being used
/opt/cray/xt-asyncpe/2.0/bin/cc: INFO: linux target is being used
```

- Build the instrumented binary

```
pat_build -Drtenv=PAT_RT_HWPC=0 -g mpi -w –o xcpbls-cp+apa ./xcpbls-cp
```

- The run script

```
#PBS -V
#PBS -l walltime=00:30:00,size=56
#PBS -A csc053
#PBS -N cpbls
#PBS -j oe

cd ${PBS_O_WORKDIR}
aprun -n 56 ./xcpbls-cp+apa 16384 16384 16384 80 7 8
```

- Run the code

```
qsub qscr-joule-apa
```

- Build the automated performance report

```
pat_report -o apa-report.txt
xcpbls-cp+apa+25678-20623tdt.xf
```

- Actual output of the automatically generated performance report

```
CrayPat/X:  Version 4.4.0 Revision 2195 (xf 2119)  10/29/08 14:13:53

 Number of PEs (MPI ranks):      56

 Number of Threads per PE:       1

 Number of Cores per Processor:  4

 Execution start time:   Tue Mar  3 02:31:12 2009

 System type and speed:   x86_64  2300 MHz

 Current path to data file:
   /tmp/work/roche/chk-perf/xcpbls-cp+apa+25678-20623tdt.ap2  (RTS)
   /tmp/work/roche/chk-perf/xcpbls-cp+apa+25678-20623tdt.xf   (RTS)

 Notes for table 1:

   Table option:
     -O profile_pe_th-h
   Options implied by table option:
     -d ti%@0.95,ti,imb_ti,imb_ti%,tr -b gr,fu,pe=HIDE

   Options for related tables not shown by default:
     -O profile_pe.th           -O callers
     -O profile_th_pe           -O callers+src
     -O profile+src             -O calltree
     -O load_balance            -O calltree+src

   The Total value for each of Time, Calls is the sum of the Group values.
   The Group value for each of Time, Calls is the sum of the Function
 values.
   The Function value for each of Time, Calls is the avg of the PE values.
     (To specify different aggregations, see:  pat_help report options s1)

   This table shows only lines with Time% > 0.95.
     (To set thresholds to zero, specify:  -T)

   Percentages at each level are of the Total for the program.
     (For percentages relative to next level up, specify:
       -s percent=r[elative])

 Table 1:  Profile by Function Group and Function (no hwpc)

  Time % |       Time |Imb. Time |   Imb. |  Calls |Group
         |            |          | Time % |        | Function
         |            |          |        |        |  PE='HIDE'

  100.0% | 110.941610 |       -- |     -- | 5109.3 |Total
 |-----------------------------------------------------------------
 |  88.9% |  98.575756 |       -- |     -- |    2.0 |USER
 ||----------------------------------------------------------------
 ||  88.9% |  98.575642 | 4.527489 |   4.5% |    1.0 |main
 ||================================================================
```

```
|  11.1% |  12.313924 |       -- |     -- | 5105.3 |MPI
||-------------------------------------------------------------
||   4.8% |   5.370178 | 1.467647 |  21.9% |    1.0 |MPI_Comm_create
||   3.6% |   3.938757 | 3.694346 |  49.3% |  930.2 |MPI_Recv
||   2.7% |   2.980756 | 3.777156 |  56.9% |  930.2 |MPI_Send
|=============================================================

Notes for table 2:

  Table option:
    -O profile
  Options implied by table option:
    -d ti%@0.95,ti,imb_ti,imb_ti%,tr,P -b gr,fu,pe=HIDE

  Options for related tables not shown by default:
    -O profile_pe.th           -O callers
    -O profile_th_pe           -O callers+src
    -O profile+src             -O calltree
    -O load_balance            -O calltree+src

  The Total value for each data item is the sum of the Group values.
  The Group value for each data item is the sum of the Function values.
  The Function value for each data item is the avg of the PE values.
    (To specify different aggregations, see:  pat_help report options s1)

  'D1 cache utilization (M)' is based on data size 8B, and refills caused
by

    misses.

  This table shows only lines with Time% > 0.95.
    (To set thresholds to zero, specify:  -T)

  Percentages at each level are of the Total for the program.
    (For percentages relative to next level up, specify:
      -s percent=r[elative])

Table 2:  Profile by Function Group and Function

Group / Function / PE='HIDE'

========================================================================
Totals for program
------------------------------------------------------------------------
  Time%                                          100.0%
  Time                                       110.941610 secs
  Imb.Time                                           -- secs
  Imb.Time%                                          --
  Calls                        46.3 /sec        5109.3 calls
  PAPI_L1_DCM                11.761M/sec    1297073220 misses
  PAPI_TOT_INS             5392.251M/sec  594676118362 instr
  PAPI_L1_DCA              2267.168M/sec  250031159666 refs
  PAPI_FP_OPS              5920.436M/sec  652926106258 ops
  User time (approx)        110.283 secs  253651943657 cycles   99.4%Time
```

```
  Average Time per Call                      0.021714 sec
  CrayPat Overhead : Time     0.0%
  HW FP Ops / User time   5920.436M/sec  652926106258 ops  64.4%peak(DP)
  HW FP Ops / WCT         5885.313M/sec
  HW FP Ops / Inst                             109.8%
  Computational intensity     2.57 ops/cycle     2.61 ops/ref
  Instr per cycle                                2.34 inst/cycle
  MIPS                  301966.08M/sec
  MFLOPS (aggregate)    331544.40M/sec
  Instructions per LD & ST   42.0% refs          2.38 inst/ref
  D1 cache hit,miss ratios   99.5% hits          0.5% misses
  D1 cache utilization (M)  192.77 refs/miss   24.096 avg uses
==========================================================================
USER
--------------------------------------------------------------------------
  Time%                                         88.9%
  Time                                       98.575756 secs
  Imb.Time                                          -- secs
  Imb.Time%                                         --
  Calls                        0.0 /sec           2.0 calls
  PAPI_L1_DCM               11.002M/sec    1082508371 misses
  PAPI_TOT_INS            5613.815M/sec  552364281560 instr
  PAPI_L1_DCA             2370.347M/sec  233227344737 refs
  PAPI_FP_OPS             6635.850M/sec  652926105327 ops
  User time (approx)        98.394 secs  226305620820 cycles   99.8%Time
  Average Time per Call                      49.287878 sec
  CrayPat Overhead : Time     0.0%
  HW FP Ops / User time   6635.850M/sec  652926105327 ops  72.1%peak(DP)
  HW FP Ops / WCT         6623.597M/sec
  HW FP Ops / Inst                             118.2%
  Computational intensity     2.89 ops/cycle     2.80 ops/ref
  Instr per cycle                                2.44 inst/cycle
  MIPS                  314373.63M/sec
  MFLOPS (aggregate)    371607.57M/sec
  Instructions per LD & ST   42.2% refs          2.37 inst/ref
  D1 cache hit,miss ratios   99.5% hits          0.5% misses
  D1 cache utilization (M)  215.45 refs/miss   26.931 avg uses
==========================================================================
USER / main
--------------------------------------------------------------------------
  Time%                                         88.9%
  Time                                       98.575642 secs
  Imb.Time                                    4.527489 secs
  Imb.Time%                                       4.6%
  Calls                        0.0 /sec           1.0 calls
  PAPI_L1_DCM               11.002M/sec    1082507665 misses
  PAPI_TOT_INS            5613.824M/sec  552364180392 instr
  PAPI_L1_DCA             2370.351M/sec  233227289198 refs
  PAPI_FP_OPS             6635.862M/sec  652926105327 ops
  User time (approx)        98.394 secs  226305210179 cycles   99.8%Time
  Average Time per Call                      98.575642 sec
  CrayPat Overhead : Time     0.0%
  HW FP Ops / User time   6635.862M/sec  652926105327 ops  72.1%peak(DP)
  HW FP Ops / WCT         6623.605M/sec
```

```
  HW FP Ops / Inst                                118.2%
  Computational intensity     2.89 ops/cycle      2.80 ops/ref
  Instr per cycle                                 2.44 inst/cycle
  MIPS                   314374.14M/sec
  MFLOPS (aggregate)     371608.25M/sec
  Instructions per LD & ST   42.2% refs           2.37 inst/ref
  D1 cache hit,miss ratios   99.5% hits           0.5% misses
  D1 cache utilization (M)  215.45 refs/miss   26.931 avg uses
==========================================================================
MPI
--------------------------------------------------------------------------
  Time%                                          11.1%
  Time                                       12.313924 secs
  Imb.Time                                          -- secs
  Imb.Time%                                         --
  Calls                      431.2 /sec       5105.3 calls
  PAPI_L1_DCM              18.020M/sec    213344554 misses
  PAPI_TOT_INS           3557.995M/sec  42123083365 instr
  PAPI_L1_DCA            1412.942M/sec  16727808051 refs
  PAPI_FP_OPS                   79 /sec      930.161 ops
  User time (approx)       11.839 secs  27229688194 cycles  96.1%Time
  Average Time per Call                     0.002412 sec
  CrayPat Overhead : Time    0.1%
  HW FP Ops / User time         79 /sec      930.161 ops  0.0%peak(DP)
  HW FP Ops / WCT               76 /sec
  HW FP Ops / Inst                                0.0%
  Computational intensity    0.00 ops/cycle       0.00 ops/ref
  Instr per cycle                                 1.55 inst/cycle
  MIPS                   199247.71M/sec
  MFLOPS (aggregate)         0.00M/sec
  Instructions per LD & ST  39.7% refs            2.52 inst/ref
  D1 cache hit,miss ratios  98.7% hits            1.3% misses
  D1 cache utilization (M)  78.41 refs/miss      9.801 avg uses
==========================================================================
MPI / MPI_Comm_create
--------------------------------------------------------------------------
  Time%                                           4.8%
  Time                                        5.370178 secs
  Imb.Time                                    1.467647 secs
  Imb.Time%                                      22.3%
  Calls                        0.2 /sec          1.0 calls
  PAPI_L1_DCM              19.450M/sec    104448581 misses
  PAPI_TOT_INS           3657.179M/sec  19639696959 instr
  PAPI_L1_DCA            1453.390M/sec   7804962080 refs
  PAPI_FP_OPS                                      0 ops
  User time (approx)        5.370 secs  12351406681 cycles  100.0%Time
  Average Time per Call                       5.370178 sec
  CrayPat Overhead : Time    0.0%
  HW FP Ops / User time                            0 ops   0.0%peak(DP)
  HW FP Ops / WCT
  HW FP Ops / Inst                                0.0%
  Computational intensity    0.00 ops/cycle       0.00 ops/ref
  Instr per cycle                                 1.59 inst/cycle
  MIPS                   204802.01M/sec
```

```
  MFLOPS (aggregate)         0.00M/sec
  Instructions per LD & ST  39.7% refs         2.52 inst/ref
  D1 cache hit,miss ratios  98.7% hits         1.3% misses
  D1 cache utilization (M)  74.73 refs/miss   9.341 avg uses
==========================================================================
MPI / MPI_Recv
--------------------------------------------------------------------------
  Time%                                        3.6%
  Time                                     3.938757 secs
  Imb.Time                                 3.694346 secs
  Imb.Time%                                   50.2%
  Calls                      267.3 /sec        930.2 calls
  PAPI_L1_DCM              16.915M/sec     58867999 misses
  PAPI_TOT_INS           3516.575M/sec  12238748109 instr
  PAPI_L1_DCA            1395.848M/sec   4857975146 refs
  PAPI_FP_OPS                                     0 ops
  User time (approx)         3.480 secs   8004697555 cycles  88.4%Time
  Average Time per Call                    0.004234 sec
  CrayPat Overhead : Time    0.0%
  HW FP Ops / User time                           0 ops  0.0%peak(DP)
  HW FP Ops / WCT
  HW FP Ops / Inst                             0.0%
  Computational intensity    0.00 ops/cycle    0.00 ops/ref
  Instr per cycle                              1.53 inst/cycle
  MIPS                  196928.21M/sec
  MFLOPS (aggregate)         0.00M/sec
  Instructions per LD & ST  39.7% refs         2.52 inst/ref
  D1 cache hit,miss ratios  98.8% hits         1.2% misses
  D1 cache utilization (M)  82.52 refs/miss  10.315 avg uses
==========================================================================
MPI / MPI_Send
--------------------------------------------------------------------------
  Time%                                        2.7%
  Time                                     2.980756 secs
  Imb.Time                                 3.777156 secs
  Imb.Time%                                   57.9%
  Calls                      313.2 /sec        930.2 calls
  PAPI_L1_DCM              16.804M/sec     49901153 misses
  PAPI_TOT_INS           3444.843M/sec  10229767713 instr
  PAPI_L1_DCA            1366.745M/sec   4058671524 refs
  PAPI_FP_OPS                 313 /sec      930.161 ops
  User time (approx)         2.970 secs   6830054698 cycles  99.6%Time
  Average Time per Call                    0.003205 sec
  CrayPat Overhead : Time    0.0%
  HW FP Ops / User time       313 /sec      930.161 ops  0.0%peak(DP)
  HW FP Ops / WCT             312 /sec
  HW FP Ops / Inst                             0.0%
  Computational intensity    0.00 ops/cycle    0.00 ops/ref
  Instr per cycle                              1.50 inst/cycle
  MIPS                  192911.20M/sec
  MFLOPS (aggregate)         0.02M/sec
  Instructions per LD & ST  39.7% refs         2.52 inst/ref
  D1 cache hit,miss ratios  98.8% hits         1.2% misses
  D1 cache utilization (M)  81.33 refs/miss  10.167 avg uses
```

```
========================================================================

Notes for table 3:

  Table option:
    -O load_balance_m
  Options implied by table option:
    -d ti%@0.95,ti,Mc,Mm,Mz -b gr,pe=[mmm]

  Options for related tables not shown by default:
    -O load_balance_sm         -O load_balance_cm

  The Total value for each data item is the sum of the Group values.
  The Group value for each data item is the avg of the PE values.
    (To specify different aggregations, see:  pat_help report options s1)

  This table shows only lines with Time% > 0.95.
    (To set thresholds to zero, specify:  -T)

  Percentages at each level are of the Total for the program.
    (For percentages relative to next level up, specify:
      -s percent=r[elative])

Table 3:  Load Balance with MPI Message Stats

 Time % |       Time |   MPI |MPI Msg Bytes |Avg MPI Msg |Group
        |            |   Msg |              |       Size | PE[mmm]
        |            | Count |              |            |

 100.0% | 110.949027 | 930.2 |  997045979.0 | 1071907.21 |Total
|-----------------------------------------------------------------
|  88.8% |  98.575759 |    -- |           -- |         -- |USER
||----------------------------------------------------------------
||   1.7% | 103.103247 |    -- |           -- |         -- |pe.1
||   1.6% |  99.183271 |    -- |           -- |         -- |pe.43
||   1.5% |  95.192528 |    -- |           -- |         -- |pe.22
||================================================================
|  11.1% |  12.321335 | 930.2 |  997045979.0 | 1071907.21 |MPI
||----------------------------------------------------------------
||   0.3% |  17.008587 | 932.0 |  981012480.0 | 1052588.50 |pe.39
||   0.2% |  12.790484 | 927.0 |  999014400.0 | 1077685.44 |pe.32
||   0.0% |   2.468221 | 929.0 | 1015193600.0 | 1092781.05 |pe.1
|=================================================================

Notes for table 4:

  Table option:
    -O mpi_callers
  Options implied by table option:
    -d Mm,Mc@,Mb1..7 -b fu,ca,pe=[mmm]

  Options for related tables not shown by default:
    -O mpi_sm_callers          -O mpi_coll_callers
```

```
  The Total value for each data item is the sum of the Function values.
  The Function value for each data item is the sum of the Caller values.
  The Caller value for each data item is the avg of the PE values.
    (To specify different aggregations, see:  pat_help report options s1)

  This table shows only lines with MPI Msg Count > 0.

Table 4:  MPI Message Stats by Caller

MPI Msg Bytes |   MPI | 64KB<= | 1MB<= |Function
              |   Msg |  MsgSz | MsgSz | Caller
              | Count |   <1MB | <16MB |  PE[mmm]
              |       |  Count | Count |

  997045979.0 | 930.2 |  243.1 | 687.0 |Total
|-----------------------------------------------
|  997045979.0 | 930.2 |  243.1 | 687.0 |MPI_Send
|              |       |        |       | pzgemm_
3              |       |        |       |  main
||||---------------------------------------------
4||| 1015808000.0 | 926.0 |   82.0 | 844.0 |pe.3
4|||  998711296.0 | 930.0 |  500.0 | 430.0 |pe.14
4|||  978452480.0 | 930.0 |  507.0 | 423.0 |pe.55
|===============================================

Notes for table 6:

  Table option:
    -O program_time
  Options implied by table option:
    -d pt,hm -b pe=[mmm]

  The Total value for each of Process Time, Process HiMem (MBytes) is the
avg
    of the PE values.
    (To specify different aggregations, see:  pat_help report options s1)

Table 6:  Program Wall Clock Time, Memory High Water Mark

    Process |  Process |PE[mmm]
       Time |    HiMem |
            | (MBytes) |

 114.987871 |      313 |Total
|------------------------------
| 115.521469 |  320.918 |pe.9
| 114.982404 |  313.098 |pe.28
| 114.475464 |  313.090 |pe.20
|==============================

=========  Additional details ============================

Experiment:  trace
```

```
Original path to data file:
  /lustre/scratch/roche/chk-perf/xcpbls-cp+apa+25678-20623tdt.xf  (RTS)

Original program:  /lustre/scratch/roche/chk-perf/./xcpbls-cp

Instrumented with:
  pat_build -Drtenv=PAT_RT_HWPC=0 -g mpi -w -o xcpbls-cp+apa \
         ./xcpbls-cp

Instrumented program:  ./xcpbls-cp+apa

Program invocation:  ./xcpbls-cp+apa 16384 16384 16384 80 7 8

Exit Status:  0  PEs:  0-55

Memory pagesize:  4096

Runtime environment variables:
  MPICHBASEDIR=/opt/mpt/3.1.0/xt
  PAT_RT_HWPC=0
  MPICH_DIR=/opt/mpt/3.1.0/xt/mpich2-pgi

Report time environment variables:
  CRAYPAT_ROOT=/opt/xt-tools/craypat/4.4.0.4/v23/cpatx

Report command line options:  -o apa-report.txt

Operating system:
  Linux 2.6.16.54-0.2.12_1.0000.3997.0-cnl #1 SMP Mon Jan 26 13:41:57 PST
2009

Hardware performance counter events:
  PAPI_L1_DCM   Level 1 data cache misses
  CYCLES_USER   User Cycles (approx, from clock ticks)
  PAPI_L1_DCA   Level 1 data cache accesses
  PAPI_TOT_INS  Instructions completed
  PAPI_FP_OPS   Floating point operations

Estimated minimum overhead per call of a traced function,
  which was subtracted from the data shown in this report
  (for raw data, use the option:  -s overhead=include):
    PAPI_L1_DCM      10.653  misses
    PAPI_TOT_INS   2019.045  instr
    PAPI_L1_DCA    1192.191  refs
    PAPI_FP_OPS       0.000  ops
    CYCLES_USER    4107.143  cycles
    Time              1.452  microseconds

Number of traced functions: 104
  (To see the list, specify:  -s traced_functions=show)
```

GNU compilation/execution process plus automatic instrumentation

- Instrument the code (from a bash shell)

  ```
  <yourcode>c via GNU gcc compiler
  ```

- Load the GNU environment

  ```
  module swap PrgEnv-pgi PrgEnv-gnu .
  ```

- Load the correct tools

  ```
  Module load xt-craypat .
  ```

- Set the environment variables to capture intended metrics

  ```
  export PAT_RT_HWPC=PAPI_TOT_CYC,PAPI_TOT_INS,PAPI_FP_INS .
  ```

- Instrument the source code by compiling with code generation hooks in the original code that CrayPAT will utilize

  ```
  cc -c -finstrument-functions <yourcode>.c .
  ```

- Build the binary including the instrumentation hooks

  ```
  cc -o x<yourcodebinary><yourcode>.o .
  ```

- Build the instrumentation (tracing example) binary

  ```
  x<yourcodebinary>+pat : pat_build -w
x<yourcodebinary>x<yourcodebinary>+pat .
  ```

- Execute the instrumented binary

  ```
  aprun -n <n> ./x<yourcodebinary>+pat .
  ```

- Build a simplistic performance report for the run

  ```
  pat_report -o <yourreport>.txt x<yourcodebinary>+pat+<processlabels>.xf .
  ```

- Report is in the text file, which will include output similar to below (note that the profiled program did essentially no floating point computations)

  ```
  ==============================================================================
   Totals for program
   ------------------------------------------------------------------------------
     Time%                                    100.0%
     Time                                   0.548266 secs
     Imb.Time                                     -- secs
     Imb.Time%                                    --
     Calls                       9.1 /sec         5.0 calls
     PAPI_TOT_INS           3648.979M/sec  1997187152 instr
     PAPI_FP_INS                  0 /sec       0.018 ops
     PAPI_TOT_CYC              0.547 secs  1258853693 cycles
     User time (approx)        0.548 secs  1260482143 cycles  99.8%Time
     Average Time per Call                   0.109653 sec
     CrayPat Overhead : Time  0.0%
  ```

```
  HW FP Ops / Cycles                        0.00 ops/cycle
  HW FP Ops / User time       0 /sec       0.018 ops  0.0%peak(DP)
  HW FP Ops / WCT             0 /sec
  HW FP Ops / Inst                          0.0%
  Instr per cycle                           1.59 inst/cycle
  MIPS                204342.81M/sec
  MFLOPS (aggregate)       0.00M/sec
========================================================================
USER
------------------------------------------------------------------------
  Time%                                    100.0%
  Time                                   0.548263 secs
  Imb.Time                                     -- secs
  Imb.Time%                                    --
  Calls                     5.5 /sec         3.0 calls
  PAPI_TOT_INS         3648.992M/sec  1997186185 instr
  PAPI_FP_INS                 0 /sec       0.018 ops
  PAPI_TOT_CYC            0.547 secs  1258848387 cycles
  User time (approx)      0.548 secs  1260482143 cycles  99.8%Time
  Average Time per Call                  0.182754 sec
  CrayPat Overhead : Time  0.0%
  HW FP Ops / Cycles                        0.00 ops/cycle
  HW FP Ops / User time       0 /sec       0.018 ops  0.0%peak(DP)
  HW FP Ops / WCT             0 /sec
  HW FP Ops / Inst                          0.0%
  Instr per cycle                           1.59 inst/cycle
  MIPS                204343.58M/sec
  MFLOPS (aggregate)       0.00M/sec
========================================================================
USER / main
------------------------------------------------------------------------
  Time%                                    100.0%
  Time                                   0.548163 secs
  Imb.Time                               0.565060 secs
  Imb.Time%                                 52.6%
  Calls                     3.7 /sec         2.0 calls
  PAPI_TOT_INS         3649.078M/sec  1997092099 instr
  PAPI_FP_INS                 0 /sec       0.018 ops
  PAPI_TOT_CYC            0.547 secs  1258759596 cycles
  User time (approx)      0.548 secs  1260482143 cycles  99.8%Time
  Average Time per Call                  0.274082 sec
  CrayPat Overhead : Time  0.0%
  HW FP Ops / Cycles                        0.00 ops/cycle
  HW FP Ops / User time       0 /sec       0.018 ops  0.0%peak(DP)
  HW FP Ops / WCT             0 /sec
  HW FP Ops / Inst                          0.0%
  Instr per cycle                           1.59 inst/cycle
  MIPS                204348.36M/sec
  MFLOPS (aggregate)       0.00M/sec
========================================================================
```

APPENDIX B. VISIT

B.1 INPUT SETTINGS

The VisIt test runs were launched using the python API using the isosurface and volume rendering scripts as given below.

Isosurface script

```
import sys
OpenDatabase("/lustre/scratch/pugmire/proj/joule/denovo/forward_out.silo",
   0 )
expr = "scalar_flux_0 *1.6151E-04"
expr = expr + "+ scalar_flux_1 *1.4451E-04"
expr = expr + "+ scalar_flux_2 *1.2704E-04"
expr = expr + "+ scalar_flux_3 *1.2811E-04"
expr = expr + "+ scalar_flux_4 *1.2984E-04"
expr = expr + "+ scalar_flux_5 *1.0343E-04"
expr = expr + "+ scalar_flux_6 *5.2655E-05"
expr = expr + "+ scalar_flux_7 *1.2861E-05"
expr = expr + "+ scalar_flux_8 *3.7358E-06"
expr = expr + "+ scalar_flux_9 *3.7198E-06"
expr = expr + "+ scalar_flux_10 *4.0086E-06"
expr = expr + "+ scalar_flux_11 *4.2945E-06"
expr = expr + "+ scalar_flux_12 *4.4731E-06"
expr = expr + "+ scalar_flux_13 *4.5656E-06"
expr = expr + "+ scalar_flux_14 *4.5597E-06"
expr = expr + "+ scalar_flux_15 *4.5210E-06"
expr = expr + "+ scalar_flux_16 *4.4873E-06"
expr = expr + "+ scalar_flux_17 *4.4660E-06"
expr = expr + "+ scalar_flux_18 *4.4342E-06"
expr = expr + "+ scalar_flux_19 *4.3316E-06"
expr = expr + "+ scalar_flux_20 *4.2028E-06"
expr = expr + "+ scalar_flux_21 *4.0974E-06"
expr = expr + "+ scalar_flux_22 *3.8398E-06"
expr = expr + "+ scalar_flux_23 *3.6748E-06"
expr = expr + "+ scalar_flux_24 *3.6748E-06"
expr = expr + "+ scalar_flux_25 *3.6748E-06"
expr = expr + "+ scalar_flux_26 *3.6748E-06"
DefineScalarExpression("dose", "%s" % expr)
AddPlot("Contour", "dose", 1, 1)
ContourAtts = ContourAttributes()
ContourAtts.contourMethod = ContourAtts.Value
ContourAtts.contourValue = (.001, .01, .1, 1, 10, 100)
SetPlotOptions(ContourAtts)
AddPlot("Contour", "dose", 1, 1)
ContourAtts = ContourAttributes()
ContourAtts.contourMethod = ContourAtts.Value
ContourAtts.contourValue = (.001, .01, .1, 1, 10, 100)
SetPlotOptions(ContourAtts)
s = SaveWindowAttributes()
s.width, s.height = (1024,1024)
```

```
SetSaveWindowAttributes(s)
DrawPlots()
SaveWindow()
sys.exit()
```

The Q4 problem script was identitcal to the Q2 script except that it references the new Q4 file. Namely:

```
OpenDatabase("/lustre/scratch/pugmire/proj/joule/denovo/forward_out.silo",
   0 )
```

Was replaced with:

```
OpenDatabase("/lustre/widow1/scratch/pugmire/proj/joule/denovo/medbig_forw
   ard_out.silo", 0 )
```

Volume Rendering script:

```
import sys
RestoreSession( "/lustre/scratch/pugmire/proj/joule/benchmark0/vr/VR-
   4000samp.session", 0 )
s = SaveWindowAttributes()
s.width, s.height = (1024,1024)
SetSaveWindowAttributes(s)
DrawPlots()
SaveWindow()
sys.exit()
```

The Q4 problem script was identical to the Q2 script, except that it references the new Q4 data file. Namely:

```
RestoreSession( "/lustre/scratch/pugmire/proj/joule/benchmark0/vr/VR-
   4000samp.session", 0 )
```

Was replaced with:

```
RestoreSession(
   "/lustre/widow1/scratch/pugmire/proj/joule/benchmark1/vr/VR-
   4000samp.session", 0 )
```

Denvo was run on Jaguar/XT5 on 4096 cores and 4096 domains with the input deck given below.

```
eq_set: sc
input: pwr_in
Pn_order: 3
Sn_order: 16
num_blocks_i: 64
num_blocks_j: 64
num_z_blocks: 27
silo_output: forward_out
pwr_in is a binary file specifying the problem setup.

The simulation output is as follows:
```

```
Tue Dec 16 23:18:29 EST 2008
>>> Finished reading problem database.
>>> Finished partitioning problem.
>>> Finished reading material database.
>>> Finished reading source database.
>>> Finished building solvers.
Database for hpc_input has:
    12 integer entries
    1 double entries
    2 bool entries
    4 string entries
    0 vector<int> entries
    5 vector<double> entries
    1 nested database entries

====================================
Entries in       hpc_input database
====================================
integer entries
----------------------------------------
                 Pn_order             3
                 Sn_order            16
               aztec_diag             0
             aztec_kspace            20
             aztec_output             0
              first_group             0
               last_group            26
                   max_itr          1000
             num_blocks_i            64
             num_blocks_j            64
               num_groups            46
             num_z_blocks            27
double entries
----------------------------------------
                tolerance          1e-06
bool entries
----------------------------------------
                  adjoint             0
              downscatter             1

string entries
----------------------------------------
                 boundary         vacuum
                    input         pwr_in
             problem_name            pwr
      within_group_solver          GMRES
====================================
Entries in            silo database
====================================
bool entries
----------------------------------------
         silo_out_current             0
           silo_out_sigma             0
```

```
string entries
----------------------------------------
              silo_output    forward_out
Denovo Setup complete, ready to solve using   SC spatial differencing
   option.
---------------------------------------------------------------------------
   -

>>> Forward group   0 finished in    8 GMRES iterations.
>>> Forward group   1 finished in    9 GMRES iterations.
>>> Forward group   2 finished in    9 GMRES iterations.
>>> Forward group   3 finished in    9 GMRES iterations.
>>> Forward group   4 finished in   12 GMRES iterations.
>>> Forward group   5 finished in   17 GMRES iterations.
>>> Forward group   6 finished in   23 GMRES iterations.
>>> Forward group   7 finished in   21 GMRES iterations.
>>> Forward group   8 finished in   27 GMRES iterations.
>>> Forward group   9 finished in   26 GMRES iterations.
>>> Forward group  10 finished in   27 GMRES iterations.
>>> Forward group  11 finished in   23 GMRES iterations.
>>> Forward group  12 finished in   20 GMRES iterations.
>>> Forward group  13 finished in   20 GMRES iterations.
>>> Forward group  14 finished in   15 GMRES iterations.
>>> Forward group  15 finished in   13 GMRES iterations.
>>> Forward group  16 finished in    9 GMRES iterations.
>>> Forward group  17 finished in    8 GMRES iterations.
>>> Forward group  18 finished in   10 GMRES iterations.
>>> Forward group  19 finished in   14 GMRES iterations.
>>> Forward group  20 finished in    9 GMRES iterations.
>>> Forward group  21 finished in   10 GMRES iterations.
>>> Forward group  22 finished in   13 GMRES iterations.
>>> Forward group  23 finished in    9 GMRES iterations.
>>> Forward group  24 finished in    8 GMRES iterations.
>>> Forward group  25 finished in    9 GMRES iterations.
>>> Forward group  26 finished in    8 GMRES iterations.

====================
Final Timing Report
====================
                                  Routine   Max Fraction   Min Fraction
=================================================================
                           Build_solver     1.5522e-05     5.1841e-06
                                 Output     9.5824e-02     4.0892e-04
                                  Setup     3.6749e-01     3.6746e-01
                                 Solver     5.3672e-01     5.3669e-01
                                  Sweep     5.1994e-01     4.8821e-01
                     Within_group_solver     5.3672e-01     5.3669e-01
=================================================================
Total execution time : 4.1919e+03 seconds.
Application 1843223 resources: utime 0, stime 7
```

B.2 COMPILATION

Compilation of VisIt was done on the currently released version of VisIt, 1.11.1 using the g++ compiler (in /opt/gcc/4.2.0.quadcore/bin/g++).The following compiler options were used:

```
CC="gcc"
CXX="g++"
CFLAGS=" -m64 -fPIC -DMPICH_IGNORE_CXX_SEEK -DPAPI"
CXXFLAGS=" -m64 -fPIC -DMPICH_IGNORE_CXX_SEEK -DPAPI"
#Get these via CC -v
LDFLAGS="-L/opt/fftw/3.1.1/cnos/lib $LDFLAGS"
LDFLAGS="-L/opt/mpt/3.1.0/xt/mpich2-gnu/lib $LDFLAGS"
LDFLAGS="-L/opt/xt-libsci/10.3.1/gnu/snos64/lib $LDFLAGS"
LDFLAGS="-L/opt/mpt/3.1.0/xt/sma/lib $LDFLAGS"
LDFLAGS="-L/opt/mpt/3.1.0/xt/util/lib $LDFLAGS"
LDFLAGS="-L/opt/mpt/3.1.0/xt/pmi/lib $LDFLAGS"
LDFLAGS="-L/opt/xt-pe/2.1.41HD/lib/snos64 $LDFLAGS"
LDFLAGS="-L/opt/xt-service/2.1.41HD/lib/snos64 $LDFLAGS"
LDFLAGS="-L/opt/mpt/3.1.0/xt/mpich2-gnu/lib $LDFLAGS"
CPPFLAGS="-I/opt/mpt/3.1.0/xt/mpich2-gnu/include -I/opt/xt-
   tools/papi/3.6.2/v23/linux/include $CPPFLAGS"
MPI_LIBS="-Bstatic -lfftw3 -lfftw3f -lsci_quadcore -lsci -lfftw3 -lfftw3f
   /opt/mpt/3.1.0/xt/sma/lib/libsma.a /opt/mpt/3.1.0/xt/mpich2-
   gnu/lib/libmpichcxx.a /opt/mpt/3.1.0/xt/mpich2-gnu/lib/libmpich.a -lrt
   --start -lpct /opt/mpt/3.1.0/xt/pmi/lib/libpmi.a /opt/xt-
   mpt/2.1.41HD/lib/snos64/libalpslli.a /opt/xt-
   mpt/2.1.41HD/lib/snos64/libalpsutil.a /opt/xt-
   service/2.1.41HD/lib/snos64/libportals.a /opt/xt-
   tools/papi/3.6.2/v23/linux/lib/libpapi.a /opt/xt-
   tools/papi/3.6.2/v23/linux/lib/libpfm.a -lpthread -lm --end -lm -lgcc -
   lgcc_eh -lc -lgcc -lgcc_eh -lc"
```

The VisIt parallel engine links to the Silo, python, VTK, mesa and HDF5 libraries:

```
silo/4.6.1/linux-x86_64_gcc-4.2.0
python/2.5/linux-x86_64_gcc-4.2.0
vtk/5.0.0c/linux-x86_64_gcc-4.2.0
mesa/5.0/linux-x86_64_gcc-4.2.0
hdf5/1.6.5/linux-x86_64_gcc-4.2.0
```

B.3 BATCH SCRIPT

The batch script is available upon request.

B.4 RUNTIME ENVIRONMENT

Modules used:

```
PrgEnv-gnu
xt-papi
```

APPENDIX C. CAM

Here we report the important elements of the software environment at the time of the Q2 run. This includes version specification of the operating system, compiler, and required software libraries. We also report important settings in the Makefile and other parts of the model build procedure (configuration and compilation), including optimization flags passed to the compiler. Finally, the model run script and critical Fortran name-list settings are also included. We have archived all these files and settings locally (along with the model source code), in order to isolate what was changed between Q2 and Q4. We did not include full Makefile, name list, or "make" output here due to the vast volume of data inclusion that would be required.

A.1 INPUT SETTINGS

Shown below are performance tuning settings from the input Fortran name-list that were applied in the Q2 and Q4 runs. Changes in the Q4 settings are as a result of exploring optimal values, and the fact that the optimal values can change based on code changes.

```
&cam_inparm
 phys_loadbalance = 2
 phys_alltoall = 1
/
```

Q4 settings:

```
&cam_inparm
 phys_loadbalance = 3
/
```

A.2 COMPILATION

The following are the critical settings from the model Makefile used in the Q2 and Q4 runs, respectively.

Q2 Makefile:

```
MODEL_EXEDIR:=/autofs/na1_home/rosinski/cam3.5.55/models/atm/cam/joulebase
INC_NETCDF   := /sw/xt5/netcdf/3.6.2/sles10.1_pgi7.2.3/include
LIB_NETCDF   := /sw/xt5/netcdf/3.6.2/sles10.1_pgi7.2.3/lib
MOD_NETCDF   := /sw/xt5/netcdf/3.6.2/sles10.1_pgi7.2.3/include
USER_CPPDEFS := -DHAVE_PAPI -DFORTRANUNDERSCORE \
                -DTROPCHEM -DCOUP_DOM -DPLON=1024 \
                -DPLAT=512 -DPLEV=26 -DPCNST=3 -DPCOLS=16 \
                -DPTRM=341 -DPTRN=341 -DPTRK=341
FORTRAN_OPTIMIZATION := -fast -Mvect=nosse –Kieee
```

Q4 Makefile:

```
MODEL_EXEDIR:=/autofs/na1_home/rosinski/cam3.5.55/models/atm/cam/joulebase
INC_NETCDF   := /opt/cray/netcdf/4.0.0.3/netcdf-pgi/include
```

```
LIB_NETCDF   := /opt/cray/netcdf/4.0.0.3/netcdf-pgi/lib
MOD_NETCDF   := /opt/cray/netcdf/4.0.0.3/netcdf-pgi/include
USER_CPPDEFS := -DHAVE_PAPI -DFORTRANUNDERSCORE \
                -DTROPCHEM -DCOUP_DOM -DPLON=1024 \
                -DPLAT=512 -DPLEV=26 -DPCNST=3 -DPCOLS=16 \
                -DPTRM=341 -DPTRN=341 -DPTRK=341
FORTRAN_OPTIMIZATION := -fast -fastsse -Mvect=sse
```

A.3 BATCH SCRIPT

The following is the Q2 model run script, minus the Fortran name lists.

```
!/bin/csh -fvx
#PBS -N jb8192
#PBS -V
#PBS -A CSC053CAM
#PBS -j oe
#PBS -l walltime=3:00:00
#PBS -l size=8192
#PBS -q batch
##PBS -q debug

setenv OMP_NUM_THREADS 8
setenv MPSTKZ 384M
setenv MPICH_UNEX_BUFFER_SIZE 250M

cd /tmp/work/rosinski/cam3.5.55.t341.withmods.adv3/joulebase  || exit 1

set iter = 2
set dir = npes8192.iter$iter
mkdir $dir
cd $dir
cp /ccs/home/rosinski/cam3.5.55/models/atm/cam/joulebase/cam .
aprun -n 1024 -d $OMP_NUM_THREADS ./cam >&! out.init
exit 0
```

Paths to model source and run directories were different for the Q4 runs. But there were no changes to environment variables or other aspects of the build system.

C.4 RUNTIME ENVIRONMENT

The runtime environment is available in an archived file on the Jaguar/XT5 file system.

APPENDIX D. XGC1

D.1 INPUT SETTINGS

```
Input namelist file
&sml_param
sml_machine=1             ! 0 circular, 1:D3D,
sml_node_file='d3d_g096333_2mm_16_mr10.1.node'
sml_ele_file='d3d_g096333_2mm_16_mr10.1.ele'
sml_use_pade=.true.
sml_bfollow=1
sml_bfollow_read=0

sml_special=0             ! 0: normal simulation, 1: single particle
   simulation
sml_dt=0.002            ! delta-t for one time step - unit of toroidal
   transit time.
sml_mstep=500          ! totoal time step
sml_deltaf=0              ! delta-f simulation switch - incomplete
sml_turb_efield=1
sml_electron_on=0
sml_nphi_total=16
sml_canonical_maxwell=0   ! cononical maxwellian initial loading switch -
   incomplete
sml_bounce=0              ! Particle motion boundary condition
                          ! 1 for edge simulation (including open field
   line region)
                          ! 2 for core simulation (excluding open field
   line region)
sml_limiter=0             ! Limiter on/off
sml_fem_matrix=1
sml_inpsi=0.0d0             ! inner boundary of simulation - unit of
   eq_x_psi
sml_outpsi=1.10d0           ! outter boundary of simulation - unit of
   eq_x_psi
sml_push_mode=3
sml_pc_order=2
sml_restart_write_period=500
sml_restart=0
sml_zero_inner_bd=0
sml_guess_table_size=1500
sml_no_00_efield=0
sml_input_file_dir='../XGC1_inputs/'
sml_bd_ext_delta2=-0.01
sml_bd_ext_delta1=-0.003
sml_bd_ext_delta3=0.001
sml_bd_ext_delta4=0.03
sml_max_mat_width=300
sml_bd_Te_mode=0
sml_bd_Te_width=0.01D0
sml_sheath_mode=0
sml_sheath_init_pot_factor=2.5
```

```
sml_rgn1_pot0_only=.true.
sml_add_pot0=1
sml_add_pot0_file='pot0_0327_d3d_g096333_2mm_16_mr10_177236_fac.5.dat'
sml_zero_out_total_charge=.false.
sml_pol_decomp=.false.
sml_heat_on=.true.
sml_iter_solver=.false.
sml_iter_solver_niter=3
sml_bt_sign=1
/

&ptl_param
ptl_mass_au=2D0             ! 1 for hydrogen, 2 for deutron
ptl_charge_eu=1D0            ! ion charge
ptl_num=450000             ! number of particle for simulation
ptl_maxnum=550000
/

&eq_param ! Initial equilibrium profile - Tanh profile
eq_filename='d3d096333.eqd'
eq_den_shape=-1
eq_den_edge=4.0D20          ! inside value of density m^-3
eq_den_out=0.5D20           ! outside value of density  m^-3
eq_den_ped_c=0.96D0      ! pedestal center
eq_den_ped_width=0.09D0    ! pedestal width
eq_den_val3=6.0D20
eq_den_psi3=0D0

eq_tempi_shape=-1
eq_tempe_shape=-1
eq_tempi_ped_c=0.91D0
eq_tempe_ped_c=0.91D0
eq_tempi_ped_width=0.14D0
eq_tempe_ped_width=0.14D0
eq_tempi_ev_edge=1D3      ! ion temperature (inside) - eV
eq_tempi_ev_out=5d1     ! ion temperature (outside) - eV
eq_tempe_ev_edge=1D3
eq_tempe_ev_out=5D1
eq_tempi_val3=4.5D3
eq_tempi_psi3=0D0
eq_tempe_val3=4.5D3
eq_tempe_psi3=0D0

eq_den_file='d3d_white_pop_2008_den.prf'
eq_tempi_file='d3d_white_pop_2008_tempi.prf'
eq_tempe_file='d3d_white_pop_2008_tempe.prf'
/

&efld_param ! E-field calculation
efld_mode=2              ! 0 zero efield, 1 static efield, 2 self-
   consistent
efld_cutoff=0
/
```

```
&col_param  ! Collision
col_mode=3  ! 0 : off , 1 monte-carlo (non-conserving) 2: monte-carlo
   (conserving)
col_accel=.true.
col_accel_n=1
col_accel_factor1=10.
col_accel_pout1=0.08
/

&diag_param ! diagnosis
diag_f_on=0
diag_tracer_period=1
diag_tracer_n=1
diag_binout_period=10
diag_pot_period=200000
diag_ptl_on=0
diag_ptl_begin=10
diag_ptl_num=1000
diag_gam_on=0
diag_avg_on=1
diag_avg_outperiod=10
diag_flow_period=10
diag_rect_rmin=1.7
diag_rect_rmax=2.3
diag_rect_zmin=-0.03
diag_rect_zmax=0.03
diag_rect_nr=100
diag_rect_nz=3
diag_stress_on=.true.
/

&neu_param            ! neutral collision
neu_col_mode=0
/

&lim_param            ! limiter

/
&smooth_param
smooth_mode_in=0
smooth_n_in=2
smooth_H_mode_in=2
smooth_H_n_in=2

smooth_r1_n_in=-1
smooth_r1_d0_in=0.0042
smooth_r1_type_in=1

smooth_diag_mode_in=-1
/

&tbl_param
/
```

```
&heat_param
heat_narea=1
heat_power=50D6
heat_period=10
heat_outpsi=0.04
heat_decay_width=0.05
/

&mon_param
mon_flush_count=100
/

&prof_inparam
profile_papi_enable=.true.
profile_outpe_num = -1
profile_single_file = .false.
/
&papi_inparam
papi_ctr1_str="PAPI_TOT_CYC"
papi_ctr2_str="PAPI_TOT_INS"
papi_ctr3_str="PAPI_FP_INS"
/

PETSc input
-log_summary
-pc_type hypre
%-pc_type jacobi
-ksp_type cg
-pc_hypre_type boomeramg
-mat_partitioning_type current
-s2_mat_partitioning_type current
-s2_ksp_type cg
-s2_pc_type hypre
-s2_pc_hypre_type boomeramg
```

D.2 COMPILATION

```
/usr/bin/make  all-am
make[1]: Entering directory
   `/autofs/na1_home/shku/xgc/trunk/XGC1/jaguarpf'
source='../../camtimers/GPTLget_memusage.c' object='libtimers_wpapi_a-
   GPTLget_memusage.o' libtool=no \
DEPDIR=.deps depmode=none /bin/sh ../../acfiles/depcomp \
cc -DHAVE_CONFIG_H -I. -I.. -I.    -DLINUX -DFORTRANUNDERSCORE -DSPMD -
   DHAVE_NANOTIME -DBIT64 -I../../camtimers  -DHAVE_PAPI -I/opt/xt-
   tools/papi/3.6.2/v23/xt-cnl/include -fastsse -Kieee -mp -c -o
   libtimers_wpapi_a-GPTLget_memusage.o `test -f
   '../../camtimers/GPTLget_memusage.c' || echo
   '../'`../../camtimers/GPTLget_memusage.c
/opt/cray/xt-asyncpe/2.0/bin/cc: INFO: linux target is being used
source='../../camtimers/GPTLprint_memusage.c' object='libtimers_wpapi_a-
   GPTLprint_memusage.o' libtool=no \
```

```
DEPDIR=.deps depmode=none /bin/sh ../../acfiles/depcomp \
cc -DHAVE_CONFIG_H -I. -I.. -I.    -DLINUX -DFORTRANUNDERSCORE -DSPMD -
   DHAVE_NANOTIME -DBIT64 -I../../camtimers  -DHAVE_PAPI -I/opt/xt-
   tools/papi/3.6.2/v23/xt-cnl/include -fastsse -Kieee -mp -c -o
   libtimers_wpapi_a-GPTLprint_memusage.o `test -f
   '../../camtimers/GPTLprint_memusage.c' || echo
   '../'`../../camtimers/GPTLprint_memusage.c
/opt/cray/xt-asyncpe/2.0/bin/cc: INFO: linux target is being used
source='../../camtimers/GPTLutil.c' object='libtimers_wpapi_a-GPTLutil.o'
   libtool=no \
DEPDIR=.deps depmode=none /bin/sh ../../acfiles/depcomp \
cc -DHAVE_CONFIG_H -I. -I.. -I.    -DLINUX -DFORTRANUNDERSCORE -DSPMD -
   DHAVE_NANOTIME -DBIT64 -I../../camtimers  -DHAVE_PAPI -I/opt/xt-
   tools/papi/3.6.2/v23/xt-cnl/include -fastsse -Kieee -mp -c -o
   libtimers_wpapi_a-GPTLutil.o `test -f '../../camtimers/GPTLutil.c' ||
   echo '../'`../../camtimers/GPTLutil.c
/opt/cray/xt-asyncpe/2.0/bin/cc: INFO: linux target is being used
source='../../camtimers/f_wrappers.c' object='libtimers_wpapi_a-
   f_wrappers.o' libtool=no \
DEPDIR=.deps depmode=none /bin/sh ../../acfiles/depcomp \
cc -DHAVE_CONFIG_H -I. -I.. -I.    -DLINUX -DFORTRANUNDERSCORE -DSPMD -
   DHAVE_NANOTIME -DBIT64 -I../../camtimers  -DHAVE_PAPI -I/opt/xt-
   tools/papi/3.6.2/v23/xt-cnl/include -fastsse -Kieee -mp -c -o
   libtimers_wpapi_a-f_wrappers.o `test -f '../../camtimers/f_wrappers.c'
   || echo '../'`../../camtimers/f_wrappers.c
/opt/cray/xt-asyncpe/2.0/bin/cc: INFO: linux target is being used
source='../../camtimers/gptl.c' object='libtimers_wpapi_a-gptl.o'
   libtool=no \
DEPDIR=.deps depmode=none /bin/sh ../../acfiles/depcomp \
cc -DHAVE_CONFIG_H -I. -I.. -I.    -DLINUX -DFORTRANUNDERSCORE -DSPMD -
   DHAVE_NANOTIME -DBIT64 -I../../camtimers  -DHAVE_PAPI -I/opt/xt-
   tools/papi/3.6.2/v23/xt-cnl/include -fastsse -Kieee -mp -c -o
   libtimers_wpapi_a-gptl.o `test -f '../../camtimers/gptl.c' || echo
   '../'`../../camtimers/gptl.c
/opt/cray/xt-asyncpe/2.0/bin/cc: INFO: linux target is being used
source='../../camtimers/gptl_papi.c' object='libtimers_wpapi_a-
   gptl_papi.o' libtool=no \
DEPDIR=.deps depmode=none /bin/sh ../../acfiles/depcomp \
cc -DHAVE_CONFIG_H -I. -I.. -I.    -DLINUX -DFORTRANUNDERSCORE -DSPMD -
   DHAVE_NANOTIME -DBIT64 -I../../camtimers  -DHAVE_PAPI -I/opt/xt-
   tools/papi/3.6.2/v23/xt-cnl/include -fastsse -Kieee -mp -c -o
   libtimers_wpapi_a-gptl_papi.o `test -f '../../camtimers/gptl_papi.c' ||
   echo '../'`../../camtimers/gptl_papi.c
/opt/cray/xt-asyncpe/2.0/bin/cc: INFO: linux target is being used
PGC-W-0095-Type cast required for this conversion
   (../../camtimers/gptl_papi.c: 952)
PGC/x86-64 Linux 7.2-5: compilation completed with warnings
source='../../camtimers/threadutil.c' object='libtimers_wpapi_a-
   threadutil.o' libtool=no \
DEPDIR=.deps depmode=none /bin/sh ../../acfiles/depcomp \
cc -DHAVE_CONFIG_H -I. -I.. -I.    -DLINUX -DFORTRANUNDERSCORE -DSPMD -
   DHAVE_NANOTIME -DBIT64 -I../../camtimers  -DHAVE_PAPI -I/opt/xt-
   tools/papi/3.6.2/v23/xt-cnl/include -fastsse -Kieee -mp -c -o
```

```
   libtimers_wpapi_a-threadutil.o `test -f '../../camtimers/threadutil.c'
   || echo '../'`../../camtimers/threadutil.c
/opt/cray/xt-asyncpe/2.0/bin/cc: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.    -Mfree -i4 -Mdalign -Mextend -
   byteswapio -DLINUX -DFORTRANUNDERSCORE -DSPMD -DHAVE_NANOTIME -DBIT64 -
   I../../camtimers  -DHAVE_PAPI -I/opt/xt-tools/papi/3.6.2/v23/xt-
   cnl/include -fastsse -Kieee -mp -c -o libtimers_wpapi_a-perf_utils.o
   `test -f '../../camtimers/perf_utils.F90' || echo
   '../'`../../camtimers/perf_utils.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.    -Mfree -i4 -Mdalign -Mextend -
   byteswapio -DLINUX -DFORTRANUNDERSCORE -DSPMD -DHAVE_NANOTIME -DBIT64 -
   I../../camtimers  -DHAVE_PAPI -I/opt/xt-tools/papi/3.6.2/v23/xt-
   cnl/include -fastsse -Kieee -mp -c -o libtimers_wpapi_a-perf_mod.o
   `test -f '../../camtimers/perf_mod.F90' || echo
   '../'`../../camtimers/perf_mod.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
rm -f libtimers_wpapi.a
/usr/bin/ar crus libtimers_wpapi.a libtimers_wpapi_a-GPTLget_memusage.o
   libtimers_wpapi_a-GPTLprint_memusage.o libtimers_wpapi_a-GPTLutil.o
   libtimers_wpapi_a-f_wrappers.o libtimers_wpapi_a-gptl.o
   libtimers_wpapi_a-gptl_papi.o libtimers_wpapi_a-threadutil.o
   libtimers_wpapi_a-perf_utils.o libtimers_wpapi_a-perf_mod.o
/usr/bin/ranlib libtimers_wpapi.a
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-module.o `test -f 'module.F90' || echo '../'`module.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-search.o `test -f 'search.F90' || echo '../'`search.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-charge.o `test -f 'charge.F90' || echo '../'`charge.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
```

```
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-read.o `test -f 'read.F90' || echo '../'`read.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-pol_decomp.o `test -f 'pol_decomp.F90' || echo
   '../'`pol_decomp.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-push.o `test -f 'push.F90' || echo '../'`push.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-setup.o `test -f 'setup.F90' || echo '../'`setup.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-efield.o `test -f 'efield.F90' || echo '../'`efield.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-diagnosis.o `test -f 'diagnosis.F90' || echo '../'`diagnosis.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-limiter.o `test -f 'limiter.F90' || echo '../'`limiter.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
```

```
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-bounce.o `test -f 'bounce.F90' || echo '../'`bounce.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-diagnosis2.o `test -f 'diagnosis2.F90' || echo
   '../'`diagnosis2.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-collision.o `test -f 'collision.F90' || echo '../'`collision.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-collision2.o `test -f 'collision2.F90' || echo
   '../'`collision2.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-diagnosis-f.o `test -f 'diagnosis-f.F90' || echo '../'`diagnosis-
   f.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-heat.o `test -f 'heat.F90' || echo '../'`heat.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
```

```
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-turbulence.o `test -f 'turbulence.F90' || echo
   '../'`turbulence.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-neutral.o `test -f 'neutral.F90' || echo '../'`neutral.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-neutral2.o `test -f 'neutral2.F90' || echo '../'`neutral2.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-linearsolver.o `test -f 'linearsolver.F90' || echo
   '../'`linearsolver.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-therm2d.o `test -f 'therm2d.F90' || echo '../'`therm2d.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-poisson.o `test -f 'poisson.F90' || echo '../'`poisson.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-taus88.o `test -f 'taus88.F90' || echo '../'`taus88.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
```

```
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-derf.o `test -f 'derf.F90' || echo '../'`derf.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-datanh.o `test -f 'datanh.F90' || echo '../'`datanh.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-fmin.o `test -f 'fmin.F90' || echo '../'`fmin.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-bspline90_22.o `test -f 'bspline90_22.F90' || echo
   '../'`bspline90_22.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-mpi.o `test -f 'mpi.F90' || echo '../'`mpi.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-interpolation.o `test -f 'interpolation.F90' || echo
   '../'`interpolation.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
```

```
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-load.o `test -f 'load.F90' || echo '../'`load.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn -DHAVE_CONFIG_H -I. -I.. -I.     -DADIOS -
   I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include  -DCAM_TIMERS -
   I../../camtimers -DPSPLINE -I/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/mod   -
   I/opt/petsc/2.3.3a/real/PGI/linux -
   I/opt/petsc/2.3.3a/real/PGI/linux/bmake/cray-xt -
   I/opt/petsc/2.3.3a/real/PGI/linux/include -fastsse -Kieee -mp -c -o
   xgc1-main.o `test -f 'main.F90' || echo '../'`main.F90
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
ftn  -fastsse -Kieee -mp   -o xgc1  xgc1-module.o xgc1-search.o xgc1-
   charge.o xgc1-read.o xgc1-pol_decomp.o xgc1-push.o xgc1-setup.o xgc1-
   efield.o xgc1-diagnosis.o xgc1-limiter.o xgc1-bounce.o xgc1-
   diagnosis2.o xgc1-collision.o xgc1-collision2.o xgc1-diagnosis-f.o
   xgc1-heat.o xgc1-turbulence.o xgc1-neutral.o xgc1-neutral2.o xgc1-
   linearsolver.o xgc1-therm2d.o xgc1-poisson.o  xgc1-taus88.o xgc1-derf.o
   xgc1-datanh.o xgc1-fmin.o xgc1-bspline90_22.o xgc1-mpi.o xgc1-
   interpolation.o xgc1-load.o xgc1-main.o    -
   L/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/lib -ladios -lmxml -
   I/sw/xt5/hdf5/1.6.8/cnl2.1_pgi7.2.3_par/include -
   L/sw/xt5/hdf5/1.6.8/cnl2.1_pgi7.2.3_par/lib -lhdf5 -
   L/sw/xt5/szip/2.1/sles10.1_pgi7.2.3/lib -lsz -lz  -L. -ltimers_wpapi -
   L/sw/xt/pspline/1.0/cnl2.0_pgi7.0.7/lib -lpspline -lezcdf -lportlib -
   L/sw/xt/netcdf/3.6.2/sles9.2_pgi7.0.7/lib -lnetcdf -
   L/sw/xt/netcdf/3.6.2/sles9.2_pgi7.0.7/lib -lnetcdf
/opt/cray/xt-asyncpe/2.0/bin/ftn: INFO: linux target is being used
/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/lib/libadios.a(libadios_a-
   adios_socket.o): In function `adios_set_socket_address':
/autofs/na1_sw/xt5/adios/0.9.8/cnl2.1_pgi8.0.3/adios-
   0.9.8/src/adios_socket.c:46: warning: Using 'gethostbyaddr' in
   statically linked applications requires at runtime the shared libraries
   from the glibc version used for linking
/autofs/na1_sw/xt5/adios/0.9.8/cnl2.1_pgi8.0.3/adios-
   0.9.8/src/adios_socket.c:41: warning: Using 'gethostbyname' in
   statically linked applications requires at runtime the shared libraries
   from the glibc version used for linking
make[1]: Leaving directory `/autofs/na1_home/shku/xgc/trunk/XGC1/jaguarpf
```

D.3 BATCH SCRIPT

```
#PBS -N pf9
#PBS -l walltime=24:00:00,size=29952
#PBS -j eo
##PBS -q debug
#PBS -A csc053xgc1
cd $PBS_O_WORKDIR
date
rm finished.sim
mkdir restart_dir
lfs setstripe restart_dir -s 39845888  -c 40 -i -1
```

```
aprun -n 29952 ../xgc1exe/xgc1+apa3 >& output.out
date
touch finished.sim
```

D.4 RUNTIME ENVIRONMENT

```
FFTW_POST_LINK_OPTS= -L/opt/fftw/3.1.1/cnos/lib -lfftw3 -lfftw3f
MODULE_VERSION_STACK=3.1.6
LESSKEY=/etc/lesskey.bin
PAPI_POST_LINK_OPTS= -L/opt/xt-tools/papi/3.6.2/v23/linux/lib -lpapi -lpfm
NNTPSERVER=news
INFODIR=/usr/local/info:/usr/share/info:/usr/info
MANPATH=/sw/xt5/totalview/8.6.0-
   1/sles10.1_binary/man:/opt/petsc/2.3.3a/man:/opt/xt-
   tools/papi/3.6.2/man:/sw/xt5/pspline/1.0/cnl2.1_pgi7.2.3/share/man:/sw/
   xt5/netcdf/3.6.2/sles10.1_pgi7.2.3/share/man:/sw/xt5/subversion/1.5.0/s
   les10.1_gnu4.2.4/share/man:/opt/xt-
   pe/2.1.50HD/papi/man:/opt/mpt/3.1.0/xt/man:/opt/xt-
   libsci/10.3.1/man:/opt/fftw/3.1.1/cnos/man:/opt/pgi/7.2.5/linux86-
   64/7.2/man:/opt/xt-lustre-
   ss/2.1.50HD.PS04.lus.1.6.5.steve.8062_1.6.5/usr/man:/opt/xt-
   os/2.1.50HD/cnselect/man:/opt/xt-os/2.1.50HD/ros/man:/opt/xt-
   libc/2.1.50HD/xt3_glibc/man:/opt/MySQL/5.0.45/man:/opt/moab/man:/opt/to
   rque/default/man:/sw/xt5/man:/usr/local/man:/usr/share/man:/usr/X11R6/m
   an:/opt/gnome/share/man:/opt/xt-pe/2.1.50HD/pe/man
HOSTNAME=jaguarpf-login1
GNOME2_PATH=/usr/local:/opt/gnome:/usr
XKEYSYMDB=/usr/X11R6/lib/X11/XKeysymDB
PAPI_VERSION=3.6.2
PE_ENV=PGI
PATHSCALE_POST_COMPILE_OPTS= -march=barcelona
_MODULESBEGINENV_=/ccs/home/shku/.modulesbeginenv.jaguarpf-login1
HOST=jaguarpf-login1
TERM=xterm
SHELL=/bin/bash
XTOS_VERSION=2.1.50HD
PROFILEREAD=true
HISTSIZE=1000
TOTALVIEW_VERSION=8.6.0-1
PETSC_ARCH=cray-xt
PERFMON_VERSION=v23
SSH_CLIENT=128.122.81.37 40601 22
LIBRARY_PATH=/sw/xt5/pspline/1.0/cnl2.1_pgi7.2.3/lib:/sw/xt5/netcdf/3.6.2/
   sles10.1_pgi7.2.3/lib:/sw/xt5/subversion/1.5.0/sles10.1_gnu4.2.4/lib
MPT_DIR=/opt/mpt/3.1.0/xt
FFTW_INC=/opt/fftw/3.1.1/cnos/include
MORE=-sl
BOOT_DIR=/opt/xt-boot/2.1.50HD
QTDIR=/usr/lib/qt3
INCLUDE_PATH=/sw/xt5/pspline/1.0/cnl2.1_pgi7.2.3/mod:/sw/xt5/netcdf/3.6.2/
   sles10.1_pgi7.2.3/include:/sw/xt5/subversion/1.5.0/sles10.1_gnu4.2.4/in
   clude
```

```
PRGENV_DIR=/opt/xt-prgenv/2.1.50HD
SSH_TTY=/dev/pts/26
TVMEMDEBUG_POST_LINK_OPTS= -L/sw/xt/totalview/8.6.0-
   1/sles10.1_binary/linux-x86-64/lib -ltvheap_cnl_static
PSPLINE_INCLUDE_OPTS=-I/sw/xt5/pspline/1.0/cnl2.1_pgi7.2.3/mod
FFTW_DIR=/opt/fftw/3.1.1/cnos/lib
ASYNCPE_DIR=/opt/cray/xt-asyncpe/2.0
BUILD_OPTS=/opt/cray/xt-asyncpe/2.0/bin/build-opts
GROFF_NO_SGR=yes
JRE_HOME=/usr/lib/jvm/jre
USER=shku
LD_LIBRARY_PATH=/opt/xt-
   tools/papi/3.6.2/v23/linux/lib:/sw/xt5/pspline/1.0/cnl2.1_pgi7.2.3/lib:
   /sw/xt5/netcdf/3.6.2/sles10.1_pgi7.2.3/lib:/sw/xt5/subversion/1.5.0/sle
   s10.1_gnu4.2.4/lib:/opt/xt-
   pe/2.1.50HD/lib:/opt/fftw/3.1.1/cnos/lib:/opt/pgi/7.2.5/linux86-
   64/7.2/libso:/opt/pgi/7.2.5/linux86-64/7.2/lib:/opt/xt-
   os/2.1.50HD/lib:/opt/xt-
   libc/2.1.50HD/amd64/lib:/opt/MySQL/5.0.45/lib/mysql
LS_COLORS=no=00:fi=00:di=01;34:ln=00;36:pi=40;33:so=01;35:do=01;35:bd=40;3
   3;01:cd=40;33;01:or=41;33;01:ex=00;32:*.cmd=00;32:*.exe=01;32:*.com=01;
   32:*.bat=01;32:*.btm=01;32:*.dll=01;32:*.tar=00;31:*.tbz=00;31:*.tgz=00
   ;31:*.rpm=00;31:*.deb=00;31:*.arj=00;31:*.taz=00;31:*.lzh=00;31:*.zip=0
   0;31:*.zoo=00;31:*.z=00;31:*.Z=00;31:*.gz=00;31:*.bz2=00;31:*.tb2=00;31
   :*.tz2=00;31:*.tbz2=00;31:*.avi=01;35:*.bmp=01;35:*.fli=01;35:*.gif=01;
   35:*.jpg=01;35:*.jpeg=01;35:*.mng=01;35:*.mov=01;35:*.mpg=01;35:*.pcx=0
   1;35:*.pbm=01;35:*.pgm=01;35:*.png=01;35:*.ppm=01;35:*.tga=01;35:*.tif=
   01;35:*.xbm=01;35:*.xpm=01;35:*.dl=01;35:*.gl=01;35:*.wmv=01;35:*.aiff=
   00;32:*.au=00;32:*.mid=00;32:*.mp3=00;32:*.ogg=00;32:*.voc=00;32:*.wav=
   00;32:
LC_PE_ENV=pgi
TVDSVRLAUNCHCMD=ssh
XNLSPATH=/usr/X11R6/lib/X11/nls
PGI_VERS_STR=7.2.5
MPICH_DIR=/opt/mpt/3.1.0/xt/mpich2-pgi
ENV=/etc/bash.bashrc
GOTO_NUM_THREADS=1
HOSTTYPE=x86_64
RCLOCAL_PRGENV=true
MPT_VERSION=3.1.0
PE_PRODUCT_LIST=TOTALVIEW:TOTALVIEW-
   SUPPORT:PSPLINE:ASYNCPE:XTMPT:LIBSCI:PGI:LUSTRE:XTPE_QUADCORE:FFTW:PAPI
   :PETSC
FROM_HEADER=
PGI_VERSION=7.2
FFTW_SYSTEM_WISDOM_DIR=/opt/xt-libsci/10.3.1
PAGER=less
PAPI_INCLUDE_OPTS= -I/opt/xt-
   tools/papi/3.6.2/v23/${XTPE_COMPILE_TARGET}/include
OS_DIR=/opt/xt-os/2.1.50HD
CSHEDIT=emacs
PSPLINE_DIR=/sw/xt5/pspline/1.0/cnl2.1_pgi7.2.3
MPICHBASEDIR=/opt/mpt/3.1.0/xt
XDG_CONFIG_DIRS=/usr/local/etc/xdg/:/etc/xdg/:/etc/opt/gnome/xdg/
```

```
PGI=/opt/pgi/7.2.5
MINICOM=-c on
PETSC_INCLUDE_OPTS= -
   I/opt/petsc/2.3.3a/real/${PE_ENV}/${XTPE_COMPILE_TARGET} -
   I/opt/petsc/2.3.3a/real/${PE_ENV}/${XTPE_COMPILE_TARGET}/include
YOD_LOGFILE=syslog
MODULE_VERSION=3.1.6
MAIL=/var/mail/shku
PATH=/sw/xt5/totalview/8.6.0-1/sles10.1_binary/totalview-
   support/1.0.6/bin:/sw/xt5/totalview/8.6.0-
   1/sles10.1_binary/bin:/sw/xt5/totalview/8.6.0-
   1/sles10.1_binary/bin2:/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/bin:/opt/xt-
   tools/papi/3.6.2/v23/linux/bin:/sw/xt5/pspline/1.0/cnl2.1_pgi7.2.3/bin:
   /sw/xt5/netcdf/3.6.2/sles10.1_pgi7.2.3/bin:/sw/xt5/hdf5/1.6.8/cnl2.1_pg
   i7.2.3_par/bin:/sw/xt5/subversion/1.5.0/sles10.1_gnu4.2.4/bin:/opt/cray
   /xt-asyncpe/2.0/bin:/opt/xt-pe/2.1.50HD/bin/snos64:/opt/xt-
   pe/2.1.50HD/cnos/linux/64/bin:/opt/fftw/3.1.1/cnos/bin:/opt/pgi/7.2.5/l
   inux86-64/7.2/bin:/opt/xt-lustre-
   ss/2.1.50HD.PS04.lus.1.6.5.steve.8062_1.6.5/usr/sbin:/opt/xt-lustre-
   ss/2.1.50HD.PS04.lus.1.6.5.steve.8062_1.6.5/usr/bin:/opt/xt-
   boot/2.1.50HD/bin/snos64:/opt/xt-os/2.1.50HD/bin/snos64:/opt/xt-
   service/2.1.50HD/bin/snos64:/opt/xt-
   prgenv/2.1.50HD/bin:/opt/MySQL/5.0.45/etc:/opt/MySQL/5.0.45/libexec:/op
   t/MySQL/5.0.45/bin:/opt/moab/bin:/opt/torque/default/bin:/sw/xt5/bin:/o
   pt/modules/3.1.6/bin:/ccs/home/shku/bin:/usr/local/bin:/usr/bin:/usr/X1
   1R6/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:.:/us
   r/lib/qt3/bin:/opt/bin:/opt/public/bin:/ccs/proj/e2e/wf/bin:/ccs/proj/e
   2e/wf/Workflows/XGC/monitor:/ccs/proj/e2e/wf/bin:/ccs/proj/e2e/wf/Workf
   lows/XGC/monitor
HDF5_CLIB=-I/sw/xt5/hdf5/1.6.8/cnl2.1_pgi7.2.3_par/include -
   L/sw/xt5/hdf5/1.6.8/cnl2.1_pgi7.2.3_par/lib -lhdf5 -
   L/sw/xt5/szip/2.1/sles10.1_pgi7.2.3/lib -lsz -lz
CPU=x86_64
JAVA_BINDIR=/usr/lib/jvm/jre/bin
SSH_SENDS_LOCALE=yes
OCTAVE=sku@depot.cims.nyu.edu:octave
ASYNCPE_VERSION=2.0
GNU_POST_COMPILE_OPTS= -march=barcelona
XTPE_COMPILE_TARGET=linux
RCLOCAL_MYSQL=true
INPUTRC=/etc/inputrc
PWD=/ccs/home/shku/joule/Q2
_LMFILES_=/opt/modulefiles/modules/3.1.6:/sw/xt5/modulefiles/DefApps:/opt/
   modulefiles/torque/2.3.2-
   snap.200807092141:/opt/modulefiles/moab/5.2.4:/opt/cray/xt-
   asyncpe/2.0/modulefiles/xtpe-
   quadcore:/opt/modulefiles/MySQL/5.0.45:/opt/modulefiles/xt-
   service/2.1.50HD:/opt/modulefiles/xt-libc/2.1.50HD:/opt/modulefiles/xt-
   os/2.1.50HD:/opt/modulefiles/xt-boot/2.1.50HD:/opt/modulefiles/xt-
   lustre-
   ss/2.1.50HD.PS04.lus.1.6.5.steve.8062_1.6.5:/opt/modulefiles/xtpe-
   target-cnl:/opt/modulefiles/Base-
   opts/2.1.50HD:/opt/modulefiles/pgi/7.2.5:/opt/modulefiles/fftw/3.1.1:/o
```

```
   pt/modulefiles/xt-libsci/10.3.1:/opt/modulefiles/xt-
   mpt/3.1.0:/opt/modulefiles/xt-pe/2.1.50HD:/opt/modulefiles/xt-
   asyncpe/2.0:/opt/modulefiles/PrgEnv-
   pgi/2.1.50HD:/sw/xt5/modulefiles/subversion/1.5.0:/sw/xt5/modulefiles/h
   df5/1.6.8_par:/sw/xt5/modulefiles/netcdf/3.6.2:/sw/xt5/modulefiles/pspl
   ine/1.0:/opt/modulefiles/xt-
   papi/3.6.2:/opt/modulefiles/petsc/2.3.3a:/sw/xt5/modulefiles/adios/0.9.
   8:/sw/xt5/modulefiles/totalview/8.6.0-1
JAVA_HOME=/usr/lib/jvm/jre
EDITOR=vi
FFTW_INCLUDE_OPTS= -I/opt/fftw/3.1.1/cnos/include
C_DIR=/opt/xt-libc/2.1.50HD
SYSTEM_USERDIR=/tmp/work/shku
LANG=en_US.UTF-8
MODULEPATH=/opt/cray/xt-
   asyncpe/2.0/modulefiles:/opt/modulefiles:/opt/modules/3.1.6:/sw/xt5/mod
   ulefiles
PYTHONSTARTUP=/etc/pythonstart
ADIOS_LIB=-L/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/lib -ladios -lmxml
PETSC_FORTRAN_INCPATH_CNL=-lmpichf90
LOADEDMODULES=modules/3.1.6:DefApps:torque/2.3.2-
   snap.200807092141:moab/5.2.4:xtpe-quadcore:MySQL/5.0.45:xt-
   service/2.1.50HD:xt-libc/2.1.50HD:xt-os/2.1.50HD:xt-boot/2.1.50HD:xt-
   lustre-ss/2.1.50HD.PS04.lus.1.6.5.steve.8062_1.6.5:xtpe-target-
   cnl:Base-opts/2.1.50HD:pgi/7.2.5:fftw/3.1.1:xt-libsci/10.3.1:xt-
   mpt/3.1.0:xt-pe/2.1.50HD:xt-asyncpe/2.0:PrgEnv-
   pgi/2.1.50HD:subversion/1.5.0:hdf5/1.6.8_par:netcdf/3.6.2:pspline/1.0:x
   t-papi/3.6.2:petsc/2.3.3a:adios/0.9.8:totalview/8.6.0-1
PGI_POST_COMPILE_OPTS= -tp barcelona-64
LM_LICENSE_FILE=/sw/sources/totalview/license.dat:/opt/pgi/7.2.5/license.d
   at
XTPE_QUADCORE_ENABLED=ON
MPICH_PTL_UNEX_EVENTS=400000
DEPOT1=sku@depot.cims.nyu.edu:svn/xgc/trunk/XGC1
TEXINPUTS=:/ccs/home/shku/.TeX:/usr/share/doc/.TeX:/usr/doc/.TeX
NETCDF_CLIB=-I/sw/xt5/netcdf/3.6.2/sles10.1_pgi7.2.3/include -
   L/sw/xt5/netcdf/3.6.2/sles10.1_pgi7.2.3/lib -lnetcdf
QT_SYSTEM_DIR=/usr/share/desktop-data
SHLVL=1
HOME=/ccs/home/shku
ADIOS_DIR=/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3
NETCDF_DIR=/sw/xt5/netcdf/3.6.2/sles10.1_pgi7.2.3
PTL_SNOS_NAL=SS
PGI_PATH=/opt/pgi/7.2.5
LESS_ADVANCED_PREPROCESSOR=no
OSTYPE=linux
SE_DIR=/opt/xt-service/2.1.50HD
LIBLUSTRE_DEBUG_CONSOLE=0
LS_OPTIONS=-N --color=tty -T 0
WINDOWMANAGER=
GTK_PATH=/usr/local/lib/gtk-2.0:/opt/gnome/lib/gtk-2.0:/usr/lib/gtk-2.0
PSPLINE_LIB=-I/sw/xt5/pspline/1.0/cnl2.1_pgi7.2.3/mod -
   L/sw/xt5/pspline/1.0/cnl2.1_pgi7.2.3/lib -lpspline -lezcdf -lportlib
G_FILENAME_ENCODING=@locale,UTF-8,ISO-8859-15,CP1252
```

```
LESS=-M -I
MACHTYPE=x86_64-suse-linux
LOGNAME=shku
CIMS=sku@access.cims.nyu.edu:data
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=128.122.81.37 40601 160.91.205.194 22
PETSC_POST_LINK_OPTS= -L
   /opt/petsc/2.3.3a/real/${PE_ENV}/${XTPE_COMPILE_TARGET}/lib -lcraypetsc
   -lHYPRE -lparmetis -lmetis -lcmumps -ldmumps -lsmumps -lzmumps -lpord -
   lsuperlu_3.0 -lsci -lmpich
MODULESHOME=/opt/modules/3.1.6
PKG_CONFIG_PATH=/usr/local/lib/pkgconfig:/usr/local/share/pkgconfig:/usr/l
   ib64/pkgconfig:/usr/share/pkgconfig:/opt/kde3/lib64/pkgconfig:/opt/gnom
   e/lib64/pkgconfig:/opt/gnome/lib64/pkgconfig:/opt/gnome/share/pkgconfig
LESSOPEN=lessopen.sh %s
LIBSCI_BASE_DIR=/opt/xt-libsci/10.3.1
TOTALVIEW_SUPPORT_LIB=/sw/xt5/totalview/8.6.0-1/sles10.1_binary/totalview-
   support/1.0.6/lib
HDF5_FLIB=-module . -module /sw/xt5/hdf5/1.6.8/cnl2.1_pgi7.2.3_par/lib -
   I/sw/xt5/hdf5/1.6.8/cnl2.1_pgi7.2.3_par/include -
   L/sw/xt5/hdf5/1.6.8/cnl2.1_pgi7.2.3_par/lib -lhdf5_fortran -lhdf5 -
   L/sw/xt5/szip/2.1/sles10.1_pgi7.2.3/lib -lsz -lz
LIBSCI_VERSION=10.3.1
INFOPATH=/opt/MySQL/5.0.45/info:/usr/local/info:/usr/share/info:/usr/info:
   /opt/gnome/share/info
TV_EXTRA_OPTIONS=-use_interface ss
NETCDF_FLIB=-module . -module
   /sw/xt5/netcdf/3.6.2/sles10.1_pgi7.2.3/include -
   I/sw/xt5/netcdf/3.6.2/sles10.1_pgi7.2.3/include -
   L/sw/xt5/netcdf/3.6.2/sles10.1_pgi7.2.3/lib -lnetcdf
DISPLAY=localhost:25.0
ADIOS_INC=-I/sw/xt5/adios/0.9.8/cnl2.1_pgi7.2.3/include
PSPLINE_POST_LINK_OPTS=-L/sw/xt5/pspline/1.0/cnl2.1_pgi7.2.3/lib -lpspline
   -lezcdf -lportlib
XAUTHLOCALHOSTNAME=jaguarpf-login1
PETSC_DIR=/opt/petsc/2.3.3a/real/PGI/linux
PE_DIR=/opt/xt-pe/2.1.50HD
LIBSCI_POST_LINK_OPTS= -lsci_quadcore
LESSCLOSE=lessclose.sh %s %s
DEPOT=sku@depot.cims.nyu.edu:scratch/tmp
G_BROKEN_FILENAMES=1
LUSTRE_DIR=/opt/xt-lustre-ss/2.1.50HD.PS04.lus.1.6.5.steve.8062_1.6.5
JAVA_ROOT=/usr/lib/jvm/jre
COLORTERM=1
OLDPWD=/ccs/home/shku/joule
_=/usr/bin/env

Loaded module
Currently Loaded Modulefiles:
```

```
 1) modules/3.1.6
 2) DefApps
 3) torque/2.3.2-snap.200807092141
 4) moab/5.2.4
 5) xtpe-quadcore
 6) MySQL/5.0.45
 7) xt-service/2.1.50HD
 8) xt-libc/2.1.50HD
 9) xt-os/2.1.50HD
10) xt-boot/2.1.50HD
11) xt-lustre-ss/2.1.50HD.PS04.lus.1.6.5.steve.8062_1.6.5
12) xtpe-target-cnl
13) Base-opts/2.1.50HD
14) pgi/7.2.5
15) fftw/3.1.1
16) xt-libsci/10.3.1
17) xt-mpt/3.1.0
18) xt-pe/2.1.50HD
19) xt-asyncpe/2.0
20) PrgEnv-pgi/2.1.50HD
21) subversion/1.5.0
22) hdf5/1.6.8_par
23) netcdf/3.6.2
24) pspline/1.0
25) xt-papi/3.6.2
26) petsc/2.3.3a
27) adios/0.9.8
28) totalview/8.6.0-1
```

APPENDIX E. RAPTOR

E.1 INPUT SETTINGS

```
&solvr
  cfl    = 0.1000000000000000e+00,
  vnn    = 0.0100000000000000e+00,
  ptf    = 1.0000000000000000e+00,
  sor    = 1.0000000000000000e+00,
  src    = 1.0000000000000000e+00,
  tme    = 0.0000000000000000e+00,
  dtm    = 0.0010000000000000e+00,
  nrtitr = 100,
  nrtout = 100,
  nrtprt = 1,
  nrtrew = 5000,
  nptitr = 20,
  nptout = 1,
  nptprt = 1,
  nmpitr = 1000,
  nmpout = 1,
  nmpprt = 1,
  nblkio = 1,
 /
&vtmvl
   cfl_t = 1.0000000000000000e+00,
   vnn_t = 0.1000000000000000e+00,
   rtf_t = 1.0000000000000000e+00,
   dtm_x = 0.0100000000000000e+00,
 /
&lhsqv
   idtrk = 0,
   irkms = 0,
   irkcy = 0,
   iomga = 0,
   islib = 1,
   ivtme = 0,
   inorm = 0,
   iscvc = 0,
   idecc = 0,
   imecp = 0,
 /
&rhsqv
   iturb = 0,
   isgsm = 0,
   ichem = 0,
   ispry = 0,
   ihsrc = 0,
   itvgm = 0,
   iflux = 0,
   ilmeq = 1,
   ilmtr = 2,
```

```
   icsrc = 1,
   ivisc = 1,
 /
&flags
   irsrt = 0,
   ishot = 0,
   ibcnd = 000001,
   ibcwf = 0,
   ilfnc = 333333,
   idbsh = 1,
   itmsh = 1,
   itset = 0,
   iarch = 000000,
   idata = 0,
   ianim = 0,
   istsh = 0,
   ipost = 0,
   icsys = 0,
 /
&refvl
   p_ref = 99300.000000000000e+00,
   T_ref = 294.00000000000000e+00,
 rho_ref = 1.7912921543918080e+00,
   U_ref = 8.0240600000000000e+00,
   L_ref = 8.0000000000000000e-03,
  mu_ref = 8.0402554092161456e-06,
  Cp_ref = 1648.3392027401540e+00,
   c_ref = 250.19015210483540e+00,
   g_ref = 9.8100000000000000e+00,
 /
&refbc
   u_ave = 0.0000000000000000e+00,
   v_ave = 0.0000000000000000e+00,
   w_ave = 0.0000000000000000e+00,
   u_rms = 0.0000000000000000e+00,
   v_rms = 0.0000000000000000e+00,
   w_rms = 0.0000000000000000e+00,
   m_dot = 785.39810000000000e-03,
   S_fac = 0.0000000000000000e+00,
   qwall = 0.0000000000000000e+00,
   f_sto = 0.0000000000000000e+00,
   T_sto = 0.0000000000000000e+00,
   p_tot = 0.0000000000000000e+00,
   T_tot = 0.0000000000000000e+00,
   Rmgas = 0.0000000000000000e+00,
   gCpCv = 0.0000000000000000e+00,
   c_rf0 = 0.0000000000000000e+00,
   M_rf1 = 0.0000000000000000e+00,
   M_rf2 = 0.0000000000000000e+00,
 /
&rftme
   tme_1 = 0.0000000000000000e+00,
   tme_2 = 0.0000000000000000e+00,
   tme_3 = 0.0000000000000000e+00,
```

```
    tme_4 = 0.0000000000000000e+00,
    tme_5 = 0.0000000000000000e+00,
    tme_6 = 0.0000000000000000e+00,
  /
 &rftol
  epsilon_tau = 1.0000000000000000e-04,
  epsilon_inv = 0.1000000000000000e-04,
  epsilon_vis = 0.1000000000000000e-08,
  epsilon_sor = 1.0000000000000000e-16,
  epsilon_tol = 1.0000000000000000e-16,
  epsilon_src = 1.0000000000000000e-16,
  epsilon_chm = 1.0000000000000000e-99,
  /
```

E.2 COMPILATION

RAPTOR was compiled using the default Portland Group Fortran compiler. Output from the compilation is included in Sect. 3.4.7. Here we show only the skeletal output, which includes the options used for optimization of the code. Note that the complete output, which includes all information related to the optimization, is also available but spans 21,807 lines and has thus been omitted in the interest of space. The code was profiled using CrayPAT 4.2 using the following recipe to build the executable:

```
module load xt-craypat
make
pat_build -w -Drtenv=PAT_RT_HWPC=0 DTMS.out DTMS_pat.out.
```

The instrumented executable (DTMS pat.out) was run using the batch script listed in Sect. E.3. The corresponding run time environment is listed in Sect. E.4. Performance data was generated by issuing the commands

```
module load xt-craypat
pat_report DTMS_pat.out+xxxyyy > report.out,
```

where DTMS pat.out+xxxyyy is the name of the directory created by CrayPAT after the run completed. Output from the build is shown below. Here we show only the skeletal output, which includes the options used for optimization of the code. Note that the complete output, which includes all information related to the optimization, is also available but spans 21,807 lines and has thus been omitted in the interest of space.

```
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c MDLS.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c DTMS.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c abrt.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c allocate_grid.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c allocate_xyzh.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c allocate_jcch.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c allocate_jcuh.f90
```

```
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c allocate_jcvh.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c allocate_jcwh.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c allocate_main.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c allocate_halo.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c allocate_lpsh.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c allocate_bcwf.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c grid_MPI.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c gdim.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c mtrc.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c mtrc_scg.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c tvgm.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c tvgm_vlp.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c tvgm_spk.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c pole.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c pole_jcc.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c qref.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c qref_aux.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c init.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c init_aux.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c rsrt_MPI.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c rsrt_aux.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c anim_MPI.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c arch_MPI.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c arch_ascii_MPI.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c arch_ascii_opt.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c qvbt.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c dtdt.f90
```

```
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c vtme.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c dtme.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c dtau.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c norm.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c halo.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c halo_dqv.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c halo_flx.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c halo_jcc.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c halo_jcu.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c halo_jcv.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c halo_jcw.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c halo_qmp.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c halo_sgs.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c halo_qvp.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c halo_src.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c halo_xyz.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c bcnd.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c bcnd_aux.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c bcnd_dqv.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c bcnd_flx.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c bcnd_jcc.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c bcnd_jcu.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c bcnd_jcv.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c bcnd_jcw.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c bcnd_qmp.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c bcnd_sgs.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c bcnd_qvp.f90
```

```
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c bcnd_src.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c bcnd_xyz.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c bcdq.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c bctd.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c inlt.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c inlt_aux.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c estr.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c estr_lib.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c estr_mix.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c estr_Z.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c dqvc.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c dsgs.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c dsgs_c.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c dsgs_m.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c dsgs_r.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c dsgs_t.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c bcwf.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c tble.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c tbls.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c upyp.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c urms.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c ures.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c sgsh.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c rk_1.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c rk_4.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c rk_4J.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c rk_5J.f90
```

```
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c dqdt.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c dddt.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c dflx.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c lmtr.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c scvc.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c sidq.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c sidd.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c mask.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c mocQ.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c flxM.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c flxM_M.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c flxM_C.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c flxM_tvg.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c flxM_L.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c flxM_D.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c flxM_D0.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c flxM_P.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c flxP_C.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c flxQ.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c flxQ_K.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c flxQ_C.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c flxQ_tvg.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c flxQ_L.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c flxQ_D.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c srcM.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c srcM_C.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c srcM_tvg.f90
```

```
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c srcM_tvg_ddt.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c srcM_D.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c srcP.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c srcP_C.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c srcQ.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c srcQ_hv.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c srcQ_tvg.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c dqmp.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c tmsh.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c lgfc.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c dbsh.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c stsh.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c stsh_arch_MPI.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c stsh_arch_ascii_MPI.ftn -
   target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast -
   Minfo=all -DPARALLEL -DSPECIES -c stsh_arch_ascii_opt.ftn -target=linux
   -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast -Minfo=all -
   DPARALLEL -DSPECIES -c stgp.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c stlp.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c lpsh.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c lptd.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c lpdt.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c lpqs.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c lphx.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c lphxM.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c lphxQ.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c lpli.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c lpdf.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c lprk.f90
```

```
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c lpdq.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c lpcf.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c lpbc.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c lppk.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c lpcm.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c lpsh_arch_MPI.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c lpsh_arch_ascii_MPI.ftn -
   target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast -
   Minfo=all -DPARALLEL -DSPECIES -c lpsh_arch_ascii_opt.ftn -target=linux
   -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast -Minfo=all -
   DPARALLEL -DSPECIES -c frmp.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c prof.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c tdst.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c zdst.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c ydis.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c bisc.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c bnbk.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c bndc.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c circ.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c conv.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c cube.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c cycl.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c diam.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c hunt.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c inth.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c li_1.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c li_2.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c li_3.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c lubk.f90
```

```
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c ludc.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c ndev.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c sfcn.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c simp.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c spln.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c thms.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -c udev.f90
ftn -target=linux -Kieee -Mpreprocess -byteswapio -r8 -i4 -fast -Mipa=fast
   -Minfo=all -DPARALLEL -DSPECIES -o DTMS.out MDLS.o DTMS.
```

E.3 BATCH SCRIPT (IDENTICAL FOR BOTH Q2 AND Q4 EXCEPT FOR CORE SIZE)

```
#================================================================ #
# Oak Ridge National Laboratories NCCS Systems. ================= #
#================================================================ #
#PBS -A CSC057
#PBS -N RAPTOR
#PBS -M oefelei@sandia.gov
#PBS -m abe
#PBS -o Std.out
#PBS -e Std.err
#PBS -l walltime=04:00:00,size=47616
set -x
source /opt/modules/default/init/bash
cd $PBS_O_WORKDIR
date
export PAT_RT_HWPC=0
export MPICH_ENV_DISPLAY=1
export MPICH_VERSION_DISPLAY=1
module list
module avail
env
aprun -n 47616 ./dtms.e
#================================================================ #
#================================================================ #
#================================================================ #
```

E.4 RUNTIME ENVIRONMENT

```
LESSKEY=/etc/lesskey.bin
MODULE_VERSION_STACK=3.1.6
FFTW_POST_LINK_OPTS= -L/opt/fftw/3.1.1/cnos/lib -lfftw3 -lfftw3f
MANPATH=/opt/xt-pe/2.1.50HD/papi/man:/opt/mpt/3.1.0/xt/man:/opt/xt-
   libsci/10.3.1/man:/opt/fftw/3.1.1/cnos/man:/opt/pgi/7.2.5/linux86-
   INFODIR=/usr/local/info:/usr/share/info:/usr/info
NNTPSERVER=news
HOSTNAME=jaguarpf-batch4
XKEYSYMDB=/usr/X11R6/lib/X11/XKeysymDB
GNOME2_PATH=/usr/local:/opt/gnome:/usr
_MODULESBEGINENV_=/ccs/home/oefelei/.modulesbeginenv
PATHSCALE_POST_COMPILE_OPTS= -march=barcelona
PE_ENV=PGI
SHELL=/bin/bash
HOST=jaguarpf-batch4
BATCH_ALLOC_COOKIE=0
HISTSIZE=1000
PROFILEREAD=true
XTOS_VERSION=2.1.50HD
PBS_JOBNAME=RAPTOR
MPT_DIR=/opt/mpt/3.1.0/xt
FFTW_INC=/opt/fftw/3.1.1/cnos/include
BATCH_JOBID=71541
```

```
PBS_ENVIRONMENT=PBS_BATCH
MORE=-sl
OLDPWD=/autofs/na1_home/oefelei
QTDIR=/usr/lib/qt3
BOOT_DIR=/opt/xt-boot/2.1.50HD
PBS_O_WORKDIR=/ccs/home/oefelei/scratch/FY09JouleQ2
PRGENV_DIR=/opt/xt-prgenv/2.1.50HD
FFTW_DIR=/opt/fftw/3.1.1/cnos/lib
USER=oefelei
PBS_TASKNUM=1
JRE_HOME=/usr/lib/jvm/jre
GROFF_NO_SGR=yes
BUILD_OPTS=/opt/cray/xt-asyncpe/2.0/bin/build-opts
ASYNCPE_DIR=/opt/cray/xt-asyncpe/2.0
LS_COLORS=
LD_LIBRARY_PATH=/opt/xt-
   pe/2.1.50HD/lib:/opt/fftw/3.1.1/cnos/lib:/opt/pgi/7.2.5/linux86-
   64/7.2/libso:/opt/pgi/7.2.5/linux86-64/7.2/PBS_O_HOME=/ccs/home/oefelei
LC_PE_ENV=pgi
XNLSPATH=/usr/X11R6/lib/X11/nls
TVDSVRLAUNCHCMD=ssh
PBS_NNODES=47616
ENV=/etc/bash.bashrc
MPICH_DIR=/opt/mpt/3.1.0/xt/mpich2-pgi
PGI_VERS_STR=7.2.5
HOSTTYPE=x86_64
GOTO_NUM_THREADS=1
RCLOCAL_PRGENV=true
PBS_MOMPORT=15003
FROM_HEADER=
PE_PRODUCT_LIST=ASYNCPE:XTMPT:LIBSCI:PGI:LUSTRE:XTPE_QUADCORE:FFTW
MPT_VERSION=3.1.0
PAGER=less
FFTW_SYSTEM_WISDOM_DIR=/opt/xt-libsci/10.3.1
PGI_VERSION=7.2
CSHEDIT=emacs
OS_DIR=/opt/xt-os/2.1.50HD
PBS_O_QUEUE=batch
XDG_CONFIG_DIRS=/usr/local/etc/xdg/:/etc/xdg/:/etc/opt/gnome/xdg/
MPICHBASEDIR=/opt/mpt/3.1.0/xt
MINICOM=-c on
PGI=/opt/pgi/7.2.5
PATH=/opt/cray/xt-asyncpe/2.0/bin:/opt/xt-pe/2.1.50HD/bin/snos64:/opt/xt-
   pe/2.1.50HD/cnos/linux/64/bin:/opt/fftw/3.1.1/cnos/bin:/opt/PBS_O_LOGNA
   ME=oefelei
MAIL=/var/spool/mail/oefelei
MODULE_VERSION=3.1.6
YOD_LOGFILE=syslog
PBS_O_LANG=en_US.UTF-8
CPU=x86_64
PBS_JOBCOOKIE=C036D87E89EB27BDBAA67C293634D3AC
JAVA_BINDIR=/usr/lib/jvm/jre/bin
GNU_POST_COMPILE_OPTS= -march=barcelona
ASYNCPE_VERSION=2.0
```

```
PWD=/ccs/home/oefelei/scratch/FY09JouleQ2
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/2.3.2-
   snap.200807092141:/opt/modulefiles/MPICH_VERSION_DISPLAY=1
C_DIR=/opt/xt-libc/2.1.50HD
FFTW_INCLUDE_OPTS= -I/opt/fftw/3.1.1/cnos/include
LANG=en_US.UTF-8
PBS_NODENUM=0
SYSTEM_USERDIR=/tmp/work/oefelei
PYTHONSTARTUP=/etc/pythonstart
MODULEPATH=/opt/cray/xt-
   asyncpe/2.0/modulefiles:/opt/modulefiles:/opt/modules/3.1.6:/sw/xt5/mod
   ulefiles
LOADEDMODULES=modules/3.1.6:DefApps:torque/2.3.2-
   snap.200807092141:moab/5.2.4:xtpe-quadcore:MySQL/5.0.45:xt-
   service/2.1.50HD:xt-libc/PBS_O_SHELL=/bin/bash
PGI_POST_COMPILE_OPTS= -tp barcelona-64
PBS_SERVER=jaguarpf-login2.ccs.ornl.gov
PBS_JOBID=71541.nid17924
XTPE_QUADCORE_ENABLED=ON
LM_LICENSE_FILE=/opt/pgi/7.2.5/license.dat
PAT_RT_HWPC=0
ENVIRONMENT=BATCH
TEXINPUTS=:/ccs/home/oefelei/.TeX:/usr/share/doc/.TeX:/usr/doc/.TeX
HOME=/ccs/home/oefelei
SHLVL=2
QT_SYSTEM_DIR=/usr/share/desktop-data
OSTYPE=linux
LESS_ADVANCED_PREPROCESSOR=no
PGI_PATH=/opt/pgi/7.2.5
PTL_SNOS_NAL=SS
PBS_O_HOST=jaguarpf-login2.ccs.ornl.gov
XCURSOR_THEME=Industrial
LS_OPTIONS=-N --color=none -T 0
LIBLUSTRE_DEBUG_CONSOLE=0
SE_DIR=/opt/xt-service/2.1.50HD
WINDOWMANAGER=
MPICH_ENV_DISPLAY=1
PBS_VNODENUM=0
GTK_PATH=/usr/local/lib/gtk-2.0:/opt/gnome/lib/gtk-2.0:/usr/lib/gtk-2.0
LOGNAME=oefelei
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
BATCH_PARTITION_ID=1
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
PBS_O_MAIL=/var/mail/oefelei
LESSOPEN=lessopen.sh %s
PKG_CONFIG_PATH=/usr/local/lib/pkgconfig:/usr/local/share/pkgconfig:/usr/l
   ib64/pkgconfig:/usr/share/pkgconfig:/opt/kde3/lib64/pkgconfig:/
LIBSCI_BASE_DIR=/opt/xt-libsci/10.3.1
INFOPATH=/opt/MySQL/5.0.45/info:/usr/local/info:/usr/share/info:/usr/info:
   /opt/gnome/share/info
LIBSCI_VERSION=10.3.1
LESSCLOSE=lessclose.sh %s %s
LIBSCI_POST_LINK_OPTS= -lsci_quadcore
PE_DIR=/opt/xt-pe/2.1.50HD
PBS_NODEFILE=/var/spool/torque/aux//71541.nid17924
G_BROKEN_FILENAMES=1
PBS_O_PATH=/opt/xt-tools/craypat/4.4.0.4/v23/bin:/opt/cray/xt-
   asyncpe/2.0/bin:/opt/xt-pe/2.1.50HD/bin/snos64:/opt/xt-
   pe/2.1.50HD/cnos/COLORTERM=1
JAVA_ROOT=/usr/lib/jvm/jre
LUSTRE_DIR=/opt/xt-lustre-ss/2.1.50HD.PS04.lus.1.6.5.steve.8062_1.6.5
_=/usr/bin/env
------------------------------------------------------------------------
++ source /opt/modules/default/init/bash
+++ '[' 3.1.6 = '' ']'
+++ MODULE_VERSION_STACK=3.1.6
+++ export MODULE_VERSION_STACK
+++ MODULESHOME=/opt/modules/3.1.6
+++ export MODULESHOME
+++ '[' modules/3.1.6:DefApps:torque/2.3.2-
   snap.200807092141:moab/5.2.4:xtpe-quadcore:MySQL/5.0.45:xt-
   service/2.1.50HD:xt-libc/2.1.50HD:+++ '[' /opt/cray/xt-
   asyncpe/2.0/modulefiles:/opt/modulefiles:/opt/modules/3.1.6:/sw/xt5/mod
   ulefiles = '' ']'
++ cd /ccs/home/oefelei/scratch/FY09JouleQ2
++ date
++ export PAT_RT_HWPC=0
++ PAT_RT_HWPC=0
++ export MPICH_ENV_DISPLAY=1
++ MPICH_ENV_DISPLAY=1
++ export MPICH_VERSION_DISPLAY=1
++ MPICH_VERSION_DISPLAY=1
++ module list
+++ /opt/modules/3.1.6/bin/modulecmd bash list
Currently Loaded Modulefiles:
1) modules/3.1.6
2) DefApps
3) torque/2.3.2-snap.200807092141
4) moab/5.2.4
5) xtpe-quadcore
6) MySQL/5.0.45
7) xt-service/2.1.50HD
8) xt-libc/2.1.50HD
```

```
9) xt-os/2.1.50HD
10) xt-boot/2.1.50HD
11) xt-lustre-ss/2.1.50HD.PS04.lus.1.6.5.steve.8062_1.6.5
12) xtpe-target-cnl
13) Base-opts/2.1.50HD
14) pgi/7.2.5
15) fftw/3.1.1
16) xt-libsci/10.3.1
17) xt-mpt/3.1.0
18) xt-pe/2.1.50HD
19) xt-asyncpe/2.0
20) PrgEnv-pgi/2.1.50HD
++ eval
++ module avail
+++ /opt/modules/3.1.6/bin/modulecmd bash avail
------------------ /opt/cray/xt-asyncpe/2.0/modulefiles ------------------
xtpe-quadcore xtpe-target-native
---------------------------- /opt/modulefiles ----------------------------
Base-opts/2.1.27HD
Base-opts/2.1.27HD.lusrelsave
Base-opts/2.1.29HD
Base-opts/2.1.29HD.lusrelsave
Base-opts/2.1.41HD
Base-opts/2.1.41HD.lusrelsave
Base-opts/2.1.50HD(default)
Base-opts/2.1.50HD.lusrelsave
MySQL/5.0.45
PrgEnv-cray/1.0.0(default)
PrgEnv-gnu/2.1.27HD
PrgEnv-gnu/2.1.29HD
PrgEnv-gnu/2.1.41HD
PrgEnv-gnu/2.1.50HD(default)
PrgEnv-pathscale/2.1.27HD
PrgEnv-pathscale/2.1.29HD
PrgEnv-pathscale/2.1.41HD
PrgEnv-pathscale/2.1.50HD(default)
PrgEnv-pgi/2.1.27HD
PrgEnv-pgi/2.1.29HD
PrgEnv-pgi/2.1.41HD
PrgEnv-pgi/2.1.50HD(default)
acml/4.0.1a
acml/4.1.0(default)
acml/4.2.0
apprentice2/4.3.0
apprentice2/4.4.0(default)
apprentice2/4.4.0.1
blcr/0.7.3
cce/7.0.0(default)
cce/7.0.1
cray/audit/1.0.0-1.0000.15784.0
dwarf/8.2.0
dwarf/8.4.0
dwarf/8.6.0
dwarf/8.8.0(default)
```

```
elf/0.8.10(default)
fftw/2.1.5
fftw/3.1.1(default)
gcc/4.1.2
gcc/4.2.0.quadcore(default)
gcc/4.2.3
gcc/4.2.4
gcc-catamount/3.3
gnet/2.0.5
iobuf/1.0.6(default)
java/jdk1.6.0_05(default)
libfast/1.0(default)
libfast/1.0.2
libscifft-pgi/1.0.0(default)
moab/5.2.3
moab/5.2.4(default)
moab/5.3.0
modules/3.1.6(default)
pathscale/3.2(default)
petsc/2.3.3a(default)
petsc-complex/2.3.3a(default)
pgi/6.2.5
pgi/7.0.7
pgi/7.1.6
pgi/7.2.3
pgi/7.2.4
pgi/7.2.5(default)
pgi/8.0.1
pgi/8.0.2
pkgconfig/0.15.0(default)
torque/2.3.2-snap.200807092141(default)
xt-asyncpe/1.0c
xt-asyncpe/1.1
xt-asyncpe/1.2
xt-asyncpe/2.0(default)
xt-asyncpe/2.0.34
xt-boot/2.1.27HD
xt-boot/2.1.29HD
xt-boot/2.1.41HD
xt-boot/2.1.50HD
xt-catamount/2.1.27HD
xt-catamount/2.1.29HD
xt-catamount/2.1.41HD
xt-catamount/2.1.50HD
xt-craypat/4.3.1
xt-craypat/4.3.3
xt-craypat/4.4.0
xt-craypat/4.4.0.2
xt-craypat/4.4.0.4(default)
xt-libc/2.1.27HD
xt-libc/2.1.29HD
xt-libc/2.1.41HD
xt-libc/2.1.50HD
xt-libsci/10.2.1
```

```
xt-libsci/10.3.0
xt-libsci/10.3.1(default)
xt-libsci/10.3.2
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_1.6.5
xt-lustre-ss/2.1.50HD_PS04_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-mpt/2.1.27HD
xt-mpt/2.1.29HD
xt-mpt/2.1.41HD
xt-mpt/2.1.50HD
xt-mpt/3.0.1
xt-mpt/3.0.2
xt-mpt/3.0.4
xt-mpt/3.1.0(default)
xt-mpt/3.1.0.4
xt-mpt/3.1.0.6
xt-mpt/3.1.0.7
xt-os/2.1.27HD
xt-os/2.1.29HD
xt-os/2.1.41HD
xt-os/2.1.50HD
xt-papi/3.5.99c
xt-papi/3.6
xt-papi/3.6.1a
xt-papi/3.6.2(default)
xt-pe/2.1.27HD
xt-pe/2.1.29HD
xt-pe/2.1.41HD
xt-pe/2.1.50HD
xt-service/2.1.27HD
xt-service/2.1.29HD
xt-service/2.1.41HD
xt-service/2.1.50HD
xtgdb/1.0.0(default)
xtpe-target-catamount
xtpe-target-cnl
---------------------------- /opt/modules/3.1.6 ----------------------------
```

```
modulefiles/modules/dot modulefiles/modules/modules
modulefiles/modules/module-cvs modulefiles/modules/null
modulefiles/modules/module-info modulefiles/modules/use.own
-------------------------- /sw/xt5/modulefiles ---------------------------
DefApps lapack/3.1.1-dualcore
MiscApps lapack/3.1.1-fPIC
adios/0.9.8(default) liblut/0.9.6
arpack/2008.03.11 m4/1.4.11
atlas/3.8.2 matlab/7.5
atlas/3.8.2-fPIC-dualcore mercurial/1.0.2
autoconf/2.63 metis/4.0
automake/1.10.1 mpe2/1.0.6
aztec/2.1 mpip/3.1.2
blas/ref(default) mumps/4.7.3_par
blas/ref-dualcore namd/2.6
bugget/2.0 ncl/5.0.0
cmake/2.6.1(default) nco/3.9.4
cmake/2.6.2 ncview/1.93c
cpmd/3.13.1 nedit/5.5
cpmd/3.13.2 netcdf/3.6.2(default)
doxygen/1.5.6 netcdf/4.0.0
doxygen/1.5.8 netcdf/4.0.0_par
ferret/6.1 ompi/ADTR65
fftpack/5-r4i4 ompi/ADTR77
fftpack/5-r8i4 ompi/ADTR78
fftpack/5-r8i8 ompi/DTR56
fftw/3.1.2 ompi/DTR59
fftw/3.1.2-dualcore ompi/routing-pgi
fftw/3.2 p-netcdf/1.0.2(default)
fftw/3.2-dualcore p-netcdf/1.0.3
fpmpi/1.0 parmetis/3.1
fpmpi/1.1 petsc/2.3.3-debug
fpmpi_papi/1.0 petsc-complex/2.3.3-debug
fpmpi_papi/1.1 pgplot/5.2
gamess/2008Mar04 pspline/1.0
git/1.6.0 python/2.5.2
git/1.6.0.4 python/2.5.2-netcdf
globalarrays/4.0.8 qt/4.3.4
gnuplot/4.2.3 ruby/1.8.7
gnuplot/4.2.4(default) ruby/1.9.1
gptl/3.4.1 spdcp/0.3.6
gptl/3.4.3 sprng/2.0b
gptl/3.4.7(default) stagesub/1.0.2
grace/5.1.21 stagesub/1.0.3(default)
gromacs/3.3.3 subversion/1.4.6
gsl/1.11 subversion/1.5.0(default)
gsl/1.11-dualcore sundials/2.3.0
hdf5/1.6.7(default) superlu/3.0
hdf5/1.6.7_par superlu_dist/2.2
hdf5/1.6.8 swig/1.3.36
hdf5/1.6.8_par szip/2.1
hdf5/1.8.1 tau/2.17.2
hdf5/1.8.1_par tau/2.17.3
hdf5/1.8.2 tkdiff/4.1.4
```

```
hdf5/1.8.2_par totalview/8.6.0-1(default)
hypre/2.0.0 trilinos/8.0.3
idl/6.4 udunits/1.12.4
imagemagick/6.4.2(default) udunits/1.12.9
java-jdk/1.5.0.06 umfpack/5.1.1
java-jdk/1.6.0.06 valgrind/3.3.1
java-jre/1.5.0.06 vim/7.1
lammps/4Mar08 vim/7.2
lammps/May08 visit/1.11.1
lapack/3.1.1(default)
++ eval
++ env
++ aprun -n 47616 ./dtms.e
MPI VERSION : CRAY MPICH2 XT version 3.1.0 (ANL base 1.0.6)
BUILD INFO : Built Thu Nov 20 11:14:12 2008 (svn rev 7246)
PE 0: MPICH environment settings:
PE 0: MPICH_ENV_DISPLAY = 1
PE 0: MPICH_VERSION_DISPLAY = 1
PE 0: MPICH_ABORT_ON_ERROR = 0
PE 0: MPICH_CPU_YIELD = 0
PE 0: MPICH_RANK_REORDER_METHOD = 1
PE 0: MPICH_RANK_REORDER_DISPLAY = 0
PE 0: MPICH_MAX_THREAD_SAFETY = single
PE 0: MPICH_MSGS_PER_PROC = 16384
PE 0: MPICH/SMP environment settings:
PE 0: MPICH_SMP_OFF = 0
PE 0: MPICH_SMPDEV_BUFS_PER_PROC = 32
PE 0: MPICH_SMP_SINGLE_COPY_SIZE = 131072
PE 0: MPICH_SMP_SINGLE_COPY_OFF = 0
PE 0: MPICH/PORTALS environment settings:
PE 0: MPICH_MAX_SHORT_MSG_SIZE = 4301
PE 0: MPICH_UNEX_BUFFER_SIZE = 142848000
PE 0: MPICH_PTL_UNEX_EVENTS = 104755
PE 0: MPICH_PTL_OTHER_EVENTS = 11904
PE 0: MPICH_VSHORT_OFF = 0
PE 0: MPICH_MAX_VSHORT_MSG_SIZE = 1024
PE 0: MPICH_VSHORT_BUFFERS = 32
PE 0: MPICH_PTL_EAGER_LONG = 0
PE 0: MPICH_PTL_MATCH_OFF = 0
PE 0: MPICH_PTL_SEND_CREDITS = 0
PE 0: MPICH/COLLECTIVE environment settings:
PE 0: MPICH_FAST_MEMCPY = 0
PE 0: MPICH_COLL_OPT_OFF = 0
PE 0: MPICH_COLL_SYNC = 0
PE 0: MPICH_BCAST_ONLY_TREE = 1
PE 0: MPICH_ALLTOALL_SHORT_MSG = 1024
PE 0: MPICH_REDUCE_SHORT_MSG = 65536
PE 0: MPICH_ALLREDUCE_LARGE_MSG = 262144
PE 0: MPICH_ALLGATHER_VSHORT_MSG = 2048
PE 0: MPICH_ALLTOALLVW_FCSIZE = 32
PE 0: MPICH_ALLTOALLVW_SENDWIN = 20
PE 0: MPICH_ALLTOALLVW_RECVWIN = 20
PE 0: MPICH/MPIIO environment settings:
PE 0: MPICH_MPIIO_HINTS_DISPLAY = 0
```

```
PE 0: MPICH_MPIIO_CB_ALIGN = 0
PE 0: MPICH_MPIIO_HINTS = NULL
CrayPat/X: Version 4.4.0 Revision 2195 10/29/08 14:13:53
Experiment data directory written:
/lustre/scratch/oefelei/FY09JouleQ2/dtms.e+25388-13808tdt
```

E.5 COMPARISON OF TOTAL RUN TIME VS INITIALIZATION TIME

In running the Q4 case, we observed an anomaly associated with the time required for the initialization stage of the calculation (which is not compute intensive) compared to the integration stage (which is compute intensive). This anomaly was traced to CrayPAT. In all cases, executables that were instrumented with CrayPAT exhibited a wide range of initialization times compared to those that were not. In the results for Q2, for example, the total run time reported by CrayPAT was 1,423 seconds, as shown in Table E.1. However, the time spent in the integration part of the calculation, as given by the internal timer in the code was only 1,034 seconds, which implies that approximately 389 seconds were required for initialization. To verify this we reran the Q2 case with the integration loop bypassed to isolate the time associated with initialization. Results from this run are provided shown in Table E.2. Comparing these data verifies that a negligible amount of floating point operations occurred during initialization for the selected cases and that the internal timer used to measure the amount of time spent in the integrator was accurate. As a second test, we ran both the Q2 benchmark and Q4 cases without CrayPAT installed and verified that the initialization times for both became negligible (i.e., less than 10 percent of the total integration time). The combined set of tests confirmed that the integration times and estimated floating point operation rates reported are accurate.

Table E.1. Counter data acquired from CrayPAT 4.2 for Q2 benchmark run using RAPTOR

```
Totals for program
----------------------------------------------------------------------------
  Time%                                          100.0%
  Time                                        1425.761880 secs
  Imb.Time                                             -- secs
  Imb.Time%                                            --
  Calls                          0.0 /sec            4.0 calls
  PAPI_L1_DCM                  20.674M/sec    26457314029 misses
  PAPI_TOT_INS               3379.668M/sec  4325136094614 instr
  PAPI_L1_DCA                1348.943M/sec  1726311709236 refs
  PAPI_FP_OPS                   6.204M/sec     7939032813 ops
  User time (approx)         1279.752 secs  2943428628772 cycles  89.8%Time
  Average Time per Call                        356.440470 sec
  CrayPat Overhead : Time    0.0%
  HW FP Ops / User time         6.204M/sec     7939032813 ops  0.1%peak(DP)
  HW FP Ops / WCT               5.568M/sec
  HW FP Ops / Inst                                    0.2%
  Computational intensity    0.00 ops/cycle           0.00 ops/ref
  Instr per cycle                                     1.47 inst/cycle
  MIPS                 160926295.28M/sec
  MFLOPS (aggregate)       295389.35M/sec
  Instructions per LD & ST  39.9% refs                2.51 inst/ref
  D1 cache hit,miss ratios  98.5% hits                1.5% misses
```

```
D1 cache utilization (M)   65.25 refs/miss     8.156 avg uses
```

Table E.2. Counter data acquired from CrayPAT 4.2 for Q2 benchmark run using RAPTOR but with the integration loop bypassed

```
Totals for program
---------------------------------------------------------------------------
  Time%                                          100.0%
  Time                                        398.057434 secs
  Imb.Time                                            -- secs
  Imb.Time%                                           --
  Calls                        0.0 /sec            4.0 calls
  PAPI_L1_DCM                23.802M/sec     8862182513 misses
  PAPI_TOT_INS             3505.363M/sec  1305131854712 instr
  PAPI_L1_DCA              1393.542M/sec   518849673837 refs
  PAPI_FP_OPS                 0.002M/sec         574298 ops
  User time (approx)        372.324 secs   856345890248 cycles  93.5%Time
  Average Time per Call                       99.514358 sec
  CrayPat Overhead : Time    0.0%
  HW FP Ops / User time       0.002M/sec         574298 ops  0.0%peak(DP)
  HW FP Ops / WCT             0.001M/sec
  HW FP Ops / Inst                                  0.0%
  Computational intensity    0.00 ops/cycle         0.00 ops/ref
  Instr per cycle                                   1.52 inst/cycle
  MIPS                 166911368.33M/sec
  MFLOPS (aggregate)          73.45M/sec
  Instructions per LD & ST  39.8% refs              2.52 inst/ref
  D1 cache hit,miss ratios  98.3% hits              1.7% misses
  D1 cache utilization (M)  58.55 refs/miss        7.318 avg uses
```