Study of Intra- and Inter-Job Interference with Different Job allocations

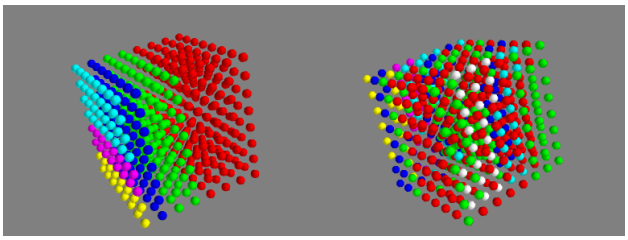
Xu Yang*, John Jenkins†, Misbah Mubarak†, Robert B. Ross†, Zhou Zhou*, Zhiling Lan*
*Department of Computer Science, Illinois Institute of Technology, Chicago, Illinois, USA 60616
{xyang56, zzhou}@hawk.iit.edu, lan@iit.edu
†Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, USA 60439
{jenkins,rross}@mcs.anl.gov, mmubarak@anl.gov

***Abstract*—Network contention between concurrently running jobs on HPC system is one of the primary causes of performance variability. Optimizing job allocation and avoiding network sharing are proven techniques to alleviate network contention and performance degradation. In this work, we propose a novel job communication pattern aware allocation strategy which serves as a critical module in our future HPC batch scheduler. The novelty of our new strategy is that it makes allocation decisions based on jobs communication pattern in addition to system topology and resource availability information. The scheduler uses job communication patterns information to decide whether its performance relies heavily on the network or not, and what the potential intra- and inter-job interference would be for a given allocation. Thus, the scheduler can pick a preferable resource allocation based on a specific job's communication pattern. We validate the effectiveness of our new design using traces collected from three representative parallel applications. The idea presented in this paper is widely applicable: HPC parallel applications have common communication patterns, and allocating resource with communication pattern awareness can improve performance on these systems with a variety of network topologies.**

I. Introduction

The scale of supercomputers keeps growing to accommodate the the demand for computing power required by scientific research areas. Supercomputers have hundreds of thousands of nodes, and serve as irreplaceable research vehicle for scientific problems with increasing size and complexity. Supercomputers are usually employed as a shared resource to accommodate many parallel applications (also known as jobs) running concurrently [31] and maintain high system utilization. These communication and I/O intensive jobs share the system infrastructure such as network and I/O bandwidth, and inevitably there is contention over these shared resources. As supercomputers continue to evolve, these shared resources are increasingly the bottleneck for performance.

Users make job submissions to the HPC system with specification about job's required number of nodes and expected runtime. Batch scheduler will pick several jobs and dispatch them to the system for running. These jobs run concurrently and utilize the system in a space sharing way. One of the most prominent problems is network contention among concurrently running jobs. The network sharing among concurrently running jobs causes communication variability that results in jobs running slower [4]. The delay of running jobs also increases the queueing time of the following submitted jobs, thus leading to low system throughput and utilization [15]. This adverse effect of network sharing can be mitigated by providing jobs with isolated allocations and exclusive network resource.



(a) Contiguous **(b) Non-contiguous**

Fig. 1 Multiple jobs running concurrently with different allocations. Each job is represented by a specific color. a) shows the effect of contiguous allocation, which reduce the inter-job interference. b) shows non-contiguous allocation, which introduces both intra and inter-job interferences.

On the widely used torus-connected HPC systems [6][1][19], two allocation strategies are commonly used. In the contiguous allocation, the scheduler assigns each user job a compact and contiguous set of computing nodes, as shown in Figure 1a. The partition-based allocation used in Blue Gene series systems is one of this kind [31]. This strategy favors application performance by providing application with isolated partitions and exclusive network connections, thus reducing network contention from concurrently running jobs. However, this strategy can cause internal fragmentation (when more nodes are allocated to a job than it requests) and external fragmentation (when sufficient nodes are available for a request, but they can not be allocated contiguously), therefore leading to lower system utilization that is otherwise possible. The other strategy is non-contiguous allocation, used by Cray XT/XE series [2], in which free nodes are assigned to jobs whether they are contiguous or not. Figure 1b shows the effect of non-contiguous allocation. Non-contiguous allocation eliminates internal and external fragmentation as seen in contiguous allocation systems, thereby leading to high system utilization. However, it introduces other problems, for example scattering application processes all over the system, causing both inter-job and intra-job contention. The non-contiguous

node allocation can significantly reduce job performance, especially for communication-intensive ones.

We envision that future HPC will adopt a flexible job allocation mechanism which combines the best of both contiguous and non-contiguous allocation. Such flexible mechanism would make allocation for jobs based on their communication patterns. With detail analysis of application's communication patterns, it can be identified which processes are tightly connected and conduct intensive communication within each job. Rather than assign each job a whole chunk of compact contiguous allocation, only the "neighborhood communication" consists of such tightly coupled processes need to get a compact allocation. The new allocation mechanism only needs to guarantee the compactness of the subset of the nodes where those "neighborhood communication" would reside.

In this work, we focus on an in-depth analysis of intra- and inter-job communication interferences with different job allocations on torus-connected HPC systems. Torus networks have been extensively used in the current generation of supercomputers because of their good scalability and competitive communication performance. On the recent Top500 list, 6 of the top 10 supercomputers use a high-radix torus-interconnected network [26]. The current generation of IBM Blue Gene/Q (BG/Q) supercomputer, such as Mira at Argonne Nation Laboratory and Sequoia at Lawrence Livermore National Laboratory, has its nodes connected in a 5D torus network [6]. The K computer from Japan uses the "Tofu system, which is a 6D mesh/torus topology [1]. Titan, a Cray XK7 supercomputer located at the Oak Ridge Leadership Computing Facility (OLCF), has nodes connected in a 3D torus within the compute partition [19]. Although the analysis are based on torus network, the idea conveyed from this work is applicable to networks with different topologies.

We selected three signature applications from DOE Design Forward Project[10] as examples to conduct detailed study about their communication patterns. We use a sophisticated simulation toolkit named CODES, Co-Design of Multi-layer Exascale Storage Architecture[7], from Argonne National Laboratory as a research vehicle to evaluate the performance of these applications with various allocations. We analyze the intra- and inter-job interference by simulating these applications running exclusively and concurrently with different allocations. The major finds and insights of this work are the following:

- We study the communication behaviors and identify the communication patterns of three representative HPC applications.
- We conduct detailed analysis about the intra- and inter-job interference by running three applications with different allocations. And we find the relation between application's communication pattern and the cause of interference.
- We show a novel and flexible way to make allocation with awareness of job's communication pattern, which is every effective to alleviate the interference between concurrently running jobs.

We believe the finds and insights presented in this work would be very useful for the design of future HPC batch job scheduler and resource management module.

The rest of this paper is organized as follows. Section II gives a detailed study of the three representative applications from DOE Design Forward Project. Section III talks about CODEs as research vehicle for our work. Section IV shows the performance analysis of three applications on different torus networks. Section V provides detailed analysis about the intra- and inter-job interference between three applications on torus network with different allocations. Section VI talks about the idea of a novel and flexible allocation strategy with job communication pattern awareness. Section VII talks about the related work in the area and conclusion will be presented in Section VIII.

II. Application Study

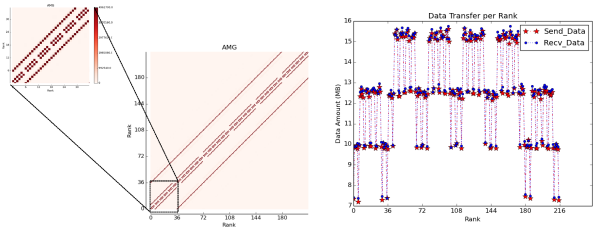
A parallel application usually conforms to a combination of several basic communication patterns [22]. At its different execution phases, the application's communication behavior may follow different basic patterns respectively. There are many profiling tools [23][17][30][25][28] available to capture information regarding communication patterns of parallel applications. They can help analyze parallel applications' communication behavior, providing information like the percentage of different MPI operations of the applications, communication topology, the amount data transferred between processes.

There are many applications running on HPC systems like Mira [6] and Titan [19]. Most of these applications are parallel and communication intensive, and conform to different communication patterns. In this work, we select three representative applications from DOE Design Forward project [10]. Each application conforms to a distinctive communication pattern that is commonly seen from applications in HPC workload. We believe these three applications can represent a big group of applications that conform to similar communication patterns. The applications studied in this work are Algebraic MultiGrid Solver(AMG), Geometric MultiGrid(MultiGrid) and Crystal-Router. The communication pattern figures we present in Section II-A, II-B, II-C are generated with the IPM [9] data from [10].

A. AMG

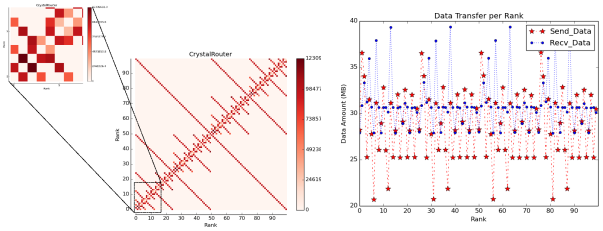
AMG is the MiniApp of Algebraic MultiGrid Solver, which is a parallel algebraic multi-grid solver for linear systems arising from problems on unstructured mesh physics packages. It has been derived directly from the BoomerAMG solver that is being developed in the Center for Applied Scientific Computing (CASC) at LLNL [13]. The dominant communication pattern is this regional communication with decreasing message size for different parts of the multi-grid v-cycle.

Figure 2a shows the communication topology of a small scale AMG with 216 MPI Ranks. Here we show the communication topology with small scale version because the dominant communication pattern of the application doesn't change with



(a) Communication Topology **(b) Data Volume**

Fig. 2 AMG. (a) rank-to-rank communication topology graph. (b) the data sent/received by each rank in AMG.



(a) Communication Topology **(b) Data Volume**

Fig. 3 CrystalRouter. (a) rank-to-rank communication topology graph. (b) the data sent/received by each rank in CrystalRouter.

scale. We can make the observation that AMG's dominant communication pattern is 3D Nearest Neighbor, each rank has intensive communication with with its 6 neighbors unless it is on the boundary. The dominant communication pattern is quite obvious when we identify this relation between ranks.

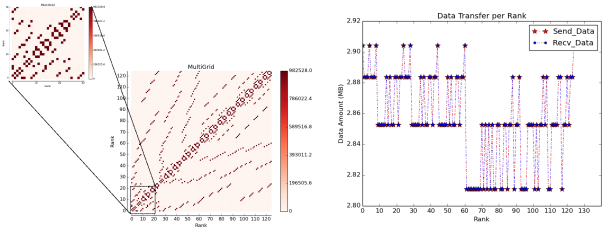
Figure 2b shows the amount of data each rank transferred in AMG. The red line shows the amount of data sent by each rank, the blue line shows the amount of received data. The data transferring of each rank is between 7MB to 16MB.

AMG represents a group of applications whose dominant communication pattern is Nearest Neighbor. Applications with such similar dominant communication pattern are like PARTISN, SNAP [10].

B. Crystal Router

The second MiniApp we have is CrystalRouter, the extracted communication kernel of the full application Nek5000 [12], which is a spectral element CFD application developed at Argonne National Laboratory. It features spectral element multi-grid solvers coupled with a highly scalable, parallel coarse-grid solver that widely used for projects including ocean current modeling, thermal hydraulics of reactor cores, and spatiotemporal chaos. CrystalRouter demonstrates the "Many-to-Many" communication pattern through scalable multi-stage communication process. The way CrystalRouter works is nodes compute for a while, then synchronize and communicate, continually alternating between these two types of activities.

The collective communication in CrystalRouter utilizing a recursive doubling approach. Ranks in CrystalRouter conform



(a) Communication Topology **(b) Data Volume**

Fig. 4 MultiGrid. (a) rank-to-rank communication topology graph. (b) the data sent/received by each rank in MultiGrid.

to a n-dimensional hypercube and recursively splitting into (n-1)-dimensional hypercubes, with communication happening with respect to the links that are severed. The pattern of this recursive communication can be found in Figure 3a. By nature of the logarithmic distribution, more data transferring are in the local neighborhood, which consists of 5 ranks.

Figure 3b shows the amount of data each rank transferred in CrystalRouter. The spiky shape indicates there is intensive data transferring in the local neighborhood. There is great variance of data transferring among the ranks in CrystalRouter, which is between 20MB to 36MB. This is due to the specific feature of CrystalRouter's communication pattern.

CrystalRouter represents a group of applications whose dominant communication is a hybrid of multi-stage local and hierarchical global communication.

C. MultiGrid

MultiGrid is geometric multi-grid v-cycle from production elliptic solver BoxLib, a software framework for massively parallel block-structured adaptive mesh refinement (AMR) codes. MultiGrid conforms to Many-to-Many communication pattern with decreasing message size and collectives for different parts of the multi-grid v-cycle. It is widely used for structured grid physics packages.

Figure 4a shows the communication topology of MultiGrid with 125 ranks. We can see intensive communication along the diagnal that conforms to 3D Nearest Neighbor communication pattern. However, there are also considerable amount of data transferring on both sides of the diagnal, which means the dominant communication pattern of MultiGrid is "Many-to-Many".

Figure4b shows the the amount of data transferred between ranks in MultiGrid. The data transferring of each rank is relatively small and stable, which is between 2.8MB to 2.9MB.

MultiGrid represents a group of applications whose dominant communication pattern is Many-to-Many (some literatures also refer it as All-to-All [22]). Applications with such similar dominant communication pattern are like FillBoundary and MiniDFT [10].

III. Research Vehicle

It is difficult to accurately and flexibly experiment with concurrently running jobs in an HPC context. One reason is

that the allocation strategy used on production machine is part of the system software, which can not be changed by user. Even the system administrator is not authorized to make that change. Another reason is that it is unrealistic to reserve the system exclusively to run the same job with desired allocation without interference then compare the results with those in the presence of interference. Therefore, we resort to simulation for this work.

A simulation toolkit named CODES enables the exploration of simulating different HPC networks with high fidelity [7][18]. CODES is built on top of Rensselaer Optimistic Simulation System (ROSS) parallel discrete-event simulator, which is capable of processing billions of events per second on leadership-class supercomputers [3]. CODES support both torus and dragonfly network with high fidelity flit-level simulation. CODES has this network workload component that capable of conducting trace-driven simulation. It can take real MPI application trace generated by SST DUMPI [25] to drive CODES network models. In this work, we focus on an in-depth analysis of intra- and inter-job communication interferences with different job allocations on torus-connected HPC systems. Since torus networks have been extensively used in the current generation of supercomputers because of their linear scaling on per-node cost and competitive communication performance [18][31].

IV. Configuration Study

In this section, we study the communication behavior of three applications on torus network with different dimensionality and bandwidth configurations.

The topology of torus network is k-ary n-cube, with k^n nodes in total arranged in an n-dimensional grid having k nodes in each dimension. Each node has 2×n direct linked neighbor nodes. The torus network performance is determined by its dimensionality and link bandwidth. As the increase of the dimensionality of torus network, so does the number of links connected with each node. The increased aggregated bandwidth of each node will definitely reduce the data transfer time of each rank in the application. Figure 5 shows the performance of the AMG, CrystalRouter and MultiGrid on a 2K node torus network model with a 3D torus (16×16×8), a 5D torus (8×4×4×4×4), and a 7D torus (4×4×4×4×2×2×2). The bandwidth of direct link between nodes is 2GiB/s in each direction [6], thus, the aggregated bandwidth is 12GiB/s per node in 3D torus, 20GiB/s per node in 5D torus, and 28GiB/s per node in 7D torus.

We can see from Figure 5 that the data transfer time of those three applications are greatly reduced as the dimensionality increases. The increased aggregated bandwidth of each node can accelerate the data transfer. We allocate each rank within each application linearly on the 3D torus. However, due to the variance of data transfer volume of each rank, AMG's data transfer time has great dispersion, shown as the outliers above the boxes in Figure 5a. There is few dispersion of MultiGrid, shown in Figure 5c, since data transfer of each rank in MultiGrid varies slightly between 2.8MB to 2.9MB. CrystalRouter also has few dispersion as shown in Figure 5b, this is due to the majority of data transferring is in the local neighborhood, which only take few hops.

Then we use fixed dimensionality to study the impact of increased bandwidth. We run these three applications on 5D torus with direct link bandwidth increases from 2GiB/s, to 4GiB/s and 8GiB/s. As shown in Figure 6, the data transfer time of each application can be greatly reduced as the bandwidth increases.

We also study the impact of dimensionality with fixed aggregated bandwidth. We use 3D, 5D and 7D torus network with the same per-node aggregated bandwidth, which is 28Gb/s. With the same aggregated bandwidth, higher dimensionality means shorter diameter and pair-wise distance between nodes. The bandwidth between nodes in each torus network are shown in Table I

TABLE I Torus networks with same aggregated bandwidth

	Bandwidth	
Dimension	direct	aggregate
3D	4.67Gb/s	28Gb/s
5D	2.8Gb/s	28Gb/s
7D	2Gb/s	28Gb/s

As we see from Figure 7a, the quartiles of "3D" box is a little bit higher (1%) than that of "5D", which means increasing dimensionality from 3D to 5D has little improvement on the average data transfer time for AMG. However, there is obvious reduction in those outliers between 3D and 5D. When dimensionality increases up to 7D, the improvement in terms of average data transfer time is still small, but more outliers above the top whisker can be spotted.

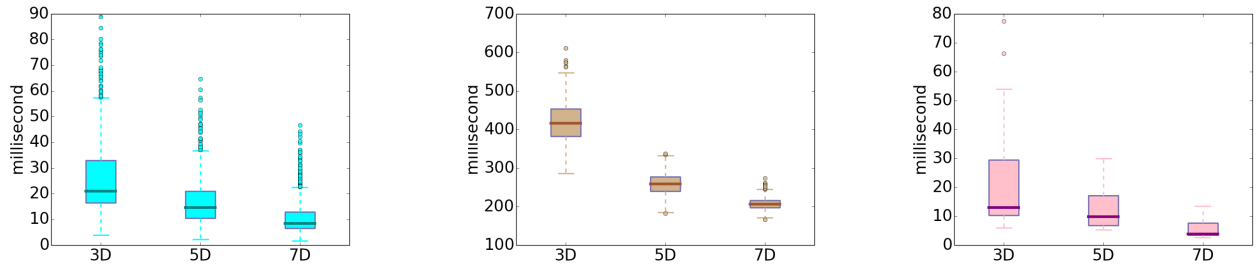
The performance of CrystalRouter actually deteriorate as the torus dimensionality increasing, shown in Figure 7b. This is due to the direct link bandwidth reduces from 4.67Gb/s to 2Gb/s, as dimension increases from 3D to 7D. CrystalRouter's large amount of intensive local data transfer benefit more from higher direct link bandwidth than its global transfer from lower diameter and shorter pair-wise distance between nodes.

The performance of MultiGrid is gradually improved as the dimensionality of torus network increases, shown in Figure 7c. The reduced diameter and shorten pair-wise distance between nodes would make the "Many-to-Many" global data transfer in MultiGrid more efficient.

We can get the observation that higher dimensionality of torus network would improve the performance of application with "Many-to-Many" communication patterns, while application with intensive local communication like "Nearest Neighbor" won't benefit much from higher dimensionality.

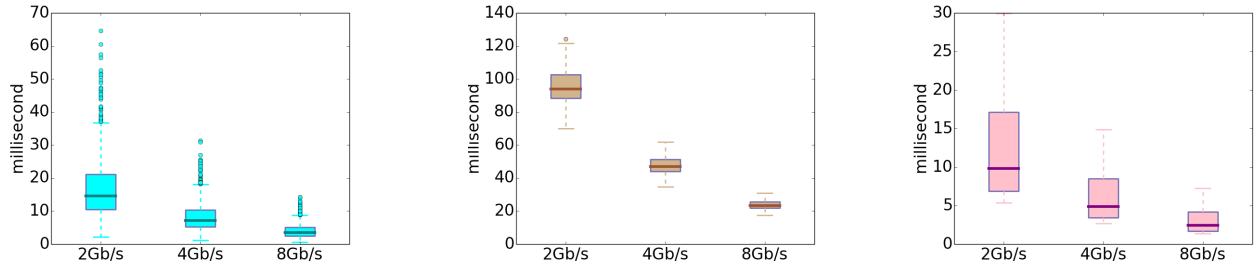
V. Interference Analysis

Communication variability due to network sharing can cause application performance degradation. ***Intra-job interference*** refers to the network contention between ranks within each



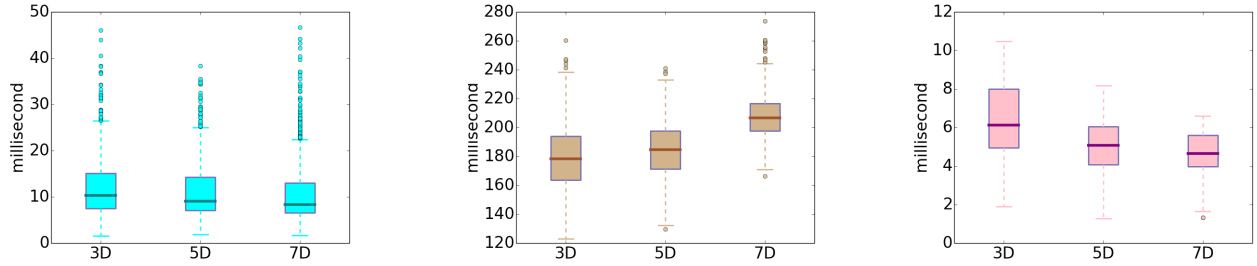
(a) AMG **(b) CrystalRouter** **(c) MultiGrid**

Fig. 5 Data Transfer Time of AMG, CrystalRouter and MultiGrid on 3D, 5D and 7D torus network. The network bandwidth increases as the dimensionality grows. Due to the data volume sent/received by each rank greatly varies, there is great dispersion of AMG's data transfer time, shown as outliers above the boxes in Figure 5a.



(a) AMG **(b) CrystalRouter** **(c) MultiGrid**

Fig. 6 Data transfer time of AMG, CrystalRouter and MultiGrid on 5D torus network with 2Gb/s, 4Gb/s and 8Gb/s direct link bandwidth.



(a) AMG **(b) CrystalRouter** **(c) MultiGrid**

Fig. 7 Data transfer time of AMG, CrystalRouter and MultiGrid on 3D, 5D and 7D torus network with same aggregated bandwidth.

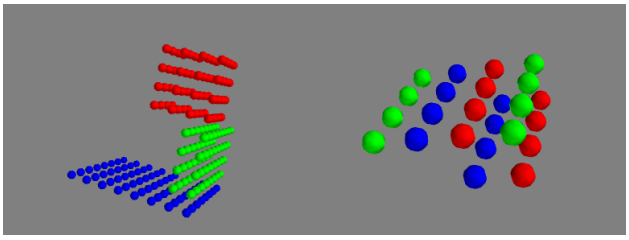
application. ***Inter-job interference*** is introduced by concurrently running jobs that adjacent to each other sharing network resources.

In this section we will study both kinds of interference in the context of torus network. Since application's communication pattern won't change with its scale, we choose AMG with 216 ranks, CrystalRouter with 100 ranks, and MultiGrid 125 ranks. And in order to accommodate three applications running concurrently with different allocations, we simulate a 2K-node ($16 \times 16 \times 8$) 3D torus network for the experiments.

A. Intra-Job Interference Analysis

There are lots of research works [29][27] try to come up with better task mapping algorithms to alleviate the intra-job interference, which is out the scope of our work. In this work, we focus on the analysis of such interference with different allocations. To study the intra-job interference of each job, we simulate each job running on 3D torus network with different allocation shapes. We assign each application with three different shapes allocation, i.e. 3D balanced-cube, 3D unbalanced-cube, 2D mesh.

3D balanced-cube, shown as red in Figure 8a, can guarantee the minimum *average pair-wise distance* within the allocation.



(a) (b)

Fig. 8 (a) Illustrates the contiguous allocation for 64-node job in three different shapes. "Blue" is 2D mesh, "Green" is 3D-unbalanced cube and "Red" is 3D-balanced cube. (b) Illustrates the non-contiguous allocation. Each job is represented by a specific color. The nodes assigned to different jobs are interleaved, the allocation unit is 4-node.

There are some research work like [16] [4] try to prove that compact allocation can guarantee job with better performance. They use many metrics to evaluate the compactness of the allocation, such as *average pair-wise distance*, *diameter* and *contiguity*. In this work, we select 3D balanced-cube as the most compact allocation on a 3D torus network.

3D unbalanced-cube, shown as green in Figure 8a, is a rectangular prism, which is the possible allocation shape on system with asymmetric networks. For example, Blue Waters Cray XE6/XK7 system network is 3D torus with Gemini routers. The network connections in the *y*direction have only half the bandwidth of the cables used in the x and z directions. In order to take advantage of the faster links in the x and z directions, job allocation is always start from X-Z plane, which leads to a rectangular prism shaped allocation [11].

2D mesh, shown as blue in Figure 8a, can be cut out from a single layer of the 3D torus. 2D mesh is a very common allocation shape in torus network for both contiguous and non-contiguous allocation strategies. For example, Cray Application Level Placement Scheduler indexes the nodes in torus network into a list and make allocation by simply works off the this list [2]. When the list is obtained by sorting the nodes based on their spacial coordinates in the torus, allocation made off from this list will be 2D mesh. The IBM Blue/Gene Q supercomputer Mira at Argonne Leadership Computing Facility also allow its allocation partition configured into mesh [31].

Figure 9a shows that for AMG, who conforms to 3D "Nearest Neighbor" communication pattern, spent less time when being assigned with 3D-unbalanced allocation than 3D-balanced. And MultiGrid gets the best performance (shortest data transfer time) when running on 3D-balanced allocation, as shown in Figure 9c. Since MultiGrid's communication pattern is "Many-to-Many" dominant, 3D-balanced is the most compact allocation with shortest pair-wise distance between nodes, which can reduce the aggregated hops for transferring message between ranks in MultiGrid. CrystalRouter has with both dominant local communication and multi-stage global data transfer, the 3D-balance is also the best allocation, but its advantage over 3D-unbalanced is not that obvious as to MultiGrid, as shown in Figure 9b.

There are lots of research work try to design fancy allocation algorithms to provide application with most compact allocation. However, providing cubical allocation without considering application's communication pattern won't guarantee the best performance for every application. Compact allocation should be provided to application with global data transfer, such as those conforms to "Many-to-Many" communication pattern. Application with dominant communication pattern like "Nearest Neighbor" won't fully utilize the compact allocation and could get even better performance on a relaxed allocation shape.

In non-contiguous allocation systems, the nodes assigned to each job could be scatter anywhere in the system. Some HPC systems predefine the minimum number of nodes consist of the allocation unit. The nodes within each unit are contiguous, and units assigned to each job may have arbitrary locations. We assign each application with a non-contiguous set of allocation units, refers to a small number(like 4) of adjacent nodes as shown in Figure 8b. We compare application's performance when it runs with contiguous allocation and non-contiguous allocation with unit size of 16, as shown in Figure 10.

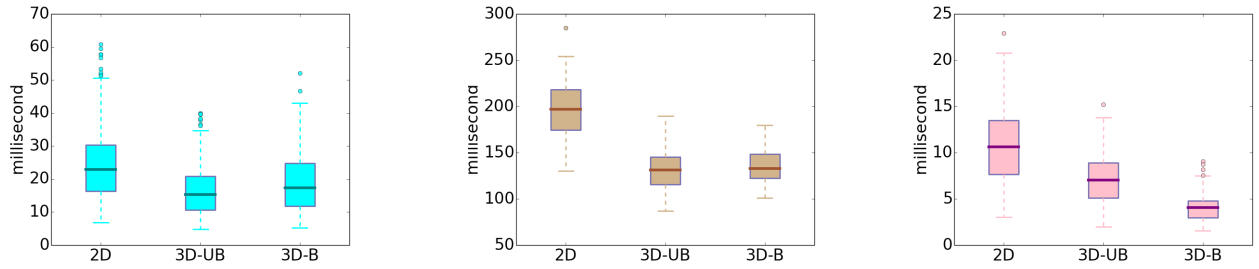
Figure 10a shows that AMG is insensitive to the allocation's contiguity. Non-contiguous allocation can serve AMG equally well as contiguous allocation, as long as the allocation unit in non-contiguous allocation is big enough to preserve the locality of AMG. CrystalRouter shows preference in choosing between contiguous and non-contiguous allocation. As shown in Figure 10b, there is obvious degradation in terms data transfer time when CrystalRouter running with non-contiguous allocation. The global data transfer in CrystalRouter will take more hops on non-contiguous allocation, thus result in longer communication time. When it comes to MultiGrid shown in Figure 10c, this degradation is even worse. Since the global data transfer take a great portion of MultiGrid "Many-to-Many" communication pattern.

The performance of application conforms to Nearest Neighbor communication pattern keep relatively stable with different allocations. Non-contiguous allocation won't aggravate intra-job interference as long as the unit size can preserve the locality in application's communication topology graph.

B. Inter-Job Interference Analysis

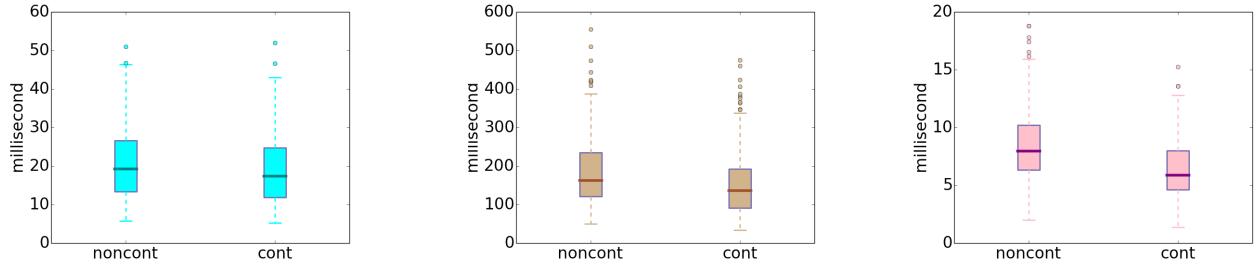
Inter-job interference has been identified as one of the major culprits for application's performance variability [4][24][21]. Inter-job interference is a more prominent issue for system adopted non-contiguous allocation policy than system with contiguous allocation. Application communication times have been demonstrated to vary from 36% faster to 69% slower due to job interference when running on non-contiguous allocation [4].

We assign each application with non-contiguous allocation and run them concurrently on the same network. The allocation unit belongs to different jobs are interleaved as shown in Figure 8b. The unit size is critical to the application's



(a) AMG **(b) CrystalRouter** **(c) MultiGrid**

Fig. 9 Intra-job Study: Data Transfer Time of 3 Apps on 3 different shapes allocation



(a) AMG **(b) CrystalRouter** **(c) MultiGrid**

Fig. 10 Intra-job study: data transfer time of 3 apps with Non/Contiguous allocation

performance. Figure 11 shows the results of each application data transfer time with different allocation unit size.

The data transfer time of AMG in Figure 11a keeps stable between allocation unit size of 16 (big) and 8 (medium), since the locality of AMG is 6-rank based. When the unit size reduce to 2 (small), AMG suffers prolong data transfer time by about 10%. CrystalRouter is more sensitive to allocation unit size. The best unit size would be the one big enough to accommodate its local neighborhood. Figure 11b shows unit size of 16 and 8 can guarantee the same average data transfer time, while some rank spend more time with allocation unit size 8 than 16. When the unit size reduce to 2-node, the communication become less efficiency and takes about 15% more times for transferring data.

The data transfer time of MultiGrid with different allocation unit sizes doesn't show obvious variability in Figure 11c. This is because even big allocation unit size like 16-node will still fail to preserve MultiGrid's "Many-to-Many" pattern. The data transfer time is almost doubled when MultiGrid running concurrently with allocation unit size of 16, as shown in Figure 11c . As the unit size decreases, the data transfer time keep growing, but in a slow way, which means in terms of preserving the locality of MultiGrid, allocation unit of size 16, 8 and 2 serve equally bad.

The unit size in non-contiguous allocation policy should be chosen according to application's communication pattern. Inter-job interference is inevitable in non-contiguous based systems, but unit size that big enough to preserve the neighborhood communication of the application will alleviate such interference and improve job performance.

C. Summary

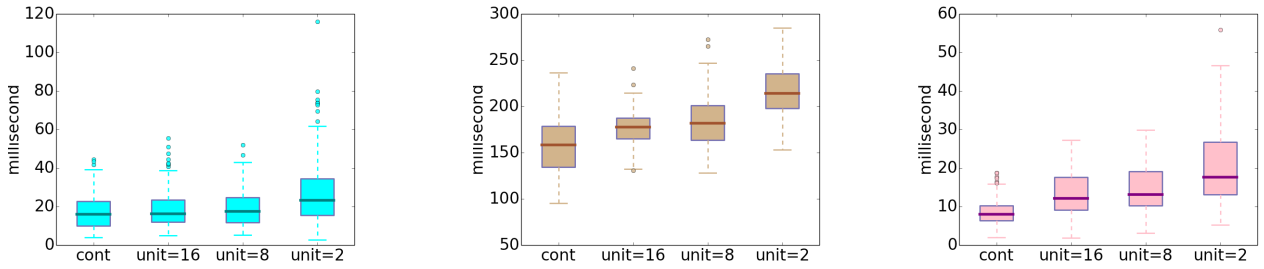
Based on our comprehensive study, we can get following observations:

- The compact allocation may not guarantee the best performance for every application.
- The performance of application conforms to Nearest Neighbor communication pattern keeps relatively stable with different shape allocations, while application conforms to "Many-to-Many" communication pattern prefers compact allocation.
- The allocation unit size should be decided according to application's communication pattern. Unit size that big enough to preserve the neighborhood communication of the application will result in better performance.

VI. Discussion

The observations we obtained by analyzing the intra- and inter-job interference can provide insights for the design of a smart and flexible job allocation strategy. By scrutinizing job's communication behavior, we can identify job's dominant communication pattern and pinpoint the "neighborhood communication". With such knowledge about jobs' communication patterns, we can analyze the possible interference between jobs and take precautions to alleviate the negative effect when making allocation decisions.

When an application with dominant communication that is intensive "Many-to-Many" being submitted to the system, it



(a) AMG **(b) CrystalRouter** **(c) MultiGrid**

Fig. 11 Inter-job interference study. "cont" indicates three applications running side by side concurrently on the same network with contiguous allocation. To study the impact of non-contiguous allocation on inter-job interference, applications running concurrently with interleaved allocation of different unit sizes, which are 16-node, 8-node, 2-node.

should be granted with compact node allocation and exclusive network provision. The compact allocation can guarantee the shortest pair-wise distance between all the ranks, and make the data transfer between ranks take less hops. On the other hand, the exclusive network provision will prevent other adjacent applications from sharing network resources, thus eliminate the performance degradation due to interference.

Not every application's preferable resource are compact node allocation and exclusive network provision. As we found in our comprehensive analysis, applications whose dominant communication patterns contain intensive "neighborhood communication" like "Nearest Neighbor" won't benefit from compact node allocation and exclusive network provision. Such applications can run with non-contiguous node allocation without performance degradation, as long as the allocation unit can accommodate the "neighborhood communication".

The allocation unit size shouldn't be fixed. Instead, the new allocation strategy should choose the proper unit size based on the scale of "neighborhood communication" of each job. The best unit size should be neither too big nor too small, jut perfectly fit for the size of that "neighborhood". Big unit size won't be fully utilized and results in fragmentation. Small unit size won't be able to accommodate the "neighborhood communication" and makes the intra-job communication less efficient.

The advantage of making allocation with consideration about job's communication pattern is obvious. First, compared with contiguous allocation policy, the new allocation strategy is more flexible, it doesn't require the system to provide a big contiguous partition that can accommodate the whole application, just small set of compact nodes enough for all the "local community" in the application. On the other hand, the new allocation strategy won't destroy the locality of application's communication pattern. On the contrary, the small compact node set (allocation unit) provided for the application's "local communities" will preserve the communication locality in the maximum extent.

VII. Related Work

There are some tools available for system monitoring and application profiling on HPC systems. Tools like TAU (Tuning and Analysis) [23], SCALASCA [30] and mpiP [17] can capture application's runtime information about application's communication and computation in event traces. However, recognizing the patterns about application's communication from those traces requires lots of extra effort. Some researchers work on the recognition and characterization of parallel application communication patterns. Oak Ridge National Laboratory has this ongoing project about developing a tool set named Oxbow, which can characterize the computation and communication behavior of 12 scientific applications and benchmarks [28]. In their recent work [22], they demonstrate a new approach to automatically characterizing a parallel application's communication behavior.

Some research institutes put great effort in characterization of scientific applications. The Department of Energy initiate this design forward project aims at the identification of the computational characteristics of the DOE MiniApps developed at various exascale co-design centers [10]. In this project, the communication patterns of several DOE full applications and associated mini-applications are studied to provide a more complete snapshot of the DOE workload. A joint project named CORAL from Oak Ridge, Argonne and Livermore provide a series of benchmarks to represent DOE workloads and technical requirements [8]. The CORAL project includes scalable science benchmarks, throughput benchmarks, data centric benchmarks, skeleton benchmarks and Micro benchmarks.

The interference between concurrently running jobs on HPC system have been identified as major culprit for job's performance variability. Abhinav et al. found that concurrently running applications on HPC can cause interference to each other, and cause communication time varied from 36% faster to 69% slower. Such interference could come from Operating System noise, shape of the allocated partition and mainly other running jobs that sharing network resources [4]. Skinner et al. found that there is a 2-3 slowdown in MPI_Allreduce due to network contention from other jobs [24]. Rosenthal

et al. found there is limited benefit for many applications by increasing network bandwidth. The applications that send mostly small messages or larger messages asynchronously are not bandwidth bounded, hence, benefit only slightly or not at all from increased bandwidth [21].

There are some research work focus on optimizing job allocation on HPC system to alleviate the interference between concurrently running jobs. Hoefler et al. propose to use performance modeling techniques to analysis factors that impact the performance of parallel scientific applications [14]. However, as the scale of HPC continue grows, the interference of concurrently running jobs is getting worse, which is hard to be quantified by performance profiling tools. Bogdan et al provide a set of guidelines how to configure a network with Dragonfly topology for workload with Nearest Neighbor communication pattern [20]. Dong et al describe IBM Blue Gene/Q's 5D torus interconnect network [5]. They developed simple benchmarks that conforms to four different communication patterns, namely ping-pong, nearest neighbor, broadcast and all_reduce, to demonstrate the effectiveness of this highly parallel 5D torus network.

Our work is different from all these works in the following ways. First, we focus on the dominant communication patterns rather than any specific application. We believe this can provide a guideline for other research work is the area. Secondly, we explored the inter-job interference between concurrently running jobs, while similar work such as [4] only focus on the single application's performance degradation due to network contention. Finally, we explored the impact of different allocation strategies to job's communication behavior. We identified the optimal allocation strategy for each application with specific dominant communication pattern. Based on our comprehensive exploration, we claim that better allocation strategy should take job's communication pattern into consideration for allocation decision making.

VIII. Conclusions

In this work, we study the communication behavior of three parallel applications, namely AMG, CrystalRouter and MultiGrid. Each application has distinctive communication pattern, that can be representative for a group of jobs in HPC workload. We use a sophisticate simulation tool named CODES from Argonne National Laboratory to simulate the running of these three parallel applications on torus network. The torus network provided by CODES has good fidelity and scalability. We analyzed the performance of each application's communication in terms of data transfer time by simulating them running on torus networks with different bandwidth and dimensionality configurations. We found that higher dimensionality of torus network would improve the performance of application with "Many-to-Many" communication patterns, while application with intensive local communication like "Nearest Neighbor" won't benefit much from higher dimensionality.

We also analysis the intra- and inter-job interference by simulating three applications running on 3D torus network. Based on our comprehensive experiments, we got three observations. 1) The compact allocation can not guarantee the best performance for every application. 2) The performance of application conforms to Nearest Neighbor communication pattern keep relatively stable with different shape allocations, while application conforms to "Many-to-Many" communication pattern prefer compact allocation. 3) The allocation unit size should be decided according to application's communication pattern. Unit size that big enough to preserve the locality of the application will result better performance.

We believe that our finding in this work can provide guidance for HPC resource management to make flexible job allocations. Rather than use pre-defined partitions and reckless non-contiguous allocation, future HPC system should assign each job with preferable resources based on job's communication pattern.

Acknowledgment

The work at Illinois Institute of Technology is supported in part by U.S. National Science Foundation grants CNS-1320125 and CCF-1422009. This work is also supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357.

References

[1] Y. Ajima, T. Inoue, S. Hiramoto, Y. Takagi, and T. Shimizu. The tofu interconnect. *IEEE Micro*, 32(1):21–31, Jan. 2012.

[2] C. Albing and M. Baker. Alps,topology, and performance: A comparison of linear orderings for application placement in a 3d torus. In *Proceedings of Cray User Group*, 2010.

[3] P. D. Barnes, Jr., C. D. Carothers, D. R. Jefferson, and J. M. LaPre. Warp speed: Executing time warp on 1,966,080 cores. In *Proceedings of the 1st ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM PADS '13, pages 327–336, New York, NY, USA, 2013. ACM.

[4] A. Bhatele, K. Mohror, S. H. Langer, and K. E. Isaacs. There goes the neighborhood: Performance degradation due to nearby jobs. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13, pages 41:1–41:12, New York, NY, USA, 2013. ACM.

[5] D. Chen, 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 network and message unit. In *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*, pages 1–10, Nov 2011.

[6] D. Chen, 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*, 32(1):32–43, Jan. 2012.

[7] J. Cope, N. Liu, S. Lang, C. D. Carothers, and R. B. Ross. Codes: Enabling co-design of multi-layer exascale storage architectures. In *Workshop on Emerging Supercomputing Technologies 2011 (WEST 2011)*, Tuscon, AZ, 2011.

[8] CORAL. Collaboration benchmark codes, Accessed October 14, 2015. Available online https://asc.llnl.gov/CORAL-benchmarks.

[9] S. David, W. Nicholas, F. Karl, and Y. Kathy. Integrated Performance Montoring(IPM), Accessed October 14, 2015. Available online http://ipm-hpc.sourceforge.net/overview.html.

[10] DOE. Characterization of the DOE mini-apps, Accessed October 14, 2015. Available online http://portal.nersc.gov/project/CAL/trace.htm.

[11] R. Fiedler and S. Whalen. Improving task placement for applications with 2d, 3d, and 4d virtual cartesian topologies on 3d torus networks with service nodes. In *Proceedings of Cray User Group*, 2013.

[12] P. Fischer, J. Lottes, D. Pointer, and A. Siegel. Petascale algorithms for reactor hydrodynamics. *Journal of Physics: Conference Series*, 125(1):012076, 2008.

[13] V. E. Henson and U. M. Yang. Boomeramg: A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 41(1):155 – 177, 2002. Developments and Trends in Iterative Methods for Large Systems of Equations - in memorium Rudiger Weiss.

[14] T. Hoefler, W. Gropp, W. Kramer, and M. Snir. Performance modeling for systematic performance tuning. In *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*, pages 1–12, Nov 2011.

[15] A. Jokanovic, J. Sancho, G. Rodriguez, A. Lucero, C. Minkenberg, and J. Labarta. Quiet neighborhoods: Key to protect job performance predictability. In *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*, pages 449–459, May 2015.

[16] 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 *Cluster Computing, 2002. Proceedings. 2002 IEEE International Conference on*, pages 296–304, 2002.

[17] M. Michael, J. Curt, C. Mike, B. Jim, R. Philip, and M. Tushar. mpiP: Lightweight, Scalable MPI profiling, Accessed October 14, 2015. Available online at http://mpip.sourceforge.net.

[18] M. Mubarak, C. Carothers, R. Ross, and P. Carns. Modeling a million-node dragonfly network using massively parallel discrete-event simulation. In *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, pages 366–376, Nov 2012.

[19] ORNL. Titan system overview, Accessed October 14, 2015. Available online https://www.olcf.ornl.gov/kbarticles/titan-system-overview.

[20] B. Prisacari, G. Rodriguez, P. Heidelberger, D. Chen, C. Minkenberg, and T. Hoefler. Efficient task placement and routing of nearest neighbor exchanges in dragonfly networks. In *Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing*, HPDC '14, pages 129–140, New York, NY, USA, 2014. ACM.

[21] E. Rosenthal and E. A. León. Characterizing application sensitivity to network performance.

[22] P. C. Roth, J. S. Meredith, and J. S. Vetter. Automated characterization of parallel application communication patterns. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '15, pages 73–84, New York, NY, USA, 2015. ACM.

[23] S. S. Shende and A. D. Malony. The tau parallel performance system. *Int. J. High Perform. Comput. Appl.*, 20(2):287–311, May 2006.

[24] D. Skinner and W. Kramer. Understanding the causes of performance variability in hpc workloads. In *Workload Characterization Symposium, 2005. Proceedings of the IEEE International*, pages 137–149, Oct 2005.

[25] SNL. SST DUMPI trace library, Accessed October 14, 2015. Available online http://sst.sandia.gov/usingdumpi.html.

[26] TOP. Top500 supercomputing web site, Accessed October 14, 2015. Available online http://www.top500.org.

[27] O. Tuncer, V. J. Leung, and A. K. Coskun. Pacmap: Topology mapping of unstructured communication patterns onto non-contiguous allocations. In *Proceedings of the 29th ACM on International Conference on Supercomputing*, ICS '15, pages 37–46, New York, NY, USA, 2015. ACM.

[28] J. Vetter, S. Lee, D. Li, G. Marin, C. McCurdy, J. Meredith, P. Roth, and K. Spafford. Quantifying architectural requirements of contemporary extreme-scale scientific applications. In S. A. Jarvis, S. A. Wright, and S. D. Hammond, editors, *High Performance Computing Systems. Performance Modeling, Benchmarking and Simulation*, volume 8551 of *Lecture Notes in Computer Science*, pages 3–24. Springer International Publishing, 2014.

[29] J. Wu, Z. Lan, X. Xiong, N. Gnedin, and A. Kravtsov. Hierarchical task mapping of cell-based amr cosmology simulations. In *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for*, pages 1–10, Nov 2012.

[30] X. Wu, F. Mueller, and S. Pakin. Automatic generation of executable communication specifications from parallel applications. In *Proceedings of the International Conference on Supercomputing*, ICS '11, pages 12–21, New York, NY, USA, 2011. ACM.

[31] 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 *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*, pages 439–448, May 2015.