Study of Intra- and Inter-Job Interference for Improving Allocation on HPC Systems

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

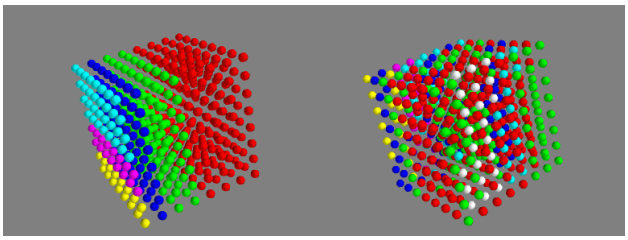
***Abstract*—Network contention between concurrently running jobs on HPC system is one of the primary causes of performance variability. Optimizing job allocation and avoiding network sharing are proven techniques to alleviate network contention and performance degradation. In this work, we propose a 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 inter/intra-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 [25] 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.

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 [19]. The delay of running jobs increases the queueing time of the following submitted jobs, thus leading to low system throughput and utilization [20].

This adverse effect of network sharing can be mitigated by providing jobs with isolated allocations and exclusive network resource. The batch scheduler is responsible for assigning job the amount of nodes it requires. On the widely used torus-connected HPC systems [34][35][36], two allocation strategies are commonly used. In the partition based allocation, used in Blue Gene series systems, he scheduler assigns each user job a compact and contiguous set of computing nodes. This strategy favors application performance by providing application with isolated partitions and exclusive network connections, thus reducing network contention caused by 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, in which free nodes are assigned to user jobs whether they are contiguous or not. Non-contiguous allocation eliminates internal and external fragmentation as seen in partition-based 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 jobs.



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

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

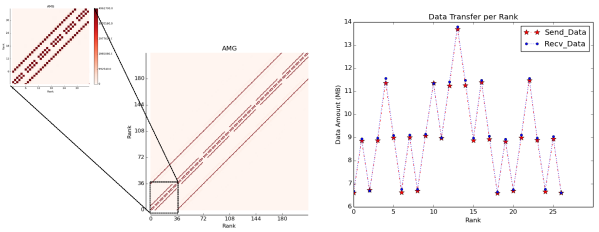
Rather than choosing blindly between partition-based or non-contiguous allocation, the new allocation strategy choses resources based on a job's communication pattern. With detail

analysis of application's communication patterns, our proposed allocation strategy obtains a clear view as to which processes are tightly connected and conduct intensive communication. The "neighborhood communication" consists of such tightly coupled processes should get a compact allocation. Then, the allocation strategy only needs to guarantee the locality of the subset of the nodes where those "neighborhood communication" would reside.

In this work, we study the impact of different allocation strategies on intra/inter-job interference on Torus network. 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. On the recent Top500 list, 6 of the top 10 supercomputers use a high-radix torus-interconnected network [39]. 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 [34]. The K computer from Japan uses the Tofu system, which is a 6D mesh/torus topology [35]. 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 [36]. 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[9] 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[30], from Argonne National Laboratory as a research vehicle to evaluate the performance of our communication pattern aware allocation policy on torus networks. We analyze each application's performance running on torus networks with different dimensionality and interconnected bandwidth. Then, we extend CODES with a Job Mapping API that supports different allocation policies on torus network. Different allocation strategies on torus network are studied with the help of this new Job Mapping API. We analyze the intra/inter-job interference by simulating those three applications running exclusively or concurrently on torus network with different allocation strategies. According to results of our comprehensive exploration, making allocation with consideration about job's communication pattern is a very promising allocation strategy. We believe the idea presented in this work would be the guideline for the design of future HPC batch job scheduler and resource management module.

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



(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.

II. Application Study

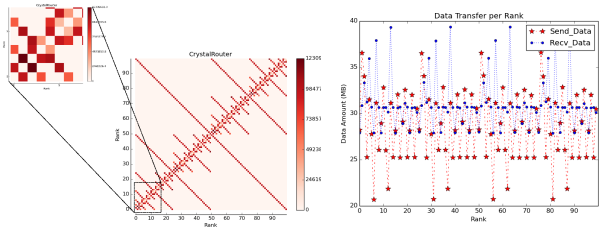
A parallel application usually conforms to a combination of several basic communication patterns[27]. At its different execution phases, the application's communication behavior may follow different basic patterns respectively. Collective communication is often used for data distribution and collection among processes. Nearest Neighbor communication is commonly seen in lots of structured and unstructured mesh physics packages. There are many profiling tools available to capture communication patterns from parallel applications, such as Tuning and Analysis Utilities(TAU)[7], mpiP[6]. They can help analysis parallel application, providing information like the percentage of different MPI operations of the applications, communication topology, the amount data transferred between processes.

In this work, we provide detailed analysis about the communication pattern of three DOE MiniApps. MiniApps are reduced proxy applications that encapsulate the salient performance of larger full size applications[12]. Since we only focus on the dominant communication pattern of parallel applications, these MiniApps are perfect candidates for analysis the application's performance on different allocations or on new network architectures. We choose three MiniApps, namely, Algebraic Multigrid Solver(AMG), Geometric MultiGrid(MultiGrid) and CrystalRouter. Their representative communication patterns that are commonly seen in the HPC system workload. The communication pattern figures we present here are generated with the IPM[11] data from [9].

A. AMG

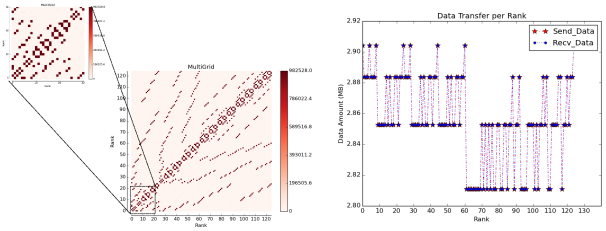
AMG is the MiniApp of Algebraic Multigrid Solver, which is a parallel algebraic multigrid solver for linear systems arising from problems on unstructured mesh physics packages. It has been derived directly from the BoomerAMG solver that is being developed in the Center for Applied Scientific Computing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

Figure 2a shows the communication topology of a small scale AMG with 27 MPI Ranks. Here we prefer to show the communication topology with small scale version of the



(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.



(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.

MiniApp because the dominant communication pattern of the application doesn't change with its scale. We can make the observation from the upper-left of the figure that each rank in AMG has intense communication between 3 other ranks, such as rank 0 between rank 1, 3 and 9, rank 1 between 2, 4 and 10, etc. And this relation is also symmetric along the main diagnal. 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 blue line shows that total data amount transferred from each rank, while the red shows the amount of received data and green shows the amount of send data. The first thing we can spot from the figure is that for each rank, the amount of data sent and received are basically the same, which also well explain the symmetry of the application's dominant communication pattern. And the second thing we can find about AMG is that the data transfer amount also has a symmetric pattern. Rank 0 has the same amount of data transfer as Rank 26, Rank 1 as of Rank 25, etc.

AMG represents a group of applications whose dominant communication pattern is Nearest Neighbor. Applications with such similar dominant communication pattern are like PARTISN, SNAP[9].

B. Crystal Router

The second MiniApp we have is Crystal Router, which is the extracted communication kernel of the full application Nek5000. Nek5000[14] is a spectral element CFD application developed at Argonne National Laboratory. It features spectral element multigrid 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. The communication kernel of Nek5000, 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.

As we can see from Figure 3a, every 5-rank is a compact community, where data transfer is very intensive. Beside the "5-rank community", there is "pair-wise" global data transfer in CrystalRouter. Obviously, there is data transfer alone the diagnal. This is because messages would be buffered on their node of origin until a synchronization point was reached when a single system call sent every message to its destination in one pass. When the scale of Crystal Router increase to 100, the "5-rank community" is still the dominant pattern in a hierarchical manner. The same pattern can be observed from the global data transfer. Our observation about the dominant communication pattern of CrystalRouter is the multi-stage "5-rank community" and "pair-wise" global data transfer happen hierarchically as the scale of the application increase.

Figure 3b shows the amount of data each rank transferred in Crystal Router. The locality pattern of "5-rank community" can still be spotted in this figure. The spiky shape occurs in the frequency of every 5 rank. Unlike AMG, the there is variance between the amount of data send and received for each rank. This is due to the specific feature of Crystal Router'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

Different from AMG, MultiGrid is geometric multigrid 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 multigrid v-cycle. It is widely used for structured grid physics packages. Figure4a shows the communication topology of MultiGrid and Figure4b shows the the amount of data transferred between ranks in MultiGrid.

MultiGrid represents a group of applications whose dominant communication pattern is Many-to-Many (some literatures also refer it as All-to-All [27]).

III. Research Vehicle

It is impossible to accurately study the interference between concurrently running jobs with different allocation strategies on production systems. There are a couple of reasons for that. First, 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. Second, it is unrealistic to reserve the system exclusively to run the same job with desired allocation without interference then compare the results to that in the presence of interference. Due to those reasons, we resort to simulation for this work.

There are couple of simulation platform to study the behavior of large scale systems. SimGrid from Inria is a versatile simulation tool sets that provide simulation for Grids, Clouds and HPC systems[38]. The Structural Simulation Toolkit (SST) from Sandia National Lab was developed to explore innovations in highly concurrent systems where the ISA, microarchitecture, and memory interact with the programming model and communications system[37].

A simulation toolkit named CODES (Co-Design of Multi-layer Exascale Storage Architecture) from Argonne National Lab enables the exploration of simulating different HPC networks with high fidelity[30][32]. 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[31]. CODES support both torus and dragonfly network with high fidelity flit-level simulation. In this work, we only use torus network to show the impact of node allocation to job with different dominant communication patterns. CODES has this network workload component that capable of conducting trace-driven simulation. It can take real MPI application trace generated by SST DUMPI[37] to drive CODES network models. We implemented a Job Mapping API for CODES, which can provide flexible job allocation strategies. The new API can help us to explore the impact of different allocation strategies to those three applications in Section II

IV. Experiment

CODES can provide Torus network model with good scalability and high fidelity[33]. In this section, we will first study the communication behavior of three applications on torus network with different dimensionality and bandwidth configurations. Then, we study the intra- and inter-job interference with these three applications by running them exclusively and concurrently with different allocation strategies.

A. Configuration Study

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 $2\times$ 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 (16x16x8), a 5D torus (8x4x4x4x4), and a 7D torus (4x4x4x4x2x2x2). The bandwidth of direct link between nodes is 2GiB/s in each direction, 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 transfer of data. This can justify the fidelity and consistency of the torus network model. 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 no such great dispersion for CrystalRouter and MultiGrid, shown in Figure 5b and 5c, since no great variance of data transfer amount of each rank in these applications.

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

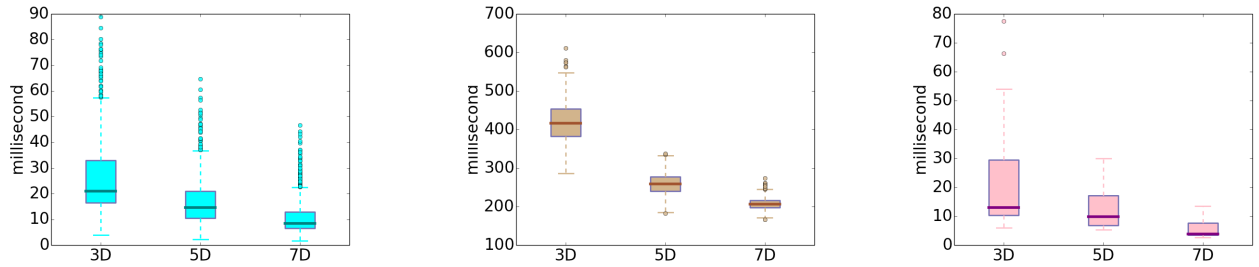
TABLE I Torus networks with same aggregated bandwidth

	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.

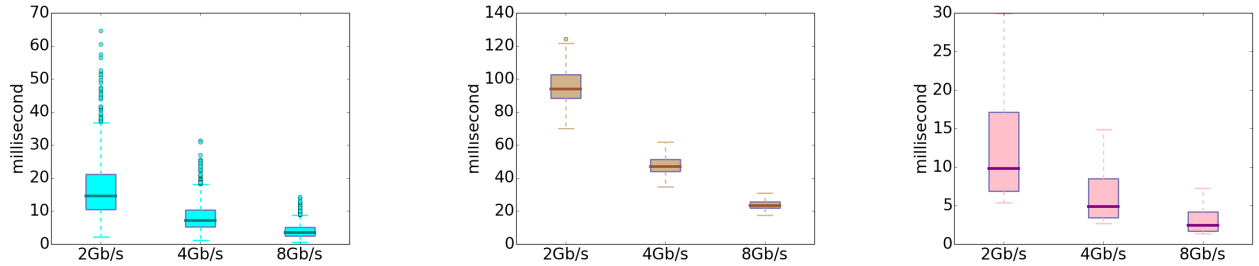
The performance of CrystalRouter actually deteriorate as the torus dimensionality increasing, shown in Figure 7b. This is due to the direct link bandwidth reduces from 4.67Gb/s to 2Gb/s, as dimension increases from 3D to 7D. CrystalRouter's large amount of intensive local data transfer benefit more from 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.



(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.

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.

B. Interference Analysis

Communication variability due to network sharing can cause application performance degradation in two ways, namely intra- and inter-job interference. The ***intra-job interference*** refers to the network contention between ranks within each application. There are lots of research works[23][24] 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.

Inter-job interference is introduced by concurrently running jobs that adjacent to each other sharing network resources. Such interference has been identified as one of the major culprits for application's performance variability[19][21][15]. Inter-job interference is a more prominent issue for system adopted non-contiguous allocation policy than partition-based system. Application's communication time can vary from 36% faster to 69% slower due to inter-job interference when running on non-contiguous allocation[19].

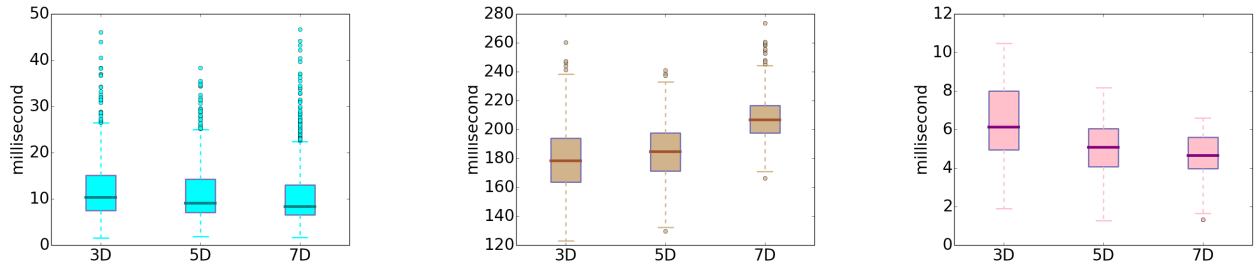
In this section we will study both kinds of interference with with AMG, CrystalRouter and MultiGrid in their medium scale. To be specific, we choose AMG with 216 ranks, CrystalRouter with 100 ranks, and MultiGrid 125 ranks and simulate their running on 2K-node ($16 \times 16 \times 8$) 3D torus network.

We can get observations that

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.

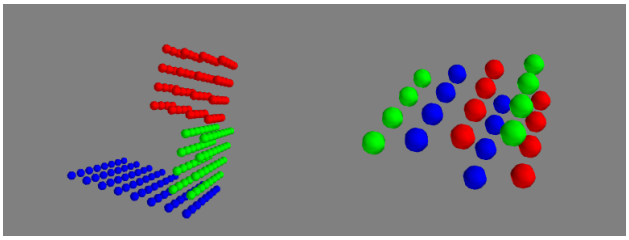
1) Intra-Job Interference Analysis: To study the intra-job interference of each job, we simulate each job running on 3D torus network with different allocation shapes. We assign each application with three different shapes allocation, i.e. 3D balanced-cube, 3D unbalanced-cube, 2D mesh.

3D balanced-cube, shown as red in Figure 8a, can guarantee



(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.



(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. Different colors represent different jobs. The nodes assigned to different jobs are interleaved, the allocation unit is 4-node.

the minimum *average pair-wise distance* within the allocation. There are some research work like [1] [19] 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[18].

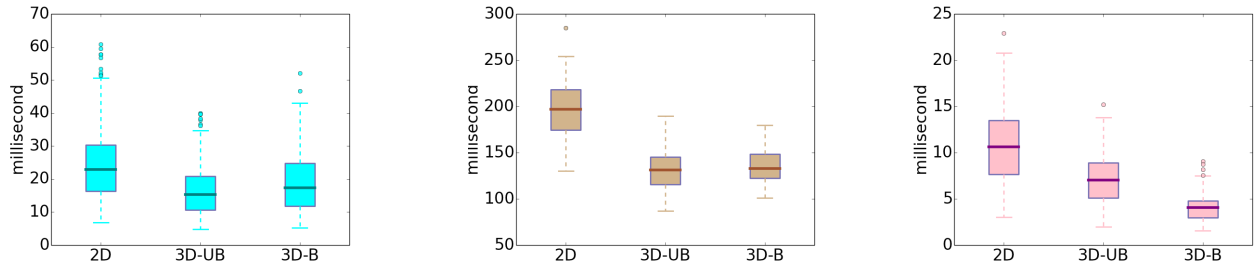
2D mesh, shown as blue in Figure 8a, can be cut out from a single layer of the 3D torus. 2D mesh is a very common allocation shape in torus network for both partition-based and non-contiguous allocation strategies. For example, Cray Application Level Placement Scheduler indexes the nodes in torus network 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 [25].

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. Figure 9a shows that for AMG, who has this symmetric local communication pattern through all ranks, spent less time when 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.

OBSERVATION 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



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

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

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.

OBSERVATION 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.

2) Inter-Job Interference Analysis: We simulate three application running concurrently with different allocation to analysis the interference between them. First, we assign each application with contiguous allocation side by side, and compare their performance with each application running exclusively. For AMG, sharing the network with other application won't affect its performance, since all the data transfer are between neighboring ranks. CrystalRouter and MultiGrid will suffer prolonged communication time when sharing network with others. The results are shown in Figure **??**.

Then, we assign each application with non-contiguous allocation and make them running concurrently. The allocation unit belongs to different jobs are interleaved as shown in Figure 8b. The unit size is critical to the application's 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(bit) 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 5-rank locality. 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 take 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 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.

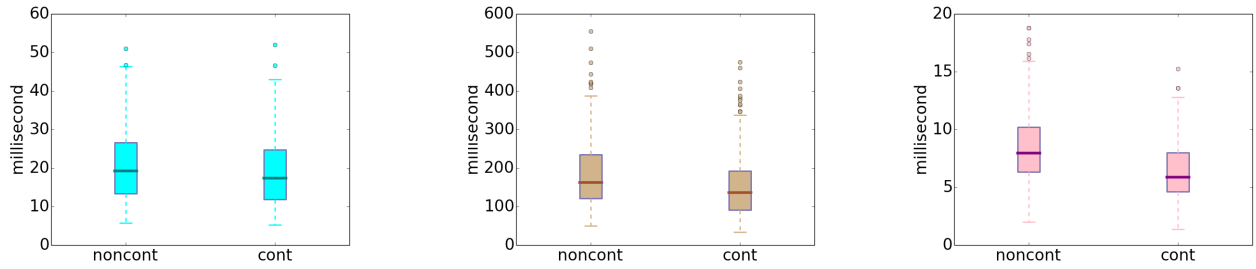
OBSERVATION The unit size in non-contiguous allocation policy should be chosen according to application's communication pattern. Inter-job interference is inevitable in non-contiguous based systems, but unit size that big enough to preserve the locality of the application will alleviate such interference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

V. Discussion

HPC system usually accommodate many parallel jobs running concurrently. Each job gets the required number of computing nodes by system resource manager. The allocated nodes for each job communicate with each other through system interconnect network in a sharing way, which cause interference as a negative side-effect. This interference can introduce performance degradation to both job and system, such as prolonged job running time and reduced system throughput.

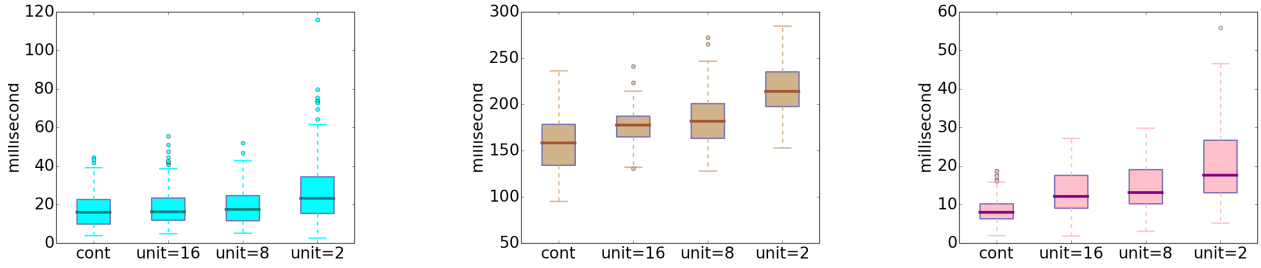
The knowledge about HPC job's interference is extremely useful for improving system and job's performance. By scrutinizing job's communication behavior, we can locate the critical path with job's communication graph, identify job's dominant communication pattern. With such knowledge about job's communication characteristics, we can analysis the possible interference between jobs through simulation or even predict such interference when job running on different allocations.

The most promising way to utilize such knowledge is to make smart allocation based on job's dominant communication pattern. Most existing batch schedulers take so few initiative



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

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



(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, applications running concurrently with interleaved allocation of different unit sizes, which are 16-node, 8-node, 2-node.

because it been hold under the obscurantism policy. Whether it use partition-based or non-contiguous policy, means it chooses the benefit of either system or job over the other. Some propose compromised policies [4][5][3] to use fix allocation unit size and make the allocation for job blindly without knowing job's communication information.

Based on the findings in this work, we propose the idea of smart job allocation with consideration about job's communication pattern. As the dominant communication pattern of parallel applications on HPC system can be summarized into a couple of categories, the deep understanding of one application can be helpful to all other applications that conforms to the same communication pattern. When an application with dominant communication that is intensive all-to-all submitted to the system, it should be granted with compact allocation with exclusive network provision, so that it won't suffer performance loss due to interference. If job with dominant communication pattern like Nearest Neighbor or others with intensive small scale communication, rather than assign it with unnecessary compact allocation and exclusive network provision, non-contiguous allocation with proper unit size would serve it equally well. No degradation will be introduced to job's performance and system utilization will be improved.

The advantage of making allocation with consideration about job's communication pattern is obvious. First, compared with partition-based policy, the new allocation strategy is more flexible, it doesn't require the system to provide a big 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.

VI. Related Work

There are some tools available for system monitoring and application profiling on HPC systems. Tools like TAU (Tuning and Analysis) [7], SCALASCA[8] and mpiP[6] 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[27], 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[9]. 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[10]. 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[19]. Skinner et al. found that there is a 2-3 slowdown in MPI_Allreduce due to network contention from other jobs[21]. 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[15].

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[16]. 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[26]. Dong et al describe IBM Blue Gene/Q's 5D torus interconnect network[29]. They developed simple benchmarks that conforms to four different communication patterns, namely ping-pong, nearest neighbor, broadcast and allreduce, 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 [19] 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.

VII. 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] V.J. Leung, E.M. Arkin, M.A. Bender, D. Bunde, J. Johnston, etc, Processor allocation on Cplant: achieving general processor locality using one-dimensional allocation strategies In Proceedings of 2002 IEEE International Conference on Cluster Computing, 2002. pages 296–304, 2002

[2] Carl Albing and Mark Baker ALPS,Topology, and Performance: A Comparison of Linear Orderings for Application Placement in a 3D torus. Presented at the CUG 2010, Edinburgh, Scotland, UK, 2010.

[3] Xu Yang and Zhou Zhou and Wei Tang and Xingwu Zheng and Jia Wang and Zhiling Lan. Cluster Computing (CLUSTER), 2014 IEEE International Conference on, Balancing job performance with system performance via locality-aware scheduling on torus-connected systems Sept, 2014, pp.140-148

[4] V. Lo, K. Windisch, W. Liu, and Nitzberg. Noncontiguous processor allocation algorithms for mesh-connected multicomputers. IEEE Transactions on Parallel and Distributed Systems 8 (1997). pages 712–726

[5] JA. Pascual, J. Navaridas and J. Miguel-Alonso. Effects of Topology-Aware Allocation Policies on Scheduling Performance. 14th Workshop on Job Scheduling Strategies for Parallel Processing. May 29, 2009.Rome, Italy.

[6] mpiP: Lightweight, Scalable MPI Profiling. http://mpip.sourceforge.net, 2013.

[7] S. Shende and A. Malony. The TAU parallel performance system. International Journal of High Performance Computing Applications, 20(2):287311, 2006.

[8] X. Wu, F. Mueller, S. Pakin, "Automatic Generation of Executable Communication Specifications from Parallel Applications", In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis(SC), 2011 International Conference for SC '11, 2011,pages 12-21.

[9] Department of Energy, Characterization of the DOE Mini-apps. [Online]. Available: http://portal.nersc.gov/project/CAL/trace.htm (Accessed on: Sept 16th. 2015)

[10] CORAL collaboration benchmark codes [Online]. Available:https://asc.llnl.gov/CORAL-benchmarks/ (Accessed on: Sept 16th. 2015)

[11] David Skinner, Nicholas Wright, Karl Fuerlinger and Kathy Yelick Integrated Performance Montoring(IPM) Available http://ipm-hpc.sourceforge.net/overview.html (Accessed on: Sept 16th. 2015)

[12] Bronson Messer "Using a Developing MiniApp to Compare Platform Characteristics on Cray Systems" In Proceedings of Cray User Group 2012

[13] V. E. Henson and U. M. Yang, "BoomerAMG: A Parallel Algebraic Multigrid Solver and Preconditioner", Appl. Num. Math. 41 (2002), pp. 155-177. UCRL-JC-141495.

[14] P. Fischer, J. Lottes, D. Pointer, and A. Siegel, Petascale algorithms for reactor hydrodynamics, Journal of Physics: Conference Series, vol. 125, no. 1, p. 012076, 2008.

[15] E. Rosenthal and E.A. Leon Characterizing Application Sensitivity to Network Performance. In Proceedings of SC14: International Conference for High Performance Computing, Networking, Storage and Analysis, 2014

[16] T. Hoefler and W. Gropp and M. Snir and W. Kramer Performance Modeling for Systematic Performance Tuning, 2011 Nov. In Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis (SC'11), SotP Session,

[17] P. Roth and J. Meredith and J. Vetter, "Automated Characterization of Parallel Application Communication Patterns", In Proceedings of the 25rd International Symposium on High-performance Parallel and Distributed Computing HPDC '15, 2015,

[18] Fiedler, Robert, and Stephen Whalen. "Improving task placement for applications with 2D, 3D, and 4D virtual Cartesian topologies on 3D torus networks with service nodes." Proc. Cray Users Group (2013).

[19] Bhatele, Abhinav and Mohror, Kathryn and Langer, Steven H. and Isaacs, Katherine E. There Goes the Neighborhood: Performance Degradation Due to Nearby Jobs In Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis, 2013

[20] Jokanovic, A.; Sancho, J.C.; Rodriguez, G.; Lucero, A.; Minkenberg, C.; Labarta, J., "Quiet Neighborhoods: Key to Protect Job Performance Predictability," in Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International , vol., no., pp.449-459, 25-29 May 2015

[21] D. Skinner and W. Kramer. Understanding the Causes of Performance Variability in HPC Workloads. In Proceedings of the IEEE International Workload Characterization Symposium, 2005, pages 137149, 2005.

[22] Prisacari, Bogdan and Rodriguez, German and Heidelberger, Philip and Chen, Dong and Minkenberg, Cyriel and Hoefler, Torsten 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, 2014, Vancouver, BC, Canada

[23] Wu, Jingjin and Lan, Zhiling and Xiong, Xuanxing and Gnedin, Nickolay Y. and Kravtsov, Andrey V. Hierarchical Task Mapping of Cell-based AMR Cosmology Simulations Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis SC '12, 2012, Salt Lake City, Utah

[24] Tuncer, Ozan, Vitus J Leung, Ayse K. Coskun, "PaCMap: Topology Mapping of Unstructured Communication Patterns onto Non-contiguous Allocations," 29th International Conference on Supercomputing, June 2015.

[25] 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, Proceedings of 29th IEEE International Parallel & Distributed Processing Symposium, IPDPS'15, 2015. Hyderabad, INDIA

[26] Prisacari, Bogdan and Rodriguez, German and Heidelberger, Philip and Chen, Dong and Minkenberg, Cyriel and Hoefler, Torsten Efficient Task Placement and Routing of Nearest Neighbor Exchanges in Dragonfly Networks, Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing, HPDC '14, 2014, Vancouver, BC, Canada,

[27] Philip. C.R, Jeremy S. M, etc "Automated Characterization of Parallel Application Communication Patterns" HPDC'15

[28] J. S. Vetter, S. Lee, D. Li, G. Marin, C. McCurdy, J. Meredith, P. C. Roth, and K. Spafford. Quantifying architectural requirements of contemporary extreme-scale scientific applications. In International Workshop on Performance Modeling, Benchmarking and Simulation of HPC Systems (PMBS13), Denver, Colorado, 2013.

[29] Dong Chen; Eisley, N.A.; Heidelberger, P.; Senger, R.M.; Sugawara, Y.; Kumar, S.; Salapura, V.; Satterfield, D.L.; Steinmacher-Burow, B.; Parker, J.J., "The IBM Blue Gene/Q interconnection network and message unit," High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference, pp.1,10, 12-18 Nov. 2011

[30] Jason Cope, Ning Liu, Sam Lang, Phil Carns, Chris Carothers, Robert Ross. CODES: Enabling Co-design of Multilayer Exascale Storage Architectures, Proceedings of the Workshop on Emerging Supercomputing Technologies 2011, pp. 303-312 2011

[31] P. D. Barnes, C. D. Carothers, D. R. Jefferson, and J. M. LaPre, Warp speed: executing time warp on 1,966,080 cores, in Proc. of the 2013 ACM SIGSIM Conf. on Principles of Advanced Discrete Simulation (PADS), May 2013, pp. 327336.

[32] Mubarak, M.; Carothers, C.D.; Ross, R.; Carns, P., "Modeling a Million-Node Dragonfly Network Using Massively Parallel Discrete-Event Simulation," High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion: pp.366,376, 10-16 Nov. 2012

[33] Mubarak, Misbah, et al. "A case study in using massively parallel simulation for extreme-scale torus network codesign." in Proceedings of the 2nd ACM SIGSIM/PADS conference on Principles of advanced discrete simulation. ACM, 2014.

[34] 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, Micro, IEEE, vol. 32, no. 1, pp. 3243, 2012.

[35] Y. Ajima, Y. Takagi, T. Inoue, S. Hiramoto, and T. Shimizu, The Tofu Interconnect, in 2011 IEEE 19th Annual Symposium on High Performance Interconnects (HOTI), 2011, pp. 8794.

[36] Titan System Overview [online] Available: https://www.olcf.ornl.gov/kbarticles/titan-system-overview (Accessed on: OCT 4th, 2015)

[37] Sandia National Labs, SST DUMPI trace library. [Online]. Available: http://sst.sandia.gov/using dumpi.html (Accessed on: Sept 3rd, 2015)

[38] SIMGRID Versatile Simulation of Distributed Systems [Online]. Availabale: http://simgrid.gforge.inria.fr (Accessed on: Oct 4th, 2015)

[39] Top500 supercomputing web site. [Online]. Available: http://www.top500.org (Accessed on: Oct 4th, 2015)