Improving Resource Availability By Relaxing Network Allocation Constraints on the 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

Torus networks are prevalent on leadership-class petascale systems. While many systems use a single shared torus for a full system, Blue Gene/P systems provide the ability to partition the system torus into a series of independent, isolated tori for individual jobs. While this approach provides substantially improved network behavior for those jobs, the additional allocation constraints imposed by this scheme dramatically limits the possibilities for system scheduling. In this paper, we assess the relative performance of discrete per-job tori compared with per-job mesh networks using a series of synthetic benchmarks and several leadership class applications from the DOE INCITE program. We then simulate the scheduling impact of using dedicated meshes using job traces from the 556 TF Intrepid system at Argonne National Laboratory. This simulation shows up to a 40% improvement in job response time when using mesh partitions over torus partitions.

1 Introduction

Building efficient and high-performance interconnection networks is a key challenge in building leadership class computing systems. Due to the large number of nodes required in such systems, network costs must grow only linearly with additional node count. For this reason, torus networks have long been a popular choice for scalable computing systems. Traditional torus systems use a single shared torus for all jobs. Much work has been done to determine how to minimize job fragmentation (and hence maximize network performance) on such systems.

The IBM Blue Gene family of systems use a torus-based network, however, a unique facility for partitioning is provided. Partitioning of the system allows for discrete reservation of resources on a per-job basis. Each job has a set of nodes, a portion of the torus network and a set of I/O-related resources. When the system network is partitioned, the resulting meshes can often be "wrapped" into torus partitions, depending on the availability of a limited number of hardware resources needed to wrap the mesh. Due to the limited availability of these resources, the system torus can only be partitioned into smaller tori in a limited number of ways, resulting in partitions that cannot be wrapped into tori. In the default operational model used on BG/L and BG/P systems, such partitions are not used (except in the case of small partitions of less than 512 nodes, which can only be meshes). This limitation of using only torus partitions for jobs above a certain size imposes constraints on resource allocation, which in turn, can have appreciable impact on scheduling performance. While meshes have substantially diminished communication capacity compared with similarly-sized tori, their lack of the partitioning restrictions allow more mesh partitions to be scheduled concurrently, making them a tempting alternative to tori; provided that application performance is not impacted too severely.

In this paper, we explore the impact of network topology on application and scheduling performance, specifically, job turn-around time. We have quantified the absolute difference in network performance between dedicated torus and mesh partitions. We have also benchmarked a series of applications from the Department of Energy INCITE program, which awards time on the Intrepid system at Argonne National Laboratory. Finally, using this measured application slowdown, we simulate the results of using mesh partitions with the workload observed from Intrepid itself.

In Sections 2 and 3 we present relevant background information, including information about Blue Gene partitioning, the expected differences in performance between mesh and torus partitions, and the Intrepid system. In Section 4 we show the impact of using mesh versus torus networks on several microbenchmarks and applications. In Section 5 we present related work. In Section 6 we discuss our conclusions and make recommendations for operators of such systems.

2 Overview of Blue Gene/P

IBM Blue Gene systems [7, 11] are highly scalable massively parallel processing (MPP) systems. BG/P is the second generation in the BG family. BG/P systems are comprised of individual racks which 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 2 such midplanes. For the remainder of the paper, we will refer to partitions and jobs by node count; for example a 2K partition is comprised of four midplanes.

BG/P uses five different networks for different communication operations. The 3-D 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 bi-directional 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 using cables and link chips. Each link chip connects to a single midplane and to four unidirectional cables that connect midplanes. These cables can only be used 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 are remain available for use. This situation, referred to as passthrough, occurs frequently during partitioning. In this case, the local midplane is unable to connect to any other midplanes in this mesh dimension.

Blue Gene partitions can either be connected with a torus or 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

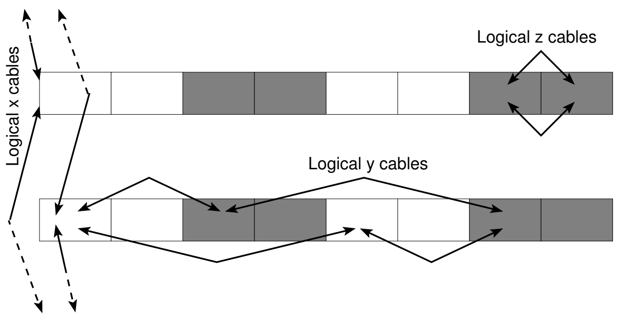

Figure 1: Blue Gene Cabling

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 quite 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 described 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.

3 The Intrepid System

Intrepid is a 556 TF Blue Gene/P system operated by Argonne National Laboratory for the Department of Energy INCITE program[8]. The system is comprised of 40 racks, 80 midplanes, containing a total of 163,840 cores. It was the entered the Top500[12] list as the number three system on the June 2008 list, and dropped to number five in the November 2008 list. Intrepid is a capability system, with single jobs frequently occupying substantial fractions of the system. 8K and 16K jobs are common on the system. Larger jobs occur often on the system as well. Jobs up to 32K nodes run without administrator assistance.

About a dozen application groups have INCITE applications on Intrepid. Each of these applications has completed a computational readiness review which measures their ability to run at large scale. Each application has different scalability; while some applications can effectively run at 40K, many can only scale to 4K or 8K before efficiency begins to drop off. Most users want to run most of their jobs on partitions between 1K and 4K. Moreover, because users are actively scaling their codes to larger sizes, job response times strongly affect user productivity.

Intrepid is a 40 rack, or 80 midplane system. The dimensions of the full system torus are 5x4x4. As described above, each row is a slice out of the X dimension. A full row (8K, 16 midplane) torus is 1x4x4. A limited number of torus configurations are possible, due to the constraints described above. A single 8K partition is possible, as are four 2K partitions. 1K and 4K partitions consume cabling resource that impact their neighbors, hence neither only a single 4K partition, or four 1K partitions can be run at the same time. The balance of the midplanes can be wired together using the remaining dimensions, so the combination of a 4K partition and two 2K partitions is valid. The combination of 4K and 1K rack partitions is particularly wasteful; the use of a single 4 rack partition and two 1K partitions requires that all four remaining midplanes are only usable individually. Use of the X dimension cables would allow connection of these midplanes, at the cost of preventing multi-row jobs from working. For operational reasons, this option is unavailable for small partitions on Intrepid.

Intrepid uses the Cobalt[2] resource manager. Cobalt is a component-based resource management suite that is 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[3].

4 Experimental Analysis

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

4.1 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.

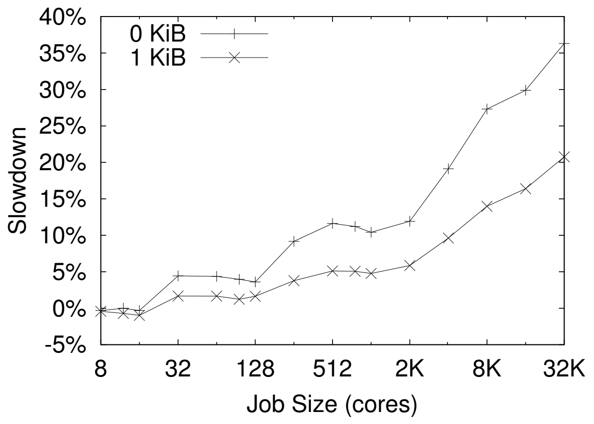

Figure 2: Latency slowdown

4.1.1 Point-to-point Latency

Figure 2 illustrates the slowdown in the maximum interprocess 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 because of 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 due to pipelining of data (i.e., the absolute difference remains constant, causing the percentage difference to reduce).

4.1.2 Effective Bisectional Bandwidth

Figure 3 illustrates the slowdown in the aggregate bandwidth reported by the B_eff benchmark (that is a part of the HPCC benchmark suite). Specifically, this benchmark presents the effective bisectional bandiwidth that is available on the system. The figure illustrates a slowdown of close to 35% for large system sizes. This is expected as the effective bisectional bandwidth would reduce when moving from a torus partition to a mesh partition due to the lesser number of communication links.

4.1.3 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`,

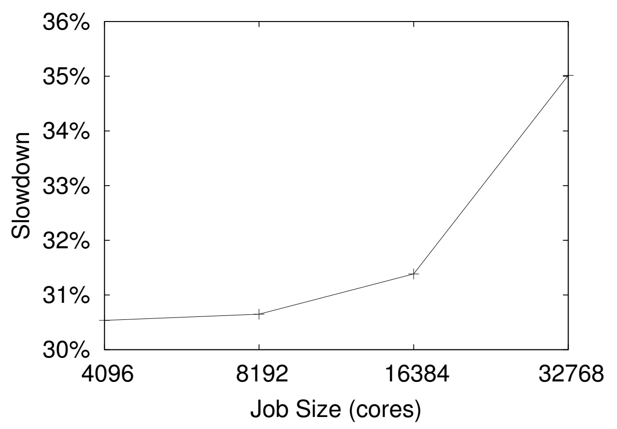

Figure 3: Slowdown in aggregate bandwidth reported by b_eff benchmark

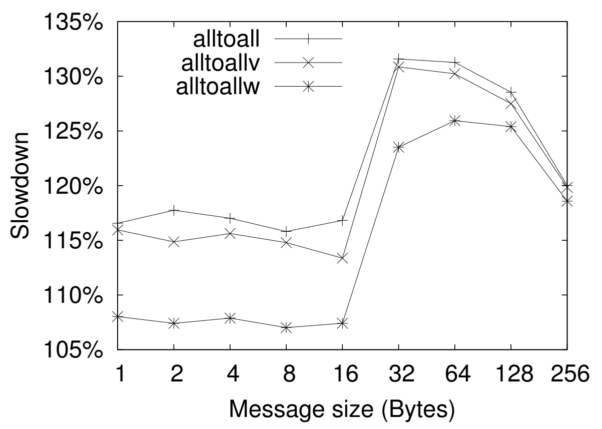

Figure 5: Alltoall, Alltoallv and Alltoallw slowdown 8192 cores

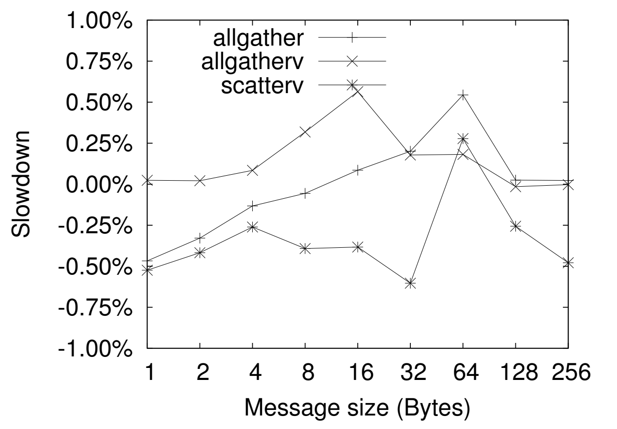

Figure 4: Allgather, Allgatherv and Scatterv slowdown 8192 cores

`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 very 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 due to space constraints). This is because all-to-all collectives perform the most amount of per-process communication; thus reducing the number of communication links causes the largest degradation for their performance.

4.2 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 using both mesh and torus networks. Likewise, we will describe P3DFFT, a 3D fast Fourier transformation library, and compare its relative performance on both networks.

NEK5000 [5] 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) [10] is an *ab-initio* light-nuclei computation code, which models nuclear structures and reactions from bare nuclear forces. This application uses the Automatic 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.

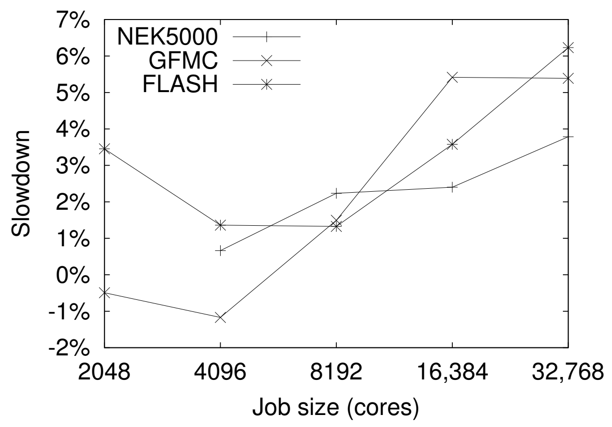

Figure 6: Slowdown of application runtimes due to using a mesh versus a torus configuration.

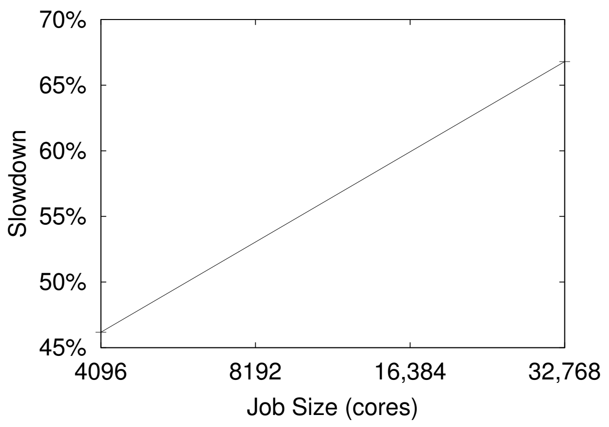

Figure 7: Slowdown of P3DFFT performance due to using a mesh versus a torus configuration.

Note, that because the servers are arranged on one side of the partition, in a torus configuration, because of wraparound, the servers have connectivity on both sides of the plane. However, in a mesh configuration, 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 [4] is the latest FLASH release from the ASC Center at the University of Chicago. The FLASH code [6] is an astrophysical MPI simulation code written in FORTRAN90 and C. We choose the Sedov 3D setup with adaptive mesh refinement module, Paramesh 4.0, with high refinement level (up to $lreine\_max = 12$) in our test runs. The Sedov 3D explosion problem (Sedov 1959) 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 2 K cores to 32 K cores, 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 32 K cores. Notice that for 2 K and 4 K 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 are either relatively localized, such as NEK5000, or are more bursty but staggered, which reduces congestion, such as GFMC.

P3DFFT[9] is a 3D fast Fourier transform library developed at SDSC. 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 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.

4.3 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. In order to evaluate whether this approach is effective, we simulate the effects of making mesh partitions available to the scheduler. Due to the cabling of Intrepid, it only makes sense to use 1K and 4K mesh partitions; if enough hardware is available to run 2K or 8K 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 3890 jobs, and reflects 2 weeks of activity. The INCITE program awards allocations on a January to January basis; hence, many new projects are just getting underway at the time of this writing. This biases this workload towards small jobs more than we typically see after the startup period for projects. Also, less 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 unitless job wait times; that is, the time a job as waited compared with the time requested. It also favors large nodecount 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 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 a realistic values for application slowdowns.

For the workload tests, utilization was effected substantially; it improved 3-5% for the cases simulated. This change had a 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 partitions were negatively impacted. This is to be expected; jobs of these sizes benefited from the resource contention experienced by 1K and 4K partitions. Response times for 1K and 4K partitions are greatly improved. Likewise, this is to be expected, because the switch from torus to mesh at these sizes reduces the resources needed to run these partitions. Finally, we note that WFP demonstrates diminished response times when applications have substantial (20%) degradation.

5 Related Work and Discussion

Mesh and torus networks have long been used on HPC platforms. Hence, much performance evaluation has been done[?]. More recently, the rivalry between Cray and Blue Gene systems has resulted in many comparisons (such as this[1] between the partitioned tori on Blue Gene systems and the shared torus available on Cray systems. While the benefits of wrapping meshes into tori have previously been studied, comparisons of meshes and tori in a partitioned environment have not been performed.

Likewise, mesh scheduling issues have also been studied. Many efforts[13] focus on efficient placement of jobs on shared tori to minimize latency and inter-job contention. However, no direct comparison of the scheduling impact of different configurations of partitioned torus networks has previously been performed.

6 Conclusions and Future Work

In this paper, we compare the costs and benefits of the use of mesh networks on Blue Gene/P systems. Using synthetic benchmarks, we demonstrated the performance penalty caused by the switch from torus to mesh networks. We were surprised by the limited impact of this change on MPI collective performance; with the exception of the alltoall collectives, most collectives were minimally impacted by the change.

When evaluating actual applications, we observed a similarly small performance penalty for three applications. The applications tested were slowed down by less than 7% at $32,768$ processes. Our benchmarks of P3DFFT showed a more substantial performance degradation, starting at 50%; we would expect this to be representative of some large scale applications. However, it is clear that a substantial fraction of applications are not substantially effected by the reduction in network performance caused by the switch from torus to mesh.

In our scheduling simulations, we found that substituting mesh partitions selectively for 1K and 4K torus partitions substantially improved scheduling performance, particularly as it relates to job wait times. Wait times improved as expected, with availability of 1K and 4K partition greatly improved. Until now, 512 node partitions and 2K partitions had benefited from the artificial scarcity of 1K and 4K partitions, so wait times for those partitions were slightly reduced.

In light of this information, we propose a hybrid scheme for production Blue Gene/P systems like Intrepid. By default, jobs running at problematic sizes (like 1K and 4K) would be routed to mesh partitions. Jobs demonstrating nontrivial slowdowns could be explicitly submitted to torus partitions of these sizes. This would allow the majority of jobs to run with a smaller resource footprint while still allowing users requiring the full torus to request it. We expect this approach to greatly improve scheduling performance in workloads where 1K and 4K jobs were common, such as the Intrepid workload.

Another factor to be considered is the sensitivity of applications to network degradation. If applications show nominal slowdown, as the applications we benchmarked did, this approach is clearly worthwhile. However, if applications with characteristics closer to P3DFFT, the benefit is less obvious. It is clear that any site considering de-

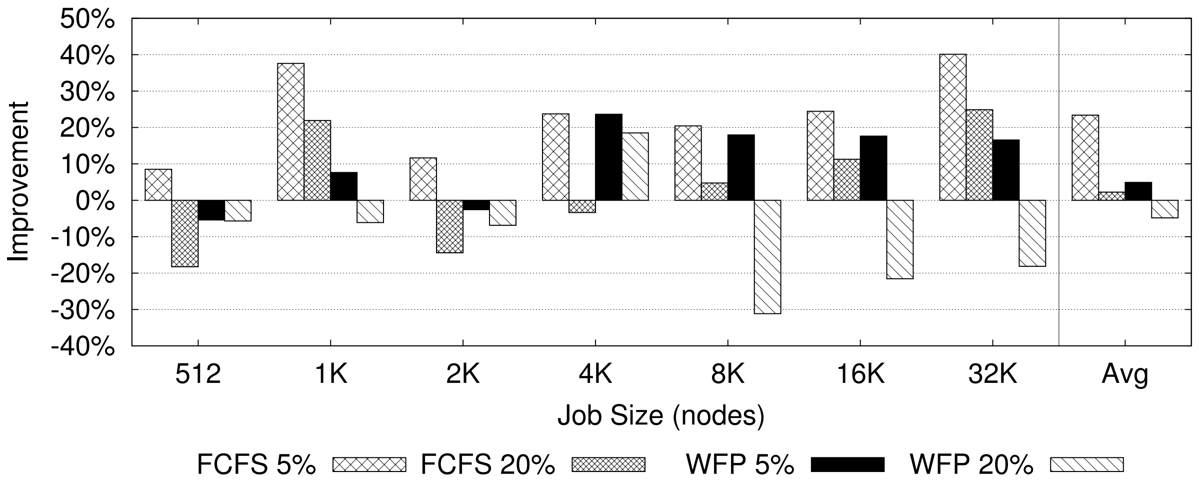

Figure 8: Response Times, by job size.

ploying such a scheduling policy should benchmark local applications, particularly those that consume large quantities of system resources.

While this work shows potential to greatly improve scheduler behavior on large Blue Gene systems, much work remains to be done. More application analysis should be done, to characterize other target applications in terms of their relative performance. Some of the performance degradation that we are seeing might be caused by collectives that are not tuned for mesh partitions, so it is possible that mesh-tuned collectives could yield less slowdown on mesh partitions.

7 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.

References

[1] Abhinav Bhatelé, Eric Bohm, and Laxmikant V. Kalé. A case study of communication optimizations on 3D mesh interconnects. Technical report, University of Illinois at Urbana-Champaign, 2008.

[2] Cobalt webpage, Feb. 2009. `http://trac.mcs.anl.gov/projects/cobalt`.

[3] Narayan Desai, Theron Voran, Ewing L. Lusk, and Andrew Cherry. The computer as software component: A mechanism for developing and testing resource management software. In *Proceedings of the 2007 IEEE International Conference on Cluster Computing, 17-20 September 2007, Austin, Texas, USA*, pages 58–63, 2007.

[4] A. Dubey, L.B. Reid, and R. Fisher. Introduction to FLASH 3.0, with application to supersonic turbulence. *Physica Scripta*, 2008. Special edition from Proceedings of the International Conference ”Turbulent Mixing and Beyond,” Trieste, Italy, August 2007.

[5] Paul Fischer, James Lottes, David Pointer, and Andrew Siegel. Petascale algorithms for reactor hydrodynamics. *J. Phys. Conf. Series*, 2008.

[6] B. Fryxell, K. Olson, P. Ricker, F. X. Timmes, M. Zingale, D. Q. Lamb, P. MacNeice, R. Rosner, J. W. Truran, and H. Tufo. FLASH: An adaptive mesh hydrodynamics code for modeling astrophysical thermonuclear flashes. *Astrophysical Journal, Supplement*, 131:273–334, November 2000.

[7] Alan Gara, Matthias A. Blumrich, Dong Chen, George L.-T. Chiu, Paul Coteus, Mark Giampapa, Ruud A. Haring, Philip Heidelberger, Dirk Hoenicke, Gerard V. Kopcsay, Thomas A. Liebsch, Martin Ohmacht, Burkhard D. Steinmacher-Burow, Todd Takken, and Pavlos Vranas. Overview of the Blue Gene/L system architecture. *IBM Journal of Research and Development*, 49(2-3):195–212, 2005.

[8] INCITE webpage, Feb. 2009. `http://www.sc.doe.gov/ascr/incite`.

[9] Dmitry Pekurovsky. P3DFFT webpage, Feb. 2009. `http://www.sdsc.edu/us/resources/p3dfft/index.php`.

[10] Steven C. Pieper and R. B. Wiringa. Quantum Monte Carlo Calculations of Light Nuclei. *Annu. Rev. Nucl. Part. Sci.*, 51:53, 2001.

[11] IBM Blue Gene Team. Overview of the IBM Blue Gene/P project. *IBM Journal of Research and Development*, 52(1-2):199–220, 2008.

[12] Top500 webpage, Feb. 2009. `http://www.top500.org`.

[13] Deborah Weisser, Nick Nystrom, Chad Vizino, Shawn T. Brown, and John Urbanic. Optimizing job placement on the Cray XT3. In *48th Cray User Group Proceedings*, 2006.