Watch Out for the Bully! Job Interference Study on Dragonfly Network

Xu Yang*, John Jenkins†, Misbah Mubarak†, Robert B. Ross†, Zhiling Lan*
* *Department of Computer Science, Illinois Institute of Technology, Chicago, Illinois, USA 60616*
{xyang56}@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—**High-radix, low-diameter dragonfly networks will be a common choice in next-generation supercomputers. Preliminary studies show that random job placement with adaptive routing should be the rule of thumb to utilize such networks, since it uniformly distributes traffic and alleviates congestion. Nevertheless, in this work we find that while random job placement coupled with adaptive routing is good at load balancing network traffic, it cannot guarantee the best performance for every job. The performance improvement of communication-intensive applications comes at the expense of performance degradation of less intensive ones. We identify this bully behavior and validate its underlying causes with the help of detailed network simulation and real application traces. We further investigate a hybrid contiguous-noncontiguous job placement policy as an alternative. Initial experimentation shows that hybrid job placement aids in reducing the worst-case performance degradation for less communication-intensive applications while retaining the performance of communication-intensive ones.**

1. Introduction

Low-latency and high-bandwidth interconnect networks play a critical role in ensuring HPC system performance. The high-radix, low-diameter dragonfly topology can lower the overall cost of the interconnect, improve network bandwidth and reduce packet latency [1], making it a very promising choice for building supercomputers with millions of cores. Even with such powerful networks, intelligent job placement is of paramount importance to the efficient use of dragonfly connected systems [2], [3].

Recent studies suggest that random node allocation for parallel jobs, coupled with adaptive routing, can alleviate local congestion, eliminate hot-spots and achieve load-balancing for dragonfly networks [3], [4], [5]. These studies explore the possible job placement and routing configurations that could optimize the overall network performance without examining how different configurations impact the progress of individual applications. *In this paper, we study the implications of contention for shared network links in the context of multiple HPC applications running on dragonfly systems when different job placement and routing configurations are in use.* Our analyses focus on the overall network performance as well as the performance of concurrently executing applications in the presence of network contention.

We choose three representative HPC applications from the DOE Design Forward Project [6] and analyze the interference among them. Our study is based on simulation with CODES, a high-fidelity, flit-level HPC network simulation toolkit [7]. For each application, we examine its performance with two job placement policies and three routing policies. We make the following observations through extensive simulations.

- Concurrently running applications on a dragonfly network interfere with each other when they share network resources. Communication-intensive applications "bully" their less intensive peers and obtain performance improvement at the expense of less intensive ones.
- Random placement of application processes in the dragonfly can improve the performance of communication-intensive applications by enabling network resource sharing, though it introduces interference causing performance degradation to the less intensive applications.
- Contiguous placement can be beneficial to the consistent performance of less communication-intensive applications by minimizing network resource sharing, because it reduces the opportunities for traffic from other applications to be loaded on links that serve as minimal routes for the less intensive application. However, this comes with the downside of reduced system performance due to load imbalance.

Based on the aforementioned key observations, one would expect that an ideal job placement policy on dragonfly systems would take relative communication intensity into account, and mix contiguous and non-contiguous placement based on application needs. To explore this expectation, we investigate *a hybrid job placement policy*, which assigns random allocations to communication-intensive applications and contiguous allocations to less intensive ones. Initial experimentation shows that hybrid job placement aids in reducing the worst-case performance degradation for

SC16; Salt Lake City, Utah, USA; November 2016
978-1-4673-8815-3/16/$31.00 ©2016 IEEE

less communication-intensive applications while retaining the performance of communication-intensive applications, though without eliminating the problem entirely.

To the best of our knowledge, using real application traces from production systems for the study of job interference on dragonfly networks has not been reported in the literature so far. We believe the observations and new placement policy presented in this paper are valuable to the HPC community.

The rest of this paper is organized as follows. Section 2 describes an implementation of the dragonfly network, introduces the placement policies and routing policies. Section 3 discusses the use of CODES as a research vehicle and three representative applications from the DOE Design Forward Project. Section 4 presents the observations and analysis of three applications running on a dragonfly network with different placement and routing configurations. Section 5 validates the observations we obtain from previous section. Section 6 presents a alternative placement policy for the dragonfly network. Section 7 discusses the related work. Finally, the conclusion is presented in Section 8.

2. Background

In this section, we review the dragonfly topology, including the placement and routing policies examined in previous work.

2.1. Dragonfly Network

The dragonfly is a two-level hierarchical topology, consisting of several groups connected by all-to-all links. Each group consists of a routers connected via all-to-all local channels. For each router, p compute nodes are attached to it via terminal links, while h links are used as global channels for intergroup connections. The resulting radix of each router is $k = a+h+p-1$. Different computing centers could choose different values for a, h, p when deploying their dragonfly network. The adoption of proper a, h, p involves many factors such as system scale, building cost and workload characteristics.

It is recommended that for load balancing purposes, a proper dragonfly configuration should follow $a = 2p = 2h$ [8]. According to this configuration, the total number of groups, denoted as g in the dragonfly network would be $g = a * h + 1$, the total number of compute nodes denoted as N in the network would be $N = p * a * g$. An example dragonfly network is illustrated in Figure 1. There are six routers in each group $(a = 6)$, three compute nodes per router $(p = 3)$, and three global channels per router $(h = 3)$. This dragonfly network consists of 19 groups and 342 nodes in total.

2.2. Routing on Dragonfly

The routing policy refers to the strategy adopted to route packets from the source router to the destination router.

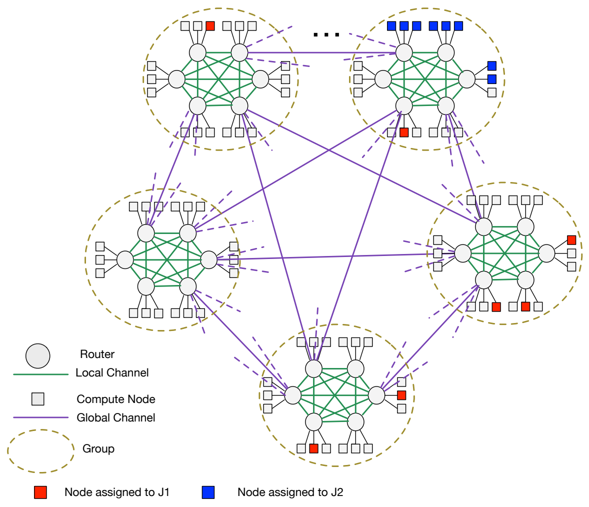


Figure 1: Five group slice of a 19-group dragonfly network. Job $J1$ is allocated using random placement, while Job $J2$ is allocated using contiguous placement.

Previously studied routing policies for dragonfly networks include minimal routing, adaptive routing [1], progressive adaptive routing [9] and variations thereof [10]. In this work we study three alternative routing policies considered by the community for dragonfly networks.

Minimal: In this policy, a packet takes the minimal (shortest) path from the source to the destination. The packet first routes locally from the source node to the global channel leading to the destination group. It traverses the global channel to the destination group and routes locally to the destination node. Minimal routing can guarantee the minimum hops a packet takes from the source to the destination. However, it usually results in congestion along the minimal paths.

Adaptive: In this policy, the path a packet takes will be adaptively chosen between minimal and non-minimal paths, depending on the congestion situation along those paths. For the non-minimal path, an intermediate router in a separate group will be randomly chosen. The packet is forwarded to the intermediate router, connecting the source and destination groups through two separate minimal paths. Adaptive routing can avoid hot-spots in the presence of congestion and collapses to minimal routing otherwise.

Progressive Adaptive: As opposed to adaptive routing, the decision to adaptively route a packet is continually re-evaluated within the source group until a non-minimal route is chosen; the re-evaluation does not occur in intermediate groups [9]. Progressive adaptive routing is capable of handling scenarios where the minimal route is congested but the source router has not been informed yet.

2.3. Job Placement on Dragonfly

For a parallel application requiring multiple compute nodes, the job placement policy refers to the way of assigning the required number of nodes to the application by system software such as the batch scheduler [11]. In this work, we study two alternative placement policies considered by the community for dragonfly systems:

Random Placement: In this policy, an application gets the required number of nodes randomly from the available nodes in the system. As illustrated in Figure 1, $J1$ is randomly allocated nodes attached to different routers in different groups. Routers may be shared by different applications and more routers are involved in serving each application when random placement is in use. Random placement can distribute the tasks of an application uniformly across the network to avoid the possible local congestion.

Contiguous Placement: In this policy, the compute nodes are assigned to an application consecutively. The assignment first fills up a group, then crosses group boundaries as necessary. As illustrated in Figure 1, $J2$ is allocated eight nodes by contiguous placement. Contiguous placement confines the tasks of an application into the same group and uses the minimum number of routers to serve each application, which may result in local network congestion and increase the possibility of hot-spots.

3. Methodology

Configurable dragonfly networks that allow us to perform the exploration presented in this paper are hard to come by for the time being. Even with access to systems with such networks, job placement and routing policies are part of system configuration, which is impossible for users to make changes at will [3], [4], [12], [13]. Therefore, we resort to simulation in our work.

3.1. Simulation Tool

We utilize the CODES simulation toolkit (Co-Design of Multilayer Exascale Storage Architectures) [7], which builds upon the ROSS parallel discrete event simulator [14], [15] to enable exploratory study of large scale systems of interest to the HPC community. CODES supports dragonfly [7], [16], torus [17], [18], and Slim Fly [19] networks with flit-level high-fidelity simulation. It can drive these models through an MPI simulation layer utilizing traces generated by the DUMPI MPI trace library available as part of the SST macro toolkit [20]. The behavior and performance of the CODES dragonfly network model has been validated by Mubarak et al. [16] against BookSim, a serial cycle-accurate interconnection network simulator [21].

3.2. Parallel Applications

We use a trace-driven approach to workload generation, choosing in particular three parallel application traces gathered to represent exascale workload behavior as part of the DOE Design Forward Project [6], [22]. Specifically, we study communication traces representing the *Algebraic MultiGrid Solver* (AMG), *Geometric Multigrid V-Cycle from Production Elliptic Solver* (MultiGrid) and *Crystal Router MiniApp* (CrystalRouter).

AMG: The Algebraic MultiGrid Solver 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 developed in the Center for Applied Scientific Computing (CASC) at LLNL [23].

MultiGrid: The geometric multi-grid v-cycle from the production elliptic solver BoxLib is a software framework for massively parallel block-structured adaptive mesh refinement (AMR) codes [24], which are used in structured grid physics packages.

CrystalRouter: The Crystal Router MiniApp is a communication kernel of Nek5000 [25], 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 is widely used for projects including ocean current modeling, thermal hydraulics of reactor cores, and spatiotemporal chaos.

3.3. System Configuration

The parameters for building the dragonfly network studied in our work are chosen based on the model proposed by Kim et al. [8]. In our dragonfly network, each group consists of $a = 8$ routers connected via all-to-all local channels. For each router, there are $p = 4$ compute nodes attached to it via terminal links. Each router also has $h = 4$ global channels used for intergroup connections. The radix of each router is hence $k = a+h+p-1 = 15$. The total number of groups is $g = a * h + 1 = 33$ and the total number of compute nodes is $N = p * a * g = 1056$. The dragonfly link bandwidths are asymmetric, 2Gib/s for the local and terminal router-ports and 4GiB/s for the global ports, indicated by the Cray Cascade system [26]. In this work, we simulate the network performance and job interference across six different job placement and routing policy combinations, which are summarized in Table 1.

TABLE 1: Nomenclature for different placement and routing configurations

	Routing Policies		
Placement Policies	Minimal	Adaptive	Progressive Adaptive
Contiguous	cont-min	cont-adp	cont-padp
Random	rand-min	rand-adp	rand-padp

We analyze both the overall network performance and the performance of each application. Our analysis focuses on the following metrics:

- **Network Traffic:** The traffic refers to the amount of data in bytes going through each router. We analyze the traffic on each terminal and on the local and global channels of each router. The network reaches optimal performance when the traffic is uniformly distributed and no particular network link is overloaded.

- **Network Saturation Time:** The saturation time refers to the time period when the buffer of a certain port in the router is full. We analyze the saturation time of ports corresponding to terminal links, local and global channels. The saturation time indicates the congestion level of routers.
- **Communication Time:** The communication time of each MPI rank refers to the time it spends in completing all its message exchanges with other ranks. Due to the use of simulation, we are able to measure the absolute (simulated) time a message takes to reach its destination. The performance of each application is measured by the communication time distribution across all its ranks.

Note that we do not model computation for each MPI rank due to both the complexities inherent in performance prediction on separate parallel architectures as well as the emphasis on the side of the Design Forward traces on communication behavior rather than compute representation; users are instructed to treat the traces as if they came from one MPI rank-per-node configuration, despite being gathered using a rank-per-core approach. We follow the recommended interpretation in our simulations.

3.4. Workload Summary

Two sets of parallel workloads are used in this study. Workload I consists of AMG, MultiGrid and CrystalRouter. As shown in Table 2, AMG has the least amount of data transfer, making it the least communication-intensive job in the workload. Workload II consists of sAMG, MultiGrid and CrystalRouter. sAMG, a synthetic version of AMG, is generated by increasing the data transferred in AMG's MPI calls by a factor of 100, making it the most communication-intensive job in the workload. We add sAMG for reasons that will become clear in the following sections.

As a significant portion of our experiments rely on nondeterministic behavior (random application allocation), we ran each configuration a total of 50 times with differing random seeds. We then chose a representative execution for presentation based on the median performance of each application. While there is variation in repeated runs of the following experiments, the resulting trends and observations are representative of the full suite of experimentation.

TABLE 2: Summary of Applications

App Name	Num. Rank	Avg. Data/Rank	Total Data
AMG	216	0.6MB	130MB
MultiGrid	125	5MB	625MB
CrystalRouter	100	35MB	3500MB
sAMG	216	60MB	13000MB

4. Study of Parallel Workload I

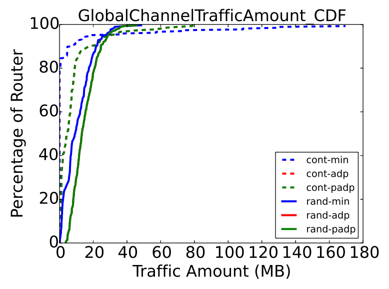
The study of Workload I consists of two parts. First, we analyze the overall network performance when Workload I is running under different placement and routing configurations. Second, we isolate each application from the workload and analyze its performance on both a per-rank basis as well as by considering router traffic resident to application ranks. The analysis allows us to identify the "bully" in Workload I.

4.1. Network Performance Analysis

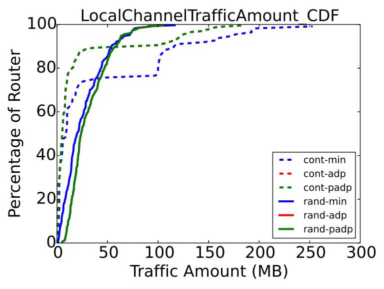
We first study the network performance at the system level by analyzing the degree of traffic and saturation seen at each router.

Figure 2 shows the aggregate traffic for terminal links, local and global channels, as well as the corresponding saturation time for Workload I under the placement and routing configurations summarized in Table 1. When contiguous placement is coupled with minimal routing (cont-min), application traffic is confined within the consecutively allocated groups, causing congestion on some routers along minimal paths to and from application nodes. Both local and global channels experience significant congestion, as applications span multiple groups. Similarly, the saturation time for both local and global channels are the highest compared with other configurations. When contiguous placement is coupled with adaptive (cont-adp) and progressive adaptive (cont-padp) routing, traffic is able to take non-minimal paths via intermediate routers, helping to alleviate congestion along the minimal paths. The resulting traffic through the most utilized local and global channels are greatly reduced, as shown in Figures 2a and 2b. Similarly, the corresponding saturation time on local and global channels is also reduced significantly, demonstrating the efficacy of adaptive routing in this case. For contiguous placement, we see no perceptible difference in behavior between adaptive and progressive adaptive routing.

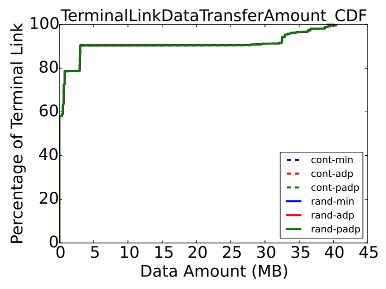
In most cases, the random placement policy behaves similarly across routing policies. Random placement uniformly distributes MPI ranks over the network, balancing the resulting traffic load. As shown in Figures 2a and 2b, no router experiences an exceptionally high volume of traffic on its local and global channels. When random placement is coupled with minimal routing (rand-min), less traffic is generated on account of the packets avoiding intermediate forwarding. At the same time, there is still significant congestion on local channels due to the lack of ability of packets to traverse non-minimal routes, falling into the same trap as the contiguous-minimal configuration. Coupled with (progressive) adaptive routing, saturation times are effectively minimized on both global and local channels when random placement is in use, as shown in Figure 2d and 2e. Further, in comparison to contiguous allocations, random allocations



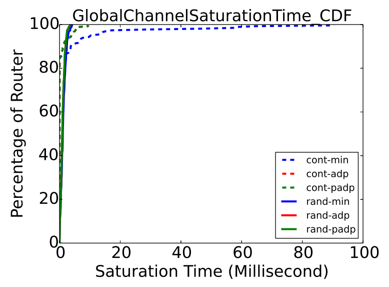
(a) Global Channel Traffic



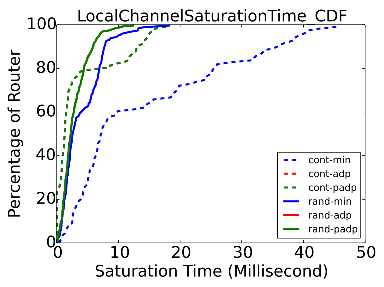
(b) Local Channel Traffic



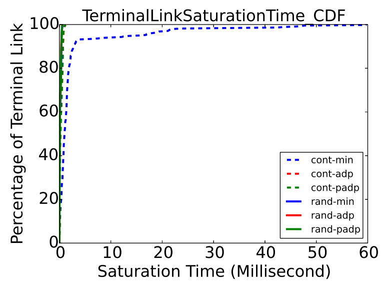
(c) Terminal Link Traffic



(d) Global Channel Saturation Time



(e) Local Channel Saturation Time



(f) Terminal Link Saturation Time

Figure 2: Aggregate traffic and saturation time for Workload I under the configurations listed in Table 1. In these plots, ”adp” and ”padp” exhibit similar behaviors, and their curves are overlapped.

result in a more evenly distributed load on the resulting channels, as expected.

Figures 2c and 2f are presented for the purpose of symmetry, showing the traffic per terminal link as well as the saturation time experienced at each terminal. The terminal traffic distribution corresponds directly to application traffic, as we are using one MPI rank per node (terminal). However, saturation times are different, resulting from the aforementioned network behavior. Particularly, contiguous allocations coupled with minimal routing results in a “long-tail” distribution of saturation time.

4.2. Individual Application Analysis

Now that the system-level view has been analyzed, we turn to evaluate the behavior of each application within Workload I. Figure 3 shows the communication time distribution across application ranks for different placement and routing configurations.

The relative behavior of contiguous allocations is roughly similar in all three applications. Contiguous placement with minimal routing results in poor relative performance across the board compared to the adaptive routing alternatives. Given the analyses in Section 4.1, this is to be expected – the contiguous-minimal configuration results in significant congestion.

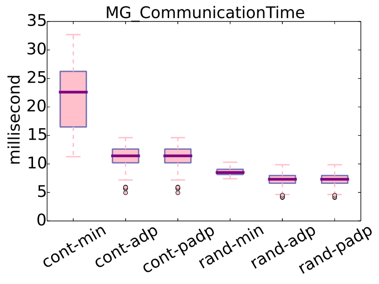
For the MultiGrid and CrystalRouter applications (Figures 3a and 3b, respectively), using random allocation with any routing method results in performance improvements over contiguous allocations, which is largely in agreement with the literature (see Section 7). The high-radix nature of the network topology ensures that the benefits from the resulting load balancing outweigh the costs of extra hops for point-to-point messages.

The AMG application (Figure 3c), however, shows markedly different behavior when using random allocation. Random allocation with minimal routing results in comparable performance to contiguous-adaptive configurations, while using adaptive routing results in significant performance regressions. As this is a counterintuitive result not discussed in other works, we investigate further.

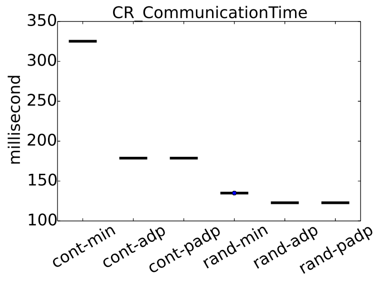
We step back to a network-level system view to identify the culprit behind AMG’s abnormal behavior with random placement. This time, however, we identify the compute nodes that each MPI rank resides on and the routers that are serving each application, and analyze the traffic on a per-application basis. The results of this experimentation are presented in Figure 4. Note that different numbers of routers are used in the contiguous and random allocation configurations, as each router serves multiple terminals.

The system behavior with respect to the CrystalRouter application arguably best matches expectations. Use of contiguous allocations results in a subset of channels with a significant traffic load while a significant portion are unused. Use of random allocations results in a comparatively smoother traffic distribution, with some variation on the margins due to the randomness.

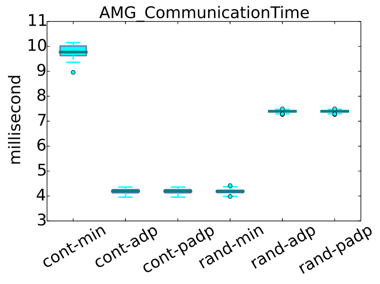
MultiGrid shows roughly similar behavior for contiguous allocations, but different behavior along the local channels. There is a significant variation in the traffic distribution on local channels, even with adaptive routing, which nevertheless has the net effect of reducing the maximal traffic



(a) MultiGrid

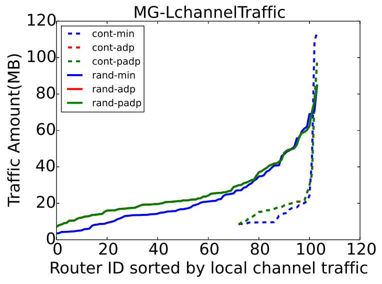


(b) CrystalRouter

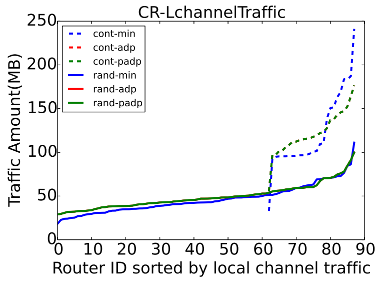


(c) AMG

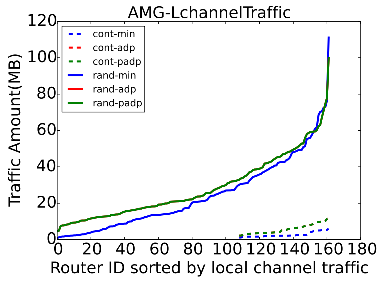
Figure 3: Communication time distribution across application ranks in Workload I.



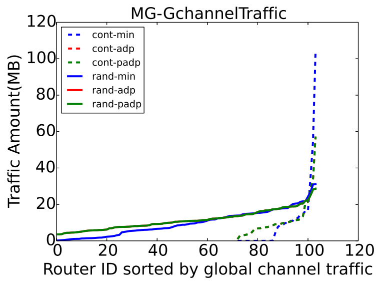
(a) MultiGrid Local Channel Traffic



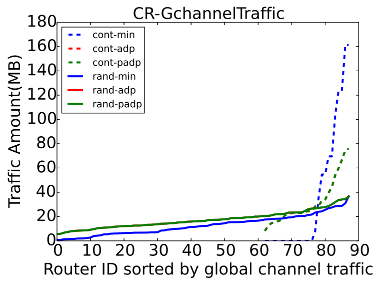
(b) CrystalRouter Local Channel Traffic



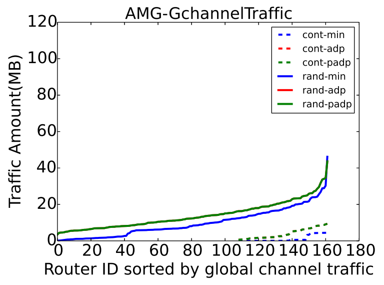
(c) AMG Local Channel Traffic



(d) MultiGrid Global Channel Traffic



(e) CrystalRouter Global Channel Traffic



(f) AMG Global Channel Traffic

Figure 4: Aggregate workload traffic for routers serving specific applications. More routers are involved in serving each application when random placement is in use, compared to contiguous placement. In these plots, ”adp” and ”padp” exhibit similar behaviors, and their curves are overlapped.

load.

AMG shows a similar level of variation to MultiGrid in this case. However, it is the least communication-intensive application of the three by a significant factor. As evidenced by the wide gap between the router traffic in contiguous and random placement configurations, the routers serving the AMG application are being utilized by MultiGrid and CrystalRouter, resulting in AMG traffic contending with traffic of other applications. The net effect, as shown in Figure 3c, is significant slowdowns for AMG. We refer to this phenomenon as AMG being “bullied” by MultiGrid and CrystalRouter.

4.3. Key Observations

In summary, based on the simulations presented in Section 4.1 and 4.2, we make the following observations.

System-level performance is significantly improved when random placement and adaptive routing are in use. Random placement can uniformly distribute MPI ranks of application over the network, and adaptive routing can redirect the traffic from congested routers to other less busy routers. The combination of the two minimizes hot-spots and promotes load-balanced distribution. The resulting increased number of hops per message was not a significant detriment in comparison. This matches what is seen in the literature.

The performance of communication-intensive jobs in the system improves through use of random allocation policies. Both CrystalRouter and MultiGrid, the two most communication-intensive jobs, saw improved distributions of communication performance when moving to a random allocation. Again, this matches what is seen in the literature.

The performance of less communication-intensive jobs *in workload regresses when random placement and adaptive routing are in use.* AMG in Workload I is "bullied" by its concurrently running communication-intensive peers, MultiGrid and CrystalRouter. AMG shares routers and groups with MultiGrid and CrystalRouter under random placement. The traffic from MultiGrid and CrystalRouter is (re)directed to the routers that serve AMG, slowing down AMG's communication. [1]

In contrast, performance consistency of each application is achieved only when contiguous placement and minimal routing are in use. As a corollary to the previous observation, router and group sharing among applications are guaranteed to be prohibited when using contiguous placement with minimal routing (sharing of spare nodes within a group notwithstanding). This renders the "bully" behavior moot, though with the downside of significant performance degradation, so such approaches must be carefully considered.

5. Study of Parallel Workload II

In this section, we use a different experimental configuration to verify and explore the observations made in the previous section. Specifically, we conduct the same sets of experiments through Workload II, which consists of sAMG, MultiGrid and CrystalRouter. The "bully" becomes the "bullied" with the presence of sAMG in Workload II.

5.1. Network Performance Analysis

Figure 5 shows aggregate traffic and saturation times for Workload II, corresponding to Figure 2 in Workload I. Replacing AMG with sAMG results in greater aggregate traffic as well as more saturation than in Workload I, but regardless, similar patterns can be observed. Contiguous placement with minimal routing results in load imbalance with respect to both traffic and saturation. The addition of adaptive routing alleviates these effects to some degree, particularly with respect to global channel usage, while trading off saturation in global channels for saturation in the local channels. Using random placement again shows roughly similar performance characteristics across routing configurations, with adaptive routing helping to balance aggregate load while increasing the aggregate traffic due to the related indirection.

5.2. Individual Application Analysis

We study the performance of each application individually in the same manner as in Section 4.2. Figure 6 shows the communication time distributions of the ranks of the three applications, running concurrently in Workload II. The "bully" in Workload I becomes the "bullied" in Workload II. MultiGrid and CrystalRouter are in this instance the less communication-intensive applications. With random placement and (progressive) adaptive routing, both MultiGrid and CrystalRouter experience prolonged communication time, as shown in Figure 6a, 6b. On the other hand, sAMG (Figure 6c) benefits from those configurations in a similar manner to CrystalRouter in Workload I. Contiguous placement coupled with minimal routing, while preventing the "bully" behavior, results in poor performance for all of the applications except CrystalRouter, which we expect is due to a higher degree of network isolation.

Once again, we look at the network-level system view to scrutinize the traffic through the routers serving each application. The routers serving sAMG have a high volume of traffic on both local and global channels when contiguous placement is in use, as shown in Figures 7c and 7f. As in previous results, the use of random placement alleviates the local congestion by uniformly distributing the traffic of sAMG over the network, getting more routers involved in serving sAMG. In this case, a majority of those less busy routers are also serving MultiGrid and CrystalRouter.

MultiGrid in Workload II is similar to AMG in Workload I when considering resident channel behavior, as shown in Figures 7a and 7d. There is a large gap in traffic volume between the contiguous and random placement approaches, due to other applications (sAMG in particular) utilizing the same links. CrystalRouter in Workload II additionally experiences more load on its channels using random allocation configurations, as shown in Figure 7b and 7b. However, the maximal load under random allocation is closer to that observed in contiguous allocations as compared to MultiGrid.

5.3. Key Observations

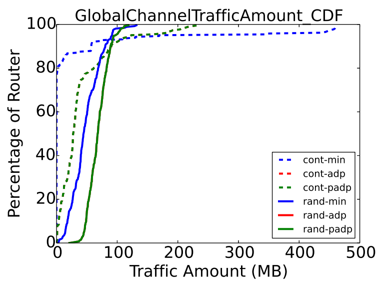
Revisiting the observations in Section 4.3, we find that those observations are held under this separate configuration. System-level performance is still much improved in terms of load-balancing with random placement. sAMG, being far and away the most communication-intensive application in Workload II, benefits greatly from random placement, whereas in Workload I AMG was effectively penalized for being less communication-intensive. CrystalRouter, being the comparatively less communication-intensive application in Workload II, experiences performance regressions in Workload II under random and adaptive policies.

Interestingly, in both Workload I and II, MultiGrid experiences a more subtle performance variatioin than the significant swings in performance observed in the other applications. These behaviors persisted across multiple runs with different random seeds. Additionally, CrystalRouter in Workload II has less drastic changes in maximal load, but still experiences performance regressions. We are continuing to work towards understanding the root causes and implications of this behavior, for which we expect application-specific communication patterns to be an important factor.

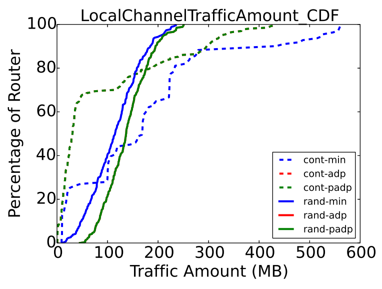
6. Hybrid Job Placement

Based on our experiments with Workloads I and II, the "bully" behavior is exhibited when the dragonfly network is

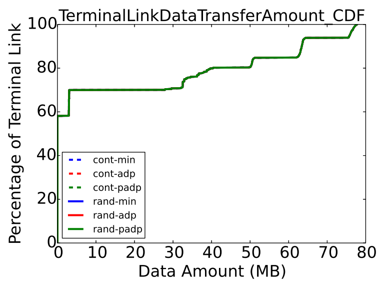
1. We have tried three different congestion "sensing" schemes in the literature for adaptive routing [10]. Although there are some variations between the results, none of the congestion sensing scheme prevent the "bully" behavior.



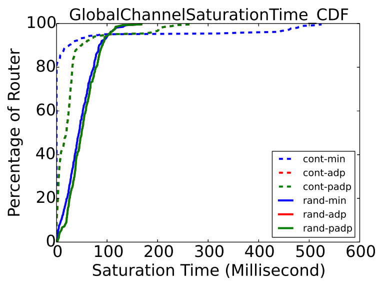
(a) Global Channel Traffic



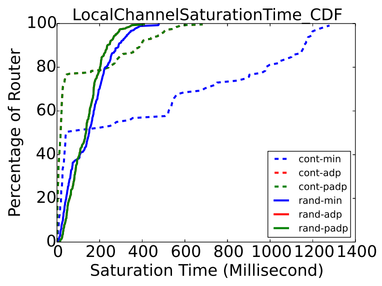
(b) Local Channel Traffic



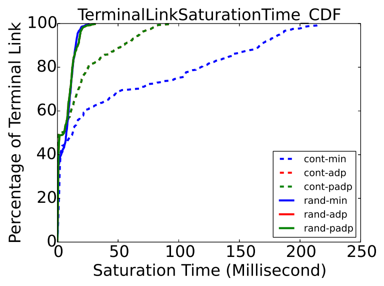
(c) Terminal Link Traffic



(d) Global Channel Saturation Time

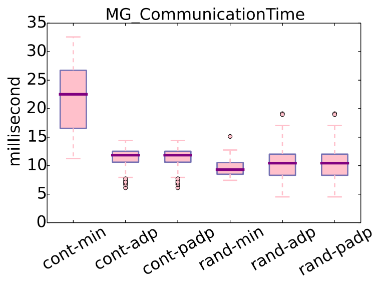


(e) Local Channel Saturation Time

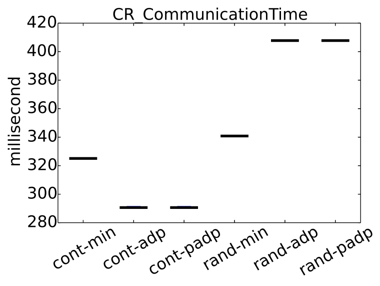


(f) Terminal Link Saturation Time

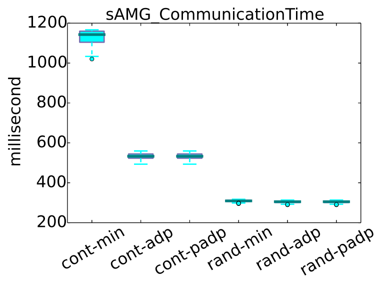
Figure 5: Aggregate traffic and saturation time for Workload II under the configurations listed in Table 1. In these plots, ”adp” and ”padp” exhibit similar behaviors, and their curves are overlapped.



(a) MultiGrid



(b) CrystalRouter



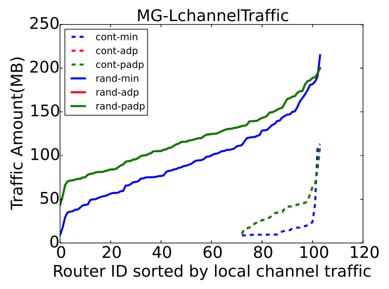
(c) sAMG

Figure 6: Communication time distribution across application ranks in Workload II. The “bully”, sAMG, benefits from random placement and adaptive routing, while the “bullied”, MultiGrid and CrystalRouter, suffer performance degradation.

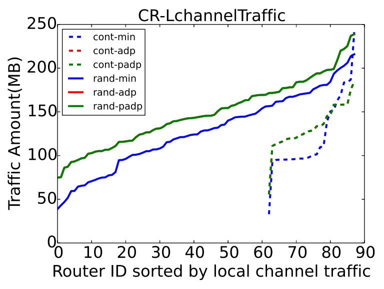
configured with random placement and (progressive) adaptive routing, and there is a large gap between the communication intensity of applications running on the network. As shown through our experimentation, contiguous placement policies give up too much in terms of congestion and load balance, hence being an impractical solution. Further, running each job with a dedicated routing policy is unrealistic, since routing policy is part of system configuration which can not be changed on the fly upon job submission.

As a natural extension of our observations, one question that arises is *whether we can combine the merits of random and contiguous placement policies in which each application receives the performance benefits from system load balancing while avoiding the “bully” behavior.* As an initial exploration of the question, we set up a mock hybrid job placement policy, in which less communication-intensive jobs receive contiguous allocations to avoid the “bully” effect, while the communication-intensive jobs are allocated randomly in order to distribute the communication load. For Workload I, this means AMG gets a contiguous allocation while MultiGrid and CrystalRouter get random allocations. Note that we do not consider challenges inherent in designing an allocation policy for production usage, such as backfilling, reserving large contiguous sets of nodes, determining a metric for communication intensity, etc., preferring a restricted-scope experiment looking at the design space of dragonfly allocation policies in light of our experimental observations.

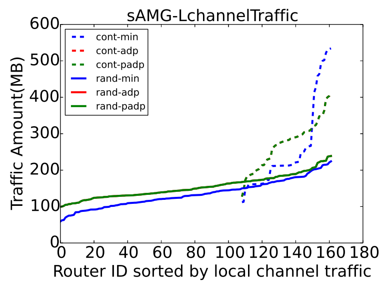
For the purpose of brevity, we only present the communication time distribution of each application under all placement and routing configurations, including the hybrid placement method. These results are presented in Figure 8.



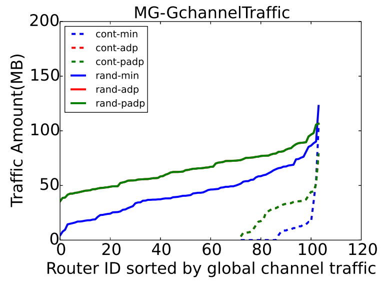
(a) MultiGrid Local Channel Traffic



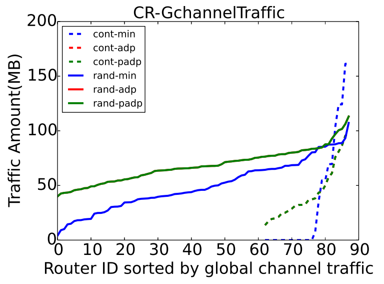
(b) CrystalRouter Local Channel Traffic



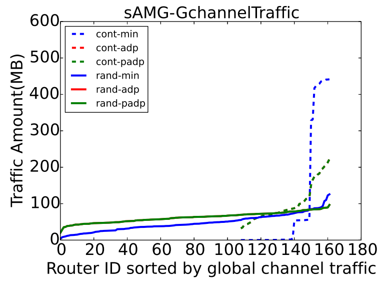
(c) sAMG Local Channel Traffic



(d) MultiGrid Global Channel Traffic

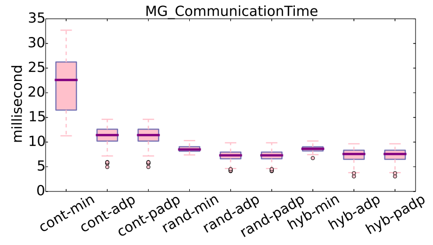


(e) CrystalRouter Global Channel Traffic

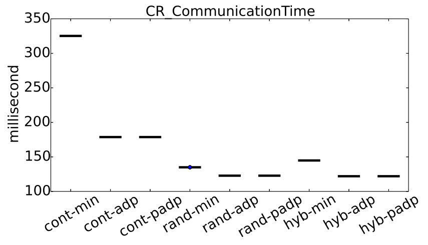


(f) sAMG Global Channel Traffic

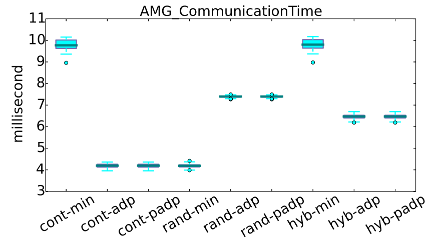
Figure 7: Aggregate workload traffic for routers serving specific applications. More routers are involved in serving each application when random placement is in use, compared to contiguous placement. In these plots, ”adp” and ”padp” exhibit similar behaviors, and their curves are overlapped.



(a) MultiGrid Communication Time



(b) CrystalRouter Communication Time



(c) AMG Communication Time

Figure 8: Application communication time. Workload I is running with all placement and routing configurations. Methods prefixed with “hyb” represent the hybrid allocation approach.

As shown in Figure 8a and 8b, MultiGrid and CrystalRouter exhibit similar performance in both hybrid and random placement, as nodes are being placed randomly in each case. While the performance of AMG under hybrid placement, shown in Figure 8c, still appears to exhibit significant communication interference on account of the other applications as opposed to the best contiguous placement policies, the effects are reduced compared to a random-adaptive policy. We believe this to be a result of more AMG-specific traffic occupying a smaller set of routers/groups, both reducing the probability of traffic entering them through adaptive routing and increasing the relative proportion of link utilization by AMG. Of course, this comes with the costs associated with contiguous allocation, in which AMG’s traffic is less likely to load balance across multiple dragonfly groups.

These initial experiments demonstrate some degree of benefit derived from using a hybrid approach, helping to alleviate the “bully” effect while retaining the performance of communication-intensive applications. However, the behavior is still not ideal in this case – AMG’s communications still experience performance degradation versus the contiguous configurations. Hence, more work in this area is needed to fully understand the intricate relationships between job scheduling and system/application communication behavior to achieve optimal network utilization and application performance in high-radix networks.

7. Related Work

The impact of job placement on system behavior and application performance has been the subject of many studies. We focus on the HPC-centric studies here. Skinner et al. identified significant performance variability due to network contention [27]. They found that performance variability is inevitable on either torus or fat-tree networks when network sharing among concurrently running applications is allowed. Bhatele et al. studied the performance variability of a specific application, p3FD, running on different HPC production systems with torus network topologies [28]. They obtained performance consistency in their application when network resources were allocated compactly and exclusively and wide variability otherwise. Jokanovic et al. studied the impact of job placement to the workload and claimed that the key to reduce performance variability is to avoid network sharing [29].

Recently, several researchers have investigated job placement and routing algorithms on dragonfly networks. Prisacari et al. proposed a communication-matrix-based analytic modeling framework for mapping application workloads onto network topologies [30]. They found that, in the context of dragonfly networks, optimizing for throughput and not workload completion time is often misleading and the notion of system balance cited as a dragonfly design parameter is not always directly applicable to all workloads. Jain et al. conducted a comprehensive analysis of various job placement and routing policies with regard to network link throughput on dragonfly network [3]. Their work is based on an analytical model and synthetic workload. Bhatele et al. used coarse-grain simulation to study the performance of synthetic workloads under the different task mapping and routing policies on two-level direct networks [4]. Mubarak et al. focused on the modeling of large-scale dragonfly networks with parallel event driven simulation. The dragonfly network model for million-node configurations presented in their work strongly scales when going from 1,024 to 65,536 MPI tasks on IBM Blue Gene/P and IBM Blue Gene/Q systems [16]. The dragonfly model used in this paper is from their work.

Our work complements the literature in the following aspects. First, our simulations are driven by real application traces intended to be representative of production-scale application patterns. Second, we holistically examine network behavior at both the overall system level as well as the individual application level, though we do not consider communication-pattern specific application mappings as Prisacari et al. did. Third, with the CODES simulation toolkit and related network models [16], [31], we are able to simulate and examine system and application behavior at a very fine grain, collecting data at the dragonfly link level with packet-level fidelity. We believe these differences allowed us to uncover the "bully" behavior, which to our knowledge is unreported in the literature. However, in a sense, Prisacari et al.'s work suggests these types of behaviors as possibilities deriving from the balance-first design rationale for the dragonfly.

8. Conclusions and Future Work

In this paper, we have conducted extensive studies of system and application behavior using various job placement and routing configurations on a dragonfly network. We took a simulation-based approach, utilizing the CODES simulation toolkit and related models for high-fidelity dragonfly simulation, driving the network with three production-representative scientific application traces. We found that, under the prevailing recommendation of random process placement and adaptive routing, network traffic can be well distributed to achieve a balanced load and strong overall performance, at the cost of impairing jobs with less intensive communication patterns. We denoted this as the "bully" effect. On the other hand, contiguous process placement prevents such effects while exacerbating local congestion, though this can be mitigated through the addition of adaptive routing. Finally, we performed initial experimentation exploring a mock "hybrid" contiguous/random job placement policy. Our preliminary study demonstrates the need of specialized job placement strategy based on application communication characteristics. We believe the findings and methodology presented in this paper provide valuable insights for efficient workload management on dragonfly networks, hence being valuable to the broad HPC community.

In the future, we plan to explore intelligent job placement and routing configurations for diversified workloads running on the dragonfly network. The dragonfly connected system could have dedicated job placement and routing configuration based on the characteristics of its workload. We plan to collect workload traces from a production HPC system, and experiment with the simulated network of the system to explore the dedicated optimal job placement and routing configuration. We envision that effectiveness of such new placement and routing polices can be examined in future on a configurable dragonfly system.

Acknowledgments

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] J. Kim, W. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *Computer Architecture, 2008. ISCA '08. 35th International Symposium on*, June 2008, pp. 77–88.

[2] A. Bhatele, N. Jain, Y. Livnat, V. Pascucci, and P.-T. Bremer, "Analyzing network health and congestion in dragonfly-based supercomputers," in *Proceedings of the IEEE International Parallel & Distributed Processing Symposium*, ser. IPDPS '16 (to appear). IEEE Computer Society, May 2016, lLNL-CONF-678293.

[3] N. Jain, A. Bhatele, X. Ni, N. J. Wright, and L. V. Kale, "Maximizing throughput on a dragonfly network," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 336–347. [Online]. Available: http://dx.doi.org/10.1109/SC.2014.33

[4] A. Bhatele, W. D. Gropp, N. Jain, and L. V. Kale, "Avoiding hot-spots on two-level direct networks," in *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*, Nov 2011, pp. 1–11.

[5] J. Brandt, K. Devine, A. Gentile, and K. Pedretti, "Demonstrating improved application performance using dynamic monitoring and task mapping," in *Cluster Computing (CLUSTER), 2014 IEEE International Conference on*. IEEE, 2014, pp. 408–415.

[6] Department of Energy, "Exascale Initiative," Accessed April 2, 2016, available online http://www.exascaleinitiative.org/design-forward.

[7] M. Mubarak, C. D. Carothers, R. B. Ross, and P. Carns, "Enabling parallel simulation of large-scale hpc network systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. PP, no. 99, pp. 1–1, 2016.

[8] J. Kim, W. Dally, S. Scott, and D. Abts, "Cost-efficient dragonfly topology for large-scale systems," *IEEE Micro*, vol. 29, no. 1, pp. 33–40, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1109/MM.2009.5

[9] N. Jiang, J. Kim, and W. J. Dally, "Indirect adaptive routing on large scale interconnection networks," in *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ser. ISCA '09. New York, NY, USA: ACM, 2009, pp. 220–231. [Online]. Available: http://doi.acm.org/10.1145/1555754.1555783

[10] J. Won, G. Kim, J. Kim, T. Jiang, M. Parker, and S. Scott, "Overcoming far-end congestion in large-scale networks," in *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*. IEEE, 2015, pp. 415–427.

[11] D. Tsafrir, Y. Etsion, and D. G. Feitelson, "Backfilling using system-generated predictions rather than user runtime estimates," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 6, pp. 789–803, June 2007.

[12] 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*, May 2015, pp. 439–448.

[13] A. Jokanovic, J. C. 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*, May 2015, pp. 449–459.

[14] C. D. Carothers, D. Bauer, and S. Pearce, "ROSS: A high-performance, low-memory, modular Time Warp system," *Journal of Parallel and Distributed Computing*, vol. 62, no. 11, pp. 1648–1669, Nov. 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731502000047

[15] 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*, ser. SIGSIM PADS '13. New York, NY, USA: ACM, 2013, pp. 327–336. [Online]. Available: http://doi.acm.org/10.1145/2486092.2486134

[16] M. Mubarak, C. D. 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:*, Nov 2012, pp. 366–376.

[17] M. Mubarak, C. D. Carothers, R. B. Ross, and P. Carns, "A case study in using massively parallel simulation for extreme-scale torus network codesign," in *Proceedings of the 2Nd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, ser. SIGSIM PADS '14. New York, NY, USA: ACM, 2014, pp. 27–38. [Online]. Available: http://doi.acm.org/10.1145/2601381.2601383

[18] N. Liu and C. D. Carothers, "Modeling billion-node torus networks using massively parallel discrete-event simulation," in *Proceedings of the 2011 IEEE Workshop on Principles of Advanced and Distributed Simulation*, ser. PADS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1–8. [Online]. Available: http://dx.doi.org/10.1109/PADS.2011.5936761

[19] N. Wolfe, C. Carothers, M. Mubarak, R. Ross, and P. Carns, "Modeling a million-node slim fly network using parallel discrete-event simulation," in *Proceedings of the 4Nd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, ser. SIGSIM PADS '16.

[20] Sandia National Labs, "SST DUMPI trace library," Accessed April 2, 2016, available online http://sst.sandia.gov/usingdumpi.html.

[21] N. Jiang, J. Balfour, D. U. Becker, B. Towles, W. J. Dally, G. Michelogiannakis, and J. Kim, "A detailed and flexible cycle-accurate network-on-chip simulator," in *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*, April 2013, pp. 86–96.

[22] Department of Energy, "Characterization of the DOE Mini-apps," Accessed April 2, 2016, available online http://portal.nersc.gov/project/CAL/designforward.htm.

[23] V. E. Henson and U. M. Yang, "Boomeramg: A parallel algebraic multigrid solver and preconditioner," *Applied Numerical Mathematics*, vol. 41, no. 1, pp. 155 – 177, 2002, developments and Trends in Iterative Methods for Large Systems of Equations - in memorium Rudiger Weiss. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168927401001155

[24] J. Bell, A. Almgren, V. Beckner, M. Day, M. Lijewski, A. Nonaka, and W. Zhang, "Boxlib users guide," Technical Report, CCSE, Lawrence Berkeley National Laboratory. Available at: https://ccse. lbl. gov/BoxLib/BoxLibUsersGuide. pdf, Tech. Rep., 2012.

[25] P. Fischer, A. Obabko, E. Merzari, and O. Marin, "Nek5000: Computational fluid dynamics code," Accessed October 14, 2015, available online http://nek5000.mcs.anl.gov.

[26] G. Faanes, A. Bataineh, D. Roweth, T. Court, E. Froese, B. Alverson, T. Johnson, J. Kopnick, M. Higgins, and J. Reinhard, "Cray cascade: A scalable hpc system based on a dragonfly network," in *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for*, Nov 2012, pp. 1–9.

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

[28] 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*, ser. SC '13. New York, NY, USA: ACM, 2013, pp. 41:1–41:12. [Online]. Available: http://doi.acm.org/10.1145/2503210.2503247

[29] 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*, May 2015, pp. 449–459.

[30] 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*, ser. HPDC '14. New York, NY, USA: ACM, 2014, pp. 129–140. [Online]. Available: http://doi.acm.org/10.1145/2600212.2600225

[31] 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.