Improving Batch Scheduling on Blue Gene/Q by Relaxing Network Allocation Constraints

Zhou Zhou, Xu Yang, *Student Member, IEEE*, Zhiling Lan, Paul Rich, Wei Tang, Vitali Morozov, and Narayan Desai

Abstract—As systems scale toward exascale, many resources will become increasingly constrained. While some of these resources have historically been explicitly allocated, many—such as network bandwidth, I/O bandwidth, or power—have not. As systems continue to evolve, we expect many such resources to become explicitly managed. This change will pose critical challenges to resource management and job scheduling. In this paper, we explore the potential of relaxing network allocation constraints for Blue Gene systems. Our objective is to improve the batch scheduling performance, where the partition-based interconnect architecture provides a unique opportunity to explicitly allocate network resources to jobs. This paper makes three major contributions. The first is substantial benchmarking of parallel applications, focusing on assessing application sensitivity to communication bandwidth at large scale. The second is three new scheduling schemes using relaxed network allocation and targeted at balancing individual job performance with overall system performance. The third is a comparative study of our scheduling schemes versus the existing scheduler on Mira, a 48-rack Blue Gene/Q system at Argonne National Laboratory. Specifically, we use job traces collected from this production system.

Index Terms—Job scheduling, resource management, network partition, torus/mesh topology

1 Introduction

The demand for more computing power seems insatiable, which continues to drive the deployment of ever-growing supercomputers. Production systems already contain hundreds of thousands of processors and are headed to millions [1]. These systems are used to tackle scientific problems of increasing size and complexity, with diverse requirements for resources. The cost to build and maintain such systems can be significant. For example, the K Computer in Japan cost more than $1 billion to build and $10 million to operate each year. Therefore, a significant number of shared resources such as the communication infrastructure are utilized to achieve high performance while controlling costs. However, it is difficult to efficiently manage these shared resources. With plans under way to achieve exascale computing within the next few decades, utilizing these shared resources is becoming increasingly important.

In order to harness the full potential of extreme-scale systems, *resource management* or *job scheduling* (i.e., effectively allocating available resources to applications) is of paramount importance. Modern resource management systems focus primarily on effective use of job-dedicated resources such as processors, memory, and accelerators. Because of a variety of ever-growing trends in computer technology, the ratio of CPU and memory to shared resources is decreasing. In the near future, *management of shared resources such as network and bandwidth will become increasingly critical*.

Torus-based networks are prevalent in high-end supercomputers because of their linear scaling on per-node cost as well as their competitive communication performance. The IBM Blue Gene/L (BG/L) [2], Blue Gene/P (BG/P) [3], [2], [4], [5], and Cray XT systems [6] use a 3D torus network for communication. The Blue Gene/Q (BG/Q) system has its nodes electrically interconnected in a 5D torus network [7], [8]. The K computer from Japan uses the "Tofu" system, which is a 6D mesh/torus topology [9]. On the recent Top 500 list, 6 of the top 10 supercomputers use a high-radix torus-interconnected network [1]. Traditionally, these systems use a single torus network, which could impact application performance because of network sharing (e.g., job interference and communication contention). In order to address this issue, Blue Gene systems [3], [2] use a *network partitioning* mechanism in which the network interconnect is reconfigured to provide private, per-job networks to compute nodes [10], [11]. In such a way, the whole torus network can be partitioned into multiple smaller tori in a limited number of ways [12]. Once a network partition is established, the job running on the partition can benefit from the dedicated synchronization network where all required hardware is dedicated to the job.

Although a partition-based system provides jobs with dedicated network resources and bandwidth, the use of partitions introduces a new problem: resource contention caused by exclusively allocating shared network resources to a single job [13]. When partitioning the system into schedulable partitions on a BG/Q system, some block wiring adjustments should be made beyond the overall geometry of the selected

- *Z. Zhou, X. Yang, and Z. Lan are with the Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616. E-mail: {zzhou1, xyang56}@hawk.iit.edu, lan@iit.edu.*
- *P. Rich and V. Morozov are with the Argonne Leadership Computing Facility, Argonne National Laboratory, Lemont, IL 60439. E-mail: richp@alcf.anl.gov, morozov@anl.gov.*
- *N. Desai is with the Ericsson Inc, San Jose, CA 95134. E-mail: narayan.desai@ericsson.com.*
- *W. Tang is with the Google Inc, New York, NY 10018. E-mail: weitang@google.com.*

Manuscript received 22 Sept. 2015; revised 12 Jan. 2016; accepted 27 Jan. 2016. Date of publication 11 Feb. 2016; date of current version 12 Oct. 2016. Recommended for acceptance by R. Brightwell.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2016.2528247

1045-9219 © 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on February 15,2021 at 23:37:19 UTC from IEEE Xplore. Restrictions apply.

nodes on the system. This operation is critical and must be performed with great caution. A common case on the BG/Q system is that a 1,024-node partition in a torus configuration is always in contention for wiring resources with other partitions around it. Such a contention can cause unusable configurations of node resources and prevent other nodes from being serviceable, regardless of node state. In other words, the hardware serviceability may be affected by inappropriate configurations which are prone to resource contention. For example, even if some nodes are idle, they still cannot be grouped together to serve a job because the wirings between them are occupied by other jobs. This issue can deteriorate both job response times and system utilization.

Inspired by the partition-based design and the flexible wiring control deployed on BG/Q, in this paper we exploit the potential of utilizing these features to relax network resource allocation, with the objective of improving job scheduling. The partition-based design of BG/Q provides a unique opportunity to explicitly allocate network resources (i.e., links or bandwidth) to jobs in a way that is impossible on other systems. When assigning wirings to carry traffic among a partition's nodes, the torus link to "wrap around" the nodes network can by dynamically turned on or omitted from the partition's specification. While this capability is currently rare, we expect it to become more common. Unlike the conventional scheduling approach that considers only the job's requirement on processor and memory for job allocation, our design also takes into account the application's communication features and treats the wirings between nodes as schedulable resources.

More specifically, this paper makes three major contributions:

1) Substantial benchmarking of applications, focusing on assessing application sensitivity to network configuration at large scale. In particular, we evaluate a number of parallel benchmarks and DOE leadership applications on the production system Mira at Argonne by analyzing their performance variation under different network configurations.
2) Design of three new scheduling schemes for BG/Q machines. These schemes have different characteristics, including network configurations and scheduling policies. In particular, we propose a communication-aware scheduling policy that selectively allocates network resource to users' jobs according to job communication characteristics.
3) Comprehensive evaluation of our scheduling schemes, through trace-based simulations using real workloads from Mira.

Our major findings are twofold:

1) Not all applications are sensitive to the network topology change. An application's performance under different network configurations depends mainly on its communication patterns and the proportion of communication over the total runtime.
2) Our new scheduling schemes can significantly improve scheduling performance by up to 60 percent in job response time and 17 percent in system utilization.

The remainder of this paper is organized as follows. Section 2 introduces background about the BG/Q system Mira at Argonne. Section 3 presents the results of benchmarking applications on Mira. Section 4 describes our new batch scheduling schemes. Section 5 presents a scheduling study. Section 6 describes system partitioning. Section 7 discusses related work. Section 8 summarizes our conclusions and points out future work.

2 Background

2.1 Mira: The IBM Blue Gene/Q System at Argonne

Mira is a 10 PFLOPS (peak) BG/Q system operated by Argonne National Laboratory for the U.S. Department of Energy [1]. It is a 48-rack system, arranged in three rows of 16 racks. Each rack contains $1{,}024$ 16-core nodes, and the whole system has a total of $786{,}432$ cores. Mira has a hierarchical structure: nodes are grouped into midplanes, each midplane contains 512 nodes in a $4 \times 4 \times 4 \times 4 \times 2$ structure, and each rack has two such midplanes. It was ranked fifth in the latest Top500 list published in November 2015 [1]. Mira uses a 5D torus-connected network. These five dimensions are referred to as the *A*,*B*,*C*,*D*,*E* dimensions. Each node in the machine has a unique set of coordinates on the full machine partition. Mira is a capability system, with single jobs frequently occupying substantial fractions of the system. The smallest production job on Mira occupies 512 nodes; 8,192-node and 16,384-node jobs are common on the system; larger jobs also occur frequently. Jobs up to the full size of Mira run without administrator assistance. Time on Mira is awarded primarily through the Innovative and Novel Computational Impact on theory and Experiment (INCITE) program [14] and the ASCR Leadership Computing Challenge (ALCC) program [15].

2.2 Partitioning on Mira

Mira uses network partitioning for job scheduling. Partitions can be constructed only in a limited set of ways. A partition must be a uniform length in each of the dimensions. Midplanes used to build partitions must be connected through a single dimension and form a roughly rectangular prism in five dimensions. Because the final dimension is used only to connect nodes within a single midplane, all partitions are length 1 in the *E* dimension. Additionally, the creation of a partition uses network resources in a dedicated fashion, preventing their use in other partitions. For a partition size to be valid, there must be a set of partition lengths in each dimension that results in a properly sized 5D prism of nodes. Partitions also require use of complete midplanes, so all partitions on the system are multiples of 512 nodes. For a given size, several partition variants may exist with different shapes.

Fig. 1 illustrates the flat view of the network topology of Mira with two halves and three rows. Each square labeled with "RXX" represents a rack. Each rack contains two vertical midplanes (not shown in the figure). As we can see, the whole machine is split into six 8-rack sections. Each node in Mira has a unique logical coordinate (A, B, C, D, E). Given the logical coordinate of a node, we can translate the logical address of a node to the midplane location. The A coordinate decides which half of the machine the node is on. The B coordinate

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on February 15,2021 at 23:37:19 UTC from IEEE Xplore. Restrictions apply.

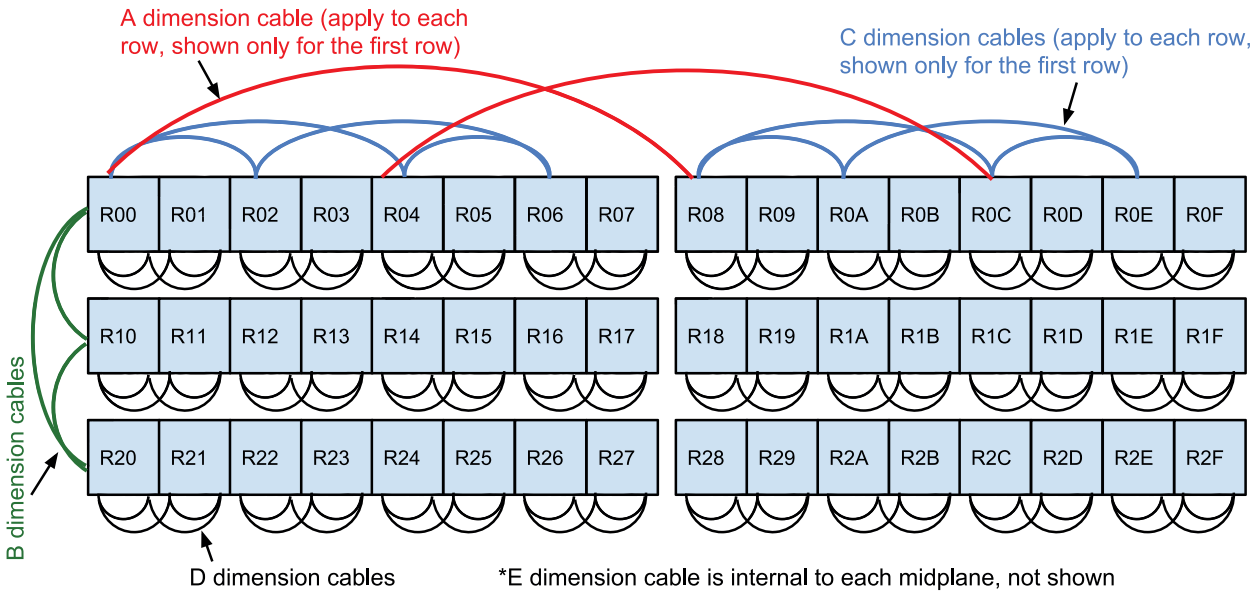


Fig. 1. Flat view of mira's network topology.

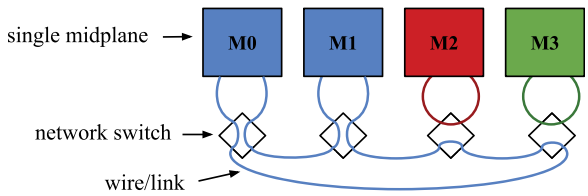
decides which row of the machine the node is on. The *C* coordinate refers to a set of four midplanes in two neighboring racks. The blue lines represent the cables that link racks together. Since the coordinate is based on a logical location and follows the midplane cable, this coordinate appears to jump around the 8-rack segment as illustrated in the figure. The *D* coordinate refers to a single midplane in two neighboring racks. Since the cable makes a loop around two racks, the coordinate loops in a clockwise direction. On Mira, a partition of the fixed size may have multiple geometry, which appears to have different "shapes". Typically, a more "compact" partition can provide better network communication performance than a relatively more "loose" partition.

2.3 Network Resource Contention

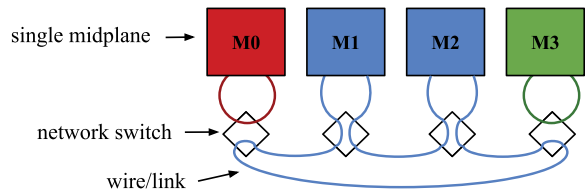
One unique feature of the BG/Q architecture is the ability to explicitly allocate network performance to jobs. When building a partition, a shared pool of network resources is allocated to a single partition at a time. If sufficient resources are dedicated to a partition, it will have a torus network. Alternatively, if fewer resources are allocated, the partition will have only a mesh network, in which the outside faces of the mesh are not connected except where the internal midplane faces connect to one another. The performance of the torus partition is considerably better than that of the mesh; the worst case and average hop counts between nodes are reduced, and more bandwidth is available to applications. This difference in performance will affect application performance, particularly for communication-intensive applications.

As we mentioned previously, a shared pool of resources is used to connect single midplanes into larger partitions, while midplanes include enough dedicated hardware to produce a full torus network in all dimensions on a single-midplane partition. Network contention is a substantial challenge for resource allocation on BG/Q systems. For example, it is possible that idle midplanes cannot be wired together to satisfy a job's resource request, because of a lack of wiring resources, as shown in Fig. 2. This situation can occur even when midplane positions meet all geometric constraints for partition creation.

Partitions have a variable dependence on the shared pool of network resources depending on their shape and the overall size of the full system. When a partition is only one midplane long in a direction, no external wiring resources are needed. In effect, with single midplane partitions, none of the shared pool will be used, while the entire pool will be consumed for a full system partition. Both cases are simple to satisfy from a paired allocation perspective, since no mismatch occurs between the free midplanes and wiring resources. Other partition sizes consume quantities of wiring resources similar to those of the full system partition, while leaving



(a) M0 and M1 as a partition



(b) M1 and M2 as a partition

Fig. 2. Wire contention between midplanes. The figure is a schematic representation of a four-midplane-long dimension showing a two-midplane torus and two single midplane tori. Because of the wiring of the two-midplane torus (blue) and the exclusivity of partitioning on Mira, this wiring prevents the formation of a torus or a mesh with the remaining two midplanes in this dimension. As shown, once two midplanes (M0 and M1 in (a) and M1 and M2 in (b)) are linked together to form a 1K partition, they will consume all the wire resources along this four-midplane-long dimension. This is representative of both the C and D dimensions on Mira.

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on February 15,2021 at 23:37:19 UTC from IEEE Xplore. Restrictions apply.

other midplanes free. In this case, however, allocation of the remaining free midplanes is constrained by the lack of wiring resources. With this metric, some partition configurations can be thought of as monopolizing wiring resources.

2.4 Batch Scheduling on Partition-Based Systems

Batch scheduling on a partition-based system comprises two parts: *network configuration* and *scheduling policy*. For network configuration, Mira uses a configuration in which all partitions are fully torus-connected. The partition size ranges from 512 nodes (single midplane) to the whole machine (48 racks). The scheduling policy consists of two phases: job prioritization and node allocation. The most commonly used priority policies include FCFS (first-come-first-served) [16], [17], [18], SJF (shortest-job-first) [19], and RR (round-robin) [20], [21], [22]. Other techniques such as conservative backfilling [23] and EASY backfilling [24], [25] were proposed to improve the system utilization and have been adopted by many production schedulers [16], [26]. On Mira, the resource manager uses a priority policy called "WFP" to order the jobs in the queue [27], [28]. The WFP policy favors large and old jobs, adjusting their priorities based on the ratio of their wait times to their requested runtimes, which was also suggested in [29]. Upon each scheduling, the job at the head of the wait queue is selected and allocated to a partition. Also the EASY backfilling [24], [25] is used to make reservations, if possible, for jobs arriving later. Next, a least-blocking (LB) scheme is used to choose the partition that causes the minimum network contention out of all candidates [28].

3 Application Benchmarking

To properly use these mesh- and contention-free partitions, we need to understand how they can affect the application's performance. We first investigate the impact of partition configuration on application performance. We choose four parallel benchmarks and three DOE leadership applications for benchmarking. The results will be used as a baseline for experimental comparison in Section 5. To quantify performance difference, we define

$$runtime\_slowdown = \frac{T_{mesh} - T_{torus}}{T_{torus}}, \tag{1}$$

where T_{mesh} is the application runtime on a mesh partition and T_{torus} is the application runtime on a torus partition.

3.1 Parallel Benchmarks and Applications

In this paper, we use the NAS Parallel Benchmarks, in particular NPB3.3, which has a larger problem size (class E), 16X size increase from the previous largest class [30]. We choose three kernel benchmarks: LU, FT, and MG. LU solves synthetic systems of nonlinear partial differential equations. FT solves a three-dimensional partial differential equation using a fast Fourier transform (FFT). MG solves a three-dimensional discrete Poisson equation using the V-cycle multigrid method.

We also study four scientific applications: Nek5000, FLASH, DNS3D, and LAMMPS. The applications are used routinely by a number of INCITE projects.

Nek 5000 [31] is a spectral-element computational fluid dynamics code developed at Argonne National Laboratory. It features spectral-element multigrid solvers coupled with 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.

FLASH 4.0 [32] is the lastest FLASH release from the Advanced Simulation and Computation (ASC) Center at the University of Chicago. The FLASH code [33] is a multiphysics simulation code written in Fortran90 and C using MPI with OpenMP. The driven turbulence setup is run using the split-PPM hydrodynamics solver and the uniform grid module, in a weak-scaling mode. This problem was run at a large scale on the BG/L at Lawrence Livermore National Laboratory [34] and is known to be highly scalable.

DNS3D is a direct numerical simulation code that solves viscous fluid dynamics equations in a periodic rectangular 3-D domain with a pseudo-spectral method of fourth-order finite differences and with the standard Runge-Kutta fourth-order time-stepping scheme [35]. DNS3D is written in Fortran and is a pure MPI application. Similar to other spectral codes, DNS3D can use either slabs or pencil domain decompositions, depending on the type of Fourier transforms executed. In both strategies, decomposition is not performed in at least one of the coordinate directions. DNS3D is highly dependent on network performance, since during each time step it executes three Fourier transforms for three 3-D scalar variables. This approach can effectively be transformed into all-to-all-type computations. Because of a relatively low memory footprint per MPI rank, DNS3D runs effectively using 64 MPI ranks per node on BG/Q; and compared with other MPI-only codes, it shows a high per-node performance on this machine.

LAMMPS ("Large-scale Atomic/Molecular Massively Parallel Simulator") [36] is a general-purpose molecular dynamics software package for massively parallel computers. Developed at Sandia National Laboratories, it is written in an exceptionally clean style that makes it one of the most popular codes for users to extend, and it currently has dozens of user-developed extensions. LAMMPS was trivially ported to BG/Q shortly after the first installations came online, and it has run on the entire Mira system using millions of MPI processes [37].

3.2 Partition Types

Here we discuss two traditional partition types: torus and mesh. Then we introduce a new mechanism to achieve contention-free partitioning.

3.2.1 Torus Partition

The most popular partition used on Mira has a five-dimensional torus network topology. Besides having direct links between the nearest neighbors in the A, B, C, D, and E dimensions, a torus partition also has a wrap-around link connecting the head and tail nodes in each dimension. The torus partition can provide the best network communication performance; however, it is also the most "expensive" partition because it consumes extra wrap-around links.

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on February 15,2021 at 23:37:19 UTC from IEEE Xplore. Restrictions apply.

dimension	A	B	C	D	E
geometry	1	1	1	2	1
connectivity	torus	torus	torus	mesh	torus

Fig. 3. Contention-free partition of 1K size.

3.2.2 Mesh Partition

A mesh partition can be built based on a torus partition by turning off every wrap-around link in each dimension. This type of mesh partition requires the least amount of wiring resource compared with a full torus partition of the same geometry. It does not cause any wiring contention in any dimension, but the cost is the sacrifice of network communication because of the doubled average maximum hop count.

3.2.3 Contention-Free Partitions on Mira

Contention for wiring resources is a critical issue on Blue Gene systems. The contention always takes place when a partition has one or multiple dimensions of which the length is larger than one. Fig. 2 shows the wiring connectivity on the *D* dimension of a 1K torus partition (blue rectangle). This 1K torus partition may cause potential contention for wiring resources with other two midplanes because its *D* dimension length is 2. Fortunately, the BG/Q system has a more flexible mechanism for controlling switches and wirings. Unlike the BG/P system with the rigid restriction on wiring resources that all dimensions must be configured as a torus or mesh, the BG/Q system can independently control a partition's wiring connectivity on each single dimension. Thus, on Mira we can build new partitions configured with mixed torus and mesh dimensions. We set contention-prone dimensions as mesh-connected while still keeping other dimensions as torus-connected. By building such partitions, we can ensure that the wiring contention does not happen uniformly in all dimensions. In this paper, we call these "contention-free" partitions. For example, as shown in Fig. 3, we turn the *D* dimension of the 1K partition into a mesh, while still having the other four dimensions torus-connected. This contention-free 1K partition does not consume any extra wiring resources compared with a torus partition, and it can provide better communication performance. Similarly to a full-mesh partition, this new type of 1K partition also does not cause any wiring contention on its torus-connected dimensions. On Mira, we build such partitions with sizes 1, 4, and 32 K. Compared with full-mesh partitions, these contention-free partitions cause less performance degradation in application runtime. The reason is that an application can still benefit from the torus links in dimensions *A*, *B*, *C*, and *E*. Our new scheduling schemes takes advantage of these contention-free partitions to improve scheduling performance.

Since we have three types of partitions—*torus*, *mesh*, and *contention-free*—we conduct experiments on application benchmarking separately. Because of the computing resource availability, for torus versus mesh partition we run these applications on the partitions of sizes 2, 4, and 8 K nodes, and for torus versus contention-free partitions we run these applications on the partitions of sizes 1 and 4 K.

TABLE 1
Application Runtime Slowdown on Mesh Partitions

Name	Runtime Slowdown		
	2 K	4 K	8 K
NPB:LU	3.25%	0.01%	0.03%
NPB:FT	22.44%	23.26%	21.69%
NPB:MG	0.00%	11.61%	19.77%
Nek5000	0.95%	0.02%	0.44%
FLASH	0.83%	5.48%	4.89%
DNS3D	39.10%	34.51%	31.29%
LAMMPS	0.02%	0.87%	0.97%

3.3 Results of Torus versus Mesh Partitions

Table 1 presents application slowdowns of NPB benchmarks on mesh partitions. Obviously, LU is not sensitive to the switching from torus to mesh. It has less than 4 percent slowdown at size 2 K and close to zero slowdown when the computing scale is increased to 4 and 8 K. The algorithm of LU is not highly parallelized, and most of its MPI routines are blocking communication. This approach leads to no performance loss when the network topology configuration is changed from torus to mesh.

MG shows no slowdown at size 2 K. When the computing scale is increased, however, we observe a 12 percent slowdown at size 4 K and nearly 20 percent slowdown at size 8 K. MG has unique communication patterns. In particular, it involves both near-neighbor communication and long-distance communication, so its performance is sensitive to network topology changes.

FT also is sensitive to the network topology. At all three sizes, its slowdown is more than 20 percent. The code performs global data communication for its FFTs [38]. This is the main reason that the performance drops significantly when using mesh partitions with reduced bisection bandwidth.

The slowdown results of the leadership applications also are presented in Table 1. For LAMMPS and Nek5000, the use of mesh partitions has minimal impact on their performance: the slowdowns are always less than 1 percent.

In Nek5000, every process is communicating to 50 to 300 geometrically neighbor processes, which in practice means about 2 to 3 hops away from the source. For a torus, the process on the "border" node does not notice any difference because, in some sense, there are no borders in torus topology. For a mesh, the process will have half the neighbors located in the same semi-plane as in the torus partition, but half the others will need to reuse the path of the semi-plane. The slowdown really depends on the level of multigrid refinement and the placement of the processes relative to each other.

In FLASH, the slowdown is no more than 5 percent on the 4 and 8 K partition. FLASH's runtime is dominated by computation, not communication, and at the larger sizes takes up on the order of 14 percent of the runtime. The reason is that the communication algorithm is largely point to point and generally fairly local. Because of the periodic boundary conditions of the physics in the problem, we do get a small but significant amount of off-node communication on the wrap-around links. For example, for 8 K partitions, the torus spends only 14 percent of its time in communication, whereas the mesh partition has communication for 17 percent of the runtime. We see a 23 percent

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on February 15,2021 at 23:37:19 UTC from IEEE Xplore. Restrictions apply.

TABLE 2
Application Runtime Slowdown on Contention-Free Partitions

Name	Runtime	Slowdown
	1K	4K
NPB:LU	1.63%	1.05%
NPB:FT	12.72%	11%
NPB:MG	1.20%	1%
Nek5000	1.00%	1%
FLASH	0.00%	0.05%
DNS3D	24.53%	18%
LAMMPS	0.02%	0%

slowdown in communication, which translates into about 5 percent slowdown of runtime.

DNS3D exhibits substantial slowdown when switching from a torus to a mesh partition. Among the three partition sizes from 2 to 8 K, the slowdown is always above 30 percent. In some cases (e.g., 2 K), the slowdown is close to 40 percent. Our MPI profiling shows that DNS3D spends 60 percent of its runtime in MPI_Alltoall(). MPI_Alltoall() is scales proportion to the bisection bandwidth of a partition. If one of the partition dimensions becomes a mesh, the bisection bandwidth of the partition is reduced by half. Therefore, it takes two times longer for MPI_Alltoall() to complete. Hence, we observe a 30 percent performance degradation here. Clearly, certain applications are sensitive to communication bandwidth, especially those heavily using MPI collective calls.

3.4 Results of Torus versus Contention-Free Partitions

Table 2 presents the application runtime slowdown on the contention-free partitions. Except FT and DNS3D, the other benchmarks are almost unaffected by the switching from torus to contention-free partitions. The largest slowdown is even less than 2 percent on LU. The slowdown of FT at both 1 and 4 K sizes is less than that in Table 1. For DNS3D, the slowdown is still notable which is around 20 percent. However, compared with the slowdown in 1 which is above 30 percent, the contention-free partition show its advantage against the mesh.

In summary, our benchmarking results demonstrate that the application's communication pattern is a key factor influencing application runtime under different network configurations. Applications dominated by local communications are not sensitive to the network topology changing from torus to mesh, whereas applications having a substantial amount of long-distance or global communications are prone to performance loss when running on mesh partitions.

4 Three New Scheduling Schemes Using Relaxed Allocation Constraints

In this section we present three new scheduling schemes to improve batch scheduling. While this work targets Mira, these new scheduling designs are applicable to all BG/Q systems and other 5D torus-connected systems. Using mesh and contention-free partitions requires fewer links than full torus partitions do. According to the benchmarking results presented in Section 3, we can observe that for many applications, the performance loss caused by mesh and contention-free configurations is not substantial. Inspired by the observation, we intend to relax resource allocation constraints and improve system scheduling performance.

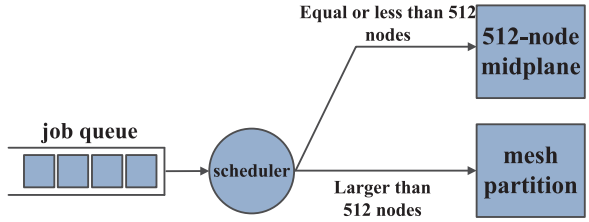


Fig. 4. The MeshSched scheduling scheme.

4.1 MeshSched

As shown in Fig. 4, we propose a mesh-based scheduling policy that uses a full mesh network configuration. This configuration is generated from the current one on Mira by turning every torus partition into a mesh partition except the 512-node partition, which must be a torus. Specifically, wrap-around torus links are turned off in each dimension, consequently reducing the potential link contention between neighboring partitions. Resources can be more freely allocated without the constraint of wrap-around links. Obviously, runtime slowdown may occur for some communication-intensive applications since mesh partitions reduce the bisection bandwidth between two nodes.

4.2 Contention-Free and Communication-Aware (CFCA)

On Mira, almost every partition can be configured to a contention-free version except the 512-node single midplane and the 48-rack whole machine. Given that possibility, the job scheduler has many more options when making scheduling decisions. Thus, using the contention-free partitions, we propose a new scheduling scheme that takes an application's communication intensity into account, as shown in Fig. 5. Compared with mesh partitions, the new contention-free partitions can preserve the application performance as much as possible without causing resource contention. We build a new network configuration on Mira by adding these contention-free partitions to the current configuration on Mira. We also develop a communication-aware scheduling policy, as shown in Fig. 5. The new scheduling scheme allocates communication-sensitive jobs to torus partitions and

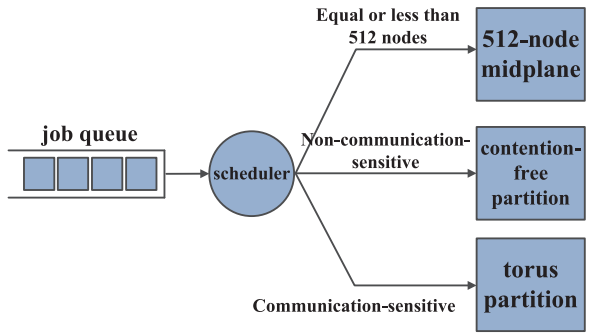


Fig. 5. The CFCA scheduling scheme.

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on February 15,2021 at 23:37:19 UTC from IEEE Xplore. Restrictions apply.

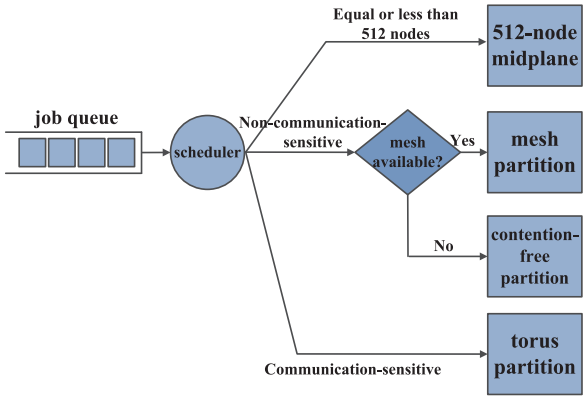


Fig. 6. The Hybrid scheduling scheme.

allocates non-communication-sensitive jobs to contention-free partitions. By doing so, this scheduler seeks to balance user requirements and system performance. Note that the single 512-node midplane must always be a torus. Any jobs requiring no more than 512 nodes should directly be routed to a single midplane. Further, with the performance monitoring support on Mira, an application's sensitivity to network topology can be determined empirically.

4.3 Hybrid

We also propose a new "hybrid" scheduling scheme that can be treated as a combination of *MeshSched* and *CFCA*. We build a complete network configuration including torus partitions and their corresponding mesh partitions, and all contention-free partitions. As shown in Fig. 6, the scheduling policy works in a way similar to that of CFCA. Communication-sensitive jobs are still routed to torus partitions, and non-communication-sensitive jobs are preferentially allocated to mesh partitions if any are available. Otherwise, the most suitable contention-free partition is chosen to run the job using the WFP policy. This scheduling scheme fully utilizes the contention-free partitions to provide performance comparable to that of the torus partition. Moreover, it take advantages of the mesh partition to minimize the resource contention.

5 Experiments

In this section, we compare our new scheduling methods under a variety of workloads using trace-based simulation. The goal is to investigate the benefit of our design compared with the current one used on Mira.

5.1 QSim Simulator

Qsim is an event-driven scheduling simulator for Cobalt, the resource management and job scheduling package used on the 48-rack Mira. Taking the historical job trace as input, Qsim quickly replays the job scheduling and resource allocation behavior and generates a new sequence of scheduling events as an output log. Qsim uses the same scheduling and resource allocation code that is used by Cobalt and thus will provide accurate resource management and scheduling simulation. Qsim is open source and available along with the Cobalt code releases [27]. It was used in our previous work [28], [40], [10].

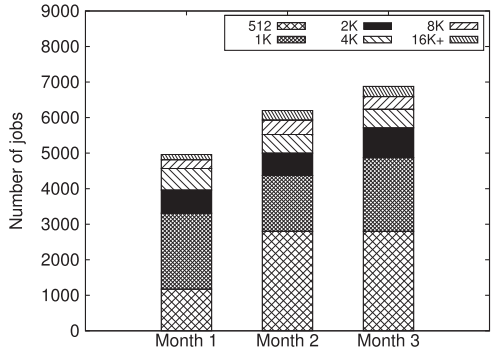


Fig. 7. Job size distribution.

5.2 Job Trace

We use a three-month workload trace collected from Mira. Fig. 7 summarizes the jobs in these months. As we can see, the 512-node, 1 K, and 4 K jobs are the majority. For months 2 and 3, 512-node jobs account for half of the jobs. While the number of large-sized jobs (more than 8 K nodes) is relatively low, these jobs consume a considerable amount of node-hours because of their sizes.

5.3 Evaluation Metrics

Four metrics are used for scheduling evaluation:

- *Average job wait time*. This metric denotes the average time elapsed between the moment a job is submitted and the moment it is allocated to run. It is commonly used to reflect the "efficiency" of a scheduling policy.
- *Average response time*. This metric denotes the average time elapsed between the moment a job is submitted and the moment it is completed. Similar to the above metric, it is often used to measure scheduling performance from the user's perspective.
- *System utilization*. System utilization rate is measured by the ratio of busy node-hours to the total node-hours during a given period of time [40], [41]. The utilization rate at the stabilized system status (excluding warm-up and cool-down phases of a workload) is an important metric of how well a system is utilized.
- *Loss of capacity (LoC.)* LoC measures system fragmentation [10]. A system incurs LoC when it has jobs waiting in the queue to execute and when it has sufficient idle nodes but still cannot execute those waiting jobs because of fragmentation. A scheduling event takes place whenever a new job arrives or an executing job terminates. Let us assume the system has N nodes and m scheduling events, which occur when a new job arrives or a running job terminates, indicated by monotonically nondecreasing times t_i, for $i = 1...m$. Let n_i be the number of nodes left idle between the scheduling event i and $i+1$. Let δ_i be 1 if any jobs are waiting in the queue after scheduling event i and at least one is smaller than the number of idle nodes n_i, and 0 otherwise. Then LoC is defined as follows:

$$LoC = \frac{\sum_{i=1}^{m-1} n_i(t_{i+1} - t_i)\delta_i}{N \times (t_m - t_1)}. \quad (2)$$

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on February 15,2021 at 23:37:19 UTC from IEEE Xplore. Restrictions apply.

TABLE 3
Scheduling Schemes Used in the Experiments

Name	Network Configuration	Scheduling Policy
Native	Current config used on Mira consisting of all torus partitions	WFP and LB (see Section II)
MeshSched	All possible mesh partitions and 512-node torus	MeshSched policy described in Fig. 4
CFCA	Current config used on Mira and additional contention-free partitions (1K, 4K, and 32K)	Contention-free and Communication-aware policy described in Fig. 5
Hybrid	Current config used on Mira plus all mesh partitions and contention-free partitions	Combination of *MeshSched* and *CFCA* described in Fig. 6

5.4 Results

In our experiments we compare our new scheduling methods (termed *MeshSched*, *CFCA*, and *Hybrid*) with the one currently used on Mira, namely, *Native*. Table 3 summarizes these scheduling methods.

We categorize jobs into communication-sensitive and non-communication-sensitive jobs. For each simulation, we set five slowdown levels for applications running on mesh partitions: 10, 20, 30, 40, and 50 percent. For example, slowdown of 10 percent means the runtime increases by 10 percent on mesh partitions. We also tune the percentage of communication-sensitive jobs in the workload. Similarly, five ratios are used: 10, 20, 30, 40, and 50 percent. We conduct experiments using the workload on a monthly base (3 months). In total we have 300 ($3 \times 4 \times 5 \times 5$) sets of experiments.

Because of space limitations, we present only a few representative results. To improve the figure readability, we present the results when the percentage of communication-sensitive jobs is 10, 30, or 50 percent. For system utilization, we present the relative improvement of *MeshSched* and *CFCA* over *Native*.

Fig. 8 shows the scheduling performance when the runtime slowdown is set to 10 percent. First, we observe that the *MeshSched*, *CFCA*, and *Hybrid* schemes can have a striking effect on job wait times and response times for all three months. The largest wait time reduction is more than 50 percent for month 1 when there are 10 percent communication-sensitive jobs. With a relatively low slowdown of 10 percent, using mesh partitions provides shorter turnaround time with affordable performance loss. The response time is also reduced substantially because of the reduction in job wait time. The relative improvement in job response time is smaller than that achieved in job wait time. It indicates that for most jobs their runtimes dominate the total response time. Second, we notice that *Hybrid* outperforms both *MeshSched* and *CFCA* regarding wait time and response time for all three months. In month 1 and month 3, *Hybrid* can reduce more than 10 percent wait time than can *CFCA*. The key to its superiority in scheduling performance is that *Hybrid* takes full advantage of *MeshSched* to reduce resource contention and *CFCA* to guarantee the performance of communication-sensitive jobs as well. Third, with respect to LoC, all three scheduling schemes perform better than *Native* does. For month 1, LoC decreases more than 10 percent when there are 20 percent communication-sensitive jobs. This decrease is significant when we consider the machine scale. For Mira in a single month, approximately $25{,}38{,}944$ $(= 0.1 \times 30 \times 24 \times 49{,}152)$ node-hours are saved, enabling the system to run 72 hours at full load. In particular, *MeshSched* achieves lower LoC in most cases than *CFCA* and *Hybrid* do. The reason is that *MeshSched* contains only mesh partitions except for 512 nodes, whereas *CFCA* and *Hybrid* still use some torus partitions, inevitably causing more resource contention than does *MeshSched*. Clearly, they all improve the overall system utilization. *MeshSched* can improve the utilization by more than 10 percent in month 2 with 40 percent communication-sensitive jobs. Although *CFCA* does not improve the utilization as much as *MeshSched* and *Hybrid* do, the average improvement is about 5 percent, with the biggest improvement in month 3 when there are 10 percent communication-sensitive jobs. The reason is similar to the case of LoC; that is, *MeshSched* has much less network resource contention.

Fig. 9 presents the scheduling performance when runtime slowdown is set to 40 percent. With respect to job wait time, the *Hybrid* scheme always outperforms the other three scheduling policies. For example, in month 1 with 10 percent communication-sensitive jobs, *CFCA* and *Hybrid* reduce the wait time by more than 50 percent. Similar performance improvement is achieved for job response time by using *CFCA* and *Hybrid* for month 3. In this case, *MeshSched* generally results in a worse job performance than *Native* does when there are more than 10 percent communication-sensitive jobs. In months 2 and 3, the job wait time is increased by up to 100 percent. The reason is that although the resource contention is reduced by using *MeshSched*, user jobs suffer from the substantial runtime expansion caused by using mesh partitions. Thus, in a scenario in which communication-sensitive jobs are the majority, one would be wise not to still schedule this jobs to mesh partitions.

Similar to Fig. 8, all three new scheduling schemes improve LoC. Especially in month 1, *CFCA* and *Hybrid* can greatly reduce LoC, much more than that achieved by using *MeshSched*. With respect to system utilization, *MeshSched* achieves more than 15 percent increase in some cases. Similar to Fig. 8, *MeshSched* improves utilization more than *CFCA* and *Hybrid* do.

Next we more closely examine performance metrics in a fine-grained way. Figs. 10 and 11 present the average wait time calculated based on job sizes. The "percentage" under each figures is the ratio of communication-sensitive jobs to the whole workload. First, we see that the wait time increases as the job size becomes larger. This result coincides with the widely acknowledged fact that in HPC systems larger jobs stay longer in the waiting queue than do smaller jobs. Second, we see that jobs at sizes equal to or larger than 1 K all benefit from the new scheduling schemes. However, jobs at size 512 nodes have their wait time increase. The reason is that the introduction of mesh and contention-free partitions improves the availability of partitions consisting of more than 1 midplane. Moreover, *MeshSched* is found to not always help reduce the average wait time. In Fig. 11, the

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on February 15,2021 at 23:37:19 UTC from IEEE Xplore. Restrictions apply.

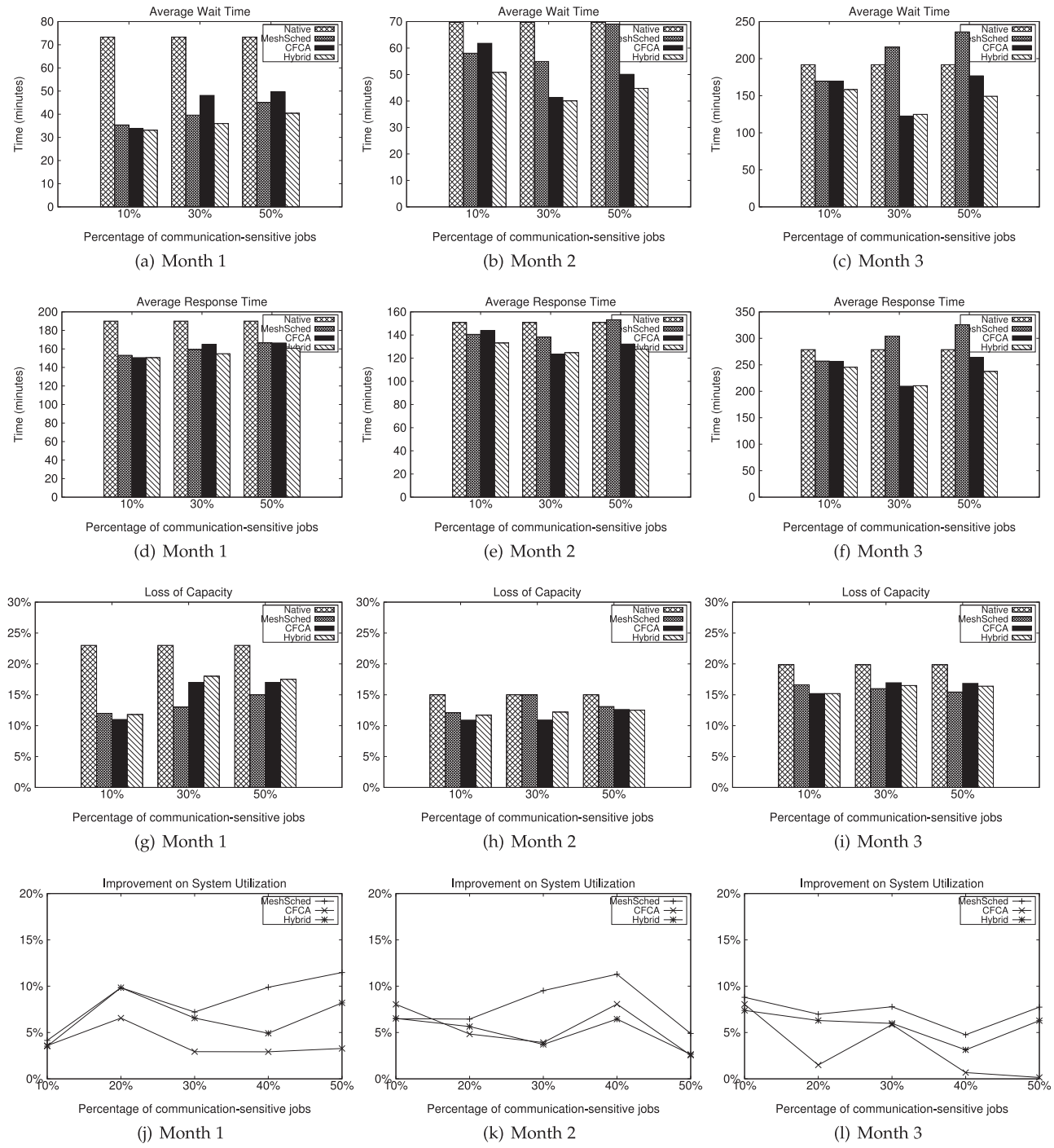


Fig. 8. Comparison of scheduling performance using different scheduling policies, where runtime slowdown is set to 10 percent for communication-sensitive jobs.

wait time greatly increases as the percentage of communication-sensitive jobs is 30 and 50 percent. This implies that if many users are submitting jobs with substantial slowdown on a mesh network, the use of full mesh topology will have more impact on large jobs (i.e., 8 K or larger). We also notice that jobs at sizes of 1, 4, and 8 K are positively affected the most. From the job distribution we know that 1K-node jobs are in the majority, which always compete with one another for the wiring resource along the D dimension. Turning the D dimension of the 1 K partition into a mesh-connected configuration enables more 1 K-node jobs to run simultaneously.

In summary, our main observations are as follows.

- Since not all applications are sensitive to communication bandwidth, we find that the existing scheduling design on Mira has much room for improvement. We believe the use of the new scheduling methods can improve the overall system performance.
- *CFCA* and *Hybrid* outperform the existing scheduling policy on Mira in almost all the simulation cases across all four scheduling metrics. Even under the extreme scenario (i.e., when there is a high percentage of communication-sensitive jobs), *CFCA* and

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on February 15,2021 at 23:37:19 UTC from IEEE Xplore. Restrictions apply.

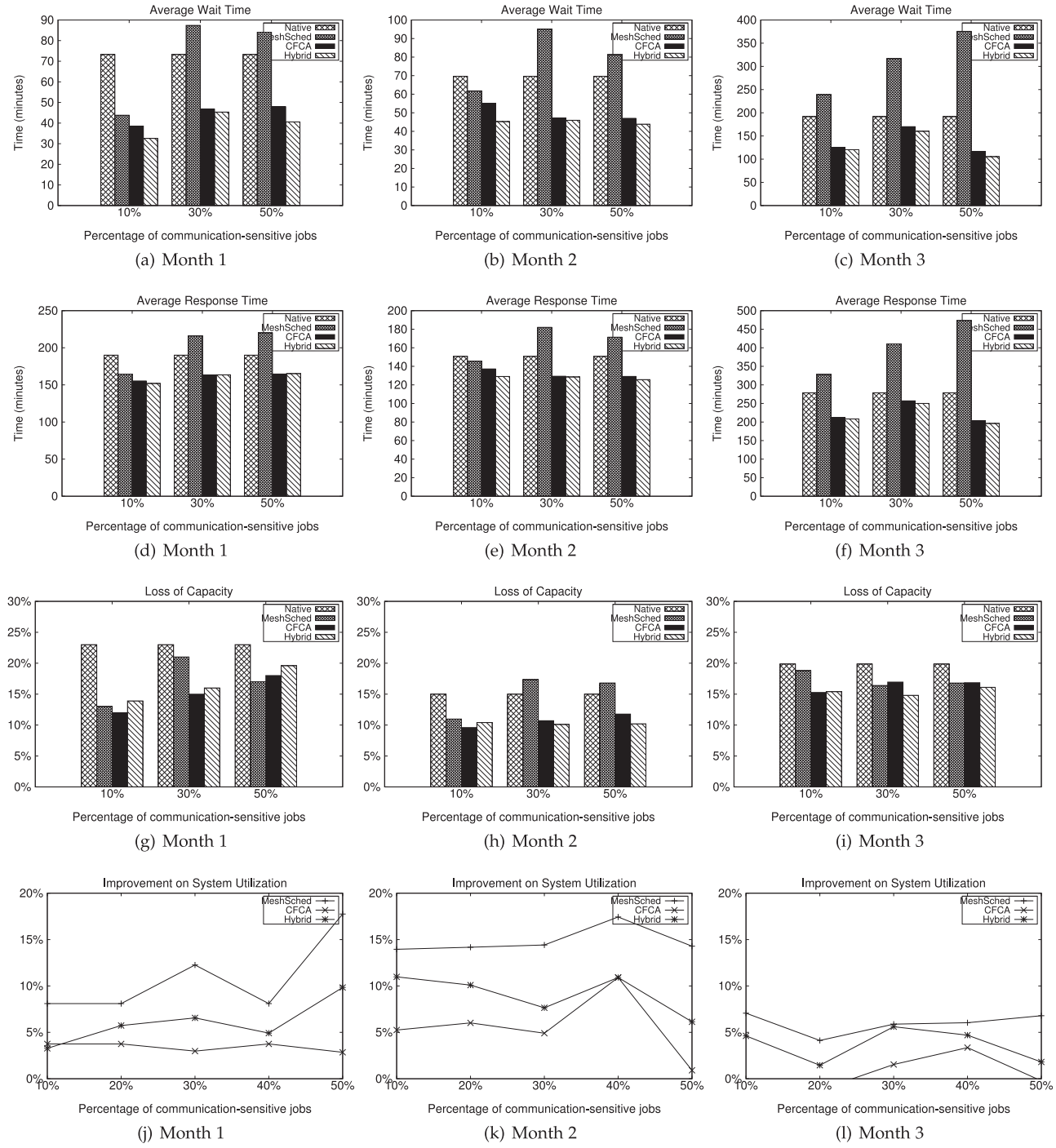
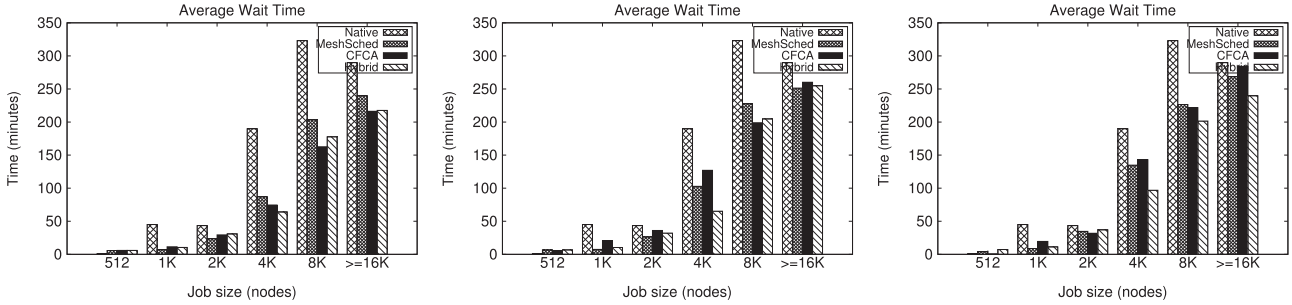


Fig. 9. Comparison of scheduling performance using different scheduling policies, where runtime slowdown is set to 40 percent for communication-sensitive jobs.

Hybrid can gain noticeable improvement on system performance and reduce fragmentation.

- *MeshSched* outperforms the current scheduler used on Mira when a small portion of jobs are communication-sensitive. When a large portion of jobs are communication-sensitive (e.g., 40 percent), *MeshSched* reduces system fragmentation and increases system utilization at the cost of increasing job wait time and response time.
- *Hybrid* has the best overall performance on reducing the wait time and response time. If possible, system administrators should be encouraged to build a network configuration containing all partition options and adopt this scheduling policy.
- When 1 and 4 K jobs are the majority of the workload as in month 1, with respect to system utilization, *CFCA* greatly reduces job wait times because of the availability of 1 and 4 K contention-free partitions. In months 2 and 3, where nearly half the jobs are 512 nodes, the performance improvement is not as good as that of month 1. In general, with a small portion of communication-sensitive jobs (e.g., no more than 10 percent), we encourage the use of *MeshSched*; otherwise, the use of contention-free partitions is a good choice.

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on February 15,2021 at 23:37:19 UTC from IEEE Xplore. Restrictions apply.



(a) Percentage=10% (b) Percentage=30% (c) Percentage=50%

Fig. 10. Average wait time with runtime slowdown=0.1 for communication-sensitive jobs.

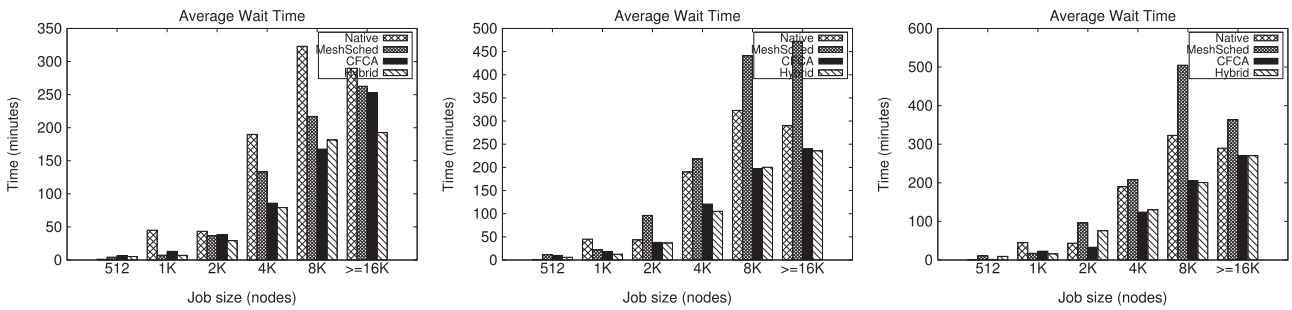
6 Discussion about System Partitioning

Our study indicates that providing multiple types of partitions on torus-based systems is critical for both improving resources utilization and guaranteeing good performance for communication-sensitive applications. Clearly the performance penalty of switching from a torus to a mesh or a contention-free partition varies with the application. A high-performance computing center typically has some mixture of both communication-sensitive and non-communication-sensitive jobs. A job's communication sensitivity can be notified to the scheduler through job submission script by the user who is fully aware of the job's behavior. In addition, the scheduler can also learn a job's communication sensitivity based on its historical record. This additional information will assist the scheduler to select the most appropriate partition for the job. Alternatively, users should be provided an option for them to select desirable partitions for their jobs, such as a torus partition for communication-sensitive applications like DNS3D and P3DFFT or a mesh/contention-free partition for non-communication-sensitive applications like Nek5000.

Hardware serviceability is another benefit on partition-based machines. Since a partition block is completely isolated from the rest of the machine, the hardware that is not involved in any running block could be replaced or repaired with no impact on running jobs. This allows for a rapid replacement of and recovery from hardware failures, removing the need for a full machine restart. Such a feature is critical for maintaining high availability of the machine. On BG/Q, compute and midplane-to-midplane interconnects are both located on the nodeboard. This is in contrast to BG/Q's predecessor, BG/P, where the midplane-to-midplane interconnects were hosted on its own card separated from the nodecard. If a block's wiring is being used just to pass through a midplane without using the compute hardware, this midplane will be isolated and can no longer be connected with other midplanes. For instance, on BG/Q, a two-midplane-wide partition is built with wrap-around links for a torus passing the third midplane in a dimension with length of three; in this case, while the node hardware on the third midplane is allocatable by the scheduler, it cannot be used without breaking the torus of the running block. Nevertheless, by using a mesh connectivity along this dimension in this case, service and recovery of the remaining compute hardware can be sped up. Consider an example of a 32 K block on Mira, which requires two-thirds of the compute nodes of the entire machine. Such a block requires passing through the remaining one-third of the machine for the wiring resources, rendering the entire machine unserviceable. By making one dimension of this 32 K block as mesh, it becomes a contention-free partition, and the remaining one-third of the machine can be serviced without impacting the job running on the 32 K block. We note that such a minor modification on system partitioning is especially important for capability computing that is featured as using a large amount of computing power to solve big scientific applications.

7 Related Work

A large amount of studies have been conducted on resource allocation and job scheduling on supercomputers. Evans et al. studied the variability of performance on clusters and claimed that tightly allocated jobs had better performance than did sparse ones [42]. Kramer and Ryan found that variability introduced by different job allocation strategies can be mitigated by periodically migrating application tasks to create larger contiguous chunks [43]. Bhateke and Kale evaluated the positive impact of locality-aware allocations



(a) Percentage=10% (b) Percentage=30% (c) Percentage=50%

Fig. 11. Average wait time with runtime slowdown=0.4 for communication-sensitive jobs.

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on February 15,2021 at 23:37:19 UTC from IEEE Xplore. Restrictions apply.

on applications performance [44]. Bhateke et al. [45] showed that with different node allocations, application performance changes dramatically. The cause of this variation could be interference from other jobs sharing the same network links. Several studies have focused on the effects of resource contention caused by concurrently running jobs. Jeff et al. [46] discovered that the high variability in performance on an SGI Origin 3,800 is due to the resource contention is caused by concurrent accesses to global memory. Skinner and Kramer [47] showed that 2-3 times improvement of MPI_Allreduce is observed by eliminating network contention from other jobs. Lo et al. and Leung et al. presented processor allocation strategies on multi-dimension network computers [48], [49].

While these studies focus mainly on performance variation caused by job interference, our work investigates application sensitivity to communication bandwidth caused by network configuration change. Moreover, our work examines a suite of parallel benchmarks and leadership applications at large scale. The results provide a foundation for the design of a communication-aware resource management system.

Arguably, a number of studies have been presented to improve resources management and scheduling on large-scale systems from various aspects. Feitelson et al. [50] provided a detailed analysis of different scheduling strategies. Zhao et al. [51] proposed network-aware caching mechanisms on large-scale systems such as IBM's Blue Gene supercomputers. Desai et al. [12] assessed application performance degradation on shared network and studied how to improve application performance while efficiently utilizing the available torus network Pedretti et al. [52] showed that one can use a large-scale parallel computer Cray XT5 to emulate the expected imbalance of future exascale systems; their results indicate that some applications experience sudden drops in performance at certain network injection bandwidth thresholds. Yang et al. [53] and Zhou et al. [54] proposed power-aware job scheduling frameworks for supercomputer systems as a 0-1 knapsack model.

To the best of our knowledge, we are among the first to systematically investigate communication awareness for resource management and job scheduling. Furthermore, we have conducted extensive trace-based simulations to quantify the benefit of communication-aware scheduling over the existing scheduling design on Mira.

8 Conclusions

In this paper, we have presented a detailed experimental study of a suite of parallel benchmarks and applications on the Mira system at Argonne. Our results show substantial variation in performance across production applications as well as microbenchmarks with different sensitivity to communication bandwidth. Based on application benchmarking, we have designed three scheduling schemes—*MeshSched*, *CFCA*, and *Hybrid*—for Mira by using partitions that require fewer link resources. Our experiments prove the performance benefit obtained by these new scheduling methods. While this study targets Mira, our design is generally applicable to all BG/Q systems as well as other 5D torus-connected machines.

Increasingly, scheduling large systems will become an exercise in multigoal optimization, considering many types of orthogonal resources. This paper demonstrates how traditional scheduling processes can be extended to efficiently manage a new resource type and the benefits to system usability and utilization that can be realized by such an approach.

Several avenues are open for future work. One is to build a model to predict whether a job is sensitive to communication bandwidth based on its historical data. We also plan to implement the proposed communication-aware policy into the production scheduler used on Mira. In addition, we are expanding this work with the aim of developing a smart resource management framework for better managing non-traditional resources including I/O and power consumption.

Acknowledgments

The work at Illinois Institute of Technology is supported in part by US National Science Foundation grants CNS-1320125 and CCF-1422009. The FLASH software used in this work was in part developed by the DOE NNSA-ASC OASCR Flash Center at the University of Chicago. This material is based in part upon work supported by the US Department of Energy, Office of Science, under contract DE-AC02-06CH11357.

References

[1] (2016). Top500 supercomputing web site [Online]. Available: http://www.top500.org.

[2] A. Gara, M. A. Blumrich, D. Chen, G. L.-T. Chiu, P. Coteus, M. E. Giampapa, R. A. Haring, P. Heidelberger, D. Hoenicke, G. V. Kopcsay, T. A. Liebsch, M. Ohmacht, B. D. Steinmacher-Burow, T. Takken, and P. Vranas, "Overview of the blue gene/L system architecture," *IBM J. Res. Dev.*, vol. 49, no. 2, pp. 195–212, Mar. 2005.

[3] IBM Journal of Research and Development staff, "Overview of the IBM Blue Gene/P project," *IBM J. Res. Dev.*, vol. 52, no. 1/2, pp. 199–220, Jan. 2008.

[4] T. Li, X. Zhou, K. Brandstatter, D. Zhao, K. Wang, A. Rajendran, Z. Zhang, and I. Raicu, "ZHT: A Light-weight reliable persistent dynamic scalable zero-hop distributed hash table," in *Proc. IEEE 27th Int. Symp. Parallel Distrib. Process.*, 2013, pp. 775–787.

[5] L. Yu, Z. Zheng, Z. Lan, T. Jones, J. M. Brandt, and A. C. Gentile, "Filtering log data: Finding the needles in the haystack," in *Proc. Dependable Syst. Netw. Workshops*, 2012, pp. 1–12.

[6] (2016, Feb. 4). Managing system software for Cray XE and Cray XT systems. Cray document [Online]. Available: http://docs.cray.com/books/S-2393-31/

[7] C. Dong, N. Eisley, P. Heidelberger, R. Senger, Y. Sugawara, S. Kumar, V. Salapura, D. Satterfield, B. Steinmacher-Burow, and J. Parker, "The IBM blue gene/Q interconnection fabric," *IEEE Micro*, vol. 32, no. 1, pp. 32–43, Jan./Feb. 2012.

[8] T. Budnik, B. Knudson, M. Megerian, S. Miller, M. Mundy, and W. Stockdell, "Blue gene/q resource management architecture," in *Proc. IEEE Workshop Many-Task Comput. Grids Supercomput.*, Nov. 2010, pp. 1–5.

[9] Y. Ajima, Y. Takagi, T. Inoue, S. Hiramoto, and T. Shimizu, "The Tofu interconnect," in *Proc. IEEE 19th Annu. Symp. High Perform. Interconnects*, 2011, pp. 87–94.

[10] W. Tang, Z. Lan, N. Desai, D. Buettner, and Y. Yu, "Reducing fragmentation on torus-connected supercomputers," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2011, pp. 828–839.

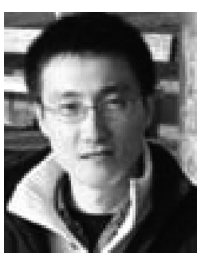
[11] Z. Zhou, X. Yang, Z. Lan, P. Rich, W. Tang, V. Morozov, and N. Desai, "Bandwidth-aware resource management for extreme scale systems," in *Int. Conf. High Perform. Comput., Netw., Storage Anal. (SC14), poster session*, 2014.

[12] N. Desai, D. Buntinas, D. Buettner, P. Balaji, and A. Chan, "Improving resource availability by relaxing network allocation constraints on Blue Gene/P," in *Proc. Int. Conf. Parallel Process.*, 2009, pp. 333–339.

[13] Z. Zhou, X. Yang, Z. Lan, P. Rich, W. Tang, V. Morozov, and N. Desai, "Improving batch scheduling on Blue Gene/Q by relaxing 5D torus network allocation constraints," in *Proc. IEEE Int. Parallel and Distrib. Process. Symp.*, May 2015, pp. 439–448.

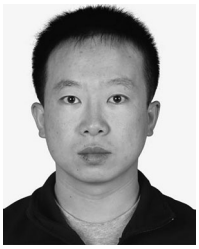
Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on February 15,2021 at 23:37:19 UTC from IEEE Xplore. Restrictions apply.

[14] (2016, Feb. 4). Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program [Online]. Available: https://www.alcf.anl.gov/incite-program
[15] (2016, Feb. 4). ASCR Leadership Computing Challenge (ALCC) [Online]. Available: http://science.energy.gov/ascr/facilities/alcc/
[16] S. Srinivasan, R. Kettimuthu, V. Subramani, and P. Sadayappan, "Characterization of backfilling strategies for parallel job scheduling," in *Proc. Int. Conf. Parallel Process. Workshops*, 2002, pp. 514–519.
[17] P. Keleher, D. Zotkin, and D. Perkovic, "Attacking the bottlenecks of backfilling schedulers," *Cluster Comput.*, vol. 3, no. 4, pp. 245–254, 2000.
[18] D. G. Feitelson. (2016, Feb. 4). Logs of real parallel workloads from production systems [Online]. Available: http://www.cs.huji.ac.il/labs/parallel/workload/logs.html
[19] S. Lupetti and D. Zagorodnov, "Data popularity and shortest-job-first scheduling of network transfers," in *Proc. Int. Conf. Digit. Telecommun.*, Aug. 2006, pp. 26–26.
[20] A. Afzal, A. S. McGough, and J. Darlington, "Capacity planning and scheduling in grid computing environments," *Future Gener. Comput. Syst.*, vol. 24, no. 5, pp. 404–414, May 2008.
[21] T. Li, D. Baumberger, and S. Hahn, "Efficient and scalable multiprocessor fair scheduling using distributed weighted round-robin," in *Proc. 14th ACM SIGPLAN Symp. Principles Practice Parallel Program.*, 2009, pp. 65–74.
[22] A. Ganapathi, Y. Chen, A. Fox, R. Katz, and D. Patterson, "Statistics-driven workload modeling for the cloud," in *Proc. IEEE 26th Int. Conf. Data Eng. Workshops*, Mar. 2010, pp. 87–92.
[23] A. Mu'alem and D. Feitelson, "Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 6, pp. 529–543, Jun. 2001.
[24] D. A. Lifka, "The ANL/IBM SP scheduling system," in *Proc. Workshop Job Scheduling Strategies Parallel Process.*, 1995, pp. 295–303.
[25] J. Skovira, W. Chan, H. Zhou, and D. A. Lifka, "The EASY - LoadLeveler API project," in *Proc. Workshop Job Scheduling Strategies Parallel Process.*, 1996, pp. 41–47.
[26] D. B. Jackson, Q. Snell, and M. J. Clement, "Core algorithms of the maui scheduler," in *Proc. Revised Papers 7th Int. Workshop Job Scheduling Strategies Parallel Process.*, 2001, pp. 87–102.
[27] (2016, Feb. 4). Cobalt resource manager [Online]. Available: https://trac.mcs.anl.gov/projects/cobalt
[28] W. Tang, Z. Lan, N. Desai, and D. Buettner, "Fault-aware, utility-based job scheduling on Blue Gene/P systems," in *Proc. IEEE Int. Conf. Cluster Comput. Workshops*, 2009, pp. 1–10.
[29] P. B. Hansen, "An analysis of response ratio scheduling," in *Proc. IFIP Congress*, 1971, pp. 479–484.
[30] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga, "The NAS parallel benchmarks," Tech. Rep., 1991.
[31] P. Fischer, J. Lottes, D. Pointer, and A. Siegel, "Petascale algorithms for reactor hydrodynamics," *J. Phys.: Conf. Series*, vol. 125, no. 1, 2008.
[32] (2016, Feb. 4). The FLASH code [Online]. Available: http://www.flash.uchicago.edu/site/flashcode/
[33] 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," *The Astrophys. J. Suppl. Series*, vol. 131, no. 1, p. 273, 2000.
[34] R. T. Fisher, L. Kadanoff, D. Q. Lamb, A. Dubey, T. Plewa, A. C. Calder, F. Cattaneo, P. Constantin, I. T. Foster, M. E. Papka, S. I. Abarzhi, S. M. Asida, P. M. Rich, C. C. Glendenin, K. Antypas, D. J. Sheeler, L. B. Reid, B. Gallagher, and S. G. Needham, "Terascale turbulence computation on BG/L using the FLASH3 code," *IBM J. Res. Develop.*, vol. 52, pp. 127–136, 12/2007 2007.
[35] M. A. Taylor, S. Kurien, and G. Eyink, "Recovering isotropic statistics in turbulence simulations: The Kolmogorov 4/5th-law," *Phys. Rev. E*, vol. 68, no. 2, p. 026310, 2003.
[36] S. Plimpton, "Fast parallel algorithms for short-range molecular dynamics," *J. Comput. Phys.*, vol. 117, no. 1, pp. 1–19, 1995.
[37] (2016, Feb. 4). Lammps [Online]. Available: https://www.alcf.anl.gov/user-guides/lammps
[38] D. Bailey, T. Harris, W. Saphir, R. Van Der Wijngaart, A. Woo, and M. Yarrow, "The NAS parallel benchmarks 2.0," Tech. Rep. NAS-95-010, 1995.
[39] W. Tang, N. Desai, D. Buettner, and Z. Lan, "Analyzing and adjusting user runtime estimates to improve job scheduling on the Blue Gene/P," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, Apr. 2010, pp. 1–11.
[40] J. Jones and B. Nitzberg, "Scheduling for parallel supercomputing: A historical perspective of achievable utilization," in *Proc. Job Scheduling Strategies Parallel Process.*, 1999, vol. 1659, pp. 1–16.
[41] Y. Xu, Z. Zhou, W. Tang, X. Zheng, J. Wang, and Z. Lan, "Balancing job performance with system performance via locality-aware scheduling on Torus-connected systems," in *Proc. IEEE Int. Conf. Cluster Comput.*, Sep. 2014, pp. 140–148.
[42] J. Evans, W. Groop, and C. Hood, "Exploring the relationship between parallel application run-time and network performance in clusters," in *Proc. 28th Annu. IEEE Int. Conf. Local Comput. Netw.*, Oct. 2003, pp. 538–547.
[43] W. T. C. Kramer and C. Ryan, "Performance variability of highly parallel architectures," in *Proc. Int. Conf. Comput. Sci.: PartIII*, 2003, pp. 560–569.
[44] A. Bhatele and L. Kale, "Application-specific topology-aware mapping for three dimensional topologies," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, Apr. 2008, pp. 1–8.
[45] A. Bhatele, K. Mohror, S. H. Langer, and K. E. Isaacs, "There goes the neighborhood: Performance degradation due to nearby jobs," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, 2013, pp. 41:1–41:12.
[46] H. Jeff, A. Robert, D. Daniel, F. Mark, H. Lee, O. Tom, W. William, B. MMarty, and B. Jeff, "Minimizing runtime performance variation with Cpusets on the SGI Origin 3,800," in *ERDC MSRC PET Preprint*, pp. 1–32, 2001.
[47] D. Skinner and W. Kramer, "Understanding the causes of performance variability in HPC workloads," in *Proc. IEEE Int. Workload Characterization Symp.*, Oct. 2005, pp. 137–149.
[48] V. Lo, K. Windisch, W. Liu, and B. Nitzberg, "Noncontiguous processor allocation algorithms for mesh-connected multicomputers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 8, no. 7, pp. 712–726, Jul. 1997.
[49] V. Leung, E. Arkin, M. Bender, D. Bunde, J. Johnston, A. Lal, J. Mitchell, C. Phillips, and S. Seiden, "Processor allocation on cplant: Achieving general processor locality using one-dimensional allocation strategies," in *Proc. IEEE Int. Conf. Cluster Comput.*, 2002, pp. 296–304.
[50] D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn, "Parallel job scheduling strategies for parallel processing," in *Proc. 10th Int. Conf. Job Scheduling Strategies Parallel Process.*, 2005, pp. 1–16.
[51] D. Zhao, K. Qiao, and I. Raicu, "HyCache+: Towards scalable high-performance caching middleware for parallel file systems," in *Proc. 14th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, 2014, pp. 267–276.
[52] K. Pedretti, R. Brightwell, D. Doerfler, K. Hemmert, and I. Laros, James H, "The impact of injection bandwidth performance on application scalability," in *Proc. 18th Eur. MPI Users' Group Conf. Recent Adv. Message Passing Interface*, 2011, vol. 6960, pp. 237–246.
[53] X. Yang, Z. Zhou, S. Wallace, Z. Lan, W. Tang, S. Coghlan, and M. E. Papka, "Integrating dynamic pricing of electricity into energy aware scheduling for HPC systems," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, 2013, pp. 60:1–60:11.
[54] Z. Zhou, Z. Lan, W. Tang, and N. Desai, "Reducing energy costs for IBM Blue Gene/P via power-aware job scheduling," in *Proc. 17th Int. Workshop Job Scheduling Strategies Parallel Process.*, 2014, pp. 96–115.



Zhou Zhou received the BS degree from Beijing Jiaotong University in 2009. He is currently working toward the PhD degree in computer science at the Illinois Institute of Technology since 2009. His main research interests are in the areas of intelligent resource management for exascale computing systems, system performance analysis and optimization, and job scheduling on large-scale systems. He is a student member of the IEEE computer society.

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on February 15,2021 at 23:37:19 UTC from IEEE Xplore. Restrictions apply.



Xu Yang received the MS degree in computer science from Beijing Normal University in 2012. He is currently working toward the PhD degree at the Computer Science Department of IIT. His current research interest is resource management and job scheduling on HPC and distributed systems. He is a student member of the IEEE.

Zhiling Lan received the PhD degree in computer engineering from Northwestern University in 2002. She has since joined the faculty of the Illinois Institute of Technology and is currently a professor at the Department of Computer Science. She is also a guest research faculty at the Argonne National Laboratory. Her research interests are in the areas of parallel and distributed systems, with particular emphasis on fault tolerance, power efficiency, resource management and job scheduling, performance analysis and optimization. She is a senior member of the IEEE computer society.

Paul Rich recieved the masters of science in computer science degree from the University of Chicago in 2006. He has since joined the staff of the Argonne National Laboratory and is currently a software developer for the Argonne Leadership Facility's Advanced Integration Group. He is currently the primary developer of the Cobalt scheduler and resource manager. His primary research interests are high performance and parallel algorithms, system scheduling, resource management, and system scalability.

Wei Tang received the PhD degree in the Department of Computer Science at the Illinois Institute of Technology and was a postdoc researcher at the Argonne National Laboratory. He is currently a software engineer at Google, Inc. His research interests include parallel and distributed computing, job scheduling and resource management for large-scale systems, etc.

Vitali Morozov received the BS degree in mathematics from Novosibirsk State University and the PhD degree in computer science from the Ershov's Institute of Information Systems, Novosibirsk, Russia. He is a principal application performance engineer at the Argonne Leadership Computing Facility. In Argonne National Laboratory since 2001, he has been working on computer simulation of plasma generation, plasma C material interactions, plasma thermal and optical properties, and applications to laser and discharge produced plasmas. In ALCF, he has been working on performance projections and studying the hardware trends in the HPC, as well as porting and tuning applications to large supercomputers.

Narayan Desai is a principal engineer at Ericsson. His primary areas of work are resource management, system management, and system design. Prior to Ericsson, he spent 14 years at Argonne National Lab, working on a variety of issues in HPC software and systems, as well as metagenomics.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on February 15,2021 at 23:37:19 UTC from IEEE Xplore. Restrictions apply.