Performance Evaluation of Adaptive Routing on Dragonfly-based Production Systems

Citation for published version:
Chunduri, S, Groves, T, Harms, K, Mendygral, P, Zarins, J, Weiland, M & Ghadar, Y 2021, Performance Evaluation of Adaptive Routing on Dragonfly-based Production Systems. in *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS).* IEEE, pp. 340-349, 35th IEEE International Parallel and Distributed Processing Symposium, Portland, United States, 17/05/21. https://doi.org/10.1109/IPDPS49936.2021.00042

Digital Object Identifier (DOI):
10.1109/IPDPS49936.2021.00042

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)

Performance Evaluation of Adaptive Routing on Dragonfly-based Production Systems

S Chunduri*, K Harms*, T Groves†, P Mendygral‡, J Zarins§, M Weiland§, Y Ghadar*
*Argonne National Laboratory, †Lawrence Berkeley Laboratory, ‡Hewlett Packard Enterprise, §EPCC

***Abstract*—Performance of applications in production environments can be sensitive to network congestion. Cray Aries supports adaptively routing each network packet independently based on the load or congestion encountered as a packet traverses the network. Software can dictate different routing policies, adjusting between minimal and non-minimal bias, for each posted message. We have extensively evaluated the sensitivity of the routing bias selection on application performance as well as whole system performance in both production and controlled conditions. We show that the default routing bias used in Aries-based systems is often sub-optimal and that using a higher bias towards minimal routes will not only reduce the congestion effects on the application but also will decrease the overall congestion on the network. This routing scheme results in not only improved mean performance (by up to 12%) of most production applications but also reduced run-to-run variability. Our study prompted the two supercomputing facilities (ALCF and NERSC) to change the default routing mode on their Aries-based systems. We present the substantial improvement measured in the overall congestion management and interconnect performance in production after making this change.**

I. Introduction

Dragonfly networks [1] are a prominent fixture in the HPC landscape. Four of the upcoming DoE systems (Perlmutter, Aurora, Frontier, and El Capitan) will feature Cray Slingshot Networks with a dragonfly topology. The dragonfly topology provides a low-diameter network for large numbers of network endpoints with many paths between two points on the network which are in distinct "groups". These network paths can be divided into "minimal" and "non-minimal" paths. Non-minimal routing, which redirects traffic through extra hops, may provide a lower latency and higher throughput as a network becomes congested. Modern networks have several adaptive routing policies to choose from. These policies adjust the congestion threshold for taking the minimal or non-minimal path. We refer to these as routing biases.

Recent work by De Sensi et al. [2] has highlighted the importance of adaptive routing biases to application performance on production dragonfly systems. This work utilized a runtime approach called application aware routing (AWR) to poll NIC counters and tune routing policies (increasing or decreasing minimal path bias). While their approach served as motivation for our work, it is limited in two respects. One is the overhead of the runtime, particularly on many-core CPU architectures. In our experiments on Intel KNL, the overhead of querying the performance counters on every message was too high for the processor to manage without impacting the overall application runtime. Secondly, De Sensi's work showed that individual bias policies often outperformed the adaptive runtime.

In this work we ask the question, "Are there fundamental application and system characteristics that prefer a minimal or non-minimal bias in Dragonfly networks?". By identifying these characteristics we are able to intelligently apply per-application routing biases and provide a set of best practices to HPC developers. As we answer this question, this study makes the following contributions:

1) Evaluates the relationship between a set of key applications' communication patterns, routing policies and performance.
2) Draws distinctions between routing performance for workloads running alongside competing congestion and benign traffic patterns.
3) Analyzes the types of congestion at the different levels of the network to understand the efficacy of minimal and non-minimal routes.
4) Determines that high minimal-route bias should be the default setting on ALCF and NERSC systems.

The rest of the paper is organized as follows. An overview of the congestion effects and routing algorithms on Aries-based Dragonfly, and the application and system configurations used for evaluation, is presented in Section II. Section III presents the experimental methodology and the metrics used for studying the sensitivity of routing changes on application performance. The performance effects of routing policies on applications in production environments are elaborated in Section IV. Section V presents system level congestion changes using controlled experiments and global system monitoring. Other literature related to this paper and our concluding remarks are presented respectively in Sections VI and VII.

II. Background

A. The Aries dragonfly network

The most common dragonfly network today is the Cray Aries network [3]. Our study utilizes two Aries systems, ALCF Theta and NERSC Cori. Though Aries and upcoming dragonfly systems (e.g. Cray Slingshot) have differences (such as group size and congestion control policies), we expect that many of the insights provided by this paper will be applicable to future dragonfly systems as well as current. This is because on any dragonfly system applications will have a preference for minimal or non-minimal routes, due to the communication patterns inherent to the application.

Cray XC-40 systems use the Cray Aries interconnect configured in a three-level dragonfly topology [3]. The first two levels (rank-1 and rank-2) are copper-based with 10.5 GB/s bidirectional bandwidth per link, and the rank-3 level is optical with 9.38 GB/s per link. The three rank-2 links are used to connect each pair of routers in the intra-group columns, and each router is connected to the 15 other routers in a row with a rank-1 link. The Rank-1 (green), Rank-2 (grey) and Rank-3 (blue) router tiles in total 40 of these are referred as network tiles. The remaining 8 router tiles, referred as Processor tiles, connect to the Aries NICs. Rank-1, Rank-2 and Rank-3 tiles are utilized by all nodes of the system as traffic passes through routers, whereas Processor tiles are only used by the 4 nodes with NICs directly connected to an individual router.

B. *Network variability*

In an isolated environment where there is a single application running on the system, the application can have access to a completely isolated network as there is no background noise. However, in order to optimize the cost of the HPC interconnect, much of the interconnect is shared by all applications running on the system. The shared nature of the network allows any application to utilize large amounts of network resources, which can optimize the performance of that single application when the network is idle. However, when applications simultaneously use the network, congestion can result when network resources become oversubscribed. This results in variable performance from run to run, and certain applications will experience lower overall performance due to the lack of available resources [4]–[7]. The Aries interconnect provides two mechanisms for mitigating network congestion. One mechanism is throttling the network, which can have a broad effect and so only occurs under extreme persistent congestion scenarios. The second mechanism is adaptive routing covered in subsection II-D.

C. *Background noise effects on application performance*

Application specific features such as communication pattern and intensity as well as system specific aspects such as topology and routing, together determine the communication performance of an application on a HPC system. The application specific characteristics are detailed in Section II-E, and the system specific aspects are discussed here.

The job placement determines the topology of the allocated nodes and how they communicate. A compact placed job can reduce the impact from other jobs as it minimizes rank-3 exposure, however, its performance can be impacted negatively due to low availability of rank-3 network bandwidth. A disperse (or random) placed job (with nodes allocated from several electrical groups) can have better availability of rank-3 bandwidth but this can induce higher noise impact from other jobs. The adaptive routing bias towards minimal or non-minimal paths has shown to have a decision impact on application performance in production [2], [8], [9]. Dragonfly network-based simulation studies [10]–[12] show the combined effects of mapping and routing on the application performance.

The background noise is determined by the placement and routing of other running applications and also by their communication characteristics. The size of the job in terms of total nodes used is also important to understand as that implies the influence of the above mentioned factors on the performance. A large-sized job occupying almost the entire system can have isolated exposure to the network without the background noise of other applications and hence is only influenced by its own communication characteristics and the routing scheme used. A small-sized job that is placed compactly avoids interference from other jobs as there are only a few possible paths between its limited endpoints. Medium-sized jobs are more prone to be affected by background noise irrespective of the job placement (compact or dispersed) because of the adaptive routing where the potential for link sharing with other jobs is high [6], [13].

Hence, in this study, we use the medium-sized jobs and experiment with the different adaptive routing modes available. The application experiments are repeated multiple times so as to capture the spectrum of possible background noise and job placement scenarios. We believe these system characteristics are representative of many HPC systems and will translate to other dragonfly based systems.

D. *Routing on Cray Aries*

The Cray Aries defines four adaptive routing modes, ADAPTIVE_0,1,2,3, (hereafter referred to as AD[0,1,2,3] with AD0 being the default [3]. Applications on a Cray Aries interconnect may select an Aries routing control mode by setting an environment variable (MPI) or making a GNI/DMAPP API call. From the manual page for Cray MPI:

The definition of an adaptive routing mode can be configured using bias value which is a combination of shift and add (with values between 0 and 15) parameters. The AD0 mode just compares the load on minimal and non-minimal paths, essential equal bias between them. The Cray MPI library uses the AD0 routing mode for most MPI operations except for MPI_Alltoall[v] variants. The AD1 [14], [15] mode provides increasingly minimal bias, a larger bias to selecting a minimal path is used as the packet traverses more hops. The MPI_Alltoall[v] implementations in Cray MPI library use AD1 mode. AD2 uses an addition of 4 to the bias to minimal with no shift value, thus it is referred as a mode with weak bias towards minimal. With AD3, a shift of 2 is used for minimal bias, thus it is referred as a strong bias toward minimal. Thus with AD3, the load on minimal paths needs to be 4X of that on the non-minimal paths, before non-minimal paths will be used. In general using the non-minimal paths is ideal when the load on the network is low as it can enable more thoroughly utilizing the available network bandwidth. Otherwise, using the minimal paths will be beneficial.

Cray does not provide guidance on the selection of adapting routing protocols or why Cray selected AD0, but we can provide some speculation based on the defined behaviors. The choice of AD0 as the default routing protocol provides a balanced approach of giving applications the opportunity to utilize significant portions of the network, making both minimal and non-minimal paths equally available at the expense to increasing aggregate load on the network. Selection of AD1 provides a larger potential bisection bandwidth for Alltoall

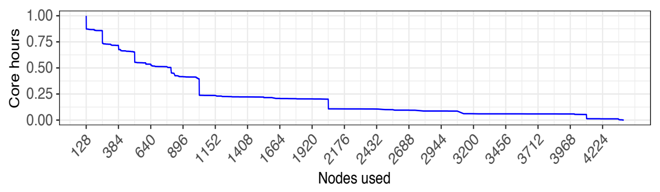

Fig. 1. Theta job size distribution

operations by allowing non-minimal paths to be selected initially but will incrementally limit the dispersion of paths the more hops messages take. Increasing the bias toward minimal limits the impact of applications using Alltoall on the entire network and will reduce potential congestion.

E. Application Characteristics

Applications can be impacted by the network performance to different extents based on their MPI usage and communication intensity. Latency-bound operations use small messages that are bound by the delay network takes to send a message. These operations are small (few KB of size) messages which require minimal network resources and perform best using a shortest path. Latency-sensitive operations can be heavily impacted by congestion as either waiting in queue or taking a longer route can incur more delay [16]. Logically, AD3 should provide the best performance for these applications by selecting the minimal paths and reducing congestion from other applications by routing messages across the non-minimal paths. Bisection bandwidth-bound operations typically use large messages that are bound by the aggregate throughput of the network rather than the delay of any individual message. Bisection bandwidth operations will favor non-minimal paths which allows increasing the total number of messages in-flight and achieving the highest overall bandwidth. Injection bandwidth or message-rate-bound operations which are limited at the NIC are less sensitive to routing mode changes because the routing mode will not impact the ability of the NIC to inject data. Applications which are mostly compute bound or use overlapped communication will also be unaffected by network behaviors.

F. System Characteristics

The experimental evaluation was performed on Cray XC-40 systems Theta and Cori for this study. Theta has 4392 compute nodes based on the Intel Xeon Phi (KNL) 7230 processors. Theta is configured with 12 dragonfly groups using 12 active optical cables (3 lanes each) [3] between each group. Cori has 9668 KNL compute nodes using similar topology as Theta, but with the distinction that only 4 cables span each group-to-group connection, resulting in a reduced bisection-to-injection ratio. Unless otherwise noted, the results presented are from experiments run on Theta.

The user base of these systems consists of a wide variety of users performing simulations and analysis from a wide variety of scientific domains [17], [18]. The selection of the applications used is a result of the INCITE and ALCC allocation award programs which select proposals submitted by scientists and that are reviewed and awarded on a merit basis. The resulting diverse workloads require that Theta perform well under many different application characteristics, indicating that the results presented here are widely applicable to the usage of other dragonfly-based systems.

Figure 1 is the complementary cumulative distribution function (CCDF) of job sizes (by number of nodes) on Theta. The figure shows that approximately 40% of all core-hours on Theta are from jobs allocated with between 128 and 512 nodes. Based on the discussion in II-C, we noted that intermediate sizes job are likely the most susceptible to network congestion.

This results in 70% of core-hours being used by jobs in the 128-512 node size range allocated with nodes from 5 or more dragonfly groups. Given the dominance of 128-512 node size range, we use this range to study routing behavior.

While the number of groups from which a job are allocated has an impact on application performance our findings suggest that it does not appear to impact the ranking or preference of a particular routing bias, that is they are largely independent.

III. Experimental Methodology and Setup

A. Experimental Methodology

In order to evaluate the effects of routing mode changes on network performance and overall perceived application performance, we examine applications running in a production environment (on Cori and Theta) as well as under controlled conditions using Theta. Along with the application study, benchmark analysis was done to analyze MPI behaviors under controlled conditions. The applications were configured using best practices presented in [6], [19] for reducing the effects of runtime noise from other components such as the CPU and OS; specifically, flat memory mode was used on Theta and OS services are moved to the last hardware thread using core-specialization. On Cori, only cache mode is available. These applications were then run in sets executing the same application several times and selecting one of the four adaptive routing modes by setting the environment variables MPICH_GNI_ROUTING_MODE and MPICH_GNI_A2A_ROUTING_MODE. The production experiments were run over the course of a four month period on different days and times to cover a wide range of production network congestion scenarios. When the production experiments were executing, all other jobs on the system were using the system default AD0 setting. This was confirmed by verifying the job logs for the environment variable usage of all jobs on Theta during the test period. Applications were instrumented with AutoPerf [20] to collect metrics on the MPI interfaces used and to collect the Cray network performance counters from the Aries router tiles.

The controlled experiments were performed during a system reservation where all nodes of the system were reserved. These experiments were performed to assess different combinations of routing modes when running in isolation and under congestion scenarios. For the controlled experiments, we also examined the effect of node placement on the routing mode. Applications were run with a compact and dispersed placement using 128, 256 and 512 nodes. The controlled experiments establish the bounds of the system behaviors.

All experiments were performed using sufficient number of samples (greater than 30) such that runtime measurements were statistically significant. Accounting for extreme congestion events such as incast and transient errors, which can disrupt the applications globally, we have removed the outliers from these samples that are above +/- 3 standard deviation of normalized runtimes. The removed samples amounts to less than 0.6% of the total run samples. The default routing mode on both Theta and Cori has been changed recently to AD3, we present the impact of this change on the overall system-level congestion on both the systems.

B. Metrics Collection

As part of the evaluation, metrics were collected using two main tools. AutoPerf is a lightweight intercept library that leverages PMPI wrapping to intercept MPI calls from the application as well as collect Cray network performance counters. AutoPerf has minimal impact on the application with less than 0.05% overhead [20]. AutoPerf reports the number of calls for each MPI interface used, the average number of bytes passed into the call and the total wallclock time used by the calls. The AutoPerf data informs the key MPI interfaces used by each application and how much data the application transfers via MPI. The counters are captured by each MPI process reading the Cray Aries router tiles connected to the node. The result is that the counter values are read for the same router many times. The values are averaged into a single result for each router tile. Collection of the counters at the application results in a local view of data as the application collects data from only routers which it is directly connected.

The other tool used is LDMS [21]. LDMS is a lightweight metric collection service that runs on the compute node and collects the Cray network counters on a (configurable) periodic rate of 1 minute. LDMS collects network counter data from every node on the entire system which provides a system (global) level view of the network congestion. By combining these two tools, we will see that network performance counter data is correlated between the two. We use the application specific coarse-grain counter data to understand network effects on a specific application while we can use the application agnostic fine-grain LDMS data to understand how the overall network behavior is effecting individual applications.

The Aries network counters can be used to approximate the congestion of the network and are an effective measure for the performance of the network and the network congestion [2], [5], [7].

C. Applications and their characterization

The five applications, **MILC**, **Nek5000**, **Qbox**, **HACC** and **Rayleigh**, selected for this study based are representative of applications running on Theta and Cori. These applications regularly run on these systems using significant CPU hours. At NERSC Lattice QCD codes such as MILC represent more than 12% of total core-hours. Both MILC and HACC are two of the top 10 codes run at NERSC by core-hours [18]. The applications also have a set of diverse communication characteristics that are representative of various classes of applications. Table I lists the applications and their MPI/communication characteristics when run using 256 nodes. These applications run in either strong or weak scaling mode. Hence, the communication characteristics could change with the scale they are run at, however the table provides a reasonable approximation.

IV. Application Performance Sensitivity to Routing in Production

We first analyze the sensitivity of the routing bias on applications that were run under normal production conditions. We use two routing modes, AD0 (the default routing mode) and AD3 (strong bias towards minimal) for experimentation. To start with, the performance of MILC, which is known to be communication intensive, is analyzed in detail. The communication pattern in MILC is a 4D stencil with frequent small message MPI_Allreduce operations. The nearest neighbor communication overlaps communication with computation, sending message sizes in the KB range that leverage throughput characteristics of the network. At the end of each neighbor exchange the application is latency bound by small message Allreduces.

A. Performance Analysis of MILC in production

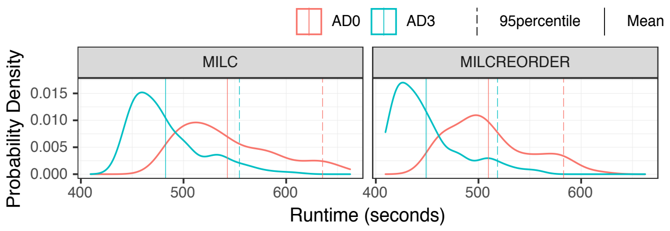

Fig. 2. MILC and MILCREORDER runtimes using 256 nodes on Theta

The runtimes for MILC application using AD0 and AD3 routing modes across several different runs using 256 nodes on Theta is presented in Figure 2. The mean runtime (482 seconds) using AD3 routing mode is 11% lower than the mean runtime (542 seconds) using AD0 routing mode. Similarly, the long tails (95 percentile of the runtimes) are lower using the AD3 routing mode, indicating that the range of variability is lower. Even for the MILCREORDER, both the mean runtime and the range of variability are lower using the AD3 routing mode.

As detailed in Section II-C, job placement, background application traffic and the routing are potential factors influencing the communication portions of the applications. In order to potentially cover the varying range of background noise scenarios, the application was run repeatedly over several months. Thus, these runs were allocated with different sets of nodes spreading across the different parts of the system. Runs organized based on the number of dragonfly groups spanned are shown in Figure 3. The jobs range from a single group up to 12 (maximum on Theta). The figure shows the runtimes of a specific job size normalized using the Z-score normalization.

App	Point-to-point	Collectives	% of MPI in total time	MPI Call1	MPI Call2	MPI Call3
MILC	heavy (KB)	allreduce (8B)	52%	MPI_Allreduce	MPI_Wait	MPI_Isend
MILC REORDER	heavy (KB)	allreduce (8B)	50%	MPI_Wait	MPI_Allreduce	MPI_Isend
Nek5000	medium (KB)	light (16B)	48%	MPI_Allreduce	MPI_Waitall	MPI_Recv
HACC	light (>1MB)	light allreduce (1KB)	22%	MPI_Wait	MPI_Waitall	MPI_Allreduce
Qbox	medium (50KB)	medium (128KB)	66%	MPI_Alltoallv	MPI_Recv	MPI_Wait
Rayleigh	none	heavy (23MB)	28%	MPI_Alltoallv	MPI_Send	MPI_Barrier

TABLE I. COMMUNICATION PROPERTIES OF EACH APPLICATION (BASED ON 256-NODE RUNS). MPI CALLS 1,2 & 3 ORDERED BY % TIME.

The normalized runtime value of zero represents the mean absolute runtime; therefore the positive values, on the y-axis indicate higher than mean runtime (slower) and negative values indicate smaller than mean runtime (faster). As the figure shows, MILC shows variability across the runs irrespective of whether the placement is localized (2 groups) or wide-spread (12 groups).

Figure 3 shows the normalized runtimes for jobs using 128 and 512 nodes as well. The MILC application is run in a strong scaling mode; thus, the absolute communication time per process decreased effectively as the number of nodes used are scaled. The performance of AD3 is consistently better than AD0 irrespective of the placement for 128 and 256 node sizes. However, MILC running using 512 nodes on Theta shows a mean performance decrease of 3% using AD3, and the performance decrease is especially seen when placement is more dispersed. In Section V, we will further discuss the MILC performance on 512 nodes on Theta. However, running on 512 nodes of Cori as shown in Figure 4, MILC shows a mean performance improvement of 6% application time with AD3. The 256-node MILC jobs on Cori showed an average 13.5% improvement in performance with preferring minimal routes. Considering the difference between Theta and Cori in terms of system scale, production workload set and job scheduler policies, the performance improvements noted are reproducible on different systems.

To confirm where in the application the performance improvement is coming from, the MPI profile data for all runs is analyzed. The stacked bar plot (one bar per run) shown in Figure 5 depicts how the runtime for MILC is distributed among MPI and non-MPI (referred as Compute in the plot) operations. The MPI time is further shown as the 3 dominant (in terms of time) MPI operations (MPI_Allreduce, MPI_Wait and MPI_Isend) and the rest of MPI operations. As shown in the figure, the time for these MPI operations is reduced using AD3 routing mode as the latency-bound operations are benefited by the minimal routes.

Next, to verify how the runtime in MPI operations improved, we examine the Aries network performance counters. Figure 6 shows the ratio of stalls to flits on the Aries router tiles broken down by the different link types. An average improvement using a routing that has strong minimal bias is seen across the network tiles. The absolute stalls (not shown here) have reduced significantly, thus reducing the ratio of stalls to flits on all the network tiles. The request and response traffic transmitted on the router tiles make use of separate virtual channels (VC) [3]. The stalls on the request traffic (data moved with Puts than Gets) on the processor tiles have increased with AD3, indicating that potential congestion on the network endpoints. The routing does not affect the response traffic. However, the overall stalls are reduced significantly with AD3.

B. Performance of all Applications in Production

Following the detailed analysis of MILC, we perform the same analysis for the remaining applications. Table II shows mean and standard deviation of the total runtimes for all the applications. The number of samples recorded for each application is also shown testifying the statistical significance. The performance improvement is up to 11.9% on Theta (13.5% on Cori) for production application mean runtime, which is a significant improvement considering that no change to the application is made. The only application which suffered a negative effect from AD3 was HACC, which is discussed in IV-C. The specific improvements in the mean MPI times of the application are also provided, showing a significant increase of up to 19% on Theta and 34% on Cori.

Figure 7 shows the normalized runtime for each application using the two routing modes. A routing mode with strong bias towards minimal shows improvement in the mean runtime as well as variability for all applications other than HACC. Corresponding improvement in the network counter metrics are also measured.

Overall, bias towards minimal shows a positive performance improvement for most production applications. These results

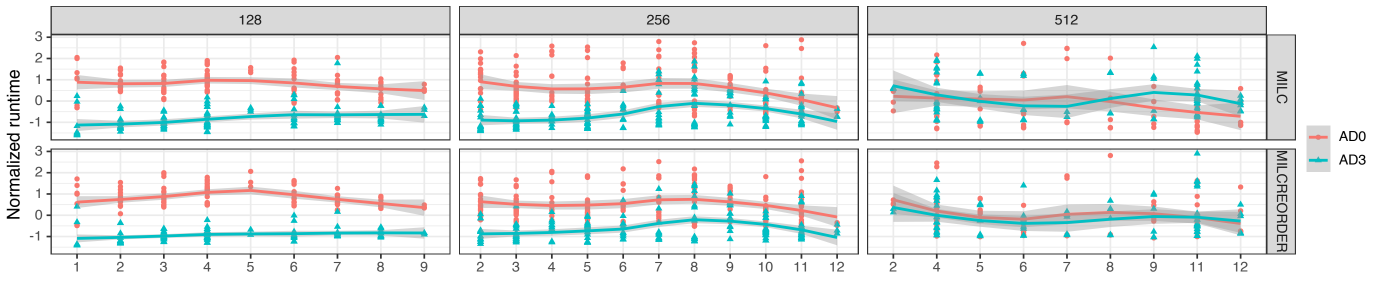

Fig. 3. MILC and MILCREORDER on three different job sizes ordered by the number of groups spanned with AD0 and AD3 routing modes.

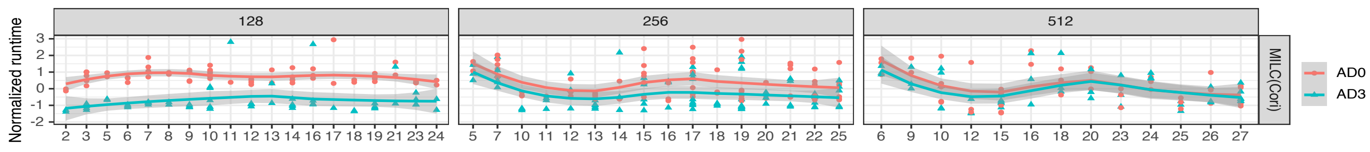

Fig. 4. Cori - MILC runtimes on three different job sizes ordered by the number of groups spanned with AD0 and AD3 routing modes

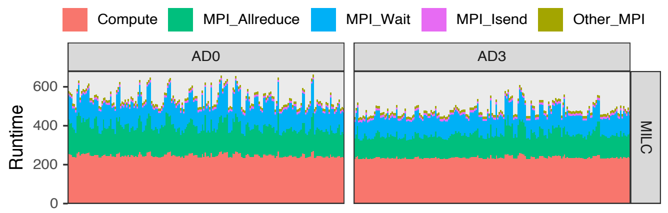

Fig. 5. MILC runtimes shown (MPI and non-MPI times) for different runs

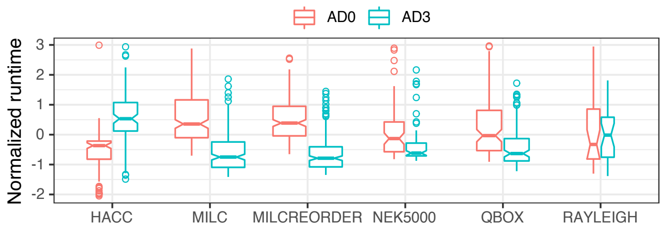

Fig. 7. Normalized runtimes for Applications with AD0 and AD3

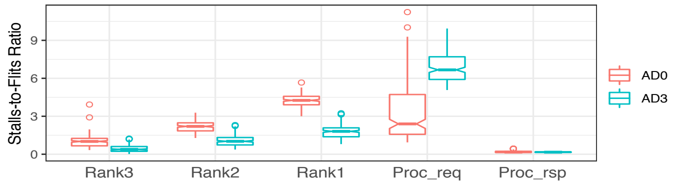

Fig. 6. Performance counters for MILC with AD0 and AD3 routing modes

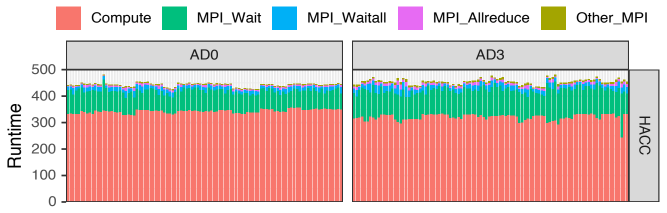

Fig. 8. HACC runtimes shown (MPI and non-MPI times) for different runs

emphasize that the default routing mode (AD0) employed on Theta is not always ideal for all the applications and in fact changing the routing mode showing a significant (up to 11.9%) performance improvement in the total application runtime.

C. *Performance Analysis of HACC in Production*

As shown earlier, HACC is the only application among those we have tested that saw a reduction in performance with strong bias towards minimal. While the decrease in runtime is small (2.7%), we wish to understand the reasons behind this so as to identify scenarios where AD0 routing is preferred.

The communication in HACC is composed of two communication patterns: 3D FFT-type communication pattern and neighbor-wise particle exchange. The 3D FFT is the dominant of the two, with communication going over random rank-pair mappings stressing the global bisection bandwidth. It uses asynchronous send/recv operations of large (1.2MB) messages; these are shown as the MPI_Wait in Figure 8. This communication stressing the global bisection of the network prefers non-minimal routes to avoid the congestion on the rank-3 optical links. Figure 12 highlights this, demonstrating high peak stalls under AD3 on a subset of the rank-3 router tiles due to the limited paths used by minimal routing. Thus, AD0 routing, which has equal bias to minimal and non-minimal, is preferred over a routing with strong minimal bias for this type of workload.

The previous analysis demonstrates the value of a high minimal bias for most applications on the system and highlights where applications should choose an equal bias for minimal and non-minimal routing. However, this production analysis was done with only the target application running with a modified routing protocol while the rest of the system was using AD0 routing. Hence, it is imperative to analyze how a different routing bias influences the overall system congestion and network performance when used by all applications. The next section discusses this along with addressing the question "Is using strong minimal bias routing for all the jobs running on the system good or not?".

V. System Level Network Noise Evaluation

In this section, we analyze the performance impact of routing biases on the overall network performance. We use the

App	AD0 $\mu \pm \sigma$	AD3 $\mu \pm \sigma$	% of improvement in time AD3 over AD0	% of improvement in MPI AD3 over AD0	number of runs
MILC	542.6 ± 46.5	482.5 ± 35.0	11	16.7	190
CORI MILC	668.6 ± 130.2	589.8 ± 102.2	11.7	NA	81
MILCREORDER	509.6 ± 40.0	448.9 ± 33.3	11.9	18.8	189
Nek5000	467.1 ± 21.1	456.7 ± 16.0	2.2	5.5	69
HACC	442.9 ± 8.1	454.9 ± 10.5	-2.7	-34	100
Qbox	677.3 ± 54.5	644.7 ± 37.5	4.8	5.7	108
Rayleigh	653.1 ± 16.6	651.7 ± 12.8	0.2	0	29

TABLE II. Mean (μ), standard deviation (σ) runtime (seconds) and the percentage of improvement in mean runtime with AD3 over AD0 for applications running on 256 nodes.

Aries network performance counters recorded by LDMS as a measure of the global congestion experienced on the network. LDMS on Theta was configured to record the counters globally at a periodic interval of one minute across all the routers on the system. We run multiple instances of the same application in a controlled manner on Theta, where all the instances are of same job size and are run with using the same adaptive routing mode. As an example, an ensemble of sixteen 256-node MILC runs starting them all simultaneously are executed. The ensemble is repeated four times with each time using one of the four adaptive routing modes. Given that the applications and their communication patterns are known, this helps in better interpreting the LDMS data, especially when we need to compare the performance effects of different routing modes on the same workload ensemble. We also have run ensembles of 128-node and 512-node jobs. In addition, job ensemble with jobs using compact placement and job ensemble with jobs using dispersed placement are used.

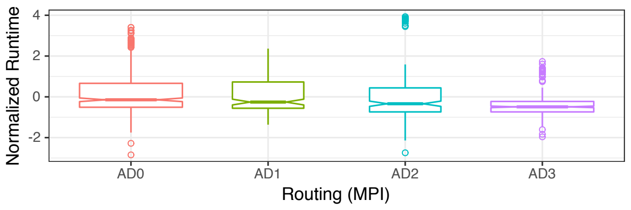

Fig. 9. All applications (256 nodes) varying adaptive routing bias

Figure 9 shows the runtimes of all the tested application jobs using 256 nodes (includes jobs with compact and random placements) in the controlled experiments. The runtimes are Z-score normalized per application using the mean and standard deviations of runtimes of that application. The AD3 routing performs the best in terms of lowest mean runtime and smallest range of run-to-run variability. AD3 which routes packets adaptively with very large bias to minimal paths limits the spread of network congestion, thus showing overall performance benefits. AD2 also routes packets with minimal bias but with a smaller bias value (as stated in Section II-D), and fares next in terms of mean runtime performance but causes few extreme outliers. Interestingly, the AD1 routing which is set as default for MPI_Alltoallv performs slightly better than AD0 for this workload set and job size combination. Overall, both the production and controlled experiments prove the benefit of strong minimal bias routing.

A. *Controlled experiments using MILC*

In order to understand how the performance improvements with strong minimal bias occur and analyze how the global network congestion changes, we run an ensemble of eight 512-node MILC jobs first with AD0 and then with AD3. Figure 10 shows the performance counters on the 48 router tiles on all 1024 Aries routers. A total of 49152 tiles inclusive of the rank-1 (green), rank-2 (grey), rank-3 (blue) and processor (red) tiles are colored accordingly in the figure showing cumulative stalls and flits on all router tiles for the duration of the ensemble. The stalls-to-flits ratio for all different tiles is also shown. Comparing the rank-1, rank-2 and processor values for stalls, we see a clear reduction in the absolute stall counts when running under AD3. This indicates an overall reduction in congestion on the network. Correspondingly, the stall-to-flit ratio is reduced almost 2x. The overall reduction in total flits on rank-1, rank-2, and rank-3 indicates fewer overall packet transmissions throughout the network, which expected when using minimal paths.

The performance improvement of 512-node MILC job with strong minimal bias is in contrast to what we observed with the same in production for Theta. Previously, we speculated that MILC at 512 nodes performed better under AD0 in production because the application was able to use more network resources if and when the system was underutilized. We validate this by demonstrating that under high network load, MILC benefits from AD3 routing.

In order to prove the validity and relevance of the above controlled experiments, it is important to analyze how representative they are with respect to the congestion seen during the production experiments.

1) Production vs. Isolated vs. Controlled experiments using MILC: We compare the congestion as represented by the stalls-to-flits ratio on the network tiles between the production, the isolated and the controlled experiments for the 256-node MILC jobs. The controlled experiments comprise of two different test cases, compact placed (nodes from 1-2 groups), and disperse placed (nodes from 11-12 groups) jobs. Figure 11 (left) shows the ratio using AD0. As per the figure, congestion experienced by isolated and production runs lies within that of the controlled (compact and disperse placed) runs. Thus, the controlled runs serve as a good proxy for the production runs. Figure 11 (right) shows the ratio using AD3. The production runs with AD3 currently lie outside the two compact and one dispersed controlled runs. While these production runs themselves are using AD3, the rest of jobs on the system are

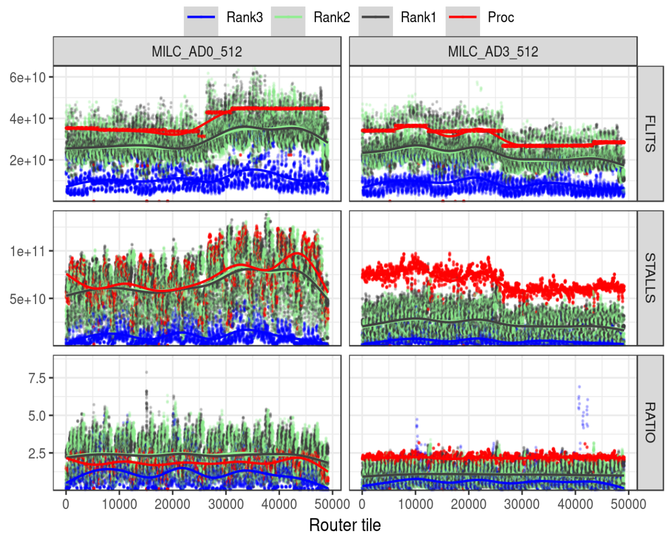

Fig. 10. Eight 512-node MILC jobs running on 4K nodes on Theta (scatter plot showing smoothed trend lines)

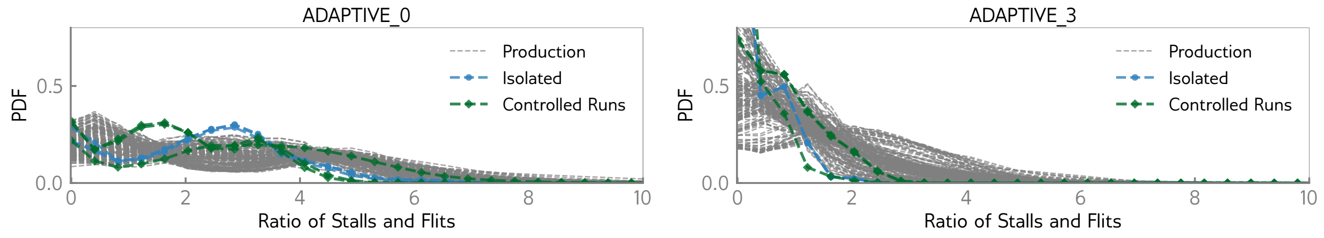

Fig. 11. Stalls-to-flits ratio on 40 Network tiles for MILC 256 node jobs in production, isolated and controlled runs

using the default (AD0) mode. We argue that by enabling AD3 for all jobs on the system, the production runs will shift to the left (higher probability of lower stalls-to-flits ratio) and thus improving the overall network performance.

B. *Controlled experiments using HACC*

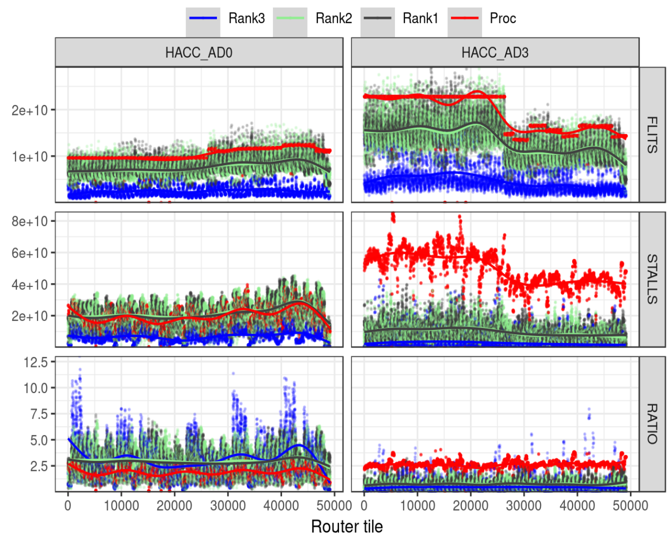

Fig. 12. 16 256-node HACC jobs running on 4K nodes on Theta (scatter plot showing smoothed trend lines)

We next analyze the controlled experiments for HACC using sixteen 256-node runs. The runtimes for HACC increase with shifting more bias towards minimal; this matches with the performance seen for HACC in the production experiments. As shown in Figure 12, with AD3 routing; the flits on all the 4 types of tiles increased. This is an indication of the backpressure seen from the rank-3 (global link) links percolating onto the other links in the network. Owing to the bisection bandwidth requirements of HACC, the more bias towards minimal is potentially leading to more packet retransmissions because of the backpressure, and thus more flit transmissions are observed (this behavior is also shown in [14]). This can be seen in the stalls counters where there are localized peaks on the rank-3 tiles due to concentration on a subset of links resulting from minimal path selections. Because of the localization of the congestion due to strong minimal bias, the stalls on the processor tiles are higher. One other insight from the figure is that analysis of the stalls-to-flits ratio in isolation can be misleading, which is why we have analyzed all the three metrics (stalls, flits and ratio).

C. *System-wide congestion in production before and after changing the default routing mode*

Motivated by our findings with the controlled experiments, both the sites, ALCF and NERSC have now changed the default routing mode in production on Theta and Cori respectively to use AD3. We present here the effects on the system-wide congestion due to this change. Figure 13 shows the system-wide counter metrics for a week of LDMS data from before and after the change to the default routing mode. Around 100 uniformly spaced samples in a one week time window each before and after the change were considered for the analysis. The FLITs communicated in the time windows are roughly in line indicating that the it is reasonable to compared these two specific time windows. There is a marked improvement in the STALLs and correspondingly the stalls-to-flits ratio after changing the default routing mode to AD3. This improvement in the system-level congestion translated into overall application performance improvement as well, which we verified by checking the performance of MILC which showed on an average around 11.8% improvement in performance after the change.

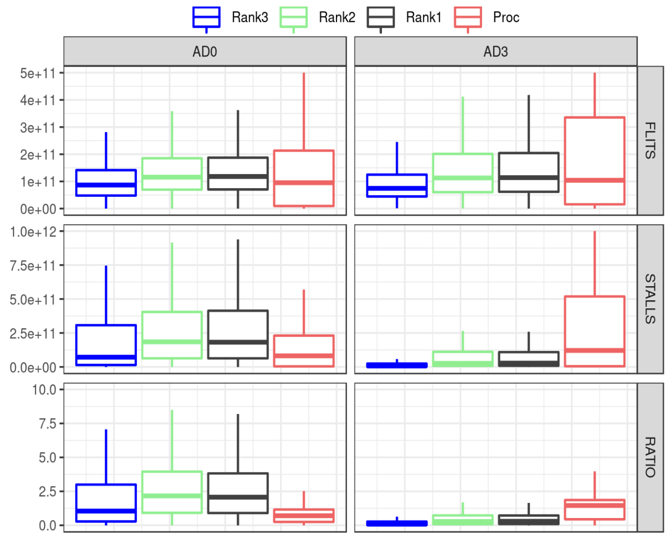

Fig. 13. System-wide counter metrics before (AD0) and after (AD3) changing the default routing on Theta

D. *System-wide packet latency sampling in production*

In order to obtain a comprehensive system-wide measurement of changes to routing bias, we utilized LDMS counters to measure average latency in a production environment.

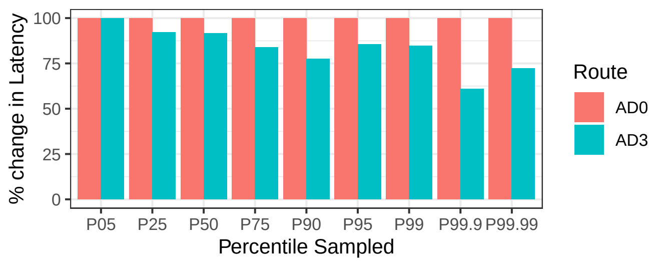

Fig. 14. System-wide packet-pair latencies measured using the NIC counters

Specifically, on Cori we utilize two Aries NIC counters[1], that can provide latency measurement for the observed request-response packet-pairs. The first counter accumulates the latency observed for every packet and the second accumulates the number of packets observed. Together, they allow us to calculate average latency at 1Hz. We then collect samples from two periods of time before and after making changes to routing bias. We sampled each of the more than 12,000 NICs on the system at 100 random points in time within a one week window before making the change to routing policy and then another one week window after the routing changes have propagated. This results in approximately 3 million total samples of average latency. This covers tens of thousands of jobs in the Cori production workload. To summarize the data we then compare the 5, 25, 50, 75, 90, 95, 99, 99.9 and 99.99 percentile of mean latency observed before and after the routing changes. The 99.99 percentile is still a common occurrence given the high overall flit rate, on the order of 1M flits per second, of these systems. This is because many jobs are limited by collective operations like Allreduce, where the collective is limited by the slowest process. Across the board, we see large improvements in mean latency, with tail latencies reduced by 20-30% (918us to 663us for P99.99). Given the latency sensitivity of the majority of the workloads on the production systems, this suggests a significant performance benefit by the switch to the minimal bias routing.

VI. Related Work

There is significant work modeling adaptive routing impact in dragonfly networks. The vast majority of it is evaluated through simulation. Though valuable, simulation does not always capture the scale and variety of workloads needed to make an informed decision regarding production systems.

a) Adaptive routing on simulated dragonfly networks: Some of the earliest simulations to evaluate different routing algorithms (e.g. Valiantand UGAL variants [22]) on dragonfly topologies were simulated by Kim et al. [1], where credit round-trip latencies were used to determine the adaptive routing threshold. Work by Bhatele et al. [23] showed that non-minimal routing on contiguous allocations could provide similar performance benefits to minimal routing with random placement on two-level direct networks – with the caveat that it created additional traffic. Garcia et al. [24] worked to improve adaptive routing by incorporating measures of congestion along intermediate hops in potential non-minimal paths. Roweth et al. [14] found low utilization of optical links in simulations of DOE workloads on dragonfly networks. They saw performance improvements with adaptive routing. Earlier work by Jain et al. [11] examined the relationship between routing, job size, and job placement through simulation. In this study Jain found that non-minimal paths were able to improve performance by distributing traffic more evenly across the network. Yang et al. [13] simulated combinations of placement and routing on dragonfly topologies. They found that system-level performance was significantly improved through the combination of random placement and adaptive routing. Finally, Rahman et al. [25] looked at customizing UGAL for individual dragonfly topologies. While interesting, real production systems are limited by a small set of vendor defined presets. A distinction of our work is that our analysis suggests minimal biased adaptive routing often outperforms routing with no bias, irrespective of whether the job placement is compact or scattered.

b) Adaptive routing on production dragonfly networks: The work that is most similar to ours is that of De Sensi et al. [2], where they use an adaptive runtime to poll Aries NIC counters, capture average latency statistics of packets and adjust the policy if latency increases beyond a threshold. Though we found this approach too expensive to effectively deploy on many-core systems (De Sensi utilized higher-frequency Haswell cores), our work complements theirs by performing a detailed evaluation of the attributes that lead to applications' preference for minimal or non-minimal paths. With this understanding, facilities are able to intelligently deploy the routing policies that best fit their workloads. Furthermore, our work differs from [2], since we focus on real applications that utilize all available cores, as would be expected in production. The UK Met office improved the performance of Weather Forecasting application by tuning the adaptive routing bias towards minimal for both MPI and IO traffic [8]. Smith et al. [26] explored deficiencies in routing algorithms on current production systems (both dragonfly and fat-tree). Using global flow information they observed a performance benefit using AFAR routing, but only evaluated on a fat-tree topology.

To the best of our knowledge, our work represents the only comprehensive study examining the role of adaptive routing bias on production dragonfly systems and the first to study system-wide performance using multiple applications. Furthermore, our study uses network hardware counters to examine how the location of congestion in the system relates to best practices in routing.

VII. Conclusions

The network performance of the dragonfly-based Aries systems is determined by the efficient operation of the adaptive routing. The congestion management approaches on networks such as Aries are not necessarily effective because the non-minimal routing can end up spreading the congestion. How-

[1]AR_NIC_ORB_PRF_NET_RSP_TRACK2:SUM_RSP_TIME_COUNT and AR_NIC_NETMON_ORB_EVENT_CNTR_RSP_NET_TRACK

ever, the use of non-minimal paths by the adaptive routing exploits the path diversity and load-balances the network channels. The key is in finding the right balance in the usage of minimal and non-minimal paths, and hence setting the routing bias parameter is performance-critical for networks that use adaptive routing.

We have shown through a detailed evaluation, using real applications on two production systems, the deficiencies in the performance of the default routing. Tuning the routing to be strongly biased towards minimal has proven to be consistently better and showed appreciable performance improvements in the total application time without making any application modifications. The benefits of minimal bias routing were observed for both compact and scattered process placement. Also, we showed that minimal biased routing resulted in reduced performance variability of applications.

This improvement in the mean runtime, as well as reduced run-to-run variability, allowed us to vouch for advocating the use of this policy as the default for Aries dragonfly-based systems. Or rather, our study motivates facilities to evaluate the adaptive routing defaults based on their workload set. Also, we have analyzed the overall system-level congestion, providing insights on the operating of the routing algorithms to understand and validate the empirical results obtained. The results of our experiments and the evaluation methodology will guide how we operate large system interconnects in the future. Dragonfly-based networks will be critical to the performance of the upcoming Exascale class supercomputing systems. Understanding how to optimize the routing policies is crucial for unlocking the full potential of these machines, and we have provided a solid basis for this understanding.

References

[1] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *Proceedings - International Symposium on Computer Architecture*, 2008, pp. 77–88.

[2] D. De Sensi, S. Di Girolamo, and T. Hoefler, "Mitigating network noise on dragonfly networks through application-aware routing," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '19. New York, NY, USA: ACM, 2019, pp. 16:1–16:32.

[3] B. Alverson, E. Froese, and D. Roweth, "Cray XC Series network," *Cray*, pp. 1–28, 2012. [Online]. Available: www.cray.com

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

[5] T. Groves, Y. Gu, and N. J. Wright, "Understanding performance variability on the Aries dragonfly network," in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, Sept 2017, pp. 809–813.

[6] 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," *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis - SC '17*, pp. 1–13, 2017.

[7] A. Bhatele, J. Thiagarajan, T. Groves, R. Anirudh, S. Smith, B. Cook, and D. Lowenthal, "The case of performance variability on dragonfly networks," in *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2020.

[8] P. Selwood, *How the Met Office Solved a Weather Forecasting Runtime Scare*, 2018 (accessed Oct 19, 2020). [Online]. Available: https://www.cray.com/blog/met-office-solved-weather-forecasting-runtime-scare

[9] A. C. Gentile, J. M. Brandt, A. M. Agelastos, J. M. Lamb, K. P. Ruggirello, and J. O. Stevenson, "Contention and congestion: Challenges and approaches to understanding application impact." in *SIAM Conference on Computational Science and Engineering*, 3 2017. [Online]. Available: https://www.osti.gov/servlets/purl/1425315

[10] 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*, 2014, pp. 129–140.

[11] N. Jain, A. Bhatele, X. Ni, N. J. Wright, and L. V. Kale, "Maximizing throughput on a dragonfly network," in *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2014, pp. 336–347.

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

[13] 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.

[14] D. Roweth, R. Barrett, and S. Hemmert, "Early results from the aces interconnection network project," in *CUG 12: Proceedings of Cray User's Group (CUG) Meeting*, 2012.

[15] A. Bataineh, T. Court, and D. Roweth, *Increasingly minimal bias routing*, 2 2017. [Online]. Available: http://patft.uspto.gov/netacgi/nph-Parser?patentnumber=9577918

[16] W. J. Dally and B. P. Towles, *Principles and practices of interconnection networks*. Elsevier, 2004.

[17] M. E. Papka, J. Collins, B. Cerny, and N. Heinonen, "2018 Annual Report - Argonne Leadership Computing Facility," *Annual Report*, 1 2018.

[18] B. Austin, "NERSC-10 Workload Analysis," *Report*, 4 2020. [Online]. Available: https://portal.nersc.gov/project/m888/nersc10/workload/N10_Workload_Analysis.latest.pdf

[19] H. Pritchard, D. Roweth, D. Henseler, and P. Cassella, "Leveraging the cray linux environment core specialization feature to realize mpi asynchronous progress on cray xe systems," *Proceedings of the Cray User Group Conference*, vol. 79, p. 130, 2012.

[20] S. Chunduri, S. Parker, P. Balaji, K. Harms, and K. Kumaran, "Characterization of mpi usage on a production supercomputer," in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2018, pp. 386–400.

[21] A. Agelastos, B. Allan, J. Brandt, P. Cassella, J. Enos, J. Fullop, A. Gentile, S. Monk, N. Naksinehaboon, J. Ogden, M. Rajan, M. Showerman, J. Stevenson, N. Taerat, and T. Tucker, "The Lightweight Distributed Metric Service: a scalable infrastructure for continuous monitoring of large scale computing systems and applications," in *International Conference for High Performance Computing, Networking, Storage and Analysis, SC*, vol. 2015-Janua, no. January, 2014, pp. 154–165.

[22] A. Singh, "Load-balanced routing in interconnection networks," Ph.D. dissertation, Stanford University, 2005.

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

[24] M. Garcia, E. Vallejo, R. Beivide, M. Odriozola, C. Camarero, M. Valero, J. Labarta, C. Minkenberg *et al.*, "On-the-fly adaptive routing in high-radix hierarchical networks," in *2012 41st International Conference on Parallel Processing*. IEEE, 2012, pp. 279–288.

[25] M. S. Rahman, S. Bhowmik, Y. Ryasnianskiy, X. Yuan, and M. Lang, "Topology-custom ugal routing on dragonfly," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019, pp. 1–15.

[26] S. A. Smith, C. E. Cromey, D. K. Lowenthal, J. Domke, N. Jain, J. J. Thiagarajan, and A. Bhatele, "Mitigating inter-job interference using adaptive flow-aware routing," in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2018, pp. 346–360.