# Understanding and Analyzing Interconnect Errors and Network Congestion on a Large Scale HPC System

Mohit Kumar[♮], Saurabh Gupta[†], Tirthak Patel[◇], Michael Wilder[°]
Weisong Shi[♮], Song Fu[§], Christian Engelmann[‡], and Devesh Tiwari[◇]

[♮]Wayne State University [†]Intel Labs [°]UT Knoxville
[§]University of North Texas [‡]Oak Ridge National Laboratory [◇]Northeastern University

***Abstract*—Today's High Performance Computing (HPC) systems are capable of delivering performance in the order of petaflops due to the fast computing devices, network interconnect, and back-end storage systems. In particular, interconnect resilience and congestion resolution methods have a major impact on the overall interconnect and application performance. This is especially true for scientific applications running multiple processes on different compute nodes as they rely on fast network messages to communicate and synchronize frequently. Unfortunately, the HPC community lacks state-of-practice experience reports that detail how different interconnect errors and congestion events occur on large-scale HPC systems. Therefore, in this paper, we process and analyze interconnect data of the Titan supercomputer to develop a thorough understanding of interconnects faults, errors and congestion events. We also study the interaction between interconnect, errors, network congestion and application characteristics.**

***Index Terms*—Cray, Gemini, Interconnect, Titan, Errors**

## I. Introduction

Fast computing devices, network interconnect, and back-end storage systems enable modern High Performance Computing (HPC) facilities to deliver performance in the order of petaflops. HPC systems consist of tens of thousands of processors which require an advanced interconnect network to minimize system latency and maximize throughput and scalability for tightly-coupled parallel scientific applications.

Performance of the interconnect depends on network topology, routing methods, flow-control algorithm, resilience mechanism, congestion reaction mechanism, and communication pattern of applications. Current scientific applications run multiple processes on different compute nodes, and thus, rely heavily on fast network messages to communicate and synchronize frequently. Subsequently, interconnect resilience and congestion resolution mechanisms have a major impact on the overall interconnect and application performance.

Unfortunately, the HPC community lacks state-of-practice experience reports that detail how different interconnect errors and congestion events occur on a large-scale HPC system. Therefore, in this paper, we study the interconnect resilience and congestion events on Titan, the fastest open-science supercomputer in the world. We used daemon services on Titan to collect useful interconnect resilience and congestion events data for over a year. We process and examine this data to develop a thorough understanding of interconnect faults, errors, and congestion events. We also investigate how these errors affect network congestion at different granularity. Our analysis addresses the following concerns:

- What are the major interconnect faults and errors?
- What are the key characteristics of different interconnect errors and network congestion events?
- What is the interaction between interconnect errors, network congestion, and application characteristics?

This study exploits various daemons, such as netwatch and nlrd that use Memory Mode Register (MMR), for collecting and logging interconnect related events. However, analysis of this data presents several challenges. First, the collected data is highly noisy and hence, needs to be filtered discreetly for accurate analysis. Second, the log patterns differ for the same type of events across different logging mechanisms. This requires the development of unified format types for different events. Finally, as data is distributed across several nodes and storage locations, it requires performing multi-source analytics to ensure consistency and accuracy. The following are the highlights of our analysis:

- **Interconnect Errors:** The magnitude of interconnect errors is very high. These errors are distributed unevenly across different types of links within and across cabinets.
- **Spatial Correlation:** Some interconnect errors have a strong spatial correlation among them. On the other hand, some errors show counter-intuitive patterns.
- **Congestion Events:** Network congestion events are highly frequent and bursty. These events are not homo-

geneously distributed across blades.

- **Application Characteristics:** Applications and users causing network congestion and high communication intensity have unique job characteristics.

In the following sections, we provide an analysis of the interconnect data and explore the above insights in detail. Given the lack of field data and analysis on interconnects, we believe our study addresses an important topic and would be useful for current and future HPC systems.

## II. Background

This study primarily studies data from the Titan supercomputer; however, it's insights are applicable to other supercomputers as well. Titan is a 27.1 petaflop supercomputer consisting of 18,688 compute nodes, each with a 16-Core AMD Opteron CPU and an NVIDIA Tesla K20x GPU. It has a total system memory of 710 TB. The supercomputer is divided into 200 cabinets in 25 rows and 8 columns. Each cabinet consists of three cages and each cage has eight blades. Each blade consists of two application specific integrated circuits (ASICs). Each ASIC has two network interface controllers (NICs) and a 48-port router. Each NIC within an ASIC is attached to one node using a HyperTransportTM 3 link [1].

### A. Titan Network Architecture

Titan follows a 3D torus topology using the Cray Gemini Interconnect in which each ASIC is connected to six of its nearest neighbors in X+, X-, Y+, Y-, Z+, and Z- dimensions. The X, Y, and Z dimensions track the rows, columns, and blades, respectively [2]. Nodes that are close physically may not be close topologically as Cray follows a "folded torus" architecture to minimize the maximum cable length. In the X and Y directions, every other cabinet is directly connected together with "loopback" cables. In the Z dimension, the uppermost chassis is connected to the lowermost chassis.

In a 3D torus design, each ASIC is connected to the network using 10 torus connections, two each in X+, X-, Z+, Z-, and one each in Y+ and Y- [1]. Each torus connection has four links where each link is composed of 3 lanes. Therefore, each connection consists of 12 lanes, providing 24 lanes in the X and Z dimensions, and 12 lanes in the Y dimension. A lane provides bi-directional communication between two ports.

### B. Interconnect Resilience

The Gemini Interconnect is tolerant to various types of failures and errors. It supports 16-bit packet Cycle Redundancy Checks (CDCs) to protect packets at each ASIC it passes through before reaching the final ASIC, packets on the receiving ASIC and packets transitioning from the router to the NIC. Link control Blocks (LCBs) on ASICs implement a sliding window protocol to provide reliable delivery of packets. Memory on each ASIC is protected using Error-Correcting Codes (ECCs). ASICs can withstand lane failures as long as there is at least one functional lane in a link. Whenever a lane fails, it is deactivated and the traffic is balanced over the remaining lanes. In such situations, the network operates in a degraded mode. The interconnect tries to reinstate the failed lane to restore the full bandwidth within a user-specified time limit.

TABLE I: Summary of Netwatch events

| Events | Count | Percent |
|---|---|---|
| All | 9367031.0 | 100.0 |
| Mode Exchanges | 5065536.0 | 54.08 |
| RX | 2146693.0 | 22.91 |
| TX | 2144221.0 | 22.89 |
| Link Inactive | 7280.0 | 0.08 |
| Bad Send EOP Error | 2548.0 | 0.03 |
| Send Packet Length Error | 366.0 | 0.004 |
| Routing Table Corruption Error | 200.0 | 0.002 |
| HSN ASIC LCB lane(s) reinit failed Error | 187.0 | 0.002 |

The lanemask value determines the current state of the lanes in a link. It is a three-bit number corresponding to the three lanes in a link. When all three lanes in a link fail and the lanes are not recovered in the configured number of attempts, the link is marked as inactive and the link failover protocol is triggered. When an entire link fails, the Cray Network Link Recovery Daemon (nlrd) on the System Management Workstation (SMW) quiesces the network traffic, computes new routing tables, and assigns them to each ASIC.

### C. Network Congestion

A network becomes congested when there is more data in the network than it can accommodate for. The Hardware Supervisory System (HSS) software manages the network congestion into the network whenever necessary. Two daemons: one on the SMW (xtnlrd), and one on the blade controller (bcbwtd), can handle network congestion by limiting the aggregate injection bandwidth across all compute nodes to less than the ejection bandwidth of a single node (also known as throttling).

### D. Dataset

The collected dataset consists of the network logs from January 2014 to January 2015. The interconnect metadata is collected by two daemons: xtnetwatch and xtnlrd. The xtnetwatch daemon logs the system High-Speed Network (HSN) faults for LCBs and router errors. These logs include details about the transmitting packets, receiving packets, mode exchanges, lane mask, link inactive and different interconnect failures data for particular nodes, along with a timestamp. The xtnetwatch data is summarized in Table I.

When the percentage of time that traffic tries to enter the network is stalled more than a high water mark threshold, the xtnlrd daemon produces log files that include various collection information. It also collects a list of the top 10 applications sorted by the aggregate ejection bandwidth whenever a congestion protection event occurs. Moreover, it estimates the top 10 most congested nodes sorted by ejection flit counts whenever a congestion protection event occurs. In both cases, it includes the job characteristics of the applications running

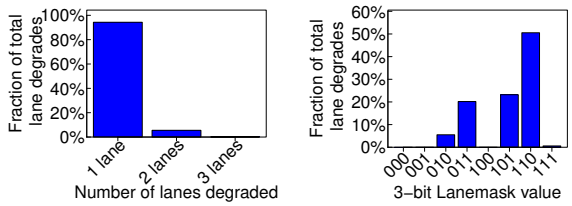


Fig. 1: Frequency distribution of lane degrade events.

on those nodes, including APID, number of nodes, the user ID, and the application name.

## III. Analysis of Interconnect Errors

In this Section, we characterize and analyze different types of interconnect faults and errors. First, we quantify and characterize lane degrade events. A lane degrade event is triggered when any one of the three lanes in a link goes down. This has a negative impact on the application performance and may cause network congestion. Unfortunately, these events occur with a very high frequency. We observed that lane degrade events take place at a high rate of one event per minute. Despite the high frequency and negative consequences of these events, the characterization of these events in an HPC system is not available to researchers, users, and system operators.

Fig. 1 (left) shows the frequency of different types of lane degrades. We observe that in more than 90% cases only one lane in a link is degraded. Two lanes are degraded in less than 10% of the cases. Three lanes are degraded relatively less frequently (<1%). When all three lanes are in a degraded state, the link is declared inactive (or failed), and an alternate route is computed for packets. While link inactive or failed events happen relatively less frequently, they do occur about 28 times per day on average, and cause more disruption than single or double lane degrades.

Fig. 1 (right) shows the frequency of lanemask values for every instance of lane degrade events. We note that a lanemask bit value of 0 indicates that the corresponding lane is degraded. For example, a lanemask value of 5 (binary value 101) indicates that the middle lane is degraded. We observe that the frequency of lanemask values indicates that even single lane failures vary significantly. Lanemask value 110 is two times more frequent than lanemask values 101 and 011. Interestingly, for two lane failures, the corner lanes failing together (010) is more likely than adjacent lanes failing together (001 and 100).

In the absence of per-lane and per-link based utilization data, we hypothesize that lane failure location indicates the utilization and load pattern on links. Given this, our results indicate that the load among lanes within a link may vary significantly. This finding should encourage designers to balance the load more homogeneously and not overload the rightmost lane. This insight could also be exploited for power optimization in interconnect links where rightmost lanes need not to be switched-on at all times.

Next, we plot the relative frequency distribution of lane degrades over time in Fig. 2. We make two important observations. First, lane degrades are not limited to a specific time

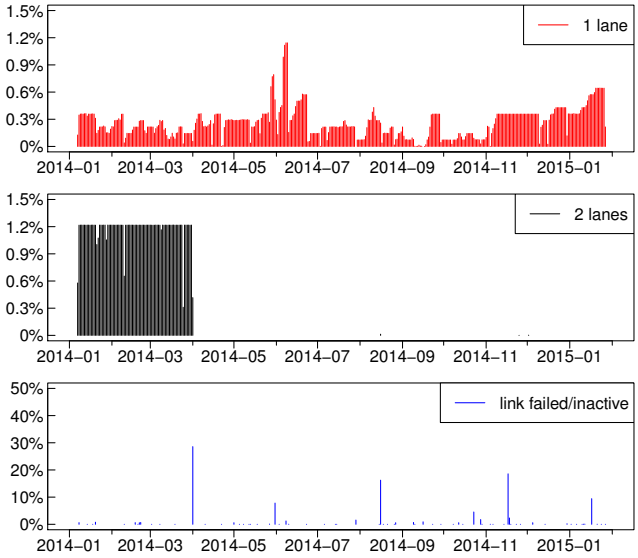


Fig. 2: Frequency of different types of lane degrades over time.

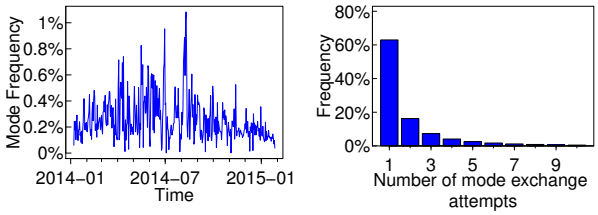


Fig. 3: Daily frequency of mode exchanges to repair lane degrades (left) and number of mode exchange attempts before successful recovery of the lane (right).

period, instead they happen continuously over time. Second, one may expect that the high single-lane degrade events will lead to an increase in the count of two-lane degrades and link failures. However, our field data suggests that this hypothesis is not necessarily true. For example, peaks in two-lane degrades are not necessarily during the high intensity of one-lane degrades. Later, we also investigate deeper to understand the correlation between network congestion and the period of high intensity of lane degrades.

When a lane goes down, the network resiliency mechanism attempts to bring the lane back up via multiple repair events, called mode exchanges. Fig. 3 shows the frequency of mode exchange events over time and the number of mode exchange attempts before a lane is brought up successfully. As expected, the frequency of mode exchange events over time is similar to that of lane degrades. System operators of Titan have set the threshold for the number of attempts allowed to restart a lane to 256. Interestingly, our result shows that more than 85% of the lanes can be restored in three or fewer attempts. Furthermore, more than 99% of the lanes can be restored within 10 attempts.

Next, we attempt to understand how lane degrade and link inactive/failed events are distributed across the system spatially. Fig. 4 shows the lane degrade events for links in and across cabinets. First, we observe that several hot spots exist for lane degrade events in the system. We conduct

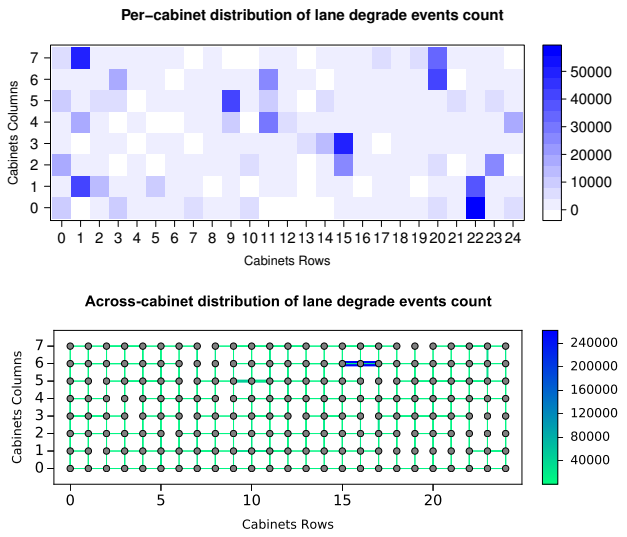


Fig. 4: Spatial distribution of lane degrades inside and across cabinets. Due to the folded 3D-torus design, cross-cabinet links connect to alternate cabinets.

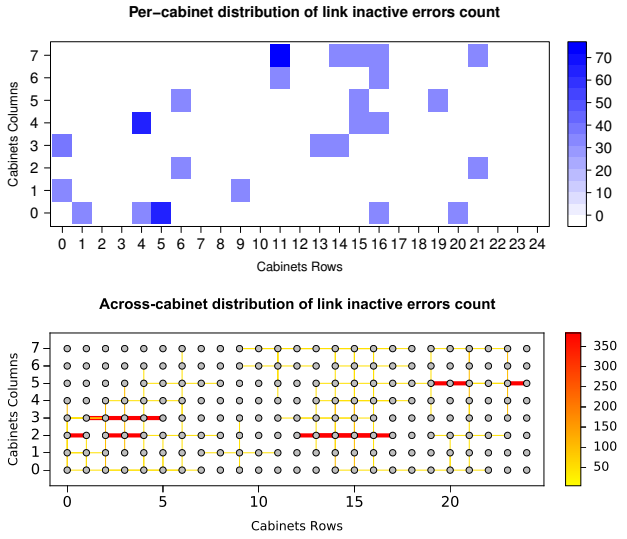


Fig. 5: Spatial distribution of link inactive errors inside and across cabinets. Due to the folded 3D-torus design, cross-cabinet links connect to alternate cabinets.

the Kolmogorov-Smirnov test (K-S test) to test whether our sample of spatial distribution of lane degrades per cabinet has a uniform distribution. The test results show D-statistic = 1, p-value = 2.2e-16. For our sample size of 200 cabinets, the critical D-value for a 0.05 level of significance is 0.0960, and therefore the null hypothesis (i.e., the sample is taken from a uniform distribution) can be rejected. This shows that the spatial distribution of lane degrades per cabinet is significantly different than uniform. This behavior can be a combination of factors, although accurate root-cause analysis is not possible. External transient effects, overloading of links, uneven usage, and complex interaction between applications and interconnect network contribute toward such a behavior.

Interestingly, we note that the hot spots for links contained within the cabinet are not the same as the hot spots for links crossing cabinet boundaries. Second, when we compare lane degrade hot spots with the hot spots of link inactive errors (Fig. 4 vs. Fig. 5), we find that they do not necessarily match. This also explains why their high intensity periods do not match (Fig. 2). This indicates that it is not possible to determine the location of link inactive/failed errors by only observing the time and location of lane degrade events.

Next, we investigate other interconnect errors: Bad Send EOP error, Send Packet Length error, Routing Table Corruption error, and HSN ASIC LCB lane reinit failed error.

**Bad Send EOP error:** Each packet in Gemini Interconnect has a single phit end-of-packet that contains the last phit of a packet, and the status bits for error handling [1]. If a packet is corrupted, the end-of-packet is marked as bad and will be discarded at its destination.

**Send Packet Length error:** Send Packet Length error occurs when the length of a packet does not match with the expected length value at destination.

**Routing Table Corruption error:** A routing table is a data table stored in a router that contains the information necessary to forward a packet along the best path toward its destination. When a packet is received, a network device examines the packet and matches it to the routing table entry providing the match for its destination. The table then helps in guiding the packet to the next hop on its route across the network. A routing table corruption results in link failure and eventually causes network congestion.

**HSN ASIC LCB lanes reinit failed error:** This error occurs when all the 256 attempts to bring up a downgraded lane are exhausted.

All these errors also show hot spots in and across cabinets, although we found that the magnitude of these errors is relatively small. For example, Routing Table Corruption error occurs only 200 times while HSN ASIC LCB lane(s) reinit failed error happens only 187 times throughout the entire observation period. On deeper investigation, we found that most of these errors are highly correlated with link inactive/failed errors. Table II shows the correlation of these interconnect errors with link inactive errors. This indicates that link inactive errors can be used to predict other interconnect errors. We also found that more than 80% of link failed errors lead to Bad Send EOP, Send Packet Length, and Routing Table Corruption errors. We also found that HSN ASIC LCB lane(s) reinit failed error has a weak correlation with link failed errors. This can be explained by our previous findings where it showed that lane degrades and link failed errors are not correlated and ASIC errors are an outcome of failed repair attempts of lane degrades.

## IV. Analysis of Network Congestion

Understanding network congestion in conjunction with interconnect errors is important since it is likely that one may cause the other. A daemon on the compute cluster monitors the percentage of time that network tiles are stalled due to increased traffic or other reasons. When these values cross a

TABLE II: Correlation factor between link inactive and other interconnect errors.

| Errors | Link Inactive |
|---|---|
| Bad Send EOP Error | 0.99 |
| Send Packet Length Error | 0.96 |
| Routing Table Corruption Error | 0.84 |
| ASIC Error | 0.04 |

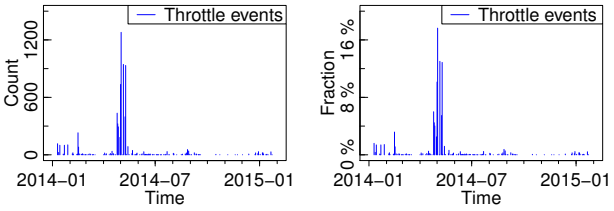


Fig. 6: Count of network throttle events over time (left), and relative frequency of throttle events over time (right).

set threshold, the daemon communicates this data to the xtnlrd daemon running on the SMW. After the congestion subsides, the daemon again passes this information to the SMW.

In this section, we first attempt to understand the characteristics of network throttle events. Fig. 6 (left) plots the network throttle events over time. We note that a large fraction of throttle events occur in a short period of time. We also note that each throttle event is typically 20-30 seconds, but it can also last up to a few minutes depending on the magnitude of the congestion observed. As shown in Fig. 6 (right), network throttle events can be quite bursty. An application that is causing network congestion can induce multiple throttles in a very short amount of time ($<$ 20 mins). Fig. 7 shows the network throttle events over time, counting only one event at maximum per hour. We experimented with multiple time windows and found that a 1-hour time window removes the skewness. However, this type of time window filtering cannot completely remove the skewness since a long-running communication-intensive application may cause multiple throttle events over multiple hours. For example, Fig. 8 shows that without 1-hour filtering the mean time between throttle events is less then 1 minute, with 90% of the events occurring within the first hour of the preceding throttle event. Even when we apply 1-hour filtering, the meantime between throttle events is approximately 22 hours. Therefore, to better understand the characteristics of network throttle events we analyze our subsequent results without any time window based filtering and with 1-hour time window based filtering.

Naturally, one may hypothesize that the lane degrades/failures may induce the network throttle events or vice versa. Therefore, we investigate the possibility of temporal correlation between the time series of throttle events and interconnect errors, in particular lane degrades and link failed events. We found the Spearman correlation coefficient to be very weak (0.03). This result indicates that lane degrades/failures alone cannot be used to predict throttling events.

Next, we plot the heatmap of compute blades that were throttled due to these network throttle events. Fig. 9 shows

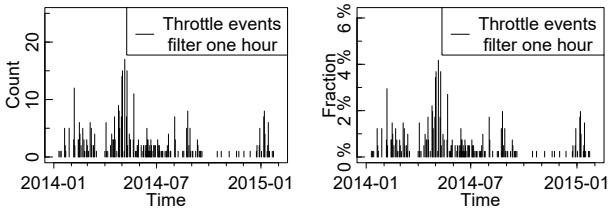


Fig. 7: Network throttle events with 1-hour filtering (left), and relative frequency of throttle events with 1-hour filter (right).

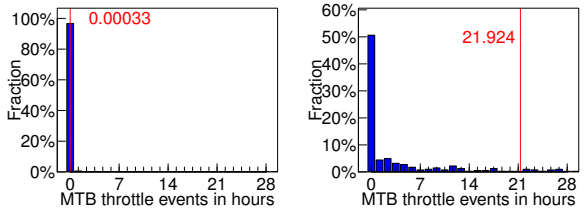


Fig. 8: Mean time between network throttling events without filter (left) and with 1-hr filter (right).

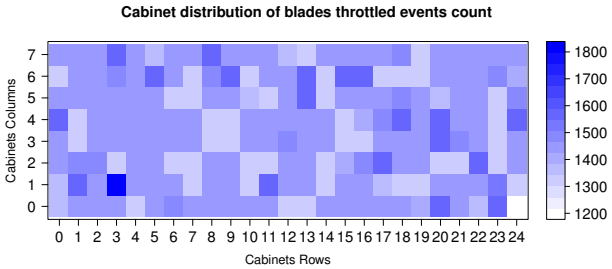


Fig. 9: Spatial distribution of throttled blades without filter.

the heatmap without any filtering and Fig. 10 shows the same heatmap with 1-hour filtering. As expected, we observe that not all blades are throttled equally over the period of observation. Interestingly, hot spots remain similar even after applying filter. We again conduct the K-S test to test whether our sample of the spatial distribution of blades throttle events per cabinet without and with one hour filter has a uniform distribution. The test results show D-statistic = 1, p-value = 2.2e-16 in both cases. For our sample size of 200 cabinets, the critical D-value for a 0.05 level of significance is 0.0960, and therefore the null hypothesis (i.e., the sample is taken from a uniform distribution) can be rejected. This shows that the spatial distribution of blade throttled events per cabinet is significantly different than uniform. As a next step in our analysis, we want to investigate the role that congestion information at the node and application levels can play in improving our understanding of congestion behavior at the

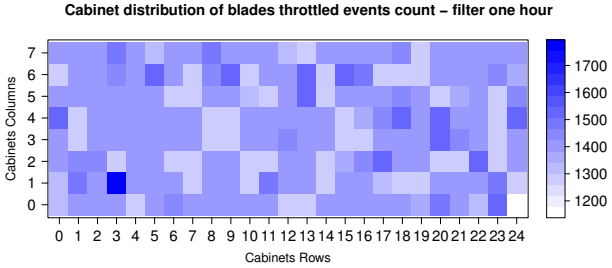


Fig. 10: Spatial distribution of throttled blades with 1-hr filter.

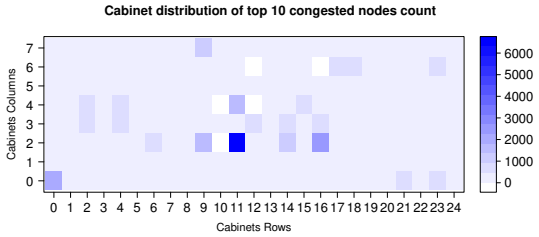


Fig. 11: Spatial distribution of congested nodes without filter. Note that the top 10 congested nodes are calculated for each throttle event. This plot is aggregated over all throttle events.

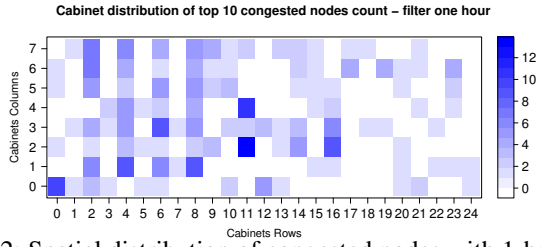


Fig. 12: Spatial distribution of congested nodes with 1-hr filter.

blade-level.

Node-level congestion data provides information about the nodes which are heavily congested at the time of throttling. Fig. 11 and 12 show the spatial distribution of congested nodes in the Titan supercomputer for the no filter and 1-hour filter cases, respectively. We make two observations. First, some nodes are much more congested than others as shown by the uneven distribution of congested nodes. This is because the applications creating significant network traffic may be repeatedly getting scheduled on the same nodes. Second, applying 1-hour filtering shows that the spatial distribution evens out compared to Fig. 11. However, interestingly it continues to show uneven distribution; skewed toward the left part of the supercomputer. This indicates that communication-intensive applications are not scheduled evenly across the cabinet. Applications scheduled on the left part are likely to see more performance impact due to network congestion.

We also calculated the correlation coefficient between the spatial distribution of congested nodes and lane degrades/failures. The Spearman correlation coefficient was close to zero (0.01); the nodes with high ejection bandwidth are not strongly correlated with the interconnect errors. This result was expected since we found the correlation between throttle events and interconnect errors were not high. However, surprisingly, there is a low correlation between the heatmap of congested nodes and throttled blades (Spearman correlation 0.01). This is because congested node information is collected after the throttle command has been issued, so it may not capture the nodes which actually caused the congestion and induced throttling. This also indicates that the aggregate network traffic at the blade can be potentially different than individual node-level traffic. Future network performance tools should focus on building more accurate and fine-grained tools that can detect the root cause in real time.

Next, we analyze the characteristics of applications running on these congested nodes. First, we plot the frequency of

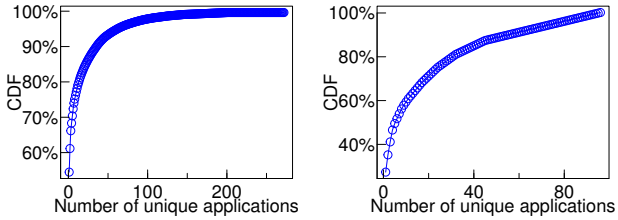


Fig. 13: Distribution of unique applications over congested node events without filter (left), and with 1-hr filter (right).

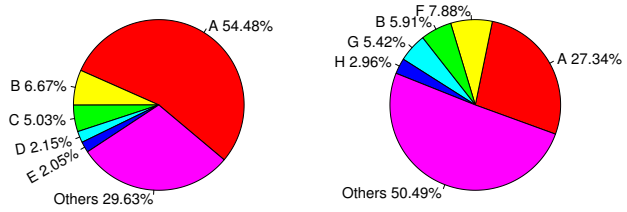


Fig. 14: Fraction of top 5 congestion-causing and other applications without filter (left), and with 1-hour filter (right).

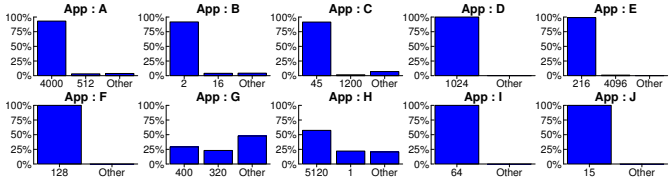


Fig. 15: Job size distribution of top 10 congestion-causing applications.

unique applications that were running on congested nodes when the throttling events occurred. Fig. 13 shows that only a few applications tend to dominate. We refer to these applications as congestion-causing applications in our discussion; however, we note that these applications may not be necessarily responsible for increasing the congestion that eventually resulted in the network throttling event. For example, 5 applications alone appear in more than 70% of congested node reporting events (Fig 14), while more than 250 unique applications are logged in total across all congested node reporting events. Interestingly, when 1-hour filtering is applied, the number of unique applications decreases significantly. The top 5 most frequently occurring applications appear only in approx. 50% of congested node reporting events (Fig 14). Total number of unique applications go down from 250 to 90. Reduction in the number of unique applications clearly indicates that when multiple throttle events occur in the small time period, they are not because of the same application. In fact, it turns out that within a 1-hour time window, multiple unique applications can cause nodes to be highly congested. We also see the same results on a per-user basis with and without filtering. These results confirm that applications and users can work as proxies for each other.

Next, we analyze the job size of these applications. We limit our discussion to the top 10 most frequent applications. As application names can be business-sensitive, we identify them with English letters. Fig. 15 shows the job size distribution of top 10 applications that appear most frequently on congested

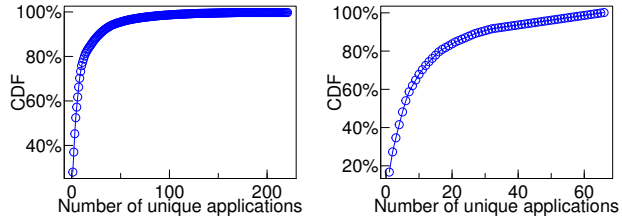


Fig. 16: Distribution of unique applications over top bandwidth-heavy application events without filter (left), and with 1-hr filter (right).

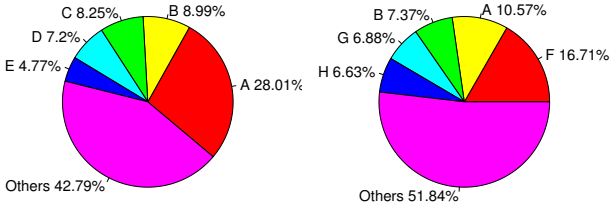


Fig. 17: Fraction of top 5 bandwidth-heavy and other applications without filter (left), and with 1-hr filter (right).

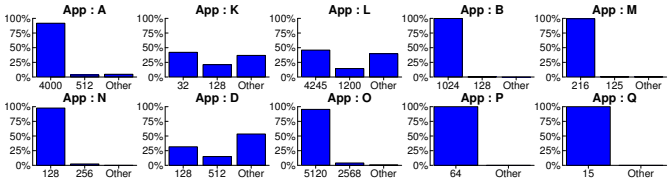


Fig. 18: Job size distribution of top 10 bandwidth-heavy applications.

nodes. We make several interesting observations. First, most of the applications tend to run on the same number of nodes every time they appear in the congested node reporting events. For example, applications A, B, and C run on 400, 2, and 45 nodes, respectively, for more than 90% of the time that they appear in the congested node reporting events.

Moreover, counter-intuitively, the job sizes of these applications are relatively small. For instance, 7 out of the 10 applications most frequently have a job size of less than 512 nodes. In fact, 5 applications have the most frequent job size of less than 128 nodes. In such cases, the many-to-few communication pattern can be responsible for congesting the nodes (high ejection bandwidth). Therefore, only focusing on large scale jobs for identifying culprit applications is an ineffective strategy. Our results show that node congestion is caused by small-scale applications in real-world scenarios.

Next, we want to extend our understanding of communication-intensive applications and their job size distributions. On every throttle event, the nlrd daemon collects the bandwidth data of all applications running on the system and lists the top 10 of these application sorted by their network bandwidth consumption (total flits/s aggregated over all nodes). Note that these bandwidth-heavy applications are different than the ones running on the top 10 heavily congested nodes.

Fig. 16 shows that a few applications tend to be heavy-hitters. For example, 5 applications alone appear in approximately 57% of the top bandwidth application reporting events (Fig. 17), while more than 200 unique applications show up in total across all top bandwidth application reporting events. Interestingly, when 1-hour filtering is applied, the number of unique applications decreases significantly. However, the top 5 most frequently occurring applications constitute 50% of top bandwidth application reporting events (Fig. 17). The total number of unique applications reduces dramatically to 60. These results indicate that focusing on the top 5-10 applications can cover 50% of the communication-intensive applications space. We also observe the same results on a per-user basis with and without filtering. These results again confirm that application and user can work as a proxy for each other, even for top bandwidth application reporting events.

Next, we want to answer two questions: (1) are these top bandwidth applications the same as the top congestion-causing applications running on congested nodes?, and (2) is the job size distribution of the top bandwidth applications different than that of the top congestion-causing applications?

Fig. 18 shows the job size distribution and anonymized application names of the top 10 bandwidth-heavy applications that appear most frequently in the top bandwidth application reporting events. We observe that most of the applications tend to run on the same number of nodes every time they appear in the top bandwidth application reporting events. However, interestingly, these applications are not the same as the top congestion-causing applications running on congested nodes. Only three applications are common between these two sets (Fig. 18 vs. Fig. 15). They are situated at positions 1, 4, and 7, in the figures. This indicates that bandwidth-heavy applications are not necessarily the ones that cause congestion or run on congested nodes. These bandwidth-heavy applications are producing a significant amount of traffic, and are likely to be spread over a large number of nodes or have a many-to-many communication pattern. We notice that only 3 applications have a job size larger than 4000 nodes, indicating that even bandwidth-heavy applications are not necessarily large in size. The communication pattern seems to be playing a critical role. As an example, App K which runs mostly on 32 and 128 nodes appears second in the bandwidth-heavy applications list, but does not appear in the congestion-heavy applications list. This could be because this particular application does not intensively exhibit a many-to-one communication pattern. Many-to-few and many-to-one communication patterns can result in high congestion due to the concentration of messages over a few nodes. In summary, bandwidth-heavy applications' job sizes are similar to that of congestion-causing applications', but there is no significant overlap between these two sets and they may differ in their communication patterns.

## V. Related Work

Various HPC interconnect networks are proposed for improving HPC systems performance - QsNET [3], SeaStar [4], Tofu [5], Blue Gene/Q [6], Aries [7], TH Express-2 [8] and others - which use different types of topology like k-Ary n-Cube, fat-tree/Clos, and dragonfly. Interconnect networks have

been a vital part of computer systems. Network resources have been a major performance bottleneck in HPC systems performance. Several studies are performed to understand [9], [10], [11], [12], [13], [14], [15], [16], [17] and improve [18], [19], [20], [21], [22], [23], [24] interconnect failures in HPC systems.

Titan is the successor of Cray X-series which use the XK7 system and 3D Gemini interconnect. The Gemini system interconnect architecture is explained in [1] and evaluated in [25], [2] using micro-benchmarks. Cray's latest XC series is implemented using the Aries interconnect which supports better bandwidth, latency, message rate and scalability [26]. Our work differs from all these studies and evaluations as none of these works evaluate how different interconnect errors and congestion events occur on a large-scale HPC system. Our field data and analysis is unique and provides useful insights that can be used by users, system architects, and operators to improve the overall efficiency of HPC systems.

## VI. Conclusion

Overall, we discussed many interesting insights derived from our analysis. Interconnect faults like lane degrades are continuous and related to heterogeneous load imbalance among lanes. Link inactive errors do not have a temporal or a spatial correlation with lane degrades, while interconnect errors have a high correlation with link inactive/failed errors. We showed that these characteristics can be exploited for different purposes. We also demonstrated that multiple applications can cause multiple congestion events within a short period of time. Moreover, these applications can be, surprisingly, small in size, not scheduled evenly across the cabinet and have a many-to-few communication pattern. Our analysis can be used in identifying such applications and users to minimize the performance impact on other applications.

**Acknowledgment** We thank reviewers and Elmootazbellah Elnozahy for their constructive feedback. The work was supported by in part through NSF Grants (#1563728, #1561216 and #1563750), Northeastern University and by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, program manager Lucy Nowell. This work also used in part the resources of the Oak Ridge Leadership Computing Facility, located in the National Center for Computational Sciences at ORNL, which is managed by UT Battelle, LLC for the U.S. DOE under contract number DE-AC05-00OR22725.

## References

[1] R. Alverson, D. Roweth, and L. Kaplan, "The gemini system interconnect," in *High Performance Interconnects (HOTI), 2010 IEEE 18th Annual Symposium on*. IEEE, 2010, pp. 83–87.

[2] M. Ezell, "Understanding the impact of interconnect failures on system operation," in *Proceedings of Cray User Group Conference (CUG 2013)*, 2013.

[3] F. Petrini, E. Frachtenberg, A. Hoisie, and S. Coll, "Performance evaluation of the quadrics interconnection network," *Cluster Computing*, vol. 6, no. 2, pp. 125–142, 2003.

[4] R. Brightwell, K. T. Pedretti, K. D. Underwood, and T. Hudson, "Seastar interconnect: Balanced bandwidth for scalable performance," *IEEE Micro*, vol. 26, no. 3, pp. 41–57, 2006.

[5] Y. Ajima, S. Sumimoto, and T. Shimizu, "Tofu: A 6d mesh/torus interconnect for exascale computers," *Computer*, vol. 42, no. 11, pp. 0036–41, 2009.

[6] D. Chen, N. A. Eisley, P. Heidelberger, R. M. Senger, Y. Sugawara, S. Kumar, V. Salapura, D. L. Satterfield, B. Steinmacher-Burow, and J. J. Parker, "The ibm blue gene/q interconnection network and message unit," in *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*. IEEE, 2011, pp. 1–10.

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

[8] Z. Pang, M. Xie, J. Zhang, Y. Zheng, G. Wang, D. Dong, and G. Suo, "The th express high performance interconnect networks," *Frontiers of Computer Science*, vol. 8, no. 3, pp. 357–366, 2014.

[9] S. L. Scott *et al.*, "The cray t3e network: adaptive routing in a high performance 3d torus," 1996.

[10] M. Blumrich, D. Chen, P. Coteus, A. Gara, M. Giampapa, P. Heidelberger, S. Singh, B. Steinmacher-Burow, T. Takken, and P. Vranas, "Design and analysis of the bluegene/l torus interconnection network," IBM Research Report RC23025 (W0312-022), Tech. Rep., 2003.

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

[12] N. R. Adiga, M. A. Blumrich, D. Chen, P. Coteus, A. Gara, M. E. Giampapa, P. Heidelberger, S. Singh, B. D. Steinmacher-Burow, T. Takken *et al.*, "Blue gene/l torus interconnection network," *IBM Journal of Research and Development*, vol. 49, no. 2.3, pp. 265–276, 2005.

[13] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection networks: an engineering approach*. Morgan Kaufmann, 2003.

[14] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 350–361.

[15] D. Abts and B. Felderman, "A guided tour of data-center networking," *Communications of the ACM*, vol. 55, no. 6, pp. 44–51, 2012.

[16] C. Di Martino, W. Kramer, Z. Kalbarczyk, and R. Iyer, "Measuring and understanding extreme-scale application resilience: A field study of 5,000,000 hpc application runs," in *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on*. IEEE, 2015, pp. 25–36.

[17] S. Jha, V. Formicola, Z. Kalbarczyk, C. Di Martino, W. T. Kramer, and R. K. Iyer, "Analysis of gemini interconnect recovery mechanisms: Methods and observations," *Cray User Group*, pp. 8–12, 2016.

[18] C. E. Leiserson, "Fat-trees: universal networks for hardware-efficient supercomputing," *IEEE transactions on Computers*, vol. 100, no. 10, pp. 892–901, 1985.

[19] W. J. Dally, "Performance analysis of k-ary n-cube interconnection networks," *IEEE transactions on Computers*, vol. 39, no. 6, pp. 775–785, 1990.

[20] ——, "Express cubes: Improving the performance of k-ary n-cube interconnection networks," *IEEE Transactions on Computers*, vol. 40, no. 9, pp. 1016–1023, 1991.

[21] D. W. Mackenthun, "Method and apparatus for automatically routing around faults within an interconnect system," Sep. 12 1995, uS Patent 5,450,578.

[22] Y. Inoguchi and S. Horiguchi, "Shifted recursive torus interconnection for high performance computing," in *High Performance Computing on the Information Superhighway, 1997. HPC Asia'97*. IEEE, 1997, pp. 61–66.

[23] V. Puente, R. Beivide, J. A. Gregorio, J. Prellezo, J. Duato, and C. Izu, "Adaptive bubble router: a design to improve performance in torus networks," in *Parallel Processing, 1999. Proceedings. 1999 International Conference on*. IEEE, 1999, pp. 58–67.

[24] J. Domke, T. Hoefler, and S. Matsuoka, "Fail-in-place network design: interaction between topology, routing algorithm and failures," in *High Performance Computing, Networking, Storage and Analysis, SC14: International Conference for*. IEEE, 2014, pp. 597–608.

[25] A. Vishnu, M. ten Bruggencate, and R. Olson, "Evaluating the potential of cray gemini interconnect for pgas communication runtime systems," in *High Performance Interconnects (HOTI), 2011 IEEE 19th Annual Symposium on*. IEEE, 2011, pp. 70–77.

[26] B. Alverson, E. Froese, L. Kaplan, and D. Roweth, "Cray xc series network," *Cray Inc., White Paper WP-Aries01-1112*, 2012.