Unveiling the Interplay Between Global Link Arrangements and Network Management Algorithms on Dragonfly Networks

Fulya Kaplan*, Ozan Tuncer*, Vitus J. Leung†, Scott K. Hemmert†, and Ayse K. Coskun*
* Boston University, Boston, MA – {fkaplan3, otuncer, acoskun}@bu.edu
† Sandia National Laboratories, Albuquerque, NM – {vjleung, kshemme}@sandia.gov

***Abstract*—Network messaging delay historically constitutes a large portion of the wall-clock time for High Performance Computing (HPC) applications, as these applications run on many nodes and involve intensive communication among their tasks. Dragonfly network topology has emerged as a promising solution for building exascale HPC systems owing to its low network diameter and large bisection bandwidth. Dragonfly includes local links that form groups and global links that connect these groups via high bandwidth optical links. Many aspects of the dragonfly network design are yet to be explored, such as the performance impact of the connectivity of the global links, i.e., *global link arrangements*, the bandwidth of the local and global links, or the job allocation algorithm.**

This paper first introduces a packet-level simulation framework to model the performance of HPC applications in detail. The proposed framework is able to simulate known MPI (message passing interface) routines as well as applications with custom-defined communication patterns for a given job placement algorithm and network topology. Using this simulation framework, we investigate the coupling between global link bandwidth and arrangements, communication pattern and intensity, job allocation and task mapping algorithms, and routing mechanisms in dragonfly topologies. We demonstrate that by choosing the right combination of system settings and workload allocation algorithms, communication overhead can be decreased by up to 44%. We also show that circulant arrangement provides up to 15% higher bisection bandwidth compared to the other arrangements; but for realistic workloads, the performance impact of link arrangements is less than 3%.

I. Introduction

In HPC systems, network communication efficiency and delay play important roles in determining performance and scalability [1]. When scaling up to tens of thousands of nodes, traditional topologies such as toroidal meshes increase the network energy consumption up to 50% of the overall system energy [2] and introduce large communication overhead as messages need to travel tens of network hops on average [3].

Recent technological advances have enabled new topology designs to overcome these energy and performance limitations. First, owing to increased-radix routers [4], a larger number of ports can be connected to a router, allowing lower-diameter networks that reduce the number of hops a packet needs to travel. A reduced number of network hops translates to shorter messaging delays as well as lower energy consumption. Second, the availability of cost- and energy-efficient optical switches [5] enables higher link bandwidth compared to electrical links. Optical links also provide longer physical distance traveled per network hop.

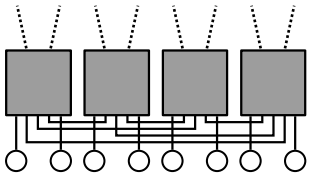


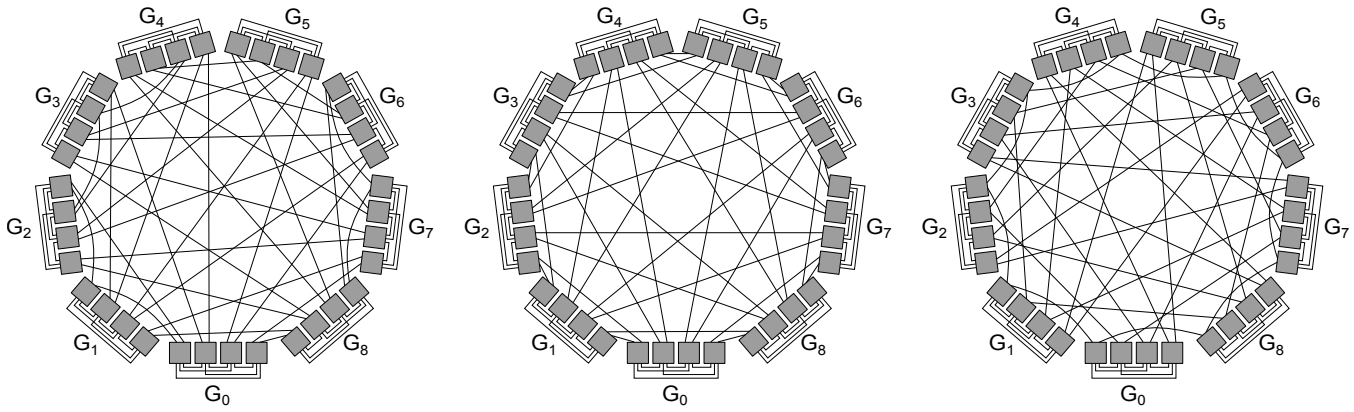
Fig. 1: A dragonfly group with all-to-all local connections. Boxes are routers, circles are nodes, solid lines are electrical local links, and dashed lines are optical global links.

Dragonfly network topology [6] exploits these technology advances mentioned above to achieve high bisection bandwidth[1] and high scalability. A dragonfly network has a two-level hierarchy, where the elements in each level are closely connected, resulting in a low network diameter. Variations of dragonfly topology are currently used in Cray XC [7], Cray Cascade [8] and IBM PERCS [9].

The dragonfly topology's two-level hierarchy is composed of *local* links forming *groups* and *global* links connecting these *groups* via optical links. Within a *group*, the routers are connected in an all-to-all or a flattened-butterfly fashion using electrical links. Overall, each router has ports connecting them to (i) the compute nodes, (ii) the other routers in the group, and (iii) the other groups in the network. Figure 1 illustrates a single group consisting of 4 routers, where each router is connected to 2 nodes, 3 other routers within the group, and 2 routers in other groups [6].

The existing literature on dragonflies focuses on analyzing the impact of routing [10], [11], [12] and job allocation algorithms [13], [14], [15] on performance. One aspect of dragonfly network topologies that has not yet been extensively studied is the *global link arrangement*, which defines the connectivity of each router in a group to the other groups. Recent work proposes three specific link arrangements to connect groups to each other: absolute, relative, and circulant-based [16]. Hastings et al. provide a theoretical study on these arrangements and analyze how the system bisection bandwidth changes with respect to the global and local link bandwidths [17]. Existing work, however, does not investigate the coupling

[1] Bisection bandwidth is the minimum bandwidth between two equally-sized parts of the system.



(a) Absolute (b) Relative (c) Circulant-based

Fig. 2: Dragonfly networks adopting different global link arrangements: a) absolute, b) relative, c) circulant-based arrangement. Each configuration uses $g = 9$, $a = 4$, $h = 2$, and all-to-all local connections. The boxes represent the routers.

of these arrangements with routing mechanisms, job allocation algorithms, or communication patterns.

This paper studies the impact of global link arrangements on network performance in tandem with routing mechanisms, job placement algorithms, and application communication patterns. To enable such analysis, we design a packet-level simulation framework that unifies network design parameters with communication patterns and accurately models wall-clock time of HPC applications. Our specific contributions are as follows:

- We introduce a packet-level simulation framework based on the Structural Simulation Toolkit (SST) [18]. Our framework estimates the performance of HPC applications with consideration of the underlying network topology by closing the gap between job allocation/task mapping and detailed network simulation. This framework allows simulating the combined impact of allocation/mapping decisions as well as network and application properties on the performance of HPC applications.
- Using our framework, we evaluate the performance of global link arrangements considering a set of job allocation and routing mechanisms for various communication patterns, and demonstrate that *circulant* global link arrangement provides up to 15% lower communication overhead compared to the other arrangements when the network is highly loaded, owing to its higher bisection bandwidth. We also show that the impact of global arrangements on application performance can be limited for common MPI patterns.
- We show that task mapping can substantially impact application running time, and the level of this impact is also affected by the routing mechanism and the network load. In our experiments with selected applications, we show that task mapping affects application running time by up to 11%.
- We demonstrate that the combined impact of the job allocation and routing mechanism highly depends on the bandwidth ratio of the global links over the local links. For high bandwidth ratios, our analysis shows up to 44% difference in communication overhead between the best and worst performing {routing, allocation} pairs.

The rest of the paper starts by providing background on dragonfly networks. Section III presents the details of our proposed simulation framework together with our target HPC machines and workload assumptions. In Section IV, we provide the results of our performance analysis regarding different aspects of dragonfly networks. Section V describes the prior work on dragonfly and Section VI concludes the paper.

II. Background on Dragonfly Networks

Dragonfly topology [6] is a two-level hierarchical direct network based on high-radix routers. We use the parameters presented in Table I to describe a dragonfly topology.

In the first hierarchical level, a routers constitute a group and are connected by local electrical links, typically with an all-to-all or a flattened-butterfly network topology. Each router is connected to p compute nodes and has h optical links that form an inter-group network. The routers in a group collectively act as a virtual router with $a \cdot p$ connections to compute nodes and $a \cdot h$ connections to other groups. The second hierarchical level consists of g of these virtual routers, typically connected with an all-to-all topology.

The rest of this section describes the link arrangements, routing mechanisms, and job placement strategies we use in our study of global link arrangements.

A. Link Arrangements

There are two link groups in a dragonfly network: global and local. Global links refer to the optical inter-group cables,

c	Number of cores per node
p	Number of nodes connected to a router
a	Number of routers in a group
g	Number of groups
h	Number of optical links on a router

TABLE I: Notation for the dragonfly parameters.

whereas local links connect routers within a single group.

1) Global: We use three different global link arrangements and comply with the terminology used by Hastings et al. [17].

Absolute arrangement: In the absolute arrangement, the first available port in group 0 is connected to the first available port in the group 1. Then, the next available port in group 0 is connected to the first available port in group 2. This continues until group 0 is linked to all other groups. We apply the same procedure to the remaining groups in order. As a result, port i of group j is connected to group i if $i < j$, and to group $i + 1$ otherwise. Figure 2(a) depicts an absolute arrangement.

Relative arrangement: All groups have identical relative connections in this arrangement. As shown in Figure 2(b), in each group, port 0 is connected to the next group, port 1 is connected to the second next group, and so on. In other words, port i of group j is connected to group $(i + j + 1)$ mod g, where g is the total number of groups.

Circulant-based arrangement: With the circulant-based arrangement, in each group, port 0 is connected to the next group, and port 1 is connected to the previous group. Port 2 is connected to the second next, and port 3 is connected to the second previous group, and so on. In other words, port i of group j is connected to group $(i/2 + j + 1)$ mod g if i is even, and to group $(-\lfloor i/2 \rfloor + j - 1)$ mod g if i is odd. This arrangement assumes that each router has an even number of optical links, and it is depicted in Figure 2(c).

2) Local: In dragonfly architectures, local links are typically arranged as a flattened-butterfly (e.g., Cray Cascade [8]) or as an all-to-all network (e.g., IBM PERCS [9]). In this paper, we assume all-to-all local link arrangements with a uniform local link bandwidth to isolate the impact of global link arrangements. We define the ratio of global link bandwidth to local link bandwidth as α.

B. Routing

A routing strategy defines the paths of message traversal through the network. Routing has been shown to play an important role in the system performance in Dragonflies [10], [11]. While a shortest-path routing strategy reduces the message latency in a dragonfly with low network utilization, it can lead to hot-spots on the global links when two groups intensively communicate with each other over a few links. In this work, we use two static routing strategies to study the impact of routing on different global link arrangements: *minimal* and *Valiant*.

1) Minimal: Minimal routing can be described as follows. Let us define the source and destination groups as G_S and G_D, and the source and destination routers as R_S and R_D, respectively. If $G_S \neq G_D$, then select an intermediate router, R_a, which is inside G_S and has a direct link to G_D. Next, use the direct link from R_a to R_b (which is the router in G_D). If $R_b \neq R_D$, use the shortest-path to R_D inside G_D. The longest communication distance with this mechanism is 3 router-to-router hops as we use all-to-all local connections.

2) Valiant [19]: Valiant routing aims to spread the traffic among the network to avoid hot-spots. If $G_S \neq G_D$, this mechanism sends the message first to a randomly-selected intermediate group, then to R_D, using minimal routing. If $G_S = G_D$, a random intermediate router is selected within the group. All messages travel at most 5 hops in Valiant routing as we use all-to-all local connections.

C. Allocation

Allocation refers to the placement of incoming jobs to the available machine nodes. Studies have shown that allocation has a significant impact on application performance on dragonflies, but there is no consensus in the community on which allocation algorithm maximizes the performance of dragonflies [13], [14], [15], [20]. In order to study the impact of global link arrangements in presence of a comprehensive set of workload management algorithms, we consider three allocation techniques that are shown to have fundamentally different characteristics [13]: *cluster*, *spread*, and *random*.

1) Cluster: Cluster first fills the available nodes in a single dragonfly group in order. Once there is no available node left, it continues with the next group.

2) Spread: This allocation strategy aims to spread a given job uniformly among groups by filling routers in a round robin manner. It first uses all nodes in router 0 of group 0, then continues with the nodes in router 0 of group 1, and so on. Once all router 0s are occupied in all groups, it continues with router 1.

3) Random: This strategy simply selects random nodes from the set of all available nodes in the system.

D. Task Mapping

Task mapping refers to the mapping of the MPI ranks of a single job onto the compute cores located in the nodes selected by the allocation algorithm. The aim of task mapping is to place closely-communicating MPI ranks next to each other to reduce the communication overhead and network load. As HPC infrastructures typically do not have access to the communication pattern of incoming jobs, task mapping is performed by the HPC application once the job starts executing.

Commonly known task mapping strategies include graph-based approaches and linear mapping, which assigns the MPI ranks to the cores in a linear order. In this work, we would like to minimize the performance impact of task mapping so as to isolate the effect of global link arrangements. Thus, for each application, we select the task mapping algorithm that suits the communication pattern in the best way. Task mapping algorithms we use are as follows:

1) Random: This task mapper randomly places application's MPI ranks onto the allocated nodes. We use the random task mapper to average out the impact of the messaging order in applications with uniform communication (e.g., *all-to-all* and *bisection* pattern; refer to Section III-C for further details).

2) Recursive Graph Bisection [21]: This algorithm recursively splits the application's communication graph and the network topology graph into equal halves using minimum weighted edge-cuts. At the end of the recursion, the remaining MPI ranks are placed in the remaining compute cores. We use this task mapper for the 3D stencil pattern as it has been shown to perform better than linear task mapping [21].

III. Experimental Methodology

Network links in real HPC machines are statically connected, and thus, it is not easy to experiment with different link connections without having an entire HPC system specifically allocated for this purpose. Hence, we use packet-level simulations (instead of experiments on real machines) in order to compare network performance for different design assumptions.

Our proposed simulation framework enables users to evaluate the combined performance impact of various design parameters such as global link arrangements, allocation/mapping algorithms, and application communication patterns. To implement our framework, we extend the Structural Simulation Toolkit (SST), which has been developed by Sandia National Laboratories to assist in the design, evaluation and optimization of HPC architectures and applications [18]. We add to SST the ability to perform packet-level simulations for custom dragonfly topologies along with dragonfly-specific routing and job allocation algorithms. Using those features we add in SST, we simulate workloads with different communication patterns to evaluate the impact of global link arrangements under a variety of scenarios. The rest of this section explains our simulation framework, the HPC machines we study, and the workloads we use in detail.

A. Simulation Framework

SST simulator has been widely used by researchers in both academic and industrial institutions. The accuracy of SST has been validated in publications and by hardware vendors [22], [23], [24]. SST incorporates individual elements to model specific aspects of a modern data center in detail, such as a scheduler and job allocator, a network simulator and a message passing simulator.

In the original version of SST, the *scheduler element* is a standalone module and is not connected to the detailed network simulator. It estimates the wall-clock time of applications through an average hop distance-based model using empirical data. The standalone *scheduler* module with hop distance-based performance model is good for estimating the relative performance improvement of new allocation/mapping strategies, but it is not sufficient in evaluating the more complex behavior that depends on network link bandwidths, message sizes, and routing strategies. In addition, the existing detailed network simulator is unaware of the job allocation/task mapping algorithms and it requires the user to manually define the nodes allocated for each job.

We propose a unified simulation framework that closes the gap between the scheduler and network simulation elements to have a holistic and accurate evaluation of the HPC data center performance. Our simulation framework is illustrated in Figure 3. We assume a job trace where jobs are defined by their arrival time, required number of processors, and the application communication pattern. We define the application communication patterns using *application phase files*. A *phase file* may include a single communication pattern (e.g., all-to-all), or a successive set of communication patterns representing the phases of an application. The *scheduler element* schedules the jobs (i.e., decides on when to start running the job), allocates nodes for the jobs, and maps individual tasks of a job onto the allocated nodes, depending on the selected scheduling, allocation, and mapping policies. The *scheduler element* includes advanced allocation/mapping algorithms that are applicable to various network topologies such as 2D/3D mesh and torus. In addition, we implement the dragonfly topology along with dragonfly-specific job allocation algorithms (i.e., cluster, spread, random) in the scheduler element.

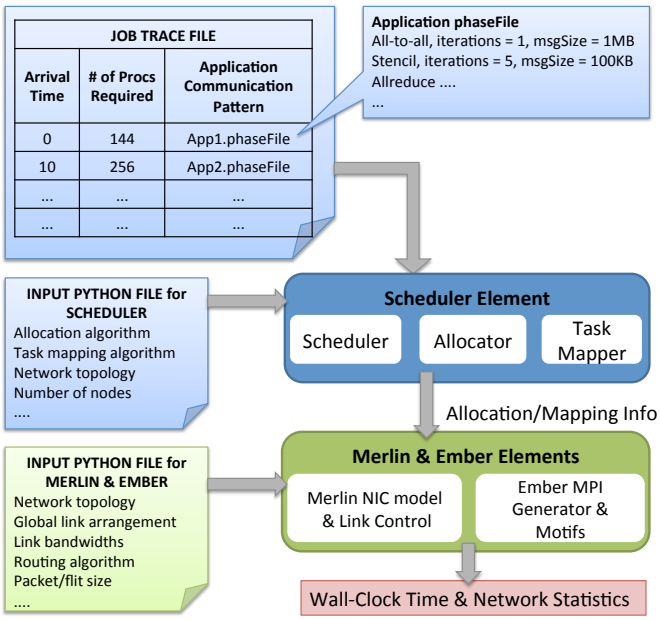


Fig. 3: Proposed simulation framework developed in SST, which integrates scheduling and network elements.

After the jobs are allocated and tasks are mapped onto the nodes, *ember* and *merlin elements* simulate the network timing for sending/receiving packets from one end point in the network to another. *Ember* models the MPI routines used in current HPC applications, such as boundary exchange (i.e., stencil), all-to-all, all-reduce. These MPI routines are named in SST as *ember motifs*. Using the *motifs*, *ember* implements the message traffic between the tasks of an application. We also add the functionality to *ember* that allows the user to simulate custom defined communication patterns. The *Merlin element* works in cooperation with *ember* and models the behavior of routers, network interface cards (NICs), and the network routing algorithms. *Merlin* can capture the transient network behavior resulting from congestion, stalls, and routing in a cycle accurate manner. We implement the *dragonfly* network

model, which accounts for the global link arrangements, in *merlin*. Apart from the link arrangements described in Section II-A, we provide the user the flexibility to define custom link arrangements. We can also define bandwidths separately for (i) the links to the hosts, (ii) the local links within the group, and (iii) the global links across groups.

Communication between the *scheduler* and *merlin & ember elements* is provided through a file interface in SST. The scheduler dumps the job allocation/task mapping information into a file. A python script then converts this information into the input format that *ember* can use.

Scalability of the SST framework to simulate exascale systems has been demonstrated in recent work [25]. In their work, Groves et al. analyze the impact of the number of global links and link bandwidths on performance and power consumption of dragonfly networks without considering allocation, task mapping, or global link arrangements. They model a dragonfly machine with 96 groups, 48 routers per group and 24 nodes per router, which corresponds to a total of 100,592 nodes. In this work, we experiment with machines with smaller intra-group networks in order to focus on the impact of global link arrangements.

B. Target HPC Machines

We study global link arrangements in 3 different target dragonfly machines. For all of our machines, we use $h = 2$ optical links per router. We change the parameters listed in Table I (specifically, a, p, c, g) to experiment with different machine sizes. Our small machine uses the same dragonfly parameters used in the theoretical study of global link arrangements by Hastings et al. [17]: $a = 4$, $p = 2$, $c = 2$, and $g = 9$, corresponding to a total of 72 nodes and 144 cores. For our medium-size machine, we use $a = 8$, $p = 2$, $c = 2$, $g = 17$, corresponding to a total of 272 nodes and 544 cores. As for the large machine, we use $a = 8$, $p = 4$, $c = 4$, $g = 17$, which adds up to a total of 544 nodes and $2,176$ cores.

We set the bandwidth of each link that is connected to the hosts and other routers within a group (i.e., local links) to 1GB/s. We sweep α (global to local link bandwidth ratio) from 0.5GB/s to 4GB/s in 0.5GB/s steps to analyze the impact of global link bandwidths. In order to isolate the effect of crossbar bandwidth on the simulation results, we set the crossbar bandwidth to the sum of the maximum link bandwidths connected to a router.

C. Workloads

We focus on three communication patterns: all-to-all, bisection, and 3D stencil. We select bisection pattern as it represents the bisection bandwidth of the network. All-to-all and 3D stencil patterns are commonly observed in real HPC applications [26].

All-to-all: It is a commonly known pattern where each task communicates with every one of the other tasks in an application. Fast Fourier Transform (FFT) is an example application with uniform all-to-all communication [27]. During our simulations, we observed that task mapping has an impact on the performance even for all-to-all communication pattern due to the scheduling order of the messages. To average out the impact of task mapping, we run all-to-all workloads 15 times with random task mapping and select the case with the median running time for our analysis.

Bisection: We use this communication pattern to account for the bisection bandwidth of the dragonfly network and to compare our results against the theoretical analysis from prior work by Hastings et al. [17]. In this pattern, the tasks are divided into two equal-sized groups (group #1 and #2), where a task from group #1 communicates with every task in group #2 and none of the tasks in group #1. In order to achieve the minimum bisection bandwidth between groups #1 and #2, we use the network cuts provided by Hastings et al. for the small machine size. These network cuts are also called *minimum cuts*; thus, the resulting application running time is a good representation of the bisection bandwidth. Once we decide on which group of tasks occupy which nodes based on the minimum cuts, we run bisection workload 15 times with random task mapping using those nodes. We select the case with maximum running time for this analysis as the bisection bandwidth represents the worst case.

For other machine sizes for which the minimum cuts are not readily available, we use the cut that takes the first half of the nodes in order, starting from node #0. Similarly, we apply random task mapping using these nodes and select the case with the maximum running time.

3D Stencil: This messaging pattern is observed in a large portion of the HPC applications and is explicitly supported by MPI [28]. Examples of real-life applications with 3D stencil communication include multi-dimensional shock-wave analysis [29] and molecular dynamics [30]. In this pattern, application tasks exchange messages with their six nearest Cartesian neighbors (i.e., in the x, y, z directions). In SST, we use the *Halo3D* motif in *ember*, which models the 3D-stencil MPI routine. As the running time of stencil jobs significantly depends on task mapping [21], we use recursive graph bisection to efficiently place application tasks.

For each of these communication patterns, we specify parameters such as the message size, the number of iterations, and the compute time per iteration. In order to focus on the communication overhead, we set the compute time to zero. We set the number of iterations for each application such that we allow enough time for the network to reach steady state. In our experiments, we experiment with 3 different message sizes: 100KB, 1000KB, and 4000KB.

We assume single job and multi-job cases, where all jobs collectively occupy 100% of the dragonfly machine. High system utilization is a common case in HPC and also makes it easier to observe how different conditions (communication patterns, message size, etc.) lead to network congestion. In the multi-job case, we assume two jobs of the same communication pattern are running simultaneously with each job occupying 50% of the machine. We focus on single-run of the jobs instead of job traces, which means that all jobs arrive and get allocated at the same time.

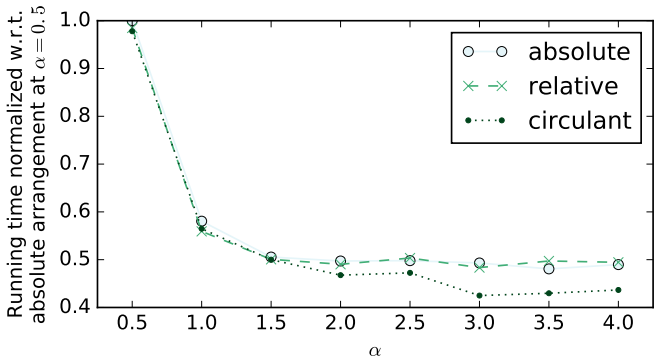


Fig. 4: Running time of the bisection communication pattern using different global link arrangements with *minimal* routing. All results are normalized with respect to absolute arrangement at $\alpha = 0.5$.

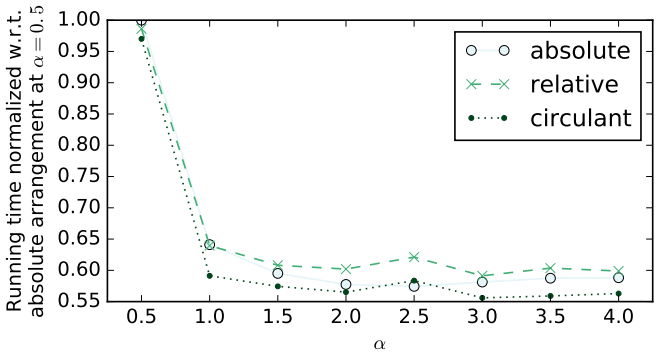


Fig. 5: Running time of the bisection communication pattern using different global link arrangements with *Valiant* routing. All results are normalized with respect to absolute arrangement at $\alpha = 0.5$.

IV. Analysis on Global Link Arrangements

This section presents our results on the impact of global link arrangements along with our findings on the coupling between link arrangements, communication patterns, routing, and job placement. We start our analysis using a single job that utilizes the entire machine to avoid the impacts of job allocation and inter-job interference, and continue with using multiple jobs.

A. Single Job Analysis

Here, we analyze the impact of global link arrangements on the communication overhead when the entire HPC machine is allocated to a single job. First, we validate the theoretical study of Hastings et al. [17] on how bisection bandwidth changes with link arrangements, and then focus on communication patterns that are commonly seen in HPC applications.

1) Bisection Bandwidth: In the bisection communication pattern, the first half of the dragonfly communicates with the second half. When used with minimal routing, the running time of this pattern is a good representation of system's bisection bandwidth where the network hot-spots are also taken into account.

Hastings et al. [17] have theoretically shown that the bisection bandwidth of the absolute global link arrangement falls behind the other two global arrangements as the ratio of global to local link bandwidth, α, is increased above 1.25. Between $\alpha = 1.25$ and $\alpha = 4$, circulant-based arrangement provides the highest bisection bandwidth; and at $\alpha = 4$, both circulant-based and relative arrangements have the same bisection bandwidth.

Our results demonstrate a similar pattern as shown in Figure 4 for 1000KB message size. We report all results in terms of normalized running time. Running time with circulant arrangement is 5-15% shorter than the other two arrangements for $\alpha > 1.5$. Circulant arrangement provides better performance with increasing α values. For all α values, performance difference between absolute and relative arrangements is very small within 4%.

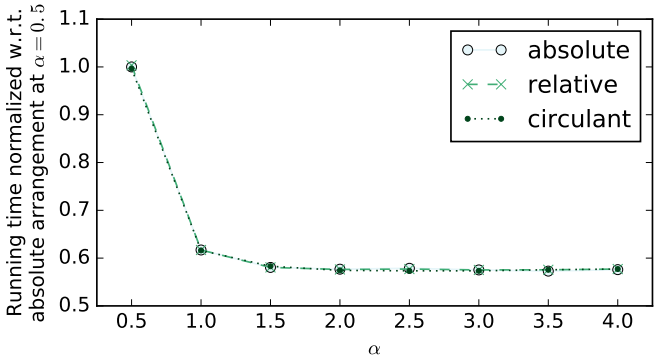


Fig. 6: Running time of the *all-to-all* communication pattern using different global link arrangements with minimal routing. All results are normalized with respect to absolute arrangement at $\alpha = 0.5$.

The impact of global link arrangements on running time decreases slightly when we use Valiant routing. As shown in Figure 5, circulant-based arrangement leads to 3-7% shorter running time in comparison to absolute arrangement. The trend for bisection bandwidth is similar for other message sizes.

The reason why circulant arrangement performs better lies behind how balanced the bisection bandwidth cuts are across the groups. A cut is a *balanced cut* if equal number of routers lie on each side of the cut in each group. For groups with a multiple of four routers, with circulant arrangement at $\alpha = 4$, the *minimum bisection cut* is completely balanced across all the groups (i.e., 2 routers per group). Moreover, the circulant arrangement achieves a *balanced cut* for a lower α than that of the relative arrangement, while absolute arrangement never reaches a balanced cut.

2) Realistic Workloads: The above results use bisection pattern, which is not typical in HPC workloads. This section presents our results with all-to-all and stencil communication patterns.

Unlike bisection pattern, global link arrangements do not

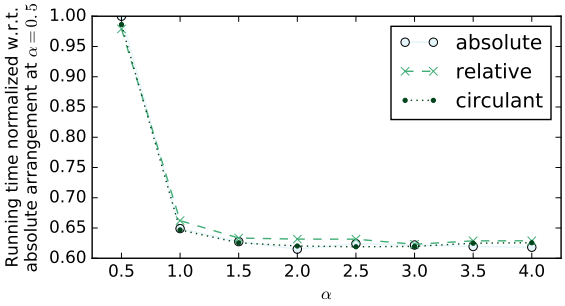


Fig. 7: Running time of the *stencil* communication pattern using different global link arrangements with minimal routing for 4000KB message size. All results are normalized with respect to absolute arrangement at $\alpha = 0.5$.

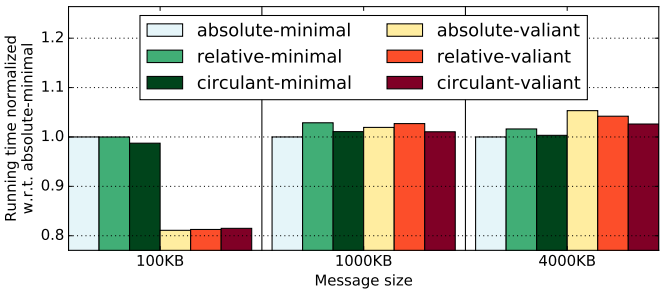


Fig. 8: Running time of all-to-all workload with different {link arrangement, routing algorithm} pairs and $\alpha = 4$. For each message size, the results are normalized with respect to the {absolute-minimal} pair of the corresponding message size.

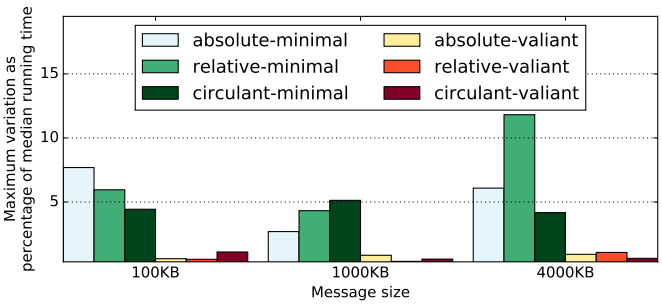


Fig. 9: Maximum running time variation of the all-to-all communication pattern under random task mapping with different {link arrangement, routing algorithm} pairs and $\alpha = 0.5$. Variation shown is relative to the median value.

have a significant impact on running time for all-to-all and stencil patterns. Figure 6 shows that the running time difference due to link arrangements is below 1% for any given α for all-to-all communication.

In Figure 7, we show the same trend for the stencil pattern. As the stencil pattern injects fewer messages into the network compared to bisection and all-to-all patterns, we present results with 4000KB message size, which achieves a similar network load. The performance difference among different global link arrangements is less than 3% for all α values.

The independence of all-to-all pattern's running time from global link arrangements is also valid for different message sizes and routing algorithms as shown in Figure 8. However, we observe that the routing mechanism has a significant impact on running time. For small message sizes such as 100KB, Valiant routing effectively spreads network traffic and avoids bottlenecks. Thus, for 100KB message size, Valiant reduces running time by 15%. On the other hand, Valiant routing increases the average traffic as it does not use the minimum path. As a result, for larger message sizes, Valiant routing introduces new network bottlenecks and results in up to 5% longer running time compared to minimal routing.

Another interesting observation of our study is regarding the impact of task mapping. As mentioned in Section III-C, our results on all-to-all pattern belongs to the case with the median running time among 15 simulations with random task mapping. In Figure 9, we present the variation in running time among these 15 simulations as a percentage of the median running time for $\alpha = 0.5$. While the variation is between 3-11% with minimal routing depending on the specific task mapping used by the random task mapper, it is below 2.5% with Valiant routing. As Valiant routing has an inherent randomization for intermediate router/group selection, it minimizes the impact of task mapping by randomly spreading traffic. For $\alpha = 4$, our results show that the variation reduces below 5% with minimal routing due to the increased network bandwidth, whereas the variation stays below 2% with Valiant routing.

3) Medium and Large Machines: For the medium machine with 17 groups, we repeat our analysis on the global link arrangements with bisection pattern. Figure 10 shows similar trends for bisection bandwidth, where we observe 2-8% shorter running time with circulant arrangement in comparison to the absolute arrangement. For medium machine size, relative arrangement starts to perform closer to the circulant arrangement than the absolute arrangement, indicating that the performance difference among global link arrangements is not specific to the small machine size.

For the large machine with 17 groups and 2,176 cores, we carry out simulations with all-to-all and bisection patterns to analyze the impact of increasing the number of tasks per local router on the resulting application running time. The results are consistent with the small and medium-size machines: The performance of the all-to-all pattern shows negligible difference across the global link arrangements (less than 2%). Similarly, for the bisection pattern, we observe that the circulant arrangement provides 3-10% shorter running time compared to the absolute arrangement.

B. Multiple Job Analysis

We next explore the dependency of performance on various network parameters when two jobs are running together. When considering multiple jobs, an important aspect comes into the picture, namely, job allocation algorithm. We focus on three main job allocation strategies: cluster, spread, and

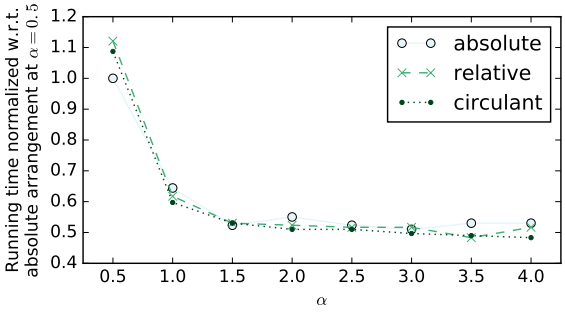


Fig. 10: Running time of the bisection communication pattern using different global link arrangements with minimal routing for 100KB message size on the large system. All results are normalized with respect to absolute arrangement at $\alpha = 0.5$.

random allocation (See Section II-C). For this analysis, we use the small machine and stencil communication pattern, and consider two jobs with the stencil pattern running together (e.g., a common example HPC scenario is when users submit the same stencil job with different inputs). We report the running time of the slower job.

In the previous results in Section IV-A, we have shown that the circulant arrangement provides slightly shorter running time for stencil application. Thus, in our next analysis, we focus on circulant arrangement. Figure 11 compares the running time for each {routing, allocation} algorithm pair over different α values. The results are normalized to {minimal, cluster} case at their corresponding α value. An interesting observation is that how good a {routing, allocation} pair performs is a function of the bandwidth ratio α. At $\alpha = 0.5$, valiant and minimal routing behave rather similarly, with {minimal, random} being the best choice. As α increases, the pairs involving Valiant algorithm start to perform poorly. The reason is that global links become less congested with increasing α. When α is large, local link congestion becomes the bottleneck; and Valiant routing amplifies this effect by increasing the overall network traffic. This results in around 44% performance difference between the best and worst pairs ({minimal, spread} and {valiant, random}, respectively) at $\alpha = 4$.

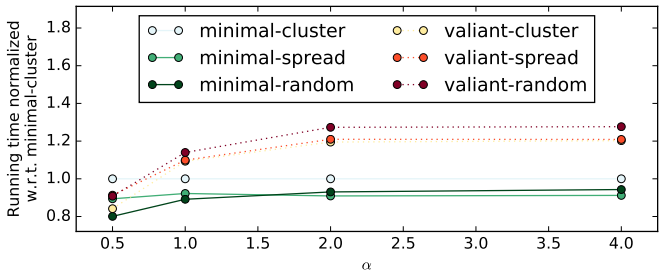


Fig. 11: Running time of the stencil communication pattern using circulant arrangements with {routing, allocation} algorithm pairs. Results are normalized with respect to the {minimal, cluster} pair at their corresponding α value.

C. Summary of Our Key Findings

Our performance analysis on dragonfly networks considering various aspects has led to the following key findings:

- Detailed network simulation closely follows the theory regarding the impact of global link arrangements on bisection bandwidth. We show that circulant arrangement provides up to 15% shorter running time compared to absolute arrangement for the bisection communication pattern with minimal routing.
- In contrast to the bisection communication pattern, global link arrangements do not have a significant impact on performance (i.e., <3%) for realistic patterns such as all-to-all and 3D stencil.
- We show that task mapping can significantly impact the application running time even for the all-to-all pattern due to the scheduling order of the messages. We observe up to 11% performance variation for all-to-all jobs due to task mapping.
- For realistic workloads, the choice of {routing, allocation} algorithm pair has a larger impact on performance compared to the impact of link arrangements. The selection of the best performing pair depends on α. As α increases, traffic on local links becomes a bottleneck. As Valiant routing creates more traffic on all links, it results in up to 44% longer running times in comparison to using minimal routing at $\alpha = 4$.

V. Related Work

As dragonfly network topologies [6] gain popularity due to their high bisection bandwidth and low diameter, researchers have investigated how to improve the performance of dragonflies. Most of these studies use simulators to easily experiment with different dragonfly settings with a clear visibility on the system. A group of existing literature utilizes high-level simulators, where the performance evaluation metrics are based on the link usage [31], [32]. Low-level simulators exist, but they either do not model different global link arrangements [33] or focus on network packet delays without considering job placement algorithms [34]. In our work, we propose a unified simulation framework that is able to simulate the combination of such factors.

Based on these simulators, various techniques have been proposed to improve dragonfly performance. Prisacari et al. [10] introduce dragonfly-specific hierarchical all-to-all exchange patterns that reduce all-to-all communication time by up to 45%. Fuentes et al. [11] focus on traffic patterns that portray real use cases and study the network unfairness caused by existing routing mechanisms with relative global link arrangement. Yébenes et al. [12] introduce a packet queuing scheme to reduce message stalls with minimal-path routing. These works specifically target improving routing-related issues in dragonflies.

Researchers have identified a strong coupling between routing and job allocation in dragonflies. Bhatele et al. [13] conduct a study on a IBM PERCS architecture [9] and conclude

that default MPI rank-ordered allocation leads to significant network congestion when used together with minimal-routing. They claim that (1) indirect routing obviates the need for intelligent job allocation and (2) random allocation gives the best performance when direct routing is used. Following up on this work, Chakaravarthy et al. [14] show that *transpose* communication patterns [35] benefit from direct routing, and Prisacari et al. demonstrate that random allocation with minimal-routing is consistently outperformed by Cartesian job placement with indirect routing for stencil communication patterns [15]. These works focus on allocation and routing strategies but do not consider different global link arrangements.

Several studies investigate global links in dragonflies. Bhatele et al. [32] investigate the impact of changing the number of router links on performance. Groves et al. [25] explore the effect of the number of global links and link bandwidths on the performance and power consumption of dragonfly networks considering the absolute arrangement only. Camarero et al. [16] introduce the three global link arrangements we use in this paper; however, they do not provide any performance comparison between them and they instead focus on the impact of the routing mechanisms. Hastings et al. conduct a theoretical analysis on global link arrangements and show that the commonly-used absolute link arrangement leads to a smaller bisection bandwidth when the ratio of global/local link bandwidths is larger than 1.25 [17]. Wen et al. propose Flexfly, a re-configurable network architecture for the global links using low-radix optical switches [36]. Flexfly modifies inter-group connections based on the observed network traffic to mitigate the need for indirect routing. However, it requires knowledge on application traffic patterns, which are not easy to extract because application type is typically unknown to HPC systems and traffic patterns may depend on application input.

To the best of our knowledge, our work is the first to experimentally evaluate the impact of different global link arrangements on performance in tandem with link bandwidths, communication patterns, job placement algorithms, and routing mechanisms.

VI. Conclusion

In this paper, we present a thorough analysis on the unexplored aspects of the dragonfly networks. For this purpose, we first propose a simulation framework that is able to evaluate the combined impact of global link arrangements, link bandwidths, job allocation and routing algorithms on the application running time. We then compare the performance of several known global link arrangements and show that circulant arrangement provides up to 15% reduction in communication time for the bisection communication pattern. We demonstrate that for common MPI communication patterns, the impact of global link arrangements is of less significance. On the other hand, for the same MPI patterns, we find that the choice of job allocation and routing algorithm is highly important, leading up to 44% difference in communication overhead, and that the best choice depends on the bandwidth ratio between global and local links. Finally, we show that task mapping in dragonfly can result in up to 11% variation in the application running time even for all-to-all communication pattern due to the scheduling order of the messages.

Acknowledgment

This work has been partially funded by Sandia National Laboratories. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

[1] J. Dongarra *et al.*, "The international exascale software project roadmap," *Int. J. High Perform. Comput. Appl.*, vol. 25, no. 1, pp. 3–60, Feb. 2011.

[2] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," in *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ser. ISCA '10, 2010, pp. 338–347.

[3] M. Besta and T. Hoefler, "Slim fly: A cost effective low-diameter network topology," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '14, 2014, pp. 348–359.

[4] J. Kim, W. J. Dally, B. Towles, and A. K. Gupta, "Microarchitecture of a high-radix router," in *Proceedings of the 32nd Annual International Symposium on Computer Architecture*, ser. ISCA '05, 2005, pp. 420–431.

[5] P. Dong, X. Liu, S. Chandrasekhar, L. L. Buhl, R. Aroca, and Y. K. Chen, "Monolithic silicon photonic integrated circuits for compact 100^{+} gb/s coherent optical receivers and transmitters," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 20, no. 4, pp. 150–157, July 2014.

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

[7] B. Alverson, E. Froese, L. Kaplan, and D. Roweth, "Cray XC series network," Tech. Rep., 2012, Cray, Inc., White paper.

[8] 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 *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Nov 2012, pp. 1–9.

[9] B. Arimilli, R. Arimilli, V. Chung, S. Clark, W. Denzel, B. Drerup, T. Hoefler, J. Joyner, J. Lewis, J. Li, N. Ni, and R. Rajamony, "The PERCS high-performance interconnect," in *2010 18th IEEE Symposium on High Performance Interconnects*, Aug 2010.

[10] B. Prisacari, G. Rodriguez, and C. Minkenberg, "Generalized hierarchical all-to-all exchange patterns," in *2013 IEEE 27th International Symposium on Parallel Distributed Processing (IPDPS)*, May 2013, pp. 537–547.

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

[12] P. Yébenes, J. Escudero-Sahuquillo, P. J. García, and F. J. Quiles, "Straightforward solutions to reduce hol blocking in different dragonfly fully-connected interconnection patterns," *The Journal of Supercomputing*, pp. 1–23, 2016.

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

[14] V. T. Chakaravarthy, M. Kedia, Y. Sabharwal, N. P. K. Katta, R. Rajamony, and A. Ramanan, "Mapping strategies for the PERCS architecture," in *2012 19th International Conference on High Performance Computing (HiPC)*, Dec 2012, pp. 1–10.

[15] 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, 2014, pp. 129–140.

[16] C. Camarero, E. Vallejo, and R. Beivide, "Topological characterization of hamming and dragonfly networks and its implications on routing," *ACM Trans. Archit. Code Optim.*, vol. 11, no. 4, pp. 39:1–39:25, Dec. 2014.

[17] E. Hastings, D. Rincon-Cruz, M. Spehlmann, S. Meyers, A. Xu, D. P. Bunde, and V. J. Leung, "Comparing global link arrangements for dragonfly networks," in *2015 IEEE International Conference on Cluster Computing*, Sept 2015, pp. 361–370.

[18] A. Rodrigues, E. Cooper-Balis, K. Bergman, K. Ferreira, D. Bunde, and K. S. Hemmert, "Improvements to the structural simulation toolkit," in *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques*, ser. SIMUTOOLS '12, 2012, pp. 190–195.

[19] L. G. Valiant and G. J. Brebner, "Universal schemes for parallel communication," in *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, ser. STOC '81, 1981, pp. 263–277.

[20] 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

[21] T. Hoefler and M. Snir, "Generic topology mapping strategies for large-scale parallel architectures," in *Proceedings of the International Conference on Supercomputing*, ser. ICS '11, 2011, pp. 75–84.

[22] A. F. Rodrigues, K. S. Hemmert, B. W. Barrett, C. Kersey, R. Oldfield, M. Weston, R. Risen, J. Cook, P. Rosenfeld, E. CooperBalls, and B. Jacob, "The structural simulation toolkit," *SIGMETRICS Perform. Eval. Rev.*, vol. 38, no. 4, pp. 37–42, Mar. 2011. [Online]. Available: http://doi.acm.org/10.1145/1964218.1964225

[23] M.-y. Hsieh, A. Rodrigues, R. Riesen, K. Thompson, and W. Song, "A framework for architecture-level power, area, and thermal simulation and its application to network-on-chip design exploration," *SIGMETRICS Perform. Eval. Rev.*, vol. 38, no. 4, pp. 63–68, Mar. 2011. [Online]. Available: http://doi.acm.org/10.1145/1964218.1964229

[24] K. D. Underwood, M. Levenhagen, and A. Rodrigues, "Simulating red storm: Challenges and successes in building a system simulation," in *2007 IEEE International Parallel and Distributed Processing Symposium*, March 2007, pp. 1–10.

[25] T. Groves, R. E. Grant, S. Hemmer, S. Hammond, M. Levenhagen, and D. C. Arnold, "(sai) stalled, active and idle: Characterizing power and performance of large-scale dragonfly networks," in *2016 IEEE International Conference on Cluster Computing (CLUSTER)*, Sept 2016, pp. 50–59.

[26] K. Antypas, "Nersc-6 workload analysis and benchmark selection process," *Lawrence Berkeley National Laboratory*, 2008.

[27] J. Meng, E. Llamosí, F. Kaplan, C. Zhang, J. Sheng, M. Herbordt, G. Schirner, and A. K. Coskun, "Communication and cooling aware job allocation in data centers for communication-intensive workloads," *J. Parallel Distrib. Comput.*, vol. 96, no. C, pp. 181–193, Oct. 2016. [Online]. Available: http://dx.doi.org/10.1016/j.jpdc.2016.05.016

[28] W. Gropp, T. Hoefler, R. Thakur, and E. Lusk, *Using Advanced MPI: Modern Features of the Message-Passing Interface*. MIT Press, Nov. 2014.

[29] E. S. Hertel, Jr., R. L. Bell, M. G. Elrick, A. V. Farnsworth, G. I. Kerley, J. M. Mcglaun, S. V. Petney, S. A. Silling, P. A. Taylor, and L. Yarrington, "Cth: A software family for multi-dimensional shock physics analysis," in *in Proceedings of the 19th International Symposium on Shock Waves, held at*, 1993, pp. 377–382.

[30] S. Plimpton, "Fast parallel algorithms for short-range molecular dynamics," *J. Comput. Phys.*, vol. 117, no. 1, pp. 1–19, Mar. 1995. [Online]. Available: lammps.sandia.gov

[31] G. Zheng, T. Wilmarth, P. Jagadishprasad, and L. V. Kalé, "Simulation-based performance prediction for large parallel machines," *International Journal of Parallel Programming*, vol. 33, no. 2, pp. 183–207, 2005.

[32] A. Bhatele, N. Jain, Y. Livnat, V. Pascucci, and P.-T. Bremer, "Evaluating system parameters on a dragonfly using simulation and visualization," Tech. Rep., July 2015, technical Report.

[33] M. Garcia, P. Fuentes, M. Odriozola, E. Vallejo, and R. Beivide. FOGSim interconnection network simulator. [Online]. Available: http://fuentesp.github.io/fogsim/

[34] P. Yebenes, J. Escudero-Sahuquillo, P. J. Garcia, and F. J. Quiles, "Towards modeling interconnection networks of exascale systems with omnet++," in *21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, Feb 2013, pp. 203–207.

[35] P. Luszczek, J. J. Dongarra, D. Koester, R. Rabenseifner, B. Lucas, J. Kepner, J. Mccalpin, D. Bailey, and D. Takahashi, "Introduction to the HPC challenge benchmark suite," Tech. Rep., 2005.

[36] K. Wen, P. Samadi, S. Rumley, C. P. Chen, Y. Shen, M. Bahadori, J. Wilke, and K. Bergman, "Flexfly: Enabling a reconfigurable dragonfly through silicon photonics," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Nov 2016.