2018 IEEE International Parallel and Distributed Processing Symposium

# Trade-Off Study of Localizing Communication and Balancing Network Traffic on a Dragonfly System

Xin Wang,* Misbah Mubarak,† Xu Yang,* Robert B. Ross,† Zhiling Lan*

*Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616, USA
{xwang149, xyang56}@hawk.iit.edu, lan@iit.edu

† Mathematics and Computer Science Division Argonne National Laboratory, Argonne, IL 60439, USA
{mmubarak, rross}@mcs.anl.gov

***Abstract*—Dragonfly networks are being widely adopted in high-performance computing systems. On these networks, however, interference caused by resource sharing can lead to significant network congestion and performance variability. We present a comparative analysis exploring the trade-off between localizing communication and balancing network traffic. We conduct trace-based simulations for applications with different communication patterns, using multiple job placement policies and routing mechanisms. We perform an in-depth performance analysis on representative applications individually and show that different applications have distinct preferences regarding localized communication and balanced network traffic. We further demonstrate the effect of external network interference by introducing background traffic and show that localized communication can help reduce the application performance variation caused by network sharing.**

***Keywords*—High-performance computing, dragonfly networks, interference, job placement**

## I. Introduction

High-bandwidth, low-latency interconnect networks play a key role in the design of modern high-performance computing systems. Dragonfly topology is a cost-efficient solution for large-scale interconnection networks [1]. Such a network has been deployed in the 29.1-petaflop Cori at the National Energy Research Scientific Computing Center [2] and the 3,624-node Theta system at the Argonne Leadership Computing Facility (ALCF) which serves as a stepping stone to the ALCF's next leadership-class supercomputer, Aurora [3] [4].

Dragonfly networks have a hierarchical, high-radix, low-diameter topology that can lower the network cost, provide high network bandwidth, and reduce packet latency. However, dragonfly systems are susceptible to *performance variability* due to network resource sharing. A recent study showed that the run-to-run variability of the execution time is frequently 15% or greater and can be up to 100% [5]. Performance variability causes significant issues for both applications and the system. If the required runtime of an application cannot be accurately estimated, the batch scheduler is likely to make poor scheduling decisions that lead to system under-utilization [6] [7].

Recent studies have identified that communication interference due to network sharing is a dominant cause of performance variability [5] [8]. On a shared network, different job placement policies lead to different traffic distributions. Contiguous job placement policy achieves *localized communication* by assigning adjacent compute nodes to the same job. Random job placement policy, on the other hand, achieves *balanced network traffic* by placing application processes sparsely across the network to uniformly distribute the message load. Localized communication and balanced network traffic have opposite advantages and drawbacks. Localizing communication reduces the number of hops for message transfers at the cost of potential network congestion, while balancing network traffic reduces potential local congestion at the cost of increased message transfer hops. Some studies emphasize localized communication [9], while others emphasize the importance of system-wide network traffic balance [10] [11]. However, the trade-off between the two has not been studied.

In this study, we conduct *an in-depth trade-off analysis of localizing communication versus balancing network traffic on dragonfly systems*. The trade-off analysis depends on a variety of factors, many of which, such as job placement and routing policies, are part of the system configuration, which is impossible for users to make changes to at will. To provide a comprehensive analysis of various factors, we conduct high-fidelity simulation using the Co-Design of Multi-layer Exascale Storage Architectures (CODES) toolkit, which has been enhanced and validated to simulate the dragonfly interconnect network on Theta [12] [13] [14]. The trade-off analysis also relies on application communication characteristics such that distinct communication patterns are subject to different degrees of communication interference. Hence, for this study we choose three representative applications with distinct communication characteristics from the DOE Design Forward Project. Our analysis is pursued in three steps:

- First, we perform an *application study under various job placement and routing policies* to identify the trade-off between localizing communication and balancing network traffic. To eliminate the interference from other jobs sharing the network, we conduct trace-based simulations of each application independently on the system. Our analysis focuses on communication performance, network traffic, and link behavior.
- Second, we perform an *application sensitivity study* by varying the communication intensity of the applications. Application behavior may change with different communication intensity because the underlying system has limited network bandwidth and buffers. The goal of tuning the message load of applications is to examine

1530-2075/18/$31.00 ©2018 IEEE
DOI 10.1109/IPDPS.2018.00120

how communication intensity affects the communication performance under various configurations.

- Third, we explore the impact on these applications from external network traffic. In a production environment, system resources are shared by multiple jobs; hence interference of external network traffic is inevitable. We simulate a multijob computing environment by adding background traffic. In this study, we explore the impacts of both uniform random background traffic and bursty background traffic.

We extend our previous work in [15] with following aspects: we adopt the interference study according to the dragonfly topology being used by Cray; we further investigated the application characteristics, including communication frequency and load; we also analyze the performance variability by simulating background communication traffic.

The rest of the paper is organized as follows. Section II introduces the background and system configuration. Section III discusses the three representative applications, job placement and routing policies, the simulation framework, and the experiment configurations. Section IV presents the results of our in-depth analysis of the trade-off between localizing communication and balancing network traffic. Section V discusses related work. Section VI summarizes our conclusions and briefly discusses future work.

## II. System Configuration

Theta is a 9.65-Petaflop production system at ALCF [3]. It is designed in collaboration with Intel and Cray and has the second-generation Intel Xeon Phi processors and a high-radix dragonfly network topology. Theta is equipped with 3,624 nodes, each containing a 64 core processor with 16 GB of high-bandwidth in-package memory (MCDRAM), 192 GB of DDR4 RAM, and a 128 GB SSD.

Figure 1 shows the system configuration studied in this work. The machine has 9 groups, each with 96 Aries routers arranged in a $6 \times 16$ grid. Each row and column of the grid is connected all-to-all by local links (green for rows and black for columns in the figure). Each router is also connected to routers in other groups via global links (blue in the figure). Four compute nodes are attached to each router via terminal links. In particular, each row of 16 routers forms a *chassis*, and 3 such chassis form a *cabinet* [16] [17].

In this work, we use CODES, which enables packet-level, high-fidelity network simulation [18] [19] [20]. CODES has been recently enhanced to support modern dragonfly configurations such as the one deployed in Theta. The enhanced CODES version has been validated against the Theta system with ping-pong and bisection pairing benchmark tests [14]. The validation results show that performance differences between CODES simulations and real experiments are less than 8%; hence CODES is a highly accurate solution for our study.

In our study, we set the parameters for CODES simulation as recorded on the Theta configuration. Specifically, each router connects to 4 compute nodes via terminal links with bidirectional bandwidth of 16 GiB/s, connects routers within the same group via local links with 5.25 GiB/s bandwidth, and connects routers in other group via global links with 4.69 GiB/s bandwidth. The buffers for the compute node virtual channel, local virtual channel, and global virtual channel are set to be 8 KiB, 8 KiB and 16 KiB respectively. The simulation currently disregards compute time since the emphasis is on communication time.

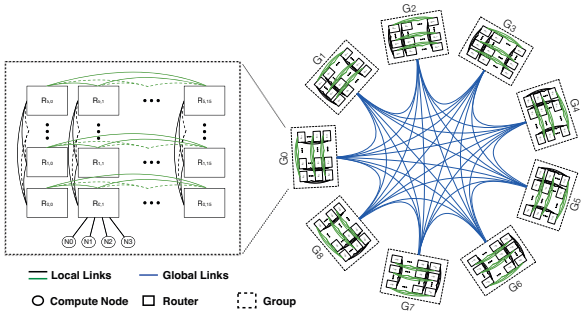



Fig. 1: Overview of the dragonfly system configuration

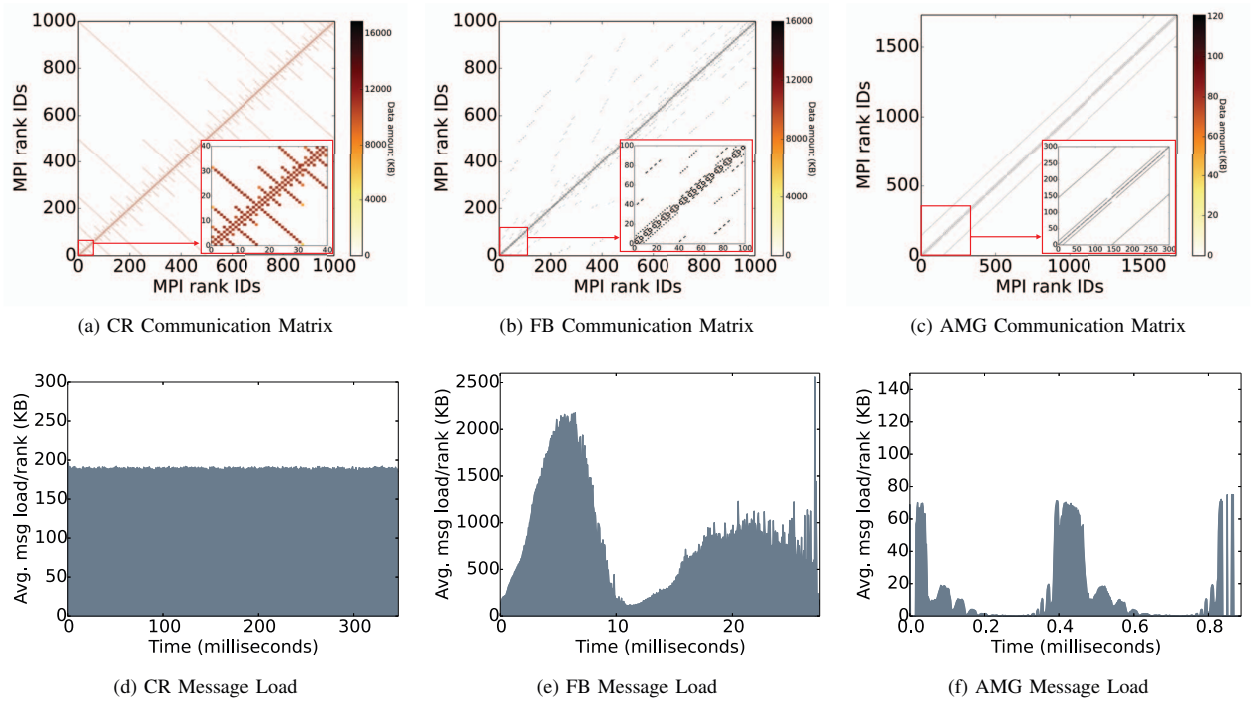
## III. Methodology

Our analysis consists of three parts. The *application study* examines individual application behavior under various job placement and routing; the *sensitivity analysis of communication intensity* investigates the effect of communication intensity on an application; and the *external traffic impact* study explores the impact of background traffic on application performance.

### A. Parallel Applications

We choose three representative parallel applications from the DOE Design Forward Project [21]: 1,000-node crystal router (CR) miniapp, 1,000-node fill boundary (FB) miniapp, and 1,728-node algebraic multigrid (AMG) Solver. The communication traces are collected by using the open source DUMPI toolkit from SST project [22]. Since our focus is on interconnect performance, a mapping of one MPI rank per node is used in all the experiments. From the application traces, we capture the *average message load per rank* by tracking the message size in bytes of the data transfer operations for each MPI rank. Average message load per rank measures the communication intensity of an application. The computation delay in the traces is ignored since we do not count computation time in our simulation.

***CR***: The crystal router miniapp is an extracted communication kernel of the full Nek5000 code, which is a computational fluid dynamics (CFD) application developed at Argonne National Laboratory [23]. It demonstrates a many-to-many communication pattern through a scalable multistage communication process, but a substantial portion of the communication occurs in small neighborhoods of MPI ranks. During execution, it has relatively constant message load at around 190 KB.




(a) CR Communication Matrix (b) FB Communication Matrix (c) AMG Communication Matrix

(d) CR Message Load (e) FB Message Load (f) AMG Message Load

Fig. 2: Communication matrix (top) and message load per rank (bottom) for selected applications. In communication matrix plot, the label of both the x- and y-axes are the index of MPI ranks; the legend bar on the right indicates the message transfer amount between ranks. (CR = Crystal Router, FB = Fill Boundary, AMG = Algebraic MultiGrid)

***FB***: The fill boundary miniapp fills periodic domain boundary and neighboring grid ghost cells in BoxLib, a software framework for massively parallel block-structured adaptive mesh refinement codes for solving time-dependent partial differential equations [24]. Its dominant communication pattern is a 3D block domain decomposition. It has intensive communication between neighbors as well as many-to-many communication across the set of MPI ranks. During execution, it continuously sends messages, the amount of which fluctuates strongly, ranging from 100 KB to 2560 KB.

***AMG***: The algebraic multiGrid solver is a parallel algebraic multigrid solver to solve large problems posed on unstructured grids. It has been derived directly from the BoomerAMG solver developed at Lawrence Livermore National Laboratory [25]. The dominant communication pattern is regional communication with decreasing message size. Each rank has intensive communication with up to six neighbors, depending on rank boundaries. During execution, its message load has three short-duration surges, the peak amount reaching 75 KB. The message load is relatively small compared with that of the other two applications.

Figures 2(a)-(c) show the application communication matrix for each application. Figures 2(d)-(f) show the message load per rank over time for each application.

### B. Job Placement Policy

In this study, we investigate five job placement policies.

*Contiguous Placement*. Each job is assigned to available nodes consecutively. This method guarantees minimum number of routers to serve each application and tends to place application processes into the same group; hence it helps preserve spatial locality but increases the chances of local link congestion.

*Random-Cabinet Placement*. This scheme reduces the level of contiguity by allocating a random selection of cabinets to a job. The nodes within a cabinet are assigned contiguously, but the cabinets are randomly arranged throughout the network.

*Random-Chassis Placement*. Each job receives a random selection of chassis. Similar to random-cabinet placement, the nodes within a chassis are assigned contiguously, but the chassis are randomly arranged.

*Random-Router Placement*. Each job is assigned with a random selection of routers. The nodes attached to a router are assigned consecutively, but the routers are ordered randomly. This scheme helps restrict the communication between nearby nodes leaving the router.

*Random-Node Placement*. Each job receives a random selection of available nodes. An application can be randomly allocated to different routers in different groups. This approach can help distribute the communication load evenly across the network, but it will increment the number of hops that packets should traverse to reach the destination.

### C. Routing Mechanism

For the routing policies, we consider minimal routing and adaptive routing.

*Minimal Routing*. In this policy, a packet takes the minimal path from the source to the destination. Within a group, a minimal route will traverse the source router, an intermediate router if source and destination routers do not share the same row or column, and the destination router. Across groups, a minimal route will take a global link directly connected to the group having the destination node. Minimal routing can guarantee the minimum hops a packet takes. With specific workloads, however, it can result in congestion along the minimal paths because it has no mechanism to sense congestion.

*Adaptive Routing*. In this policy, the path taken by a packet will be chosen based on congestion situation from up to four possible randomly selected routes, two minimal and two nonminimal. A nonminimal path is chosen by randomly selecting an intermediate router from the network. The packet is then routed minimally from the source to the randomly selected intermediate router and then minimally to the destination [16]. Adaptive routing is designed to avoid hotspots and to balance network traffic.

Table I summarizes the configurations explored in this study.

TABLE I: Nomenclature of Different Placement and Routing Configurations

| Placement Policy | Routing Mechanism | |
|---|---|---|
| | Minimal Routing | Adaptive Routing |
| Contiguous | $cont-min$ | $cont-adp$ |
| Random-cabinet | $cab-min$ | $cab-adp$ |
| Random-chassis | $chas-min$ | $chas-adp$ |
| Random-router | $rotr-min$ | $rotr-adp$ |
| Random-node | $rand-min$ | $rand-adp$ |

### D. Three-Step Trade-Off Analysis

Our trade-off analysis is pursued in three steps.

First, we investigate the application performance under a variety of combinations of job placement strategies and network routing mechanisms. Contiguous placement tends to reserve the communication locality, random-node placement tends to balance the overall network traffic, and the other three placement policies tend to keep both. Therefore, the relationship between localized communication and balanced network traffic can be revealed by analyzing the application performance for various job placement policies and routing mechanisms. To this end, we conduct simulations for the three representative applications independently, to eliminate interference from multiple jobs sharing the network.

Second, we modify the communication intensity of the three applications and perform a sensitivity study with configurable message load. The communication behavior of applications may change with different communication intensity since the network bandwidth and buffers are limited. The purpose of tuning the message load of applications is to find out how communication intensity affects the communication performance under various configurations.

Third, we analyze the behavior of each application under external network interference by introducing background traffic on the network. In a production environment, system resources are often shared by multiple jobs; hence, interference of external network traffic is inevitable. In this experiment, we introduce synthetic background traffic to simulate the multijob workload environment. We consider uniform random background traffic and bursty background traffic.

### E. Evaluation Metrics

The following metrics are used in our analysis.

- *Communication Time*: The communication time of each MPI rank is measured as the time spent in completing all its message exchanging operations.
- *Average Hops*: Each simulated MPI rank collects the average number of intermediate routers traversed by a packet from the source to destination compute nodes. Average hops is used to measure the message locality of a packet.
- *Network Traffic*: The traffic refers to the amount of data in bytes going through each router. We analyze the traffic of the local and global channels of each router.
- *Link Saturation Time*: The saturation time is defined as the total amount of time during which a link has used up all its buffers. We categorize the links into local links and global links.

## IV. Experimental Analysis

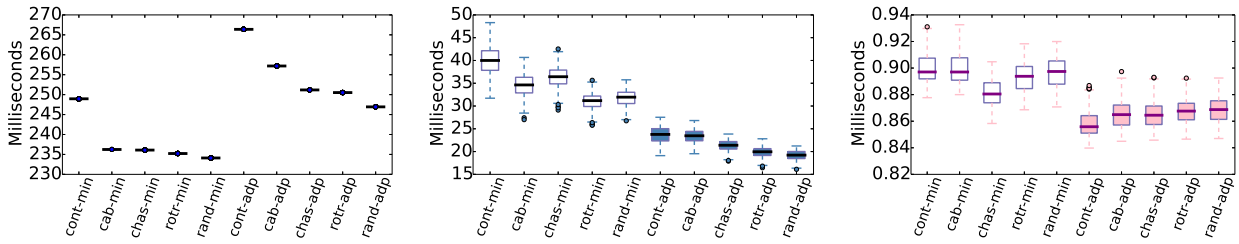
### A. Application Study

In this set of experiments, we examine application communication performance, network traffic, and link behavior under various job placement and routing policies, with the objective of exploring the trade-off between localizing communication and balancing network traffic.

Figure 3 compares the communication time distributions for each application with different placement and routing combinations. The distributions of communication time are shown as box plots; each box represents five data points, minimum, first quartile, median, third quartile, and maximum, from bottom to top. We notice that FB and AMG prefer adaptive routing, while CR favors minimal routing. In terms of placement policy, CR and FB perform better with random placement, while AMG achieves slightly better performance with contiguous placement.

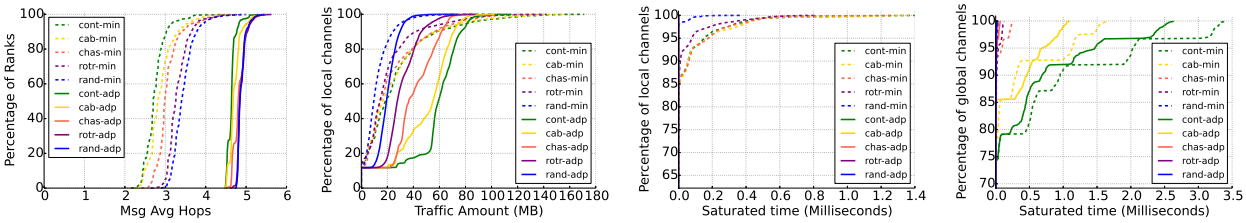
In general, the total time of sending a message is divided into two parts: the transfer time and the waiting time. A message spends less time on transferring if it traverses fewer hops. A message spends less time on waiting if the network links along its path are not saturated.

For **CR**, random-node placement with minimal routing combination results in the shortest communication time. Figure 4 shows other network metrics for CR. With contiguous placement, application ranks reside in a small group of routers,



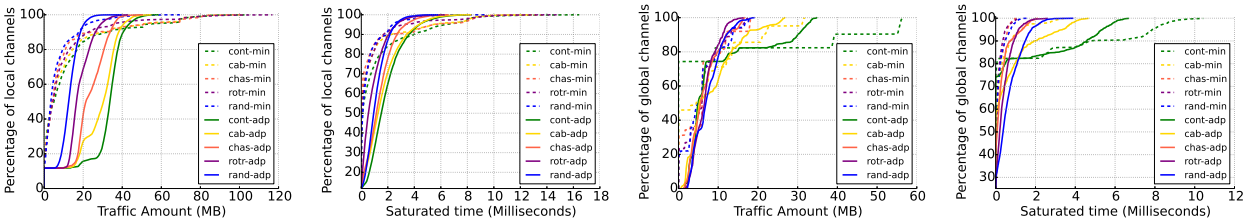

(a) CR Communication Time (b) FB Communication Time (c) AMG Communication Time

Fig. 3: Application communication times under different placement and routing configurations.




(a) Average Hops (b) Local Channel Traffic (c) Local Link Saturation (d) Global Link Saturation

Fig. 4: Average hops, network traffic, and link saturation time for CR.



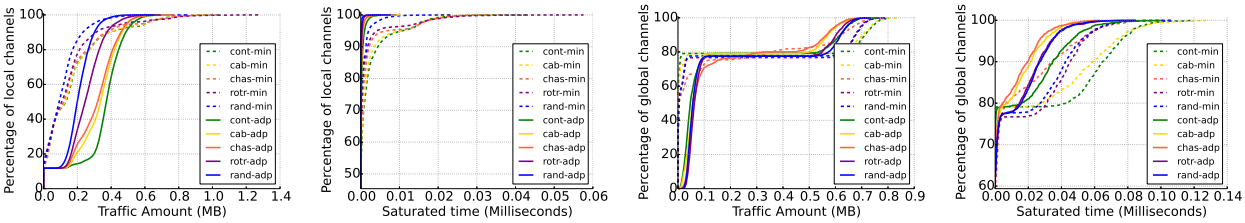

(a) Local Channel Traffic (b) Local Link Saturation (c) Global Channel Traffic (d) Global Link Saturation

Fig. 5: Network traffic and link saturation time for FB.

resulting in fewer hops as messages traverse to their destinations. As shown in Figure 4(a), contiguous placement has fewer average hops than does random-node placement, and minimal routing has fewer average hops than does adaptive routing. As shown in Figure 4(b)-(d), however, contiguous placement suffers large local link saturation time because the majority of traffic is confined within a small group of routers. Minimal routing will make the situation even worse because it is unable to avoid the saturated links by routing to nonminimal paths. Adaptive routing, with the ability to avoid hotspots, helps reduce saturation noticeably on local links, at the cost of increased message transfer time due to nonminimal path. On the contrary, random-node placement evenly distributes the ranks over the network, resulting in balanced message load and hence reducing the saturation time on the links. Thus with fewer average hops, minimal routing combined with random placement results in the shortest communication time.

For **FB**, contiguous-minimal configuration raises communication time significantly. Figure 5 illustrates the network traffic and link saturation status of FB. In Figures 5(a) and 5(c), $cont-min$ suffers from large amount of traffic clustered on a few link channels; $cont-adp$ reduces the local channel traffic by directing traffic to other links, resulting in balanced local and global channel traffic; both $rand-min$ and $rand-adp$ further balance channel traffic on both local and global channels, as shown in Figures 5(b) and 5(d). Therefore, FB achieves its best performance under random-node and adaptive configuration, since it greatly balances the network traffic and hence reduces link saturation significantly.

For **AMG**, contiguous placement with adaptive routing outperforms other configurations. Figure 6 shows the network traffic and link saturation time of AMG. In Figures 6(a) and 6(c), we observe similar behavior to that of FB. With $cont-min$, a small number of channels having a large amount of traffic. Consequently, $cont-min$ brings a longer link saturation time, as shown in Figures 6(b) and 6(d). On the




(a) Local Channel Traffic (b) Local Link Saturation (c) Global Channel Traffic (d) Global Link Saturation

Fig. 6: Network traffic and link saturation time for AMG.

other hand, $rand - adp$ evenly distributes the traffic among local and global channels, resulting in reduced local and global link saturation. However, balanced traffic load does not help with the message communication time; as we can see from the results presented in Figure 3(c), $cont - adp$ performs slightly better than $rand - min$. In Figures 6(a) and 6(c), $rand-adp$ reduces local channel traffic by 0.2 MB, but it does not noticeably reduce global channel traffic compared with $cont - adp$. As a result, $rand - adp$ does not improve much of the link saturation compared with $cont - adp$. In contrast, $cont-adp$ provides competitive results on channel traffic and link saturation, and it achieves fewer average hops. Therefore, contiguous placement combined with minimal routing results in the shortest communication time for AMG.

Based on these results, we have the following key findings.

- Localized communication reduces the number of hops, resulting in less message transfer time at the cost of potential local link saturation.
- Balanced network traffic is achieved by distributing message load across all channels, resulting in less link saturation at the cost of higher message transfer time with an extra number of hops.
- Applications that constantly exchange messages, such as CR and FB, benefit from balanced network traffic. With random-node placement, CR and FB can gain up to 8% and 24.4% improvement, respectively, in terms of communication time compared with contiguous placement.
- Applications that do not exchange messages often, such as AMG, benefit from localized communication. With contiguous placement, AMG has 2.3% improvement in communication time compared with that of random-node placement.

### B. Sensitivity Analysis of Communication Intensity

From our analysis, we find that FB and CR achieve better performance with random-node placement, while AMG prefers contiguous placement. Next, we explore how communication intensity affects application communication performance and network traffic. Specifically, we perform a sensitivity study of application performance, network traffic, and link behavior by varying application message loads.

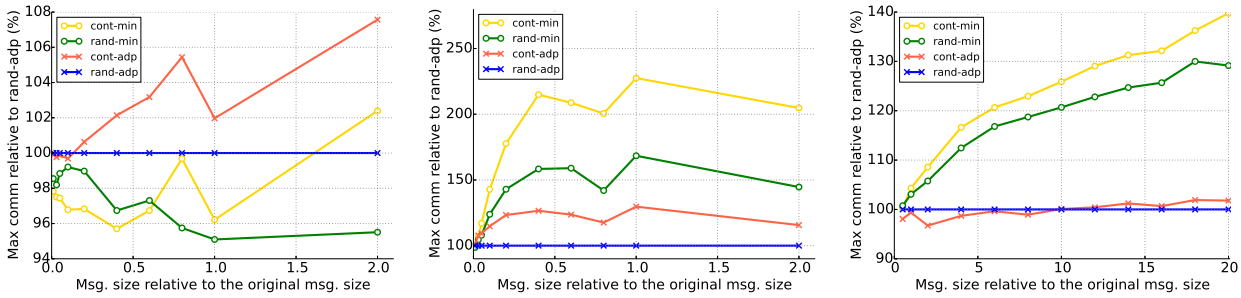
Since FB and CR have intensive communication, we wonder how the performance changes with lower communication intensity. Thus we change the message size of each message transfer operation of FB and CR to range from 1% to twice the original message size. Similarly for AMG, as we are interested in the heavy communication intensity case, we vary the message size of AMG to range from 50% to 20 times the original message size. In each configuration, we consider only contiguous and random-node placement coupled with minimal and adaptive routing, because these four combinations demonstrate the extreme cases of localized communication and balanced network traffic.

In Figure 7, we show the maximum communication time among all ranks of the three representatives applications. We use $rand - adp$ as a baseline. The y-axis represents the relative maximum communication time in percentage across application ranks, and the x-axis shows different message load settings.

Figure 7(a) presents the communication performance for various settings of message load of CR. With adaptive routing, when the message load is very small (less than 10% of the original message size), contiguous placement gains less than 0.5% improvement compared with random-node placement in terms of maximum communication time; when the message load increases (larger than 30% of the original), random-node placement outperforms contiguous placement by up to 7.5%. Similar results can be found with minimal routing. In general, CR benefits from random-node placement that balances the network traffic. In this case, minimal routing works better than adaptive routing by avoiding extra message transfer hops.

Figure 7(b) shows the maximum relative communication time for FB. We can see that random-node placement coupled with adaptive routing always gives the best communication performance with increased communication intensity, whereas the contiguous-minimal combination tends to have prolonged communication time as the message load increases. Thus, even with a light message load (1% of the original), FB benefits from random-node placement that balances the network traffic.

Figure 7(c) shows the communication performance for AMG. As the message load increases, AMG has an intensive communication load between nearby nodes. Minimal routing performs badly due to inability to traverse nonminimal paths, while adaptive routing achieves better performance with the ability to avoid local congestion. When the message load is small, contiguous placement outperforms random-node




(a) CR (b) FB (c) AMG

Fig. 7: Communication Performance with various message sizes. X-axis shows the message load relative to the original message size. Y-axis shows the maximum communication time in percentage relative to $rand - adp$.

placement with up to 3% improvement in communication performance; when the message load is large (more than 10 times that of the original), random-node placement tends to be slightly better than contiguous placement.

In summary, we make the following conclusions.

- Applications with frequent communication of a steady load, such as CR, profit from random placement for balancing the network traffic, and minimal routing for avoiding extra hops. Contiguous placement and minimal routing have a slight advantage in the case of low communication intensity.
- Applications with frequent communication of fluctuant load, such as FB, benefit from balanced network traffic. The communication performance under random-node placement and adaptive routing is always the best for all message load settings.
- Applications with bursty communication behavior, such as AMG, benefit from localized communication in the case of low communication intensity, or benefit from balanced network traffic in the case of high communication intensity.

### C. External Traffic Impact

So far, we have analyzed the trade-off between localizing the communication and balancing the network traffic for each application individually. In a production system with multijob workloads running at the same time, however, communication performance of applications suffers great variation due to network sharing [5]. Therefore, in this set of experiments, we explore the impact of external traffic on application performance.

To simulate a multijob workload environment, we create a synthetic job that occupies all the nodes in the system that are not assigned to the target application. During the execution of the target application, all the ranks of the synthetic job repeatedly issue messages at a certain interval. In this study, we investigate two types of background traffic: uniform random and bursty. In the uniform random pattern, each node of the synthetic job sends messages to a random destination, creating a balanced, external network traffic in the system. In the bursty pattern, each node of the synthetic job sends huge messages to all other nodes at a predefined interval, creating a bursty, external network traffic in the system.

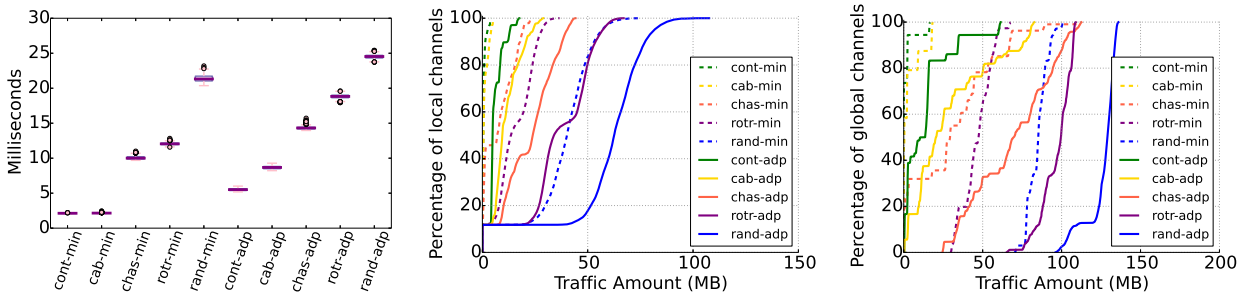
For both uniform random and bursty traffic, we specify the message size and time interval between consecutive messages to control the background traffic load in the system. Table II shows the peak background traffic load used for the target applications. The peak load is calculated as the total message load among all the ranks at a specific time interval. For each target application, we set a very small time interval (0.002 - 1 ms) for the uniform random background pattern, while setting a large time interval (0.1 - 60 ms) for the bursty background pattern. Figures 8-10 present the results for the three representative applications. The channel traffic plots present the traffic distribution on local and global channels of the routers that serve the nodes assigned to the target application.

TABLE II: Peak Background Traffic Load on the Network

| Application | Uniform Random (MB) | Bursty (GB) |
|---|---|---|
| CR | 38.38 | 92.00 |
| FB | 38.38 | 5.75 |
| AMG | 27.00 | 2.85 |

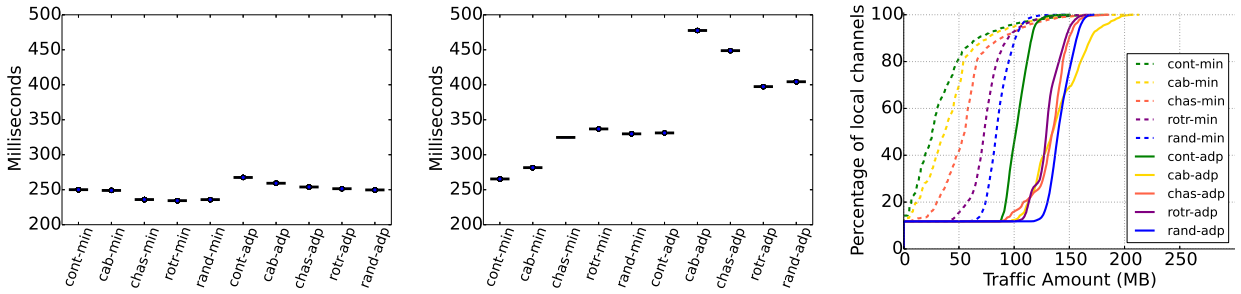
For **AMG**, Figure 8(a) shows the communication distribution with different job placement and routing. We observe that $cont - min$ and $cab - min$ achieve less communication time among all the placement and routing combinations under uniform random background traffic. The communication time is around 2.1 ms, a 125% performance degradation compare with the case without background traffic. We also observe that $rand - adp$ has the worst communication time, which is 30 times larger than the case without background traffic.

Figures 8(b) and 8(c) show that a huge amount of traffic goes through both the local and global channels of routers that serve AMG except for the $cont - min$ and $cab - min$ cases. AMG is not communication intensive; hence, with uniform random background traffic that contiguously sends messages, AMG suffers severe slowdown because the background traffic is directed to AMG's routers by using an adaptive routing



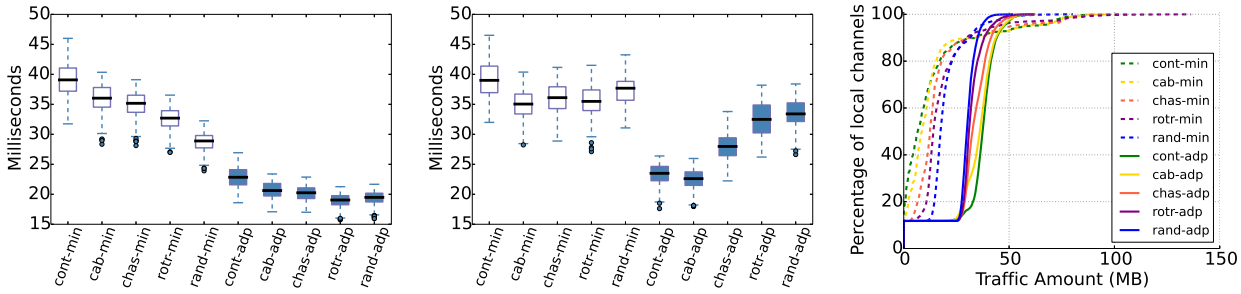

(a) Communication Time (b) Local Channel Traffic (c) Global Channel Traffic

Fig. 8: Communication time and channel traffic of AMG with uniform random background.




(a) CR with uniform random (b) CR with bursty (c) Local Channel Traffic of CR with bursty

Fig. 9: Communication time and local channel traffic of CR with background traffic.




(a) FB with uniform random (b) FB with bursty (c) Local Channel Traffic of FB with bursty

Fig. 10: Communication time and local channel traffic of FB with background traffic.

mechanism. Minimal routing, on the other hand, prevents packets from background traffic routing through the AMG routers, thus resulting in less channel traffic on these routers compared with adaptive routing. For job placement policy, contiguous and random-cabinet placements help confine the neighbor communication of AMG in nearby nodes, reducing the chance of messages from AMG being interleaved with background traffic and hence reducing the message waiting time. As a result, contiguous and random-cabinet placements suffer less performance degradation compared with other placement policies.

For **CR**, the communication performance under uniform random and bursty background traffic is presented in Figures 9(a) and 9(b) respectively. We observe that there is no obvious performance variation for all placement and routing combinations under uniform random traffic. On the other hand, with bursty background traffic, the communication time is prolonged up to 90% for CR. As shown in Figure 9(c), the local channel traffic of routers increases dramatically with all combinations of placement and routing policies except $cont-min$ and $cab-min$. Thus $cont-min$ and $cab-min$ result in less performance variation.

For **FB**, similar to CR's results, there is no obvious performance variation with uniform random traffic, as shown in Figure 10(a). The communication time increases up to 60% with bursty background traffic, as shown in 10(b). The local channel traffic of FB routers increases but not as much as with CR. Compared with minimal routing, adaptive routing suffers from higher performance variability. Again, contiguous and random-cabinet placements result in a relatively small variation under bursty background traffic.

Based on the results presented above, we have the following key findings.

- Applications with frequent communication, such as CR and FB, do not suffer much performance degradation under uniform random background traffic; however, they experience great performance variation under bursty background traffic.
- Applications with low frequency of message exchange, such as AMG, are sensitive to the background traffic, and they may suffer from high performance variability due to network interference.
- Localized communication helps reduce performance variation due to network sharing. Specifically, minimal routing and contiguous placement create a relatively "isolated" location on the shared network, resulting in less performance variation against external network interference.

## V. Related Work

Performance variability on high-performance computing (HPC) systems has been studied for a long time. Skinner and Kramer identified significant performance variability due to network contention [8]. They found that performance variability was inevitable on either torus or fat-tree networks when network sharing was allowed among concurrently running applications. Bhatele et al. studied the performance variability on different HPC production systems with torus interconnect topologies [26]. They found that the performance consistency could be obtained with compact allocation. Yang et al. [27] studied the communication behavior of three parallel applications on a torus network and analyzed interference by simulating three applications running both independently and concurrently. They found that inter-job interference was inevitable but proper allocation with communication pattern awareness could help alleviate the negative effects.

Since the dragonfly topology [1] was proposed, many studies have been conducted to explore the efficiency of routing, task mapping, and job placement on such networks [28] [29] [30]. Some research is based on an analytical model and synthetic workloads. Fuentes et al. quantified the impact of the routing mechanisms on the network congestion [11]. They injected synthetic traffic load on a modular simulator to demonstrate that a global misrouting policy is not sufficient to eradicate unfairness of the network. They showed that an explicit fairness mechanism is required to ensure an effective lack of unfairness with this traffic pattern and adaptive routing mechanisms. Prisacari et al. proposed a theoretical framework that is able to identify the bottlenecks that appear on the dragonfly network [9]. Their analytical model allows co-design of application decomposition, routing, and mapping in order to achieve optimal overall performance. Jain et al. analyzed the behavior of a dragonfly network with various job placement and routing policies [10]. They demonstrated the cost and benefit for each policy with synthetic applications and traffic models.

Our previous work [15] broadly classified applications into two categories, communication intensive and communication nonintensive, and analyzed their communication behaviors on a theoretical dragonfly network. We found that the prevailing recommendation of random process placement and adaptive routing is good at load balancing the network traffic, improving the performance of communication-intensive applications, but causing degradation for less communication-intensive ones. We identified this bully behavior of job interference and suggested a hybrid job placement methodology based on the application's communication intensity. In this work, we have adapted the interference study according to the dragonfly topology being used by Cray, we further investigated the application characteristics, including communication frequency and load, and analyzed the performance variability by simulating background communication traffic.

Our work differs from these studies at several aspects. First, unlike the studies using analytical models or high-level simulations, our study is built on high-fidelity, packet-level simulation using the exact configurations extracted from the production system. Such a simulation analysis provides us important insights into the trade-off between localizing communication and balancing network traffic that can occur in production systems; it also enables us to navigate a complex design space, such as different combinations of job placement and routing policies, many of which are inaccessible to regular users on production systems. Second, different from the studies focusing on application communication patterns or job placement or routing mechanisms, our study investigates the combined effects of various factors for three representative applications. It presents an in-depth trade-off analysis of localizing communication and balancing network traffic. We believe the key findings from this trade-off analysis provide valuable insights for the HPC community.

## VI. Conclusion

One of the most critical issues for dragonfly networks is performance variability. In this paper, we have presented a comparative analysis of the trade-off between localized communication and balanced network traffic using three representative applications with different communication patterns. By using trace-based simulation with real production system configurations, we have examined a variety of factors, including job placement policy, routing mechanism, communication pattern, communication intensity, and external network interference, that could impact the trade-off between localizing communication and balancing network traffic. Our in-depth study has provided a number of key findings. For

example, applications with low message load or low message exchange frequency benefit from localized communication, which reduces message transfer time by cutting down the number of hops; applications with high message load or high message exchange frequency benefit from balanced network traffic, which reduces message waiting time by alleviating link congestion. We also demonstrated that with external network interference, localized communication can help reduce the application performance variation.

We believe that the analysis performed in this paper provides valuable insights for efficiently utilizing dragonfly systems. We plan to investigate task mapping for diversified workloads. We will also study the joint actions among applications and system, in order to minimize performance variability on dragonfly systems.

## ACKNOWLEDGMENTS

This work is supported in part by US National Science Foundation grants CCF-1422009, CNS-1717763, and the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357. This research used data of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

## REFERENCES

[1] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *ACM SIGARCH Computer Architecture News*, vol. 36, no. 3. IEEE Computer Society, 2008, pp. 77–88.

[2] *Cori Cray XC40 System at NERSC*. [Online]. Available: http://www.nersc.gov/users/computational-systems/cori/

[3] *Theta at ALCF*. [Online]. Available: https://www.alcf.anl.gov/theta

[4] *Aurora at ALCF*. [Online]. Available: http://aurora.alcf.anl.gov

[5] S. Chunduri, K. Harms, S. Parker, V. Morozov, S. Oshin, N. Cherukuri, and K. Kumaran, "Run-to-run variability on Xeon Phi based Cray XC systems," in *SC17: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2017.

[6] W. Tang, N. Desai, D. Buettner, and Z. Lan, "Analyzing and adjusting user runtime estimates to improve job scheduling on the Blue Gene/P," in *IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*. IEEE, 2010, pp. 1–11.

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

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

[9] 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*. ACM, 2014, pp. 129–140.

[10] 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*. IEEE Press, 2014, pp. 336–347.

[11] P. Fuentes, E. Vallejo, C. Camarero, R. Beivide, and M. Valero, "Throughput unfairness in dragonfly networks under realistic traffic patterns," in *IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2015, pp. 801–808.

[12] C. Carothers, "Enabling co-design of multi-layer exascale storage architectures," Rensselaer Polytechnic Institute, Tech. Rep., 2015.

[13] M. Mubarak, P. Carns, J. Jenkins, J. K. Li, N. Jain, S. Snyder, R. Ross, C. D. Carothers, A. Bhatele, and K.-L. Ma, "Quantifying i/o and communication traffic interference on dragonfly networks equipped with burst buffers," in *IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2017, pp. 204–215.

[14] M. Mubarak and R. B. Ross, "Validation study of CODES dragonfly network model with Theta Cray XC system," Argonne National Laboratory, Tech. Rep., 04 2017. [Online]. Available: http://www.mcs.anl.gov/publication/validation-study-codes-dragonfly-network-model-theta-cray-xc-systemser

[15] X. Yang, J. Jenkins, M. Mubarak, R. B. Ross, and Z. Lan, "Watch out for the bully!: Job interference study on dragonfly network," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 64:1–64:11. [Online]. Available: http://dl.acm.org/citation.cfm?id=3014904.3014990

[16] G. Faanes, A. Bataineh, D. Roweth, E. Froese, B. Alverson, T. Johnson, J. Kopnick, M. Higgins, J. Reinhard *et al.*, "Cray Cascade: A scalable HPC system based on a dragonfly network," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press, 2012, p. 103.

[17] A. Bhatele, N. Jain, Y. Livnat, V. Pascucci, and P.-T. Bremer, "Analyzing network health and congestion in dragonfly-based supercomputers," in *IEEE International Symposium on Parallel and Distributed Processing Symposium*. IEEE, 2016, pp. 93–102.

[18] 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. 28, no. 1, pp. 87–100, 2017.

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

[20] 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*. ACM, 2014, pp. 27–38.

[21] Department of Energy, *Characterization of the DOE Mini-apps*, 2014 (accessed Nov 3, 2016). [Online]. Available: http://portal.nersc.gov/project/CAL/designforward.htm

[22] K. D. Underwood, M. Levenhagen, and A. Rodrigues, "Simulating Red Storm: Challenges and successes in building a system simulation," in *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2007, pp. 1–10.

[23] P. Fischer, A. Obabko, E. Merzari, and O. Marink, "Nek5000: Computational fluid dynamics code," Argonne National Laboratory, Tech. Rep., accessed Nov, 2016. [Online]. Available: http://nek5000.mcs.anl.gov.

[24] J. Bell, A. Almgren, V. Beckner, M. Day, and M. Lijewsk, "BoxLib users guide," Lawrence Berkeley National Laboratory, Tech. Rep., 2013. [Online]. Available: https://ccse.lbl.gov/BoxLib/BoxLibUsersGuide.pdf

[25] U. M. Yang *et al.*, "BoomerAMG: A parallel algebraic multigrid solver and preconditioner," *Applied Numerical Mathematics*, vol. 41, no. 1, pp. 155–177, 2002.

[26] 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*. ACM, 2013, p. 41.

[27] X. Yang, J. Jenkins, M. Mubarak, X. Wang, R. B. Ross, and Z. Lan, "Study of intra- and interjob interference on torus networks," in *IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, Dec 2016, pp. 239–246.

[28] P. Yébenes, J. Escudero-Sahuquillo, P. J. García, F. J. Alfaro, and F. J. Quiles, "Providing differentiated services, congestion management, and deadlock freedom in dragonfly networks," in *2nd IEEE International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)*. IEEE, 2016, pp. 33–40.

[29] M. García, E. Vallejo, R. Beivide, M. Odriozola, and M. Valero, "Efficient routing mechanisms for dragonfly networks," in *42nd International Conference on Parallel Processing (ICPP)*. IEEE, 2013, pp. 582–592.

[30] K. Wen, P. Samadi, S. Rumley, C. P. Chen, Y. Shen, M. Bahadori, K. Bergman, and J. Wilke, "Flexfly: Enabling a reconfigurable dragonfly through silicon photonics," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press, 2016, p. 15.