A Multi-Faceted Approach to Job Placement for Improved Performance on Extreme-Scale Systems

Christopher Zimmer, Saurabh Gupta, Scott Atchley, Sudharshan S. Vazhkudai
Oak Ridge Leadership Computing Facility
Oak Ridge National Laboratory
{zimmercj,guptas1,atchleyes,vazhkudaiss}@ornl.gov

Carl Albing
United States Naval Academy
Annapolis, Maryland
albing@usna.edu

***Abstract*—Job placement plays a pivotal role in application performance on supercomputers. We present a multi-faceted exploration to influence placement in extreme-scale systems, to improve network performance and decrease variability. In our first exploration, Scores, we developed a machine learning model that extracts features from a job's node-allocation and grades performance. This identified several important node-metrics that led to Dual-Ended scheduling, a means of reducing network contention without impacting utilization. In evaluations on the Titan supercomputer, we observed reductions in average hop-count by up to 50%. We also developed an improved node-layout strategy that targets a better balance between network latency and bandwidth, replacing the default ALPS layout on Titan that resulted in an average of 10% runtime improvement. Both of these efforts underscore the importance of a job placement strategy that is cognizant of workload mixture and network topology.**

I. Introduction

The Oak Ridge Leadership Computing Facility (OLCF) operates Titan, the No. 2 machine on the Top500 [1] list and the largest open science supercomputer in the world. Titan is a Cray XK-7 with 18,688 compute nodes, each of which provides 16 CPUs and a GPU. The nodes are connected via Cray's Gemini interconnect, using a 3D torus topology. Titan serves a variety of users from the U.S. Department of Energy (DOE) labs, academia, and even private industry. User jobs range from small jobs using a handful of nodes to *capability* jobs using a large portion, if not all, of the machine.

The mixture of the workload and the manner in which jobs are placed in the system can impact job performance resulting in runtime variability. For capability jobs, this manifests when having to allocate resources for the job from an already busy machine. When the job is scheduled, the system must use the nodes of jobs that previously finished, to fit the requesting job. It is often the case that incoming capability jobs will be broken into many fragments and spread throughout the machine to facilitate its reservation. Additionally, even if the job results in a single contiguous allocation, the placement of the job upon the Gemini topology may impede the performance of the network communication of the job due to the internal communication requirements.

There are many strategies for achieving high-performance allocations in extreme-scale computing. These generally come at the cost of system utilization as they are achieved by holding jobs until a suitable allocation becomes available or through over-provisioning. These solutions are not suitable for Titan because of the requirement for running large jobs and maintaining high utilization. The DOE requires OLCF to ensure that a significant portion of Titan's compute hours are consumed by capability jobs, which are defined as using 20% or more of Titan's nodes. Also, a major measure of Titan's effectiveness is the utilization, which is the ratio of consumed compute hours to available compute hours. Titan averages a high utilization above 90%.

Contributions: In this paper, we present our efforts to improve performance without impacting utilization or capability metrics. Using visual analysis and machine learning we evaluated several features of job placement. During this study, we identified a set of primary job node-allocation features that correlate to performance. In one outcome, we found that high hop-count jobs were often highly fragmented. Much of this fragmentation is caused by gaps in the resource list during scheduling. Using this knowledge, we developed a mechanism called Dual-Ended scheduling. This strategy targets a reduction of runtime variability for large jobs. It achieves this by scheduling large and small jobs from opposite ends of the resource lists, thereby reducing the number of partitions or fragments for a large job. Applying this technique, we have seen consistently reduced hop-counts, a strong indicator for network performance. In evaluations on Titan, we observed reductions in average hop-count, for common job sizes, up to 50% in relation to the measured job-size minimum. In another approach, we used this analysis to identify potential pitfalls with the default placement of jobs on Titan, and developed new techniques for improving the node-layout generated by the Application Layout Placement Scheduler (ALPS). One particular layout targeting a better balance of latency and bandwidth on the Gemini network resulted in an average of 10% runtime improvement in our evaluation.

In section II, we discuss in more detail the issues with scheduling on a 3D torus and their impacts on variability. We then discuss how visual analysis led us to understand the scheduling impacts in section III. In section IV, we discuss how we identified the characteristics that impact performance the most on Titan. Using these results, in section V, we propose a modification to the scheduling algorithm to amelio-

SC16; Salt Lake City, Utah, USA; November 2016
978-1-4673-8815-3/16/$31.00 ©2016 IEEE

rate job fragmentation's impact on variability and we review the impacts on the actual system over several months. In section VI, we dive into the details of how the scheduler views the node ordering and how altering the ordering can improve performance without negatively impacting utilization. In section VII, we discuss related work in scheduling. We summarize and conclude our effort in section VIII.

II. Background and Motivation

Performance variability of user jobs on Titan resulted in an investigation of the scheduling stack. It was during this investigation that it became apparent that job placement can significantly impact its network performance. Resource allocation is the final step in scheduling a job. It is during this step that the set of physical resources, compute nodes that a job will use, are determined. Since compute nodes are bound to specific network locations within the supercomputer, it is apparent that the characteristic of the resources selected will play a role in the network performance of that job.

To understand how placement can influence network performance on Titan, we must first understand the high-performance Gemini [2] network. Titan's Gemini network is a 3D torus with XYZ dimensions of 25x16x24. Bandwidth on the network is dimensionally anisotropic, the Z-dimension is generally composed of in-rack backplane providing up to 15 GB/s, every 8th connection is a copper cable link between racks providing 9.36 GB/s. The X-dimension is composed entirely of copper cable and performing at 9.36 GB/s. The Y-dimension alternates between mezzanine and cable. The Y-dimension differs from the other two dimensions in that cables are only trunked in groups of 4 instead of 8. This leads 1/2 of the Y hops performing at 9.36 GB/s and 1/2 performing at 4.68 GB/s. Gemini, being a 3D torus network is a high hop-count network. It is composed of 9,600 Gemini routers, each connecting two nodes to the network. This creates a total of 19,200 nodes, with 18,688 compute and 512 service nodes. The property of high-dimensionality in 3D torus networks means that they are composed of many small routing devices. This generally is favorable economically, maintaining high-bisection bandwidth with many routing decisions possible due to the high router count. This also comes at a cost of the worst-case distance a message will have to travel. For instance, on a 3-level fat-tree, messages exchanged through core-routers will have a bounded hop-distance of 5 hops. On a torus with Titan's dimensions, worst case messages could experience 12 hops in the X-dimension, 8 hops in the Y-dimension, and 12 hops in the Z-dimension for a total of 32 hops for the worst possible case routing. This adds significant latency and exposes the message to multiple points of geometric digression which significantly limits bandwidth. Geometric digression is a result of the local fairness properties used in Gemini Routers. When flows converge on a router, message arbitration effectively subdivides each competing flow. The more hops/routers a message has to travel the more potential digression the flow could potentially experience. This digression causes reduced bandwidths along contended paths.

III. Visual Analysis

Based on the above analysis and understanding of the Gemini network, it is clear that placement will have an impact on the application network performance. However, the manner in which jobs are placed in Titan was obfuscated within complex log entries. Resource log entries containing node mappings are often several megabytes in size. Even knowing the nodes of a job was not enough to understand the mapping of the coordinates into the 3D torus. Many impacts of node allocation were not clear until we engaged in visual analysis. To understand the impact of job placement on Titan we developed a model based on TorusVis [3], that enabled us to extract job information from MOAB [4] logs pushed into Splunk [5]. This model represents the 9,600 Gemini routers used in Titan. Each node maps to a set of coordinates in the 3D space. Using this mapping we are able to map the placement of the job to the physical set of resources within the 3D torus network. From studying many jobs over the course of several months, we were able to identify troubling outcomes directly caused by resource allocation.

A. Ordering

Figure 1 shows the manner in which jobs are mapped to resources on Titan. In this example, we see the first job being mapped to Titan during a test-shot, meaning the system is empty. This allocation represents the ideal situation for the scheduler in that the first set of resources checked for this job, were the first set of resources received. However, the way this job is mapped to Titan from a network perspective seems counter-intuitive. The allocation appears to favor the Z-dimension, Y-dimension, and then the X-dimension. While prioritizing the Z-dimension makes sense, as the Z links have both high-bandwidth and low-latency, however, the Y-dimension contains the lowest bandwidth links in the entire system. This would indicate that any communication traveling in the Y-dimension would be subjected to both geometric performance digression due to the large Y-distance within the job, as well as bandwidth degradation from crossing over reduced bandwidth links.

B. Fragmentation

A second concern observed from a visual analysis of Titan scheduling is that many jobs end up being scheduled with fragmented node allocations. Figure 2 shows the layout of a single job extracted from MOAB log data. As can be seen, this job has been fragmented into several discrete node groups. Considering the manner in which the job is distributed throughout Titan, communication in this job could potentially suffer increased latency due to larger hop-counts between nodes and potentially decreased bandwidth from increased exposure to geometric digression. The consequences of geometric digression on this job may be worse than in a case of a contiguous job allocation, due to the potential of having to communicate across links and routers used by other jobs. In a highly fragmented job as this, it is highly likely that communication will run between the various partitions of the

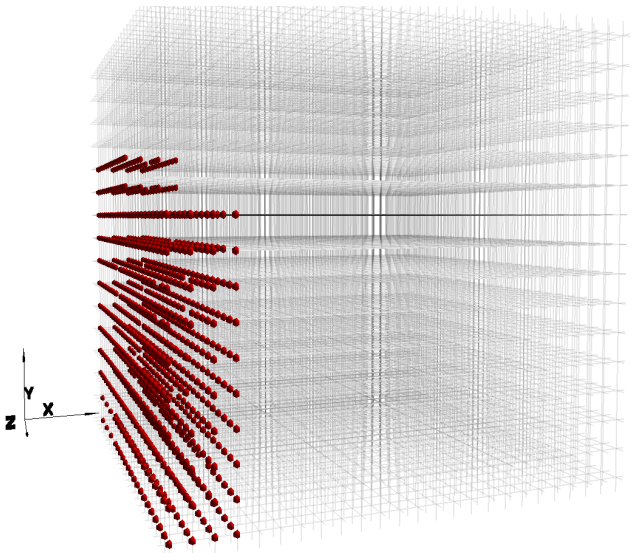

Fig. 1. Default Alps layout of a job on Titan. Dimension build priority: Z,Y,X. Increased height in the Y-dimension exposes the intra-job communication to reduced bandwidth.

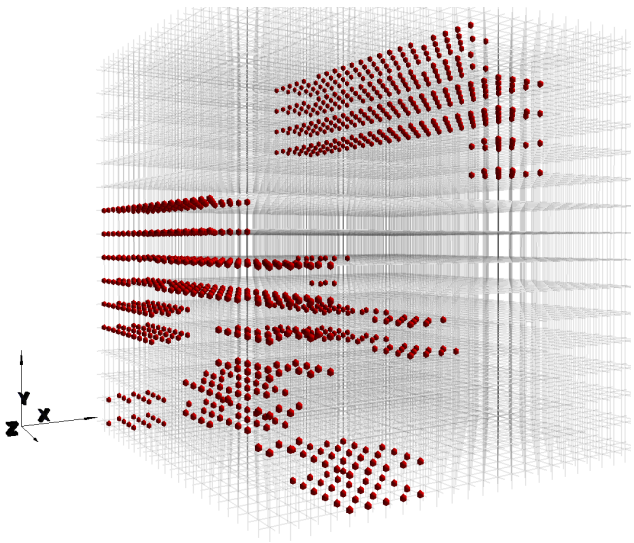

Fig. 2. Default Alps layout of a job on Titan. Highly fragmented allocation leads to increased runtime variability

job, and will have to contend with the communication of other jobs along those links.

Using visual analysis of the mapping of jobs to the system, we were able to observe several patterns that could negatively impact a job. This lead to the development of Scores that enabled us to devise a set metrics to quantify the layout of a job in the system, and then using machine learning techniques determine which of these features were strong predictors of job performance.

IV. Scores

In this section, we show how the density of the allocation affects the performance of our benchmark applications, and

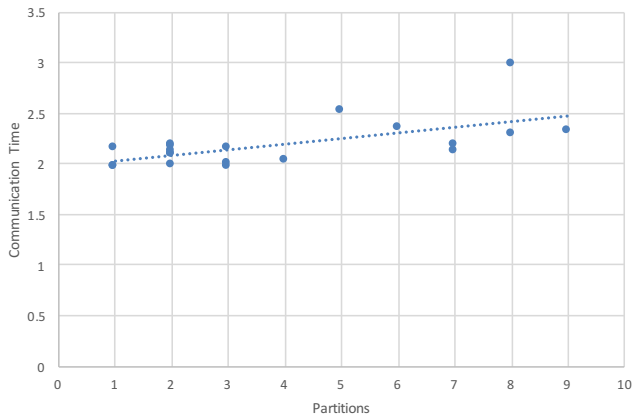

Fig. 3. IS.D.256: Average Communication time vs. Number of Partitions

what characteristics are desired in an allocation on Titan to obtain higher performance. We describe our data collection methodology, benchmark selection, machine learning based analysis, correlation results, and insights.

A. Benchmarks and Methodology

We choose the NAS benchmarks [6] for our experiments because they are well studied and show different communication patterns (all-to-all, neighbor exchange etc.). Table I describes the configuration used and the key properties of these benchmarks [7] we used in our study. In our experiments, we conduct several runs of each application on Titan in different locations (as allocated by the scheduler). The application runtime, communication time, and nodes used by the application are recorded for the analysis.

B. Regression Analysis

In our first steps, we analyzed the correlation between the observed communication times of our benchmarks with various properties of their node allocation such as average hop-count and number of partitions. This analysis drives the feature selection for our machine learning based analysis and results. In Figures 3, 4, and 5 we show scatter plots of the average communication time of benchmarks IS and MG with respect to the number of partitions and the average hop-count. The number of partitions shows the number of disconnected partitions within an allocation. Here, a partition is defined as a group of nodes such that each node is connected to at least one node among the remaining nodes, with a hop-distance of one. Average hop-count, a measure of intra-job hop-distance, is calculated assuming an all to all communication pattern. Although there exists significant variation in communication time for a given number of partitions or average hop-count, the overall trend shows that a high number of partitions and a high average hop-count results in increased communication times. A similar effect is visible for several benchmarks that show their sensitivity to their placement in the 3D torus (results not presented here due to space constraint).

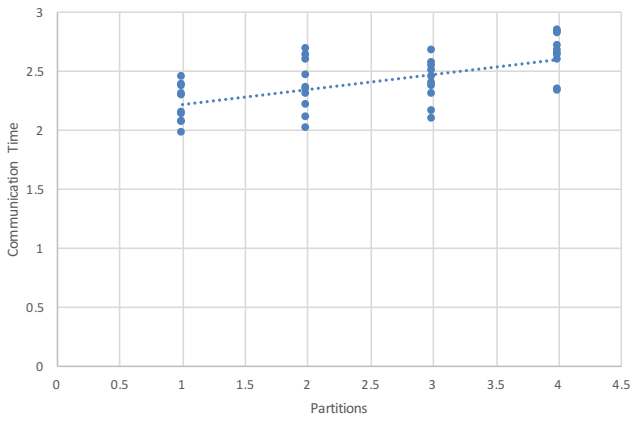

Fig. 4. MG.D.256: Average Communication time vs. Number of Partitions

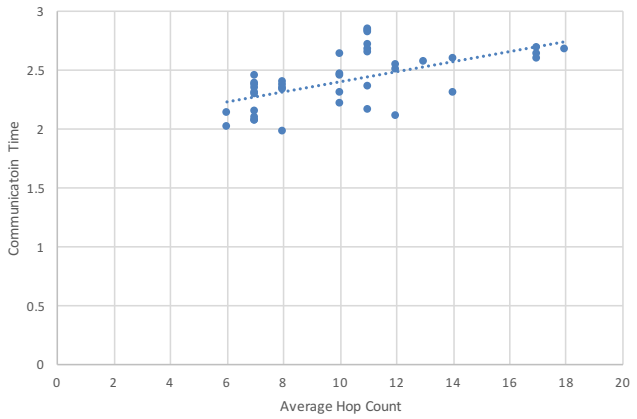

Fig. 5. MG.D.256: Average Communication time vs. Average Hop-Count

TABLE I
NAS BENCHMARKS USED IN THIS STUDY.

Benchmark	Configuration	Communication Pattern
IS	256 nodes	Collective (AlltoAll/AlltoAllv)
CG	256, 1024 nodes	Partitioned Group Exchanges
FT	256 nodes	AlltoAll and Neighbor Exchanges
MG	256 nodes	2D Neighbor Communication

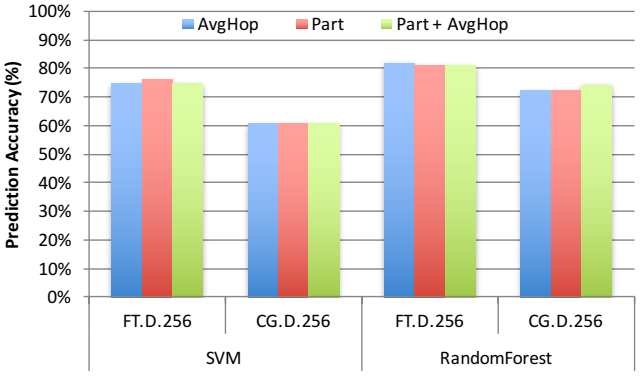

Fig. 6. Machine learning based prediction results

C. Machine Learning based Analysis

We also utilize machine learning to understand how different features of the allocation contribute towards the performance of the application. Using these techniques we try to predict the performance of each benchmark based on its allocation. The performance prediction does not involve predicting the exact execution time but instead attempts to predict the bucket in which the application performance is likely to be in. We use five grade buckets, A-F, based on mean and standard deviation. We use and compare two machine learning techniques, Support Vector Machines (SVM) [8] and Random Forest [9], both selected for their strength in categorization selection. It should be noted that our aim here is not to predict the performance, but to understand which features of the allocation affect performance significantly.

Figure 6 shows the results for FT and CG benchmarks. We divide the data into training and testing sets, and report the results of the testing set. The average hop-count and the number of partitions features are able to predict the performance bucket with 70-80% accuracy. For CG, SVM does not show good prediction accuracy while RandomForest captures the relationship better. When we combine the two features we do not obtain any significant improvement in accuracy. This is expected because the average hop-count is highly correlated with the number of partitions (as the allocation is divided into several partitions, the all to all communication hop-count increases).

Moreover, we also investigated instantaneous congestion on the Gemini routers that connect the compute nodes to the 3D torus. In our experiments, we saw no benefit in prediction accuracy when using features based on instantaneous congestion. Therefore, we conclude that average hop-count and the number of partitions should be the primary focus for higher performance. Overall, the key takeaway is that improving the packing of the job allocation on the 3D torus is an effective way of improving the application's performance. This has also been observed by some previous studies such as [10][11].

V. DUAL-ENDED SCHEDULING

In section III-B, we discussed job fragmentation as a cause of increased communication distances, resulting from the increase in the internal hop-count for communication within the job. We then demonstrated in Section IV that an increase in the average hop-count within the job strongly correlates to reduced performance. In this section, we seek to ameliorate some of this impact. As mentioned previously, strategies to completely mitigate fragmentation have not been adopted on our systems due to negative impacts on utilization. So any solution identified has to decrease fragmentation without significant impact on system utilization.

First, we seek to understand the scheduler mechanisms that lead to fragmentation. As mentioned in the previous section, ALPS is responsible for generating a list of nodes from which MOAB reserves resources during the end of job scheduling. On Titan, we reserve nodes for the two highest priority jobs on the idle queue unless a job requested specific nodes. Node

reservations are generally applied in two manners. If the system has sufficient free nodes, it will engage in a simple reservation strategy. This is a top-down traversal of the ordered ALPS list, assigning free resources to the job being scheduled on a first-come, first-serve basis.

The second manner occurs when the top priority job cannot immediately reserve nodes. In order to reserve nodes for this job, currently running allocations are analyzed for "time-remaining," which is essentially the measure of the wall-clock time requested minus the elapsed running time of the job. By evaluating time-remaining, the scheduler is able to determine the worst-case time that a set of resources will become available. In doing this, it pre-reserves a set of nodes that will be available to the job at a future time. This set of nodes represents a backfill window. As jobs within this window become available, if the scheduler were to leave them unused, it could significantly degrade system utilization as backfill windows can take hours to days to become fully available for large jobs. Instead, a secondary scheduler mechanism evaluates jobs having both high-priority and requested wall-clock times that fit within the window and will not delay the primary job's start time. This will allow the backfilled job to run, while the primary job is waiting. Thereafter, the set of backfill nodes are presented as a list, ordered in the same manner that ALPS orders nodes, and the resources are allocated in a first-come, first-serve fashion.

Where these scheduling mechanisms contribute to fragmentation is a coupling of the sequential node-selection with a workload that is mixed between long and short-lived jobs. In such an environment, small, short-lived jobs are often scheduled through backfill on any available node in the backfill nodes-list, resulting in gaps between nodes used in larger, long-lived jobs. Essentially, we view one contributing factor of fragmentation as a result of using the same starting point to look for free nodes from the node list. This results in higher use of nodes at the front of the list. This is shown in Figure 7 from an analysis of where node seconds are used on Titan. In this figure, the `NodeIndex` is the location of a particular node in the ordered list. As shown here, nodes at the head of the list tend to be scheduled the most and nodes at the tail of the list are used significantly less. In an effort to ameliorate some of this effect, we designed a new scheduling mechanism called Dual-Ended Scheduling. In this approach, we essentially change the ends from which large, long-lived jobs and small, short-lived jobs are scheduled from. Thus, large, long-lived jobs are scheduled from the front of the list and small, short-lived jobs are scheduled from the tail of the list. Ideally, this should create more entropy at the tail-end of the list, increasing gaps but doing so to smaller jobs which tend to be less impacted by fragmentation.

The question then becomes, what is the appropriate criteria for which to enable dual-ended scheduling? In our efforts, we evaluated several different demarcation points generally based on job size and requested wall-clock time. To evaluate various criteria we utilized the MOAB simulator to replay real job traces from Titan, and reschedule the jobs as they would

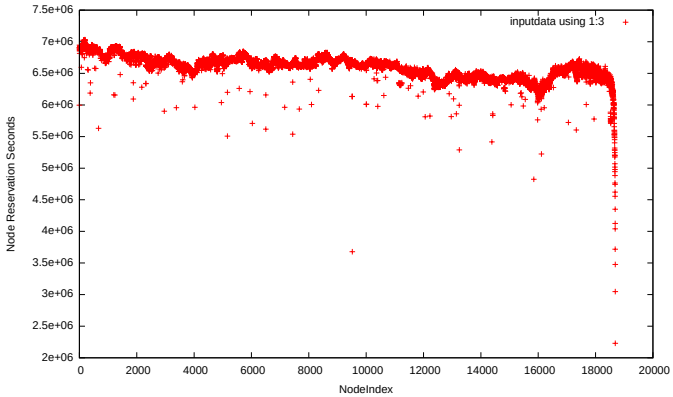

Fig. 7. Titan Reserved Seconds by Node Index: Jan - May 2015. Nodes at the tail of the Titan scheduling list are allocated far fewer hours than nodes at the head of the list.

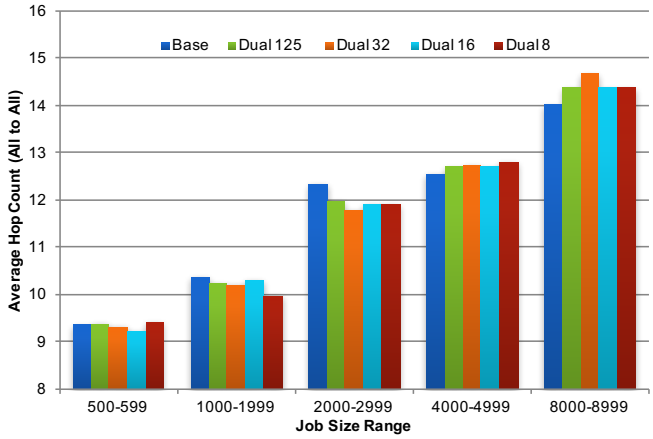

Fig. 8. Job Size Evaluation: Single Day April 15

under dual-ended scheduling. Due to simulator constraints, our preliminary evaluation only included a single day of job data, comprising of 900 jobs. Figure 8 shows this preliminary evaluation. We ran 5 simulations, using default layout, and demarcation points of 8, 16, 32 and 125 nodes. The 125 node demarcation point was selected because of existing policy within the scheduler of limiting any job less than 125 nodes to 2 hours of wall-clock time. From this preliminary analysis, we inferred that the 16 and 125 node demarcation points had the most potential. At some higher-node counts, we did see increases in average hop-count but the sample size was very limited at these ranges due to the short evaluation period.

Based on the positive results from the simulation data, it was decided to evaluate dual-ended scheduling on Titan. We began a production evaluation of the 16 node demarcation point, followed by an evaluation of the 125 node demarcation. The results in Figure 9 show the average hop-count of common job-sizes from the two months of data from the production evalutions. This data is compared against the first six months of the 2015 calendar year. As shown in the results, the trends are indicating that both the 16 and 125 node demarcation

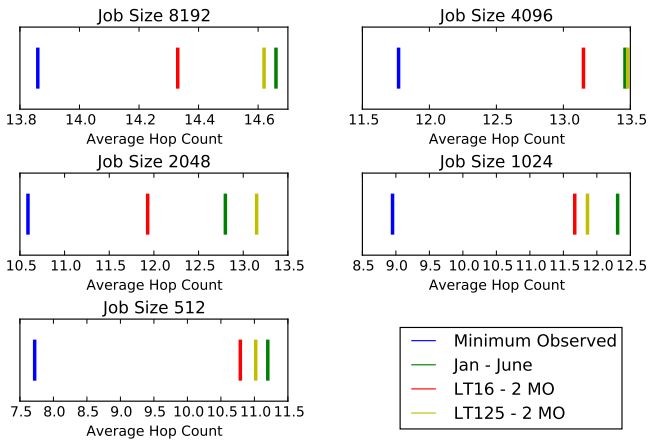

Fig. 9. Average hop-count of all jobs of specified size from production Titan. These values are compared with the single minimum hop job that occured within the evaluation logs.

strategies are resulting in reduced hop-counts for common job-sizes over the basic layout strategy on MOAB. We include the minimum job hop-count that existed in all of this data set in each of the comparison points for a frame of reference. In both large and small job sizes, the average across the entire range is moved between 40-50% closer to the observed minimum hop-count. The 16 node demarcation point scheduling strategy has remained in production on Titan.

VI. ALPS ORDER

An observation made during the initial investigation, using visual analysis, was the dimension prioritization of job placement. More specifically, jobs were stacked with prioritization in the Y-dimension over the X-dimension, as previously shown in Figure 1. From a bandwidth performance perspective, this was troubling.

The role of laying out nodes on Titan falls upon the Application Layout Placement Scheduler (ALPS). ALPS is responsible for enumerating the set of nodes in the system and generating an ordered list from which the resource manager MOAB will schedule.

The designers of ALPS use a node enumeration strategy that is both network and somewhat communication pattern aware [12]. They describe the ordering of the system as a repetition of a shape they call a "basic building block". Building blocks used this way allow sequences of nodes to be chained together (i.e., follow one after another) while integrating various techniques to optimize grouping without having to develop strategies for node packing that would be full system aware and difficult algorithmically to achieve for systems the scale of Titan.

On close examination, the building blocks enumerated on Titan are a modified form of Hilbert curves. Beginning with groupings of 2x2x2, these cubic groupings of nodes are traversed with Hilbert curves that are then connected together

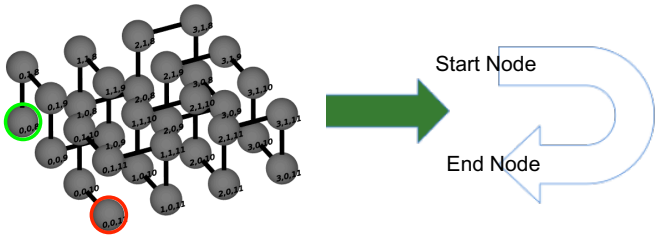

Fig. 10. Titan default basic building block and orientation. Blocks are chained together in the Z-dimension, Y-dimension, and X-dimension until the machine is fully populated. Blocks improve neighbor communication by using space-filling curves.

(i.e., followed in sequence) to form the ALPS basic building block. Figure 10 shows an example of the most common block used on Titan. These blocks are 4x2x4, with the start and end-points being in the same dimension to make it possible to chain blocks easily in the Z-dimension. Since there are 2 compute nodes per Gemini router, this enables, 16 compute nodes to be within 1 hop of each other within each 2x2x2 grouping. This packing density works well for 2D nearest neighbor communication patterns.

Basic building blocks are laid out in a Z-Y-X priority on Titan. The selection of the Z-dimension from a bandwidth and latency perspective makes sense, in that the Z-dimension is generally composed of backplane cabling with a maximum bandwidth of 15GB/s. The selection of the Y-dimension as the second dimension of priority may stem from previous attempts at improving ALPS layout strategies through maximization of the minimum dimension first, MDF. MDF strategies, in a non-anisotropic network, make sense in that an application populating the full dimension would be able to use the torus wraparound links to reduce the hop count for average communication. On Gemini, which is anisotropic, this decision forces more traffic through reduced bandwidth links, which may reduce any benefits gained from early wraparound. Stated another way, any intra-job communication on Titan having to traverse more than two hops in the Y-dimension will have its bandwidth reduced to 4.6GB/s due to the guarantee of a cable link. Jobs traversing the same distance in the X or Z dimension may suffer geometric digression only if there is a competing flow.

A. *Bandwidth Layout*

Given this manner in which ALPS orders the layout out of the machine, we consider if general applications' workloads are better off with layouts that maximize bandwidth, latency, or a bit of both. To address this we modify the manner that ALPS places basic building blocks to evaluate different layout strategies, and study their impact on the jobs Titan runs. The first layout we considered is very similar to the 2X layout developed at NCSA [10], wherein the developers worked with Cray to modify the ordering on the Blue Waters supercomputer to prioritize Z-X-Y. The authors determined that this layout would provide more bandwidth to applications on the system.

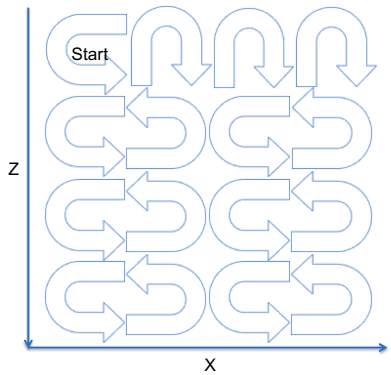

Fig. 11. ZXY Bandwidth Layout: Basic Block Placement. Block placement prioritizes X over Y dimension, leading to increased intra-job bandwidth.

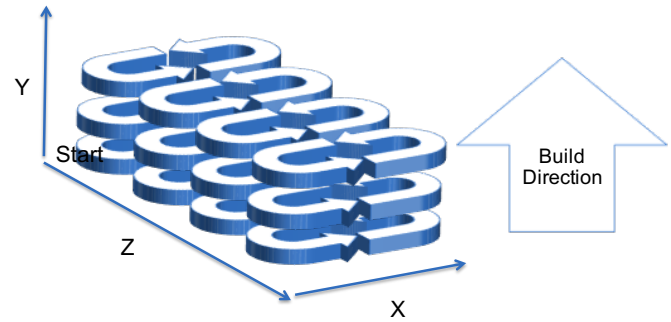

Fig. 12. Balanced Layout: Block placement alternates between Y and X dimension priority. This layout results in improved bandwidth and lower average hop-count over default placement for most large job-sizes.

Blue Waters ultimately abandoned this approach in-lieu of an approach using isolated geometries to improve application performance. While isolated geometries offer great performance improvement, the associated utilization degradation is not acceptable for OLCF resources. Similar to the NCSA effort, our first approach was to lay the machine out prioritizing the X dimension over the Y dimension. Additionally, we developed a series of transitions on the start and end-points of the basic building blocks to reduce the transition costs when growing in the X or Y dimensions. Transition costs are usually measured in terms of additional hops added to neighbor communication. The block in Figure 10 has its start and end points in the Z-dimension when these blocks are fully placed across the full Z-dimension; ALPS will place the next block in the next level of the Y-dimension, wrapping around the Z-torus. This creates a transition cost of 2Y for neighbor communication. Additionally, when block placement grows in the X-dimension the cost is 2 Y-hops and 4 X-hops.

In an effort to reduce these transition costs and improve packing density for traditional communication patterns, we evaluated using re-organized basic building blocks where the start and end points could be modified. As it turned out, there were several ways of re-organizing the layout of the 2x2x2 Hilbert curves within a basic block that enabled rotations of the blocks to better suit our needs. Figure 11 shows several of these rotated blocks and our organization of them in a ZxX plane. In this more bandwidth-centric layout, we use the same starting position for the block but use a rotated version of the block. As we grow in the X-dimension, instead of wrapping around the Torus to start the next row, we simply build in the opposite direction using a block rotation that achieves a zero cost transition. From this figure, it is evident that we can fill out the entire ZX plane without incurring any multi-hop transition costs between blocks. The only cost incurred in this layout is when building in the Y-dimension, which incurs a 2 Y-hop cost. Since Titan is 16 high in the Y-dimension we experience only 8 of these transitions.

B. Balanced Layout

Prioritizing the Y-dimension over the X-dimension in layout tends to result in allocations favoring latency above bandwidth. The same is true for favoring X-dimension over the Y-dimension for bandwidth over latency. A third option would be a layout strategy targeting a balance between latency and bandwidth. In an attempt to achieve this for our common larger job sizes we developed a balanced layout strategy. The goal of this strategy is to alternate between secondary dimensional priorities. The Z dimension remains the basic build priority, but upon every Z-transition, we alternate between growth in the X-dimension and growth in the Y-dimension. Figure 12 shows our basic approach to this new ordering. Using the balanced approach common large job sizes on the system are provided with allocations that have better internal bandwidth than traditionally ordered ALPS layouts and improved packing densities over ZXY layouts.

C. Job Placement Simulator

Evaluation of various ordering techniques on Titan is difficult due to high resource demand. To enable a larger evaluation, we first developed node-placement simulations that would allow us to develop an understanding of the impact to the scheduling features (e.g., average hop-count) derived from the Scores effort (Section IV). For this, we developed a parallel placement simulation that would enumerate every possible contiguous job allocation on Titan and calculate the average hop-count and isolated bandwidth for job placements at common job sizes. Using parallel simulations on Titan, we were able to quickly evaluate every possible placement option for jobs to allow us to predict how the lists may perform against more practical benchmarks. We additionally enabled the simulation to consider more common communication paradigms while calculating the primary metrics, e.g. Avg-hop-count under the CG communication pattern. Using this, we could infer how a particular application may run across the updated orderings.

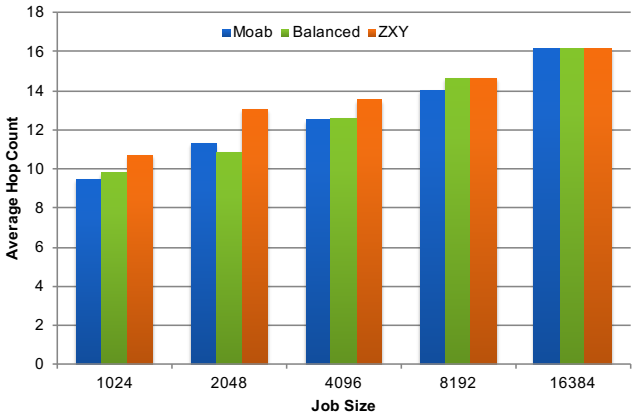

Fig. 13. All to All: Average Hop-Count

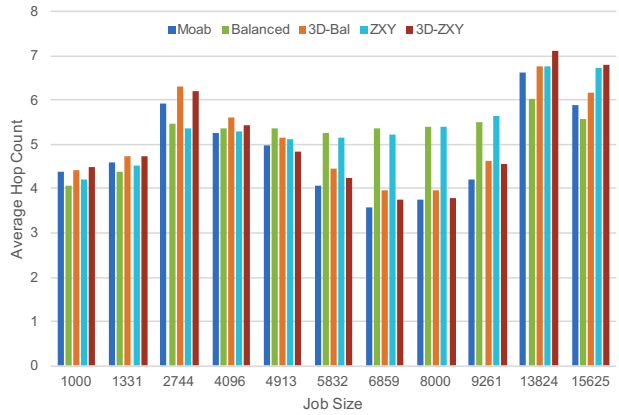

Fig. 15. 3D Exchange: Average Hop-Count

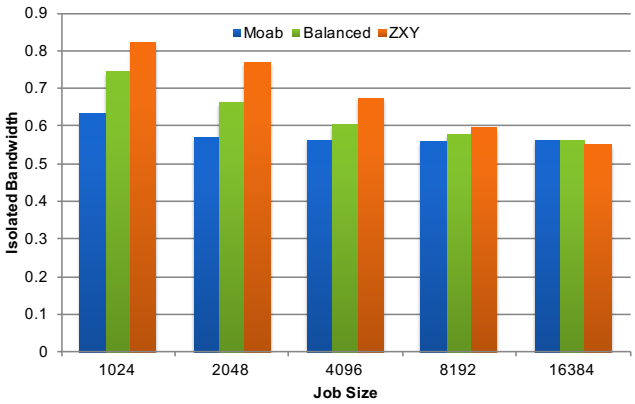

Fig. 14. All to All: Isolated Bandwidth

D. Placement Simulation Results

The results in Figure 13 show a comparison of the average hop-count for an all-to-all communication pattern when simulated for every-location within Titan. From these results, it is clear that the layout in the ZXY ordering strategy has an impact on the packing density of the jobs. As discussed in the Scores section, packing density was a strong indicator of performance for many applications. The results here show that both orderings have an impact on packing density, however, in the balanced ordering the impact is generally small. However, the trade-off is shown in Figure 14.

While the all-to-all communication pattern has shown heuristically to be a good indicator of performance, we were interested in understanding the hop-counts that would be associated with a more common communication pattern, particularly at scale. All-to-all patterns can grow in network expense as the node-count used for the application increases. A common pattern seen in several applications even within our own acceptance harnesses are the 3D 6-point stencil exchanges. S3D, a combustion application regularly run on Titan engages in such a communication pattern. The results in Figure 15 show that when using a more common communication pattern the hop-counts between communicating pairs generally track better in both the ZXY and the Balanced orderings. In fact in 5 of the 8 test-cases the Balanced ordering results in a tighter packing characteristic. However, starting around 5K nodes through 12K nodes there seems to be a local-minimum that is achieved by the traditional ALPS ordering that provides it with a significant advantage in hop-counts compared with the other lists. This is particularly troubling because, at these job sizes, the node-counts reach what we deem to be leadership quality jobs. This range of node-counts reflects our most visible job-sizes, and a negative impact due to layout would not be appropriate. After analysis of the issue, it became apparent that this artifact was created through irregularities imposed by the transition reducing wrapping built into our new lists. Essentially by not building back to front at each X and Y transition we created an artifact that leads to higher hop-counts for 3D exchanges within an important job size scope. To remedy this, we simply created two altered lists, 3D-Bal and 3D-ZXY. The results of these are shown in Figure 15 and Figure 16. While isolated bandwidth remains largely the same, the average hop-count impacts are largely reduced.

E. Titan Results

Testing on Titan required multiple phases, as taking down the machine and swapping ordering lists requires draining the machine and taking exclusive access. Due to these requirements, our testing has been scheduled during machine outages as it should not interfere with the primary science jobs being run. The infrequency of these outages has led to an extended testing duration. To date, we have performed two of three planned full system evaluations. In our first phase evaluation, we utilized a placement test-harness that would release jobs in a common ordering but not specify placement to avoid giving any of the orderings an advantage. This test harness allows us to deploy several simultaneous jobs on a quiet Titan system and automatically extract relevant

Fig. 16. 3D Exchange Isolated Bandwidth

Fig. 17. Test Shot 1: Small Jobs

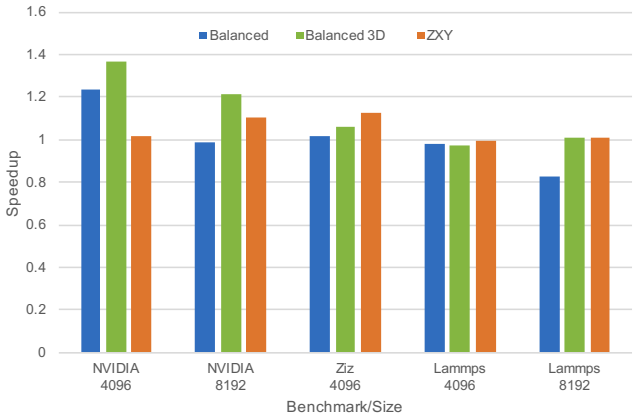

Fig. 18. Test Shot 2: Large Jobs

performance metrics. We evaluated job sizes up to 2,048 nodes. We used several applications from the Titan acceptance harness, such as LAMMPS (molecular dynamics), S3D (combustion), and CESM (climate). Additionally, we ran a few benchmark applications that were known to be placement sensitive such as the NVIDIA placement application and the Ziz (Chimera astrophysics) mini-app [13]. In our first test-phase, we had not yet identified a solution to the 3D stencil problem, and as a result, only tested using our Balanced and ZXY lists. The results in Figure 17 show the application performance improvements as a speedup over the equivalent MOAB-ordered test case. During this test-phase every single application experienced performance improvements when running against the Balanced ordering. The ZXY layout also performed well but had several instances where performance degraded, however, the ZXY layout experienced the single greatest improvement when running CESM (a Climate code) at 1,200 nodes, with an improvement of 24%. Overall, both layout changes showed improvement over the default layout for our smaller job measurements. It is important to note that during this test-phase we ran several smaller jobs, but since layout changes generally require a job to consume more than a full Z-dimension of basic blocks, there were virtually no measured difference between these jobs, and they are not reported.

In our second test shot, we focused primarily on larger jobs, ranging between 4,000 and 8,000 nodes. It was during this test-shot that we were able to perform tests against our newly added 3D-Balanced wraparound list. Unfortunately due to the time-demands of each test-case we were unable to evaluate the ZXY-3D list. For this evaluation, we ran the series of benchmarks from our acceptance harness that could scale to these sizes. The results show the promise of the 3D-balanced layout particularly as we grow towards 8K. At the 8K example, the balanced 3D list significantly outperforms the balanced list. We believe this is due to the reduction in hop-counts at this size for jobs engaging in 3D communication. Additionally, the 3D balanced layout achieved almost a 36% performance improvement in the NVIDIA placement benchmark. These results indicate that the redesigned layouts can achieve performance improvements at large-scale. We expect that as job-sizes trend toward full-machine sizes, the layout differences will converge and performance will be equivalent.

We have one more future test-shot planned to run the full acceptance harness against the MOAB, 3D-Bal, and 3D-ZXY lists. During these evaluations, we will test jobs up to the full machine size to quantify the impacts of the layouts. If the performance improvement trends continue, we will trial the lists for a set period of time on Titan, and inject known-jobs over a longer period of time to quantify the loaded system impact from layout improvements.

VII. Related Work

Researchers have looked at understanding application placement and its impact on performance extensively in past. Early work, focusing on the BlueGene/L system, by Krevat et al. [14], assumed that nodes allocated for a toroidal system "must be rectangular and contiguous". However, systems that implement such approaches either suffer potentially low

utilization or pay the high cost of process migration between nodes in order to rearrange work in contiguous blocks.

Lo et al. [15] demonstrate that better throughput can be achieved with non-contiguous allocation. They make their placements, though, in $2n$ size blocks, trying to allocate the largest block possible. For them the allocation of single nodes is the degenerate case of 2^0 size blocks. An improved allocation in non-contiguous chunks was the MC1x1 approach [16], but it required extensive calculations to evaluate numerous possible allocations.

Another approach is to accept any given allocation and attempt to map the application's task topology (communication patterns) onto the actual underlying hardware topology using embedding techniques like folding [17], [18], [19]. Such an approach typically involves requiring users to have explicit knowledge of, or have prior application runs to detect communication patterns.

More recently, Tuncer et al. [20] combine node selection with task mapping. This combined approach takes into account an application's communication pattern as well as topology when selecting the nodes. Using a clustering approach to allocation, it requires more computation to select nodes and relies on information not always readily available (communication patterns). The authors themselves note the limitation in applying their work [20] saying *"..real HPC machines do not have the infrastructure to support combined allocation and task mapping.."*

Leung et al. [21] showed that one-dimensional allocation strategies could be effective for massively parallel supercomputers. They proposed Hilbert curves [22] in two dimensions for assigning node ordering. Various other curves were evaluated by Albing et al. [23], [12] and optimal orderings were shown to be dependent on the dimensions of the torus interconnect [24]. These linear approaches have the advantage of extremely low-cost (i.e., rapid) allocations from their list of free nodes.

Some of the chunk allocations keep lists of different size free chunks and thus have more than one list to check. Only in [25] have we found any use of both ends of a single (ordered) list.

VIII. Conclusion

In this work, we have presented our efforts to decrease runtime variability and improve performance for medium to large sized jobs on Titan. Titan uses a common HPC topology, 3D torus, and a widely-used scheduling tool, MOAB. We demonstrate through the use of visual analytics and machine learning, the identification of problems and features that are traditionally obfuscated by the complexity and scope of extreme-scale resource allocation. During this effort, we discovered a common scheduling technique that due to the workload of large and small jobs, resulted in unnecessary and additional fragmentation to larger, long-lived jobs. We presented our effort to ameliorate this impact by separating large, long-lived jobs from small, short-lived jobs. In doing so, we were quickly able to reduce the average-hop-count of several major job sizes during our evaluation on Titan. In evaluations on Titan, we observed reductions in average hop-count, for common job sizes, up to 50% in relation to the measured job-size minimum.

In a second study, we demonstrate an average of 10% performance improvement across several real-applications commonly run on Titan, and used as a part of the Titan acceptance testing. This effort was the result of identifying that the default layout on Titan results in reduced intra-job bandwidth due to the anisotropic layout of the Gemini network. In our effort to improve it we designed and implemented both a layout targeting maximum bandwidth for applications and a layout designed to provide a better balance between latency within allocations.

Both of the allocation efforts presented in this work are operationally viable. Operating an extreme-scale and open-science supercomputer requires pragmatic approaches to reduce the potential negative outcomes associated with over-optimizing for a single goal. In the approaches presented in this work, there are no negative outcomes associated with 1) The utilization of the machine, a metric that largely forces scheduling to allocate jobs without delay and 2) the scheduling mechanisms, prioritizing capability jobs or the ability to schedule capability jobs into the machine.

The general insights from this work apply irrespective of topology or scheduling software. Managing a large-scale, open science cluster will result in a workload that is a mixture of small, medium, and large jobs that will vary from very short to long runtimes. Job placement that is not sensitive to the workload mixture can result in suboptimal performance. Further, resource allocation must be designed in a network topology-aware fashion. Our efforts demonstrate that the target application workloads faired better when resource allocation balanced a mixture of latency and bandwidth.

IX. Acknowledgement

This work was supported by the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is managed by UT Battelle, LLC for the U.S. DOE (under the contract No. DE-AC05-00OR22725).

References

[1] J. Dongarra, H. Meuer, and E. Strohmaier, "Top500 supercomputing sites," http://www.top500.org, 2015.

[2] R. Alverson, D. Roweth, and L. Kaplan, "The Gemini System Interconnect," in *High Performance Interconnects (HOTI), 2010 IEEE 18th Annual Symposium on*, Aug. 2010, pp. 83–87.

[3] O. Padron and D. Semeraro, "Torusvis: A topology data visualization tool," in *Proceedings of the 2014 Cray User Group International Conference*. Cray User Group, May 2015.

[4] "Adaptive Computing Moab HPC Suite." [Online]. Available: http://www.adaptivecomputing.com/products

[5] "Splunk Enterprise." [Online]. Available: http://www.splunk.com

[6] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, D. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga, "The NAS Parallel Benchmarks," *The Int'l Journal of Supercomputer Applications*, vol. 5, no. 3, pp. 63–73, 1991. [Online]. Available: citeseer.ist.psu.edu/article/bailey94nas.html

[7] I. Lee, "Characterizing communication patterns of nas-mpi benchmark programs," in *Southeastcon, 2009. SOUTHEASTCON '09. IEEE*, March 2009, pp. 158–163.

[8] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297. [Online]. Available: http://dx.doi.org/10.1007/BF00994018

[9] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32. [Online]. Available: http://dx.doi.org/10.1023/A:1010933404324

[10] J. Enos, R. Bauer, S. Islam, R. Fiedler, M. Steed, and D. Jackson, "Topology-aware job scheduling strategies for torus networks," in *Proceedings of the Cray User Group*, May 2014.

[11] Y. Aridor, T. Domany, O. Goldshmidt, J. E. Moreira, and E. Shmueli, "Resource allocation and utilization in the blue gene/l supercomputer," *IBM Journal of Research and Development*, vol. 49, pp. 425–436, 2005.

[12] C. Albing, N. Troulier, S. Whalen, R. Olson, J. Glenski, H. Pritchard, and H. Mills, "Scalable node allocation for improved performance in regular and anisotropic 3d torus supercomputers," in *Proceedings of the 18th European MPI Users' Group Meeting*, ser. EuroMPI '11, 2011, pp. 61–70.

[13] O. E. B. Messer, E. DAzevedo, J. Hill, W. Joubert, S. Laosooksathit, and A. Tharrington, "Developing miniapps on modern platforms using multiple programming models," in *Cluster Computing (CLUSTER), 2015 IEEE International Conference on*, Sept 2015, pp. 753–759.

[14] E. Krevat, J. Castaos, and J. Moreira, "Job scheduling for the BlueGene/L system," ser. LNCS. Edinburgh, Scotland: Springer, 2002, pp. 38–54.

[15] V. Lo, K. J. Windisch, W. Liu, and B. Nitzberg, "Noncontiguous processor allocation algorithms for mesh-connected multicomputers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 7, pp. 712–726, Jul. 1997.

[16] M. Bender, D. Bunde, E. Demaine, S. Fekete, V. Leung, H. Meijer, and C. Phillips, "Communication-aware processor allocation for supercomputers: Finding point sets of small average distance," *Algorithmica*, vol. 50, no. 2, pp. 279–298, 2008.

[17] H. Yu, I. Chung, and J. Moreira, "Topology mapping for blue Gene/L supercomputer," in *Proceedings of the 2006 ACM/IEEE conference on Supercomputing - SC '06*, Tampa, Florida, 2006, p. 116.

[18] T. Agarwal, A. Sharma, and L. V. Kalé, "Topology-aware task mapping for reducing communication contention on large parallel machines," in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium*, Rhodes Island, Greece, 2006, p. 110.

[19] A. Bhatele and L. V. Kalé, "Application-specific topology-aware mapping for three dimensional topologies," in *2008 IEEE International Symposium on Parallel and Distributed Processing*. Miami, FL, USA: IEEE, Apr. 2008, pp. 1–8.

[20] O. Tuncer, V. J. Leung, and A. K. Coskun, "Pacmap: Topology mapping of unstructured communication patterns onto non-contiguous allocations," in *Proceedings of the 29th ACM on International Conference on Supercomputing*. ACM, 2015, pp. 37–46.

[21] V. J. Leung, E. M. Arkin, M. A. Bender, D. Bunde, J. Johnston, A. Lal, J. S. Mitchell, C. Phillips, and S. S. Seiden, "Processor allocation on Cplant: achieving general processor locality using one-dimensional allocation strategies," in *Proc. 4th IEEE International Conference on Cluster Computing*, 2002, p. 296304.

[22] "Hilbert curve – from wolfram MathWorld," http://mathworld.wolfram.com/HilbertCurve.html, Mar. 2010.

[23] C. Albing and M. Baker, "ALPS, topology, and performance: A comparison of linear orderings for application placement in a 3D torus," in *Proceedings of the 2010 Cray User Group International Conference*. Edinburgh, Scotland, UK: Cray User Group, May 2010.

[24] C. Albing, "Characterizing Node Orderings for Improved Performance," in *Proceedings of the 6th International Workshop on Performance Modeling, Benchmarking, and Simulation of High Performance Computing Systems*, ser. PMBS '15. New York, NY, USA: ACM, 2015, pp. 6:1–6:11. [Online]. Available: http://doi.acm.org/10.1145/2832087.2832094

[25] C. K. Albing, "Improving node allocation for application placement in high-performance computing systems," Ph.D., University of Reading (UK), 2012.