Improving Resource Availability by Relaxing Network Allocation Constraints on Blue Gene/P

Narayan Desai,[1] Darius Buntinas,[1] Daniel Buettner,[2] Pavan Balaji,[1] Anthony Chan[1]

[1] Mathematics and Computer Science Division – Argonne National Laboratory
[2] Argonne Leadership Computing Facility – Argonne National Laboratory
{desai,buntinas,balaji,chan}@mcs.anl.gov, buettner@alcf.anl.gov

Abstract— **High-end computing (HEC) systems have passed the petaflop barrier and continue to move toward the next frontier of exascale computing. As companies and research institutes continue to work toward architecting these enormous systems, it is becoming increasingly clear that these systems will utilize a significant amount of shared hardware between processing units, including shared caches, memory management engines, and network infrastructure. While these systems are optimized to use all of the hardware available in a dedicated manner to achieve the best performance, in practice, the shared nature of this hardware makes scheduling applications on it difficult and wasteful. For example, while the IBM Blue Gene/P system has been designed to use a torus network for efficient communication, some of the torus links (especially those connecting different racks) are shared between multiple racks. Thus, a job running on one rack, might preclude another job from running on a second rack in spite of having its compute resources completely idle. In this paper, we assess the relative performance degradation noticed by real applications when such shared network hardware is completely unutilized for some cases. Our measurements on Intrepid, one of the largest Blue Gene/P installations in the world, demonstrate less than 5% degradation for several leadership applications commonly run on the Intrepid system. Further, we demonstrate that the additional scheduling flexibility offered by not sharing such hardware can improve the overall job turnaround time by nearly 40% in some cases.High-end computing (HEC) systems have passed the petaflop barrier and continue to move toward the next frontier of exascale computing. As companies and research institutes continue to work toward architecting these enormous systems, it is becoming increasingly clear that these systems will utilize a significant amount of shared hardware between processing units, including shared caches, memory management engines, and network infrastructure. While these systems are optimized to use all of the hardware available in a dedicated manner to achieve the best performance, in practice, the shared nature of this hardware makes scheduling applications on it difficult and wasteful. For example, while the IBM Blue Gene/P system has been designed to use a torus network for efficient communication, some of the torus links (especially those connecting different racks) are shared between multiple racks. Thus, a job running on one rack, might preclude another job from running on a second rack in spite of having its compute resources completely idle. In this paper, we assess the relative performance degradation noticed by real applications when such shared network hardware is completely unutilized for some cases. Our measurements on Intrepid, one of the largest Blue Gene/P installations in the world, demonstrate less than 5% degradation for several leadership applications commonly run on the Intrepid system. Further, we demonstrate that the additional scheduling flexibility offered by not sharing such hardware can improve the overall job turnaround time by nearly 40% in some cases.**

I. Introduction

Large-scale systems today already scale to hundreds of thousands of processing elements. With plans under way for exascale systems within the next decade, it is expected that we will soon have systems that comprise more than a million processing elements. As researchers design these enormous systems, it is becoming increasingly clear that these systems will utilize a significant amount of shared hardware. This includes shared caches, shared memory and memory management devices, and shared network infrastructure.

Such sharing is already present on systems such as the IBM Blue Gene, which use flat network topologies (e.g., torus) instead of switched cluster interconnects. While this configuration allows the network cost to increase linearly with the system size, and not superlinearly as with switched fabrics, it also results in a significant amount of shared network hardware. For example, on Blue Gene/P systems, some of the torus links (especially those connecting different racks) are shared between multiple racks. Because of the limited availability of these resources, the system torus can be partitioned into smaller tori only in a limited number of ways, resulting in partitions that cannot be wrapped into tori. Thus, a job running on one rack might preclude another job from running on a second rack in spite of having completely idle compute resources. Such shared hardware makes scheduling applications complex and can potentially waste resources.

If we were not to use such shared hardware at all, the number of resource sharing conflicts could be significantly decreased, potentially allowing applications to be scheduled more effectively. However, using less network hardware than what is available can also lead to degradation in communication performance and eventually overall application execution time. Thus, the best approach for this situation is not at all clear.

In this paper, we assess the relative performance degradation observed in real applications when such shared network hardware is completely ignored for some cases. That is, in cases where allocating a partition results in use of shared network hardware that precludes another partition from being

functional, we disable the shared network hardware and in effect use a mesh network on the partition, instead of a torus network. Having fewer links available obviously impacts communication performance. However, our experiments on Intrepid, one of the largest Blue Gene/P installations in the world, indicate that for several leadership applications commonly run on the Intrepid system this degradation is less than 5%. We also notice that the additional scheduling flexibility enabled by this configuration can allow partition allocations to be packed better and improve the overall job turnaround time by nearly 40% in some cases.

The remainder of the paper is organized as follows. In Sections II and III we present background information about Blue Gene partitioning, the expected differences in performance between mesh and torus partitions, and the Intrepid system. In Section IV we present related work. In Section V we show the impact of using mesh versus torus networks on several microbenchmarks and applications. In Section VI we make recommendations for operators of such systems. Section VII we discuss our conclusions and present future work.

II. Overview of Blue Gene/P

IBM Blue Gene systems [**?**], [**?**] are highly scalable massively parallel processing (MPP) systems. BG/P is the second generation in the BG family. BG/P systems comprise individual racks that can be connected together; each rack contains 1024 four-core nodes, for a total of 4096 cores per rack. Blue Gene systems have a hierarchical structure. Nodes are grouped into midplanes, which contain 512 nodes in an 8x8x8 structure. Each rack contains two such midplanes. For the remainder of the paper, we will refer to partitions and jobs by node count; for example, a 2K node partition comprises four midplanes.

BG/P uses five different networks for different communication operations. The 3D torus network is used for MPI point-to-point operations as well as for collective operations using irregular communication or large message sizes. Each node has six nearest neighbors. Each link provides a bandwidth of 425 MB/s per direction, for a total bidirectional bandwidth of 5.1 GB/s. Though each node has six bidirectional links on each node, there is only one shared DMA engine. All discussion in this paper will focus on the 3D torus network. The 3D torus network is also usable as a 3D mesh.

Blue Gene systems are partitioned for the purpose of job execution; this approach isolates jobs from one another. Individual midplanes can be used as 512-node partitions or built into larger partitions. Inter-midplane connectivity is implemented by using cables and link chips. Each link chip connects to a single midplane and to four unidirectional cables that connect midplanes. These cables can be used only by a single partition at once. Link chips have two major types of configurations: either the link chips connect the local midplane with a pair of cables (one in each direction), or they wrap the midplane mesh back into itself. When the midplane is wrapped, the cables connected to the link chip and the path through the link chip remain available for use. This situation, referred to as pass-through, occurs frequently during partitioning. In this case, the local midplane is unable to connect to any other midplanes in this mesh dimension.

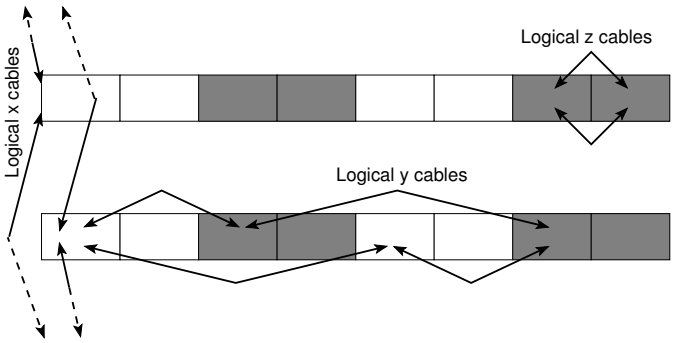

Fig. 1. Blue Gene cabling

Blue Gene partitions can be connected with either a torus or a mesh network. In general, Blue Gene systems use torus networks for partitions larger than a single midplane. In order to build a torus network, each midplane needs to be connected to six adjacent midplanes. Mesh networks are not wrapped; only interior connections are required. Hence, mesh networks cost less in terms of shared resources.

Large Blue Gene systems are constructed in rows of racks. While the torus building mechanism provided on Blue Gene systems are flexible, large systems are typically cabled in a similar fashion, described here. The X dimension of the torus connects the rows of the machine together. The Y dimension connects each midplane with a midplane in the same position two racks in either direction. The Z dimension connects the four midplanes in two adjacent racks. This scheme is depicted in Figure 1.

If a torus partition spans more than one midplane in a dimension, it monopolizes all of the cables in that dimension, so that the mesh can be properly wrapped into a torus. Mesh partitions have no such limitation; hence, multiple mesh partitions can be active on a single dimension simultaneously without interfering with each other.

III. The Intrepid System

Intrepid is a 556 TF Blue Gene/P system operated by Argonne National Laboratory for the Department of Energy INCITE program [**?**]. The system comprises 40 racks, 80 midplanes, containing a total of 163,840 cores. It was on the Top500 [**?**] June 2008 list as the number three system, and number five in the November 2008 list. Intrepid is a capability system, with single jobs frequently occupying substantial fractions of the system. Jobs of 8K and 16K nodes are common on the system. Larger jobs occur often on the system as well. Jobs up to 32K nodes run without administrator assistance.

More than a dozen application groups have INCITE applications on Intrepid. Each of these applications has completed a computational readiness review that measures their ability to run at large scale. Each application has different scalability; while some applications can effectively run at 40K nodes, many can scale only to 4K or 8K nodes before efficiency

begins to drop off. Most users want to run most of their jobs on partitions between 1K and 4K nodes. Moreover, because users are actively scaling their codes to larger sizes, job response times strongly affect user productivity.

Intrepid is a comprised of forty racks, or eighty midplanes. The dimensions of the full system torus are 5×4×4. As described above, each row is a slice out of the X dimension. A full row (8K node, 16 midplane) torus is 1×4×4. Only a limited number of torus configurations are possible, because of the constraints described above. A single 8K node partition is possible per row, as are four 2K node partitions. Partitions of 1K and 4K nodes consume cabling resources that impact their neighbors; hence, only a single 4K partition or four 1K node partitions can be run at the same time. The balance of the midplanes can be wired together by using the remaining dimensions, so the combination of a 4K node partition and two 2K node partitions is valid. The combination of 4K and 1K node rack partitions is particularly wasteful; the use of a single 4-rack partition and two 1K node partitions requires that all four remaining midplanes are usable only individually. Use of the X dimension cables would allow connection of these midplanes, at the cost of preventing multirow jobs from working. For operational reasons, this option is unavailable for small partitions on Intrepid.

Intrepid uses Cobalt [**?**], a component-based resource management suite popular on Blue Gene systems. Its architecture makes simulation of both scheduling behavior and system behavior accurate and simple. This simulation is described in detail elsewhere [**?**].

IV. Related Work

Previous literature has evaluated and presented ways to improve communication performance on systems such as the Blue Gene [**?**], [**?**]. However, this work focuses mainly on improving application performance while utilizing the available torus network. In this paper, we take a systemwide view of performance, as opposed to the more application-centric view of absolute communication performance of Blue Gene networks.

The performance difference between partitioning issues on flat networks has also been a topic of significant research recently. Specifically, in [**?**] the authors studied the performance implications between the partitioned tori on Blue Gene systems and the shared torus available on Cray systems. Similarly, researchers have studied scheduling strategies [**?**] that study the implications of not having completely dedicated hardware for each job and ways to improve such issues. However, the focus of this paper is slightly different from this research. Specifically, we do not look into using network hardware simultaneously in a shared manner by multiple applications. Rather, we concentrate on the impact of minimizing use the shared hardware. While using the hardware in a shared manner is an option too, we do not pursue that in this paper, as it is not trivial to do so in the current BG/P architecture without significant hardware changes.

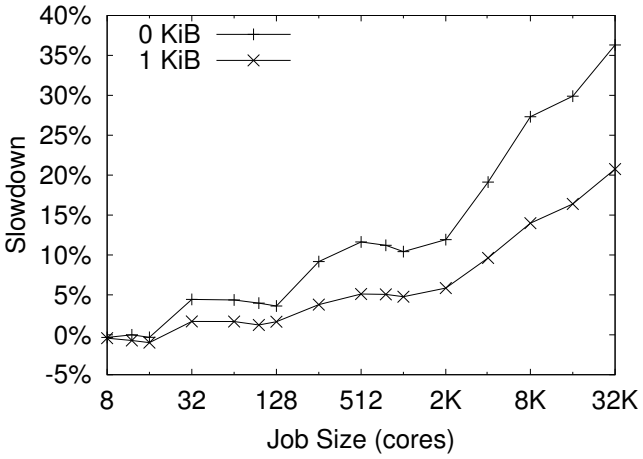

Fig. 2. Latency slowdown

In summary, this paper extends existing work and brings out interesting resource management aspects that are present on current large-scale systems and will become more prominent and visible in larger systems.

V. Experimental Analysis

Our approach has three major components. First, we validate our expectations for performance differences between wrapped and unwrapped mesh partitions. Second, we benchmark several applications from the INCITE program. Third, we simulate the scheduling effect of substituting mesh partitions for torus partitions using the workload from Intrepid.

A. Synthetic Benchmarks

In this section, we evaluate different synthetic benchmarks on torus and mesh connected topologies. These benchmarks give us an indication of the specific cases where large performance differences are expected.

1) Point-to-Point Latency: Figure 2 illustrates the slowdown in the maximum inter-process latency when using a mesh partition instead of a torus, with increasing system size. Specifically, the test measures the latency between the two farthest processes (maximum number of network hops) in the system for both topologies and presents the percentage difference between the two. For a message size of 0 bytes, we notice around 35% slowdown, while for a message size of 1K bytes, we notice about 20% slowdown. This difference is due to the increased number of hops that messages have to traverse when using a mesh instead of a torus. With increasing message sizes, the slowdown keeps decreasing because of data pipelining (i.e., the absolute difference remains constant, causing the percentage difference to reduce).

2) Effective Bisectional Bandwidth: Figure 3 illustrates the slowdown in the aggregate bandwidth reported by the B_eff benchmark (a part of the HPCC benchmark suite). Specifically, this benchmark presents the effective bisectional bandwidth that is available on the system. The figure illustrates a slowdown of close to 35% for large system sizes. This is

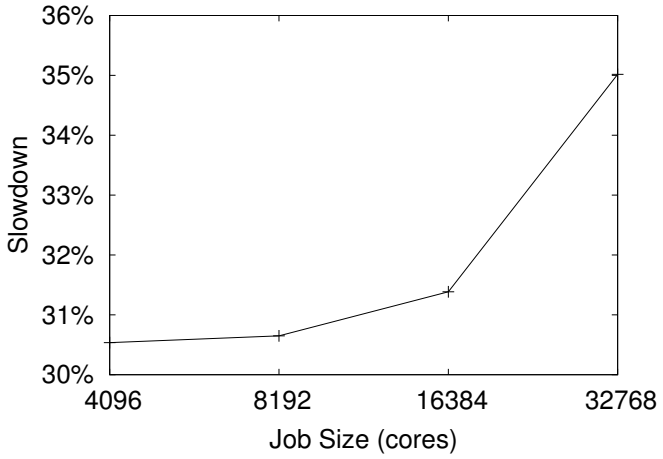

Fig. 3. Slowdown in aggregate bandwidth reported by b_eff benchmark

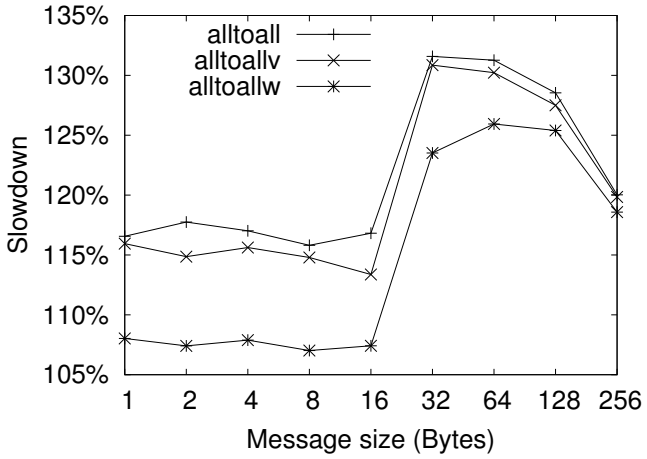

Fig. 5. Alltoall, Alltoallv, and Alltoallw slowdown

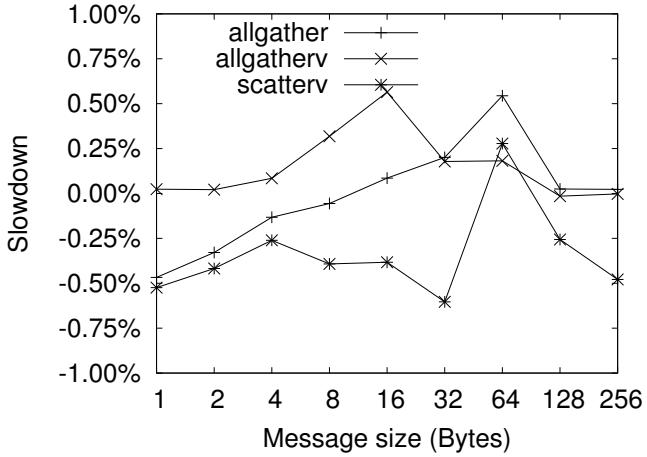

Fig. 4. Allgather, Allgatherv, and Scatterv slowdown

expected because the effective bisectional bandwidth reduces when moving from a torus partition to a mesh partition because of the smaller number of communication links.

3) Collective Communication: Figures 4 and 5 illustrate the performance of collective communication operations for the two topologies. Figure 4 shows the performance of `MPI_Allgather`, `MPI_Allgatherv`, and `MPI_Scatterv`. Figure 5 shows the performance of the different all-to-all variants (`MPI_Alltoall`, `MPI_Alltoallv`, and `MPI_Alltoallw`). As shown in the figures, for non-all-to-all collectives, there is little slowdown, while for all-to-all collectives, there is a large slowdown (we verified this for other non-all-to-all collectives as well, but the results are not shown in this paper because of space constraints). The reason is that all-to-all collectives perform the most amount of per-process communication; thus, reducing the number of communication links causes the largest degradation in their performance.

B. Application Results

In this section we describe three INCITE program applications: Nek5000, GFMC, and FLASH. Each of these has been executed on partitions ranging from 1K to 8K nodes (4K to 32K cores) using both mesh and torus networks. We also describe P3DFFT, a 3D fast Fourier transformation library, and compare its relative performance on both networks.

Nek5000 [**?**] is a spectral element CFD code developed at Argonne National Laboratory, which features spectral element multigrid solvers coupled to a highly scalable, parallel coarse-grid solver. It was recognized in 1999 with a Gordon Bell prize and is used by more than two dozen research institutions worldwide for projects including ocean current modeling, thermal hydraulics of reactor cores, and spatiotemporal chaos. Because the communication pattern is nearest-neighbor on an unstructured grid, Nek5000 is highly scalable. This communication pattern also indicates that we should not see a significant performance drop when running on a mesh versus a torus.

The second application, Green's function Monte Carlo (GFMC) [**?**], is an *ab initio* light-nuclei computation code, that models nuclear structures and reactions from bare nuclear forces. This application uses the Asynchronous Dynamic Load Balancing (ADLB) library, developed using MPI specifically for this code, to distribute work in a master-worker pattern. In GFMC, the ADLB servers, which serve as master processes, are arranged in a plane on one side of the partition. The communication pattern of this application is directed between master processes and worker processes to request and deliver work and solutions, as well as between master processes themselves to distribute pending work requests and solutions.

Note that the servers are arranged on one side of the partition. In a torus configuration, because of wrap-around links, the servers have connectivity on both sides of the plane. In a mesh configuration, however, only one of the sides is connected to the rest of the mesh. We expect that mesh performance would be improved by relocating the server processes.

FLASH 3.1.1 [**?**] is the latest FLASH release from the ASC Center at the University of Chicago. The FLASH code [**?**] is an astrophysical MPI simulation code written in FORTRAN90 and C. We choose the Sedov 3D setup with adaptive mesh

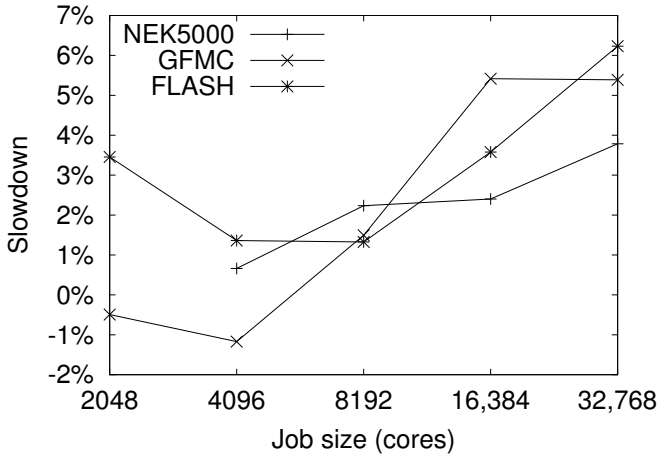

Fig. 6. Slowdown of application execution times due to using a mesh versus a torus configuration.

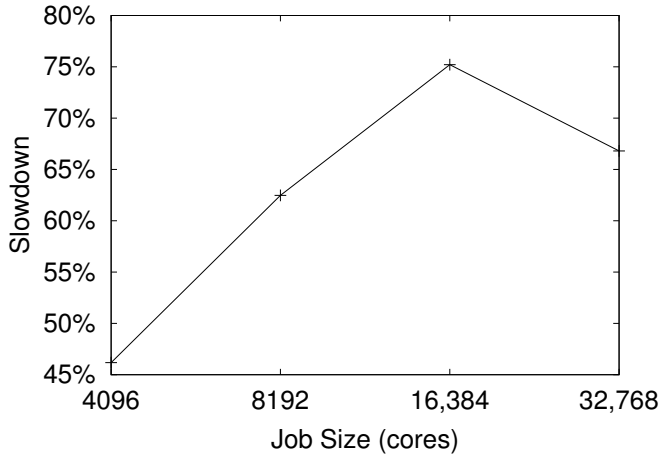

Fig. 7. Slowdown of P3DFFT performance due to using a mesh versus a torus configuration.

refinement module, Paramesh 4.0, with high refinement level (up to $lreine\_max = 12$) in our test runs. The Sedov 3D explosion problem [**?**] is a purely hydrodynamic setup that simulates the self-similar evolution of a spherical blast wave from a delta-function initial pressure perturbation in a homogeneous medium.

We ran these codes on various partition sizes, ranging from 2K cores to 32K cores (512 to 8K nodes), using mesh and torus configurations. Figure 6 shows the slowdown of the applications when using a mesh versus torus configuration for the various job sizes. We see that the largest slowdown was just over 6% for FLASH at 32K cores. Notice that for 2K and 4K core jobs, GFMC performs better in the mesh configuration than the torus; however, we believe that this is due to random variation in the execution times. These results indicate that using a mesh configuration has only a small effect on the application's execution time.

We believe that the high scalability of these application is a major reason for the relatively small impact. The communication patterns of these applications either are relatively localized, such as Nek5000, or are more bursty but staggered, which reduces congestion, such as GFMC.

P3DFFT [**?**] is a 3D fast Fourier transform library developed at the San Diego Supercomputing Center. It uses a 2D pencil decomposition, which allows for good scalability on relatively large (32768) processor counts. MPI traces show that P3DFFT depends heavily on MPI_Alltoallv for communication. This dependence explains the substantial difference in performance between mesh partitions and torus partitions. The relative performance is shown in Figure 7. While P3DFFT is not an application per se, it does demonstrate sensitivity to bisection bandwidth exhibited by one class of large-scale application.

C. Scheduling Simulation

In the previous section, we demonstrated the reduced application performance caused by the use of mesh partitions. However, using mesh partitions should provide a substantial boost to scheduler performance, because resources can be more freely allocated. To evaluate whether this approach is effective, we simulate the effects of making mesh partitions available to the scheduler. Because of the cabling of Intrepid, it makes sense to use only 1K and 4K node mesh partitions; if enough hardware is available to run 2K or 8K node partitions, sufficient cabling exists to build a torus.

For each simulation, we establish a per-run slowdown. Each job runs for the time included in the input trace, except for jobs run on mesh partitions. These jobs are expanded by the configured slowdown.

We have used a workload trace taken from Intrepid. This workload contains 3,890 jobs and reflects two weeks of activity. The INCITE program awards allocations on a January-to-January basis; hence, many new projects are just getting under way at the time of this writing. This situation biases this workload toward smaller jobs more than we typically see after the start-up period for projects. Also, fewer jobs are queued during this period in the year; job submission accelerates as groups become more experienced with the machine and improve their application scaling.

These simulations were performed using FCFS and the production scheduling policy used on Intrepid, WFP. WFP seeks to minimize unit-less job wait times, that is, the time a job has waited compared with the time requested. It also favors large node count jobs.

For each scheduling policy, we have compared the current production system (all torus) configuration used on Intrepid, with a configuration that uses mesh networks for all 1K and 4K node partitions. In the mesh configuration, we have added a performance penalty for jobs run on mesh partitions. We have simulated a uniform 5% and 20% job slowdown for these jobs. Considering the performance results for applications measured in the previous section, these seem like realistic values for application slowdowns.

For the workload tests, utilization was affected substantially; it improved 3-5% for the cases simulated. This change had a

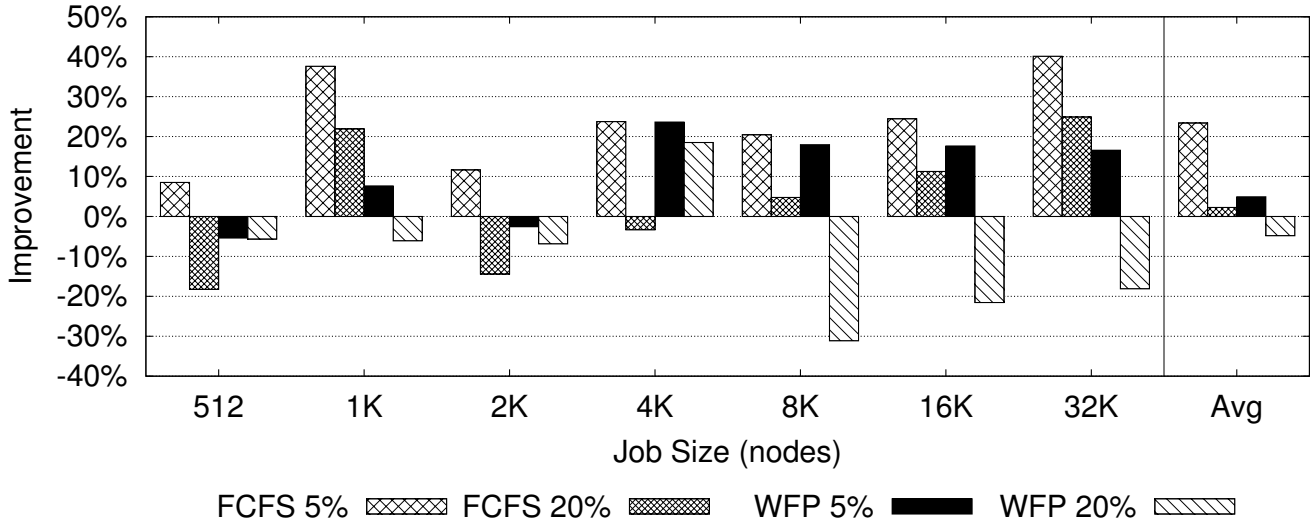

Fig. 8. Response Times, by job size.

more striking effect on job wait times. Figure 8 shows this impact. These results have several interesting characteristics. First, response times of 512 and 2K node partitions were negatively impacted. This result is to be expected; jobs of these sizes benefited from the resource contention experienced by 1K and 4K node partitions. Response times for 1K and 4K node partitions are greatly improved. Likewise, this result is to be expected because the switch from torus to mesh at these sizes reduces the resources needed to run these partitions. Furthermore, we note that WFP demonstrates diminished response times when applications have substantial (20%) degradation.

VI. Discussion

The results in this paper raise a set of questions for operators of such systems. Clearly the performance penalty of switching to a mesh partition varies with the application. The three applications benchmarked in Section V showed a range of results, from no slowdown ranging up to 7%. However, this 7% slowdown is clearly not a worst case scenario; applications based on P3DFFT or otherwise heavily dependent on all-to-all operations will show higher slowdowns, potentially rising to 100% or worse. A site will typically have some mixture of such workloads, so providing both types or partitions is important to improve resource availability while still providing the best performance for communication-intensive applications. Users should be allowed to select between a torus partition, which would provide the best communication performance for applications like those using P3DFFT, and a mesh partition for applications similar to FLASH3, GFMC, or Nek5000. Jobs using mesh partitions could be scheduled more quickly and would result in improved system utilization.

In the specific case of Intrepid's workload and current scheduling policy, this approach appears to provide a greatly improved level of service to users. Utilization is slightly improved, while job wait times are substantially reduced. While response times for large jobs are slightly increased under WFP with the 20% performance penalty, we have determined that this result is due to poor tuning of WFP with the new workload. Moreover, at least 75% of the INCITE allocations (by core hour) on Intrepid do not make heavy use of all-to-all operations, so system performance would be improved even without the hybrid solution described above. We would expect this hybrid approach to surpass the results presented here, for obvious reasons.

VII. Conclusions and Future Work

Large-scale systems currently use and will continue to use a significant amount of shared hardware between processing units. For example, on the Blue Gene/P system, which uses a torus-based network interconnect, some of the torus links are shared between multiple racks. Thus, a job running on one rack might preclude another job from running on a second rack in spite of its having completely idle compute resources. In this paper, we assessed the relative performance degradation in three applications when such shared network hardware is completely unutilized. Our experiments demonstrated that while the reduced number of network links available degrades application performance, this degradation is less than 5% for several leadership applications commonly run on Intrepid. At the same time, the additional scheduling flexibility available can improve the overall job turnaround time by nearly 40% in some cases.

As future work, we plan to characterize other target applications in terms of their relative performance. Also, some of the performance degradation we are seeing might be caused by collectives that are not tuned for mesh partitions. We need to investigate whether mesh-tuned collectives could yield improved performance on mesh partitions compared with current collectives.

ACKNOWLEDGMENTS

This work was supported in part by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

The software FLASH3 used in this work was in part developed by the DOE-supported ASC/Alliance Center for Astrophysical Thermonuclear Flashes at the University of Chicago.

This work would not have been possible without the support of the Argonne Leadership Computing Facility, in particular the ALCF Operations team. Their assistance is greatly appreciated.