Network Performance Counter Monitoring and Analysis on the Cray XC Platform

J. Brandt*, E. Froese†, A. Gentile*, L. Kaplan†, B. Allan*, and E. Walsh*
*Sandia National Laboratories
Albuquerque, NM.
Email: (brandt|gentile|baallan|ejwalsh)@sandia.gov
†Cray Inc.
Vancouver, BC. and Seattle, WA.
Email: (efroese|lkaplan)@cray.com

Abstract—**The instrumentation of Cray's Aries network ASIC, of which the XC platform's High Speed Network (HSN) is comprised, offers unprecedented potential for better understanding and utilization of platform HSN resources. Monitoring the amount of data generated on a large-scale system presents challenges with respect to synchronization, data management, and analysis. There are nearly two thousand raw counter metrics per Aries router and interface, with functional combinations of these raw metrics required for insight into network state.**

Cray and ACES (LANL/SNL) are collaborating on collection and analysis of the Aries HSN network performance counter data. In this paper we present our work to identify HSN counters of interest; perform synchronized system wide collection of this counter data; and analyze the occurrences, levels, and longevity of congestion. We also discuss the challenges and solutions associated with collection, transport, in-transit computation and derived data analysis, visualization, storage, overhead, and redundant data.

I. Introduction

The ACES (LANL/SNL) Trinity XC platform is currently comprised of approximately 10,000 hosts (nodes). In addition to the Intel Haswell based computational resources (compute nodes), the Trinity platform also provides a variety of other services which are handled by dedicated hosts with task appropriate resources. The most prominent of these are: LNet routers dedicated to routing Lustre traffic, Realm-Specific Internet Protocol (RSIP) nodes for routing outside of the Aries fabric, and NVME enhanced Burst Buffer nodes to provide a pool of high speed, high capacity, shared storage. In addition to the more traditional Haswell processors, Trinity will be doubled in size with the addition of approximately 10,000 Intel KNL based compute nodes later in 2016.

Tying all of these platform resources together is the High Speed Network (HSN) based on the Cray Aries Router [1]. The HSN is configured in a Dragonfly [2] topology. In order to efficiently utilize the Trinity computational and support resources, the HSN resources must be configured and utilized such that the high bandwidth and low latency characteristics are taken advantage of but not overtaxed. Key to proper resource management is an understanding of how the HSN is utilized by applications and other platform subsystems such as shared parallel file systems and Burst Buffers. Trinity will be undergoing substantial changes as the system is doubled in size (both number of nodes and HSN resources) and incorporates new processor technology (Intel's KNL processor) with substantially different characteristics which may in turn change how the HSN interconnect is used. Also evolution in Burst Buffer software could result in substantial changes in HSN traffic load and usage patterns. Key to understanding HSN utilization characteristics is appropriate monitoring and analysis of HSN performance counters.

In support of DOE's Exascale Program, we have the additional need to understand the level of resiliency along with failure propagation paths in current architectures that may give insight into design considerations for future architectures. The network interconnect is also of interest in this regard [3]. Gaining such understanding for the evolution of the Cray network (Gemini to Aries) is part of the scope of an Office of Science Exascale Resilience project. Collaborators on this project include Cray, NCSA, UIUC, LANL, SNL, and NERSC.

In support of both of these efforts, ACES and Cray are collaborating on system wide collection and analysis of the Cray Aries HSN network performance counter data [4]. While these counters are also accessable through Cray's *CrayPat* performance analysis tool, it is in the context of individual applications being profiled and primarily focuses on NIC based counters. Monitoring at the system wide scales of interest, however, presents challenges with respect to synchronization, data management, and analysis. There are nearly two thousand counters per Aries network ASIC, with functional combinations of these raw metrics required for useful insight into network state.

In this paper we present our work to identify network performance counters of interest and perform synchronized

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

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research.

system wide collection of this counter data. Our goal is to enable analysis of the occurrences, severity, and longevity of congestion on Cray Aries based platforms. We also discuss the challenges and solutions associated with collection, transport, in-transit computation and derived data analysis, visualization, storage, overhead, and redundant data. As part of this work, Cray is allowing Sandia to make Cray's *gpcd* source code, for reading the Aries network counters, available.

This paper is organized as follows. In Section II we present background on the Aries network. In Section III we describe the Aries router architecture in greater detail. In Section IV we present information on the counters of interest and in Section V how they are related to congestion indicators. In Section VI we present implementation details of our counter collection, transport, and run-time processing. In Section VII we present initial analyses and visualizations of network traffic related quantities from system-wide data which we collected from the Trinity Platform in production. We present related work in VIII. We conclude in Section IX with future work.

II. Network Configuration

Trinity is a Cray XC40 system which uses a distributed high speed interconnect based on the Cray Aries router chip. The predecessor Cray XE/XK platform utilized Cray's previous generation router chip (Gemini) in a 3D torus (Figure 1) configuration for its High Speed Network (HSN). The 3D torus configuration, while simple with respect to routing rules, can have congestion issues [5] that significantly impact application performance. The Aries router incorporates advanced features that should enable better network traffic balance through use of the topologically richer Dragonfly [6] network configuration and adaptive routing algorithms.

Figures 1 (from [7]) and 3 and 4 (from [6]) provide a high level view of how different these two topologies are. The routing algorithm for the 3D torus configuration is generally "route to X coordinate of destination, route to Y coordinate of destination, route to Z coordinate of destination, go to destination node". Separate applications can adversely impact each other's performance if they share links in communication routes. An example for the 3D torus is depicted in Figure 1 where traffic between compute nodes connected to two adjacent routers (Green arrow) has contention with traffic from nodes connected to routers on either side (Blue arrow). The traffic represented by the Blue arrow would actually take the shorter path of the unshown wrap between the two end nodes in such a small configuration. However, it is illustrative of an actual problem in any reasonably sized system using this network topology.

The rules for routing in the Dragonfly network are much more complex [6] and have dependencies on a forwarding router's view of congestion at its possible targets. At each

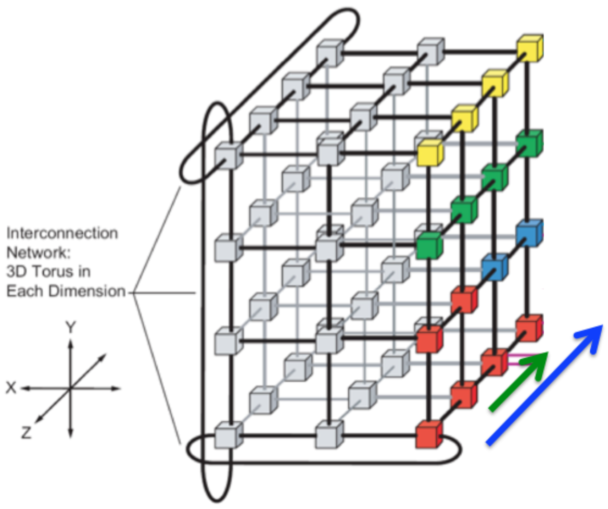


Figure 1: 3D Torus configuration of Gemini router based network fabric. The Green arrow represents traffic along the Z-axis between the two adjacent routers while the Blue represents traffic between the two adjacent to its head and tail. In such a small configuration the Blue traffic would actually take the unshown wrap but is shown as it is here to illustrate how traffic unrelated to that between compute nodes connected to and communicating through adjacent and directly connected routers can still interfere with their communications. (From [7])

router along a packet's path from point of injection to point of egress from the HSN, there are multiple viable options for which port the packet can be forwarded on to move it towards its destination. Routing rules determine the legal set of ports for the packet's next hop, and can include both choices that will send the packet along a minimal path or a non-minimal path. The decision of which port to actually choose, out of the possibly several legal choices, is made based on a combination of random selection and current congestion. In addition, biasing is provided to weight the choice between minimal and non-minimal paths.

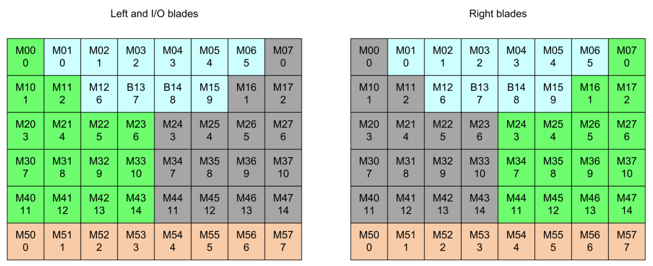
An XC system is composed of *electrical groups*, where each *electrical group* consists of two cabinets, each cabinet contains 3 chassis, each chassis contains 16 blades, and each blade contains a single Aries router. Each Aries has connections to every other Aries in the chassis (Green links). Each chassis has connections to every other chassis in the group (Black links). Each group has connections to every other group in the system (Blue links).

The dragonfly topology utilized in the XC platform uses three different levels of interconnectivity, with specific Aries router tiles being designated for each (see Figure 2). The lowest level is the chassis, consisting of 16 blades and 16 Aries routers. Within a chassis, each router utilizes a network link (Green tile) to connect directly to each other Aries router (15 links). Within an *electrical group* hereafter referred to simply as a *group* (2 cabinets consisting of 6 chassis) each blade position Aries router utilizes 3 links (Black) to connect to each of the 5 Aries routers corresponding to the same blade position in the 5 other chassis in the *group* (15 black links per Aries). Each Aries router within a *group* connects to Aries routers within other groups using up to ten of its Blue links. Trinity is currently configured with 8 connections from each *group* to each of the other 26 *groups* where each Aries router is using two to four of its Blue links for this type of connection.

III. The Aries Router

A low level description of the Aries router and its applicability to use in the Dragonfly topology at a scale of tens of thousands of compute nodes is given in [1]. Highlights from this paper are presented here to provide background for this work.

The Aries router, the fundamental building block of the Cray XC High Speed Network (HSN), is comprised of 40 router tiles each providing a bidirectional 3-lane (one bit per lane) link capable of 4.7 to 5.25 GB/s per direction. The bandwidth depends on whether the link is optical or electrical (electrical having the higher bandwidth). There are an additional 8 *processor tiles* that connect to four NICs, which are also included on the Aries device. There are no external 3 lane physical links associated with the *processor tiles*. The NICs connect to the processors in the system.


Left and I/O blades

M00 0	M01 0	M02 1	M03 2	M04 3	M05 4	M06 5	M07 0
M10 1	M11 2	M12 6	B13 7	B14 8	M15 9	M16 1	M17 2
M20 3	M21 4	M22 5	M23 6	M24 3	M25 4	M26 5	M27 6
M30 7	M31 8	M32 9	M33 10	M34 7	M35 8	M36 9	M37 10
M40 11	M41 12	M42 13	M43 14	M44 11	M45 12	M46 13	M47 14
M50 0	M51 1	M52 2	M53 3	M54 4	M55 5	M56 6	M57 7

Right blades

M00 0	M01 0	M02 1	M03 2	M04 3	M05 4	M06 5	M07 0
M10 1	M11 2	M12 6	B13 7	B14 8	M15 9	M16 1	M17 2
M20 3	M21 4	M22 5	M23 6	M24 3	M25 4	M26 5	M27 6
M30 7	M31 8	M32 9	M33 10	M34 7	M35 8	M36 9	M37 10
M40 11	M41 12	M42 13	M43 14	M44 11	M45 12	M46 13	M47 14
M50 0	M51 1	M52 2	M53 3	M54 4	M55 5	M56 6	M57 7

Figure 2: Aries tiles with color denoting one of Orange (NIC electrical), Green (intra-chassis electrical), Black (intra-group electrical), and Blue (inter-group optical) (From [8]).

IV. Counters of Interest

Aries counters of interest depend on what the user is looking for. There are nearly two thousand counters, some programmable, that provide information about latency, counts of flits across interfaces, stalls, and more. Many of these counters differ only in tile (row,column) designator or virtual channel (0-7) number. The counters available can provide information that application programmers can use to help tune code. They are also used by the system to identify areas, occurrences, and longevity of high levels of congestion. The counters also provide information that

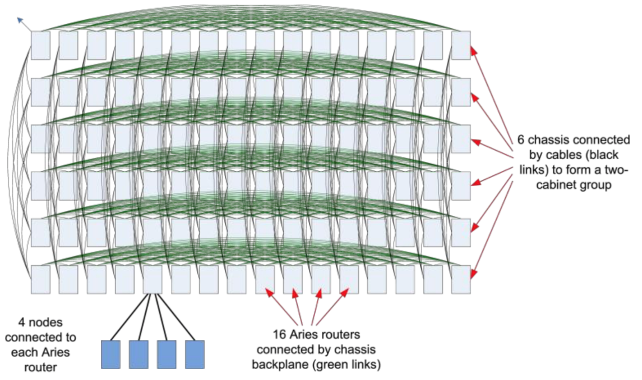

6 chassis connected by cables (black links) to form a two-cabinet group
4 nodes connected to each Aries router
16 Aries routers connected by chassis backplane (green links)

Figure 3: Aries in a blade, with 4 nodes attached to each, and connected by the chassis structure of a Cray XC *electrical group*. Each row represents the 16 backplane (Green links). Each column represents an Aries in one of the six chassis of a two-cabinet group, connected by electrical cables (Black links). (from [6])

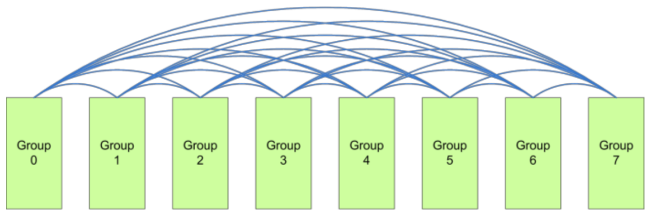


Figure 4: The global (Blue) optical links connect Dragonfly groups together. (from [6])

is useful in diagnosing certain kinds of system problems. Additionally, monitoring software can take advantage of these counters to provide long term storage of system wide synchronized measures of traffic levels and contention in the HSN for correlation with application performance variation. Counters (metrics) of interest in the system monitoring context include both node to Aries (NIC metrics) and Aries to Aries (RTR metrics) measures of traffic and contention on a per link and per virtual channel granularity. A sample of NIC and RTR metrics are given below. A more extensive list along with detailed descriptions can be found in [4].

- `AR_RTR_r_c_INQ_PRF_ROWBUS_STALL_CNT` - Number of flit-times for which the network tile identified by row `r` and column `c` is stalled while waiting to forward a flit.
- `AR_RTR_r_c_INQ_PRF_INCOMING_FLIT_VCv` - HSN traffic arriving on VC `v` of the physical link associated with the network tile identified by row `r` and column `c` in terms of network flits.
- `AR_NIC_NETMON_ORB_EVENT_CNTR_REQ_PKTS` - Request traffic being injected by the NIC into the HSN, in terms of packets.

V. Aries Congestion Indicators

We use the term congestion here to describe conditions where the amount of traffic being sent across a link or interface is curtailed due to buffer occupancy at the next hop destination. The severity of the congestion at any point in the network can be evaluated by looking at the amount of time a unit of traffic that is waiting to be forwarded is stalled due to lack of buffer space on the receiving end. The *flit* is the unit of information transmission and is a sub-component of a network packet. Thus, for a particular router link we use the ratio of stall counts to the number of flits received on that link, where the number of flits is the sum over all 8 virtual channels, to get an idea of the severity of congestion at that point in the network fabric. A ratio of 1 indicates that every other time a flit is eligible to be sent it cannot be, and is stalled, due to lack of buffer availability on the next hop receive end. As an example we utilize the ratio of `AR_RTR_r_c_INQ_PRF_ROWBUS_STALL_CNT` to the sum over all 8 virtual channels (v) of `AR_RTR_r_c_INQ_PRF_INCOMING_FLIT_VCv` on our Blue links, over a specific window of time, to get a measure of congestion faced by traffic entering a group on that link. This traffic may be destined for a target within the group or may be passthrough traffic that is only using the group for transit.

Likewise we can also estimate congestion faced by traffic coming from the other link types (i.e., Green links, Black links, and *processor tiles*). The congestion faced by traffic contending for access to a Blue link will be seen at the links where that traffic is coming from, not at the Blue link.

Ratios of up to roughly 0.25 stalls per flit are not too significant and can arise due to speed differences between different interfaces. As the ratio climbs from this point, traffic is starting to face actual congestion. The congestion might be due to contention for access to network links or it might be due to backpressure from the NIC and processor for traffic leaving the network. While some of the ratios we have seen (e.g., in the hundreds), and which we present in Section VII, seem like they should be significant enough to impact performance, we have not yet quantified the correlations between variations in application run times and these congestion measures. Severity of impact will depend on the both the severity and longevity of the congestion.

VI. Implementation

This section discusses the data collection, transport, in-transit processing, and storage methodologies and tools used in this work. Where appropriate we discuss alternatives, overhead with respect to memory and processing, and the reasons for our decisions.

A. Data Collection

In order to ease collection and calculation of the link aggregated metrics for the the Gemini [9], Cray implemented the *gpcdr* kernel module [10] which reports Gemini router metrics via user readable files in the `/sys` filesystem of all directly attached hosts [11]. A configuration file is processed, at kernel module start time, for each node in order to define the counter and counter combinations to be exposed in this manner. A kernel module restart is required for changes in the configuration file to be instantiated in the reported metrics.

The *gpcdr* kernel module is also supported for exposing the Aries performance counters. Because the Aries based dragonfly network does not use the same link aggregation scheme as Gemini, counters are reported on a per-tile basis. This results in an 8x increase in the number of metrics reported for an Aries router over the number reported for a Gemini for the same counters of interest. This increase coupled with an increase in the number of counters available for the Aries, as opposed to the Gemini [12], made the overhead for collecting the Aries metrics of interest via the `/sys` files prohibitive. As a result, we modified our HSN performance counter sampling implementation to utilize Cray's Generic Performance Counter Daemon (*gpcd*) interface for reading the Aries network performance counter information. This method also provides more flexibility for investigating counters to be collected as a change in counter specification only requires a restart of our data samplers (not a kernel module).

The general methodology for reading the *gpcd* counters applied to the Gemini has been described by Pedretti et. al. in [13]. Briefly, the *gpcd* interface provides methods for looking up a counter's memory mapped register descriptor based on the counter's name, for adding such descriptors to a data structure called a *context*, and for invoking a read of all added counters' values into the context. The counter values can then be obtained by parsing a list structure within the context. The *gpcd* source code has been expanded to include and support the necessary counter and address definitions available on the Aries. While the *gpcd* source code and library are not currently released by Cray as part of the software stack, Cray has provided it as part of an ongoing collaboration with Sandia and is allowing Sandia to release it in conjunction with the next release of our more general data collection code.

We utilize the Lightweight Distributed Metric Service (LDMS) [11], [14] for our data collection, transport, and storage infrastructure. (Version 3.0, pre-release, is used in this work). LDMS is plugin based. Configuration of the same core LDMS daemon (*ldmsd*), including plugins and their configurations, determines the functionality of a particular instantiation. Configuration of sampling plugins enables a *ldmsd* to collect data on hosts including compute nodes.

While all RTR metrics for a particular Aries router are visible from all connected network interfaces, NIC metrics are only visible to the directly connected nodes. For the purpose of management and logical separation we divide RTR and NIC metrics into two distinct sets each with its own separate *gpcd* context. From the full set of possible metrics, we collect all `FLIT`, `PKT`, and `STALL` related counters; for example, for all relevant `(r)ow`, `(c)olumn`, and `(v)irtual channel`, we collect `AR_RTR_r_c_INQ_PRF_INCOMING_FLIT_VCv` (`r=5` is not relevant to this counter as it corresponds to the processor facing tiles (ptiles)). The NIC plugin collects 13 NIC metrics while the RTR plugin collects the 800 RTR metrics of current focus.

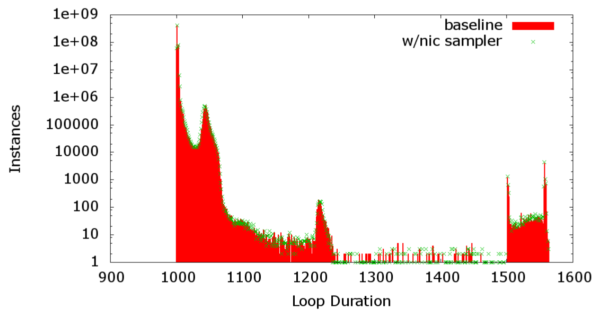
In our initial implementation, we have created four RTR sampler configuration files each defining a non-overlapping quarter of the 800 RTR metrics. Closely related metrics (only differing in VC) are collected together to minimize the time skew in their collection. We run a sampler configured with one of these four on each of the four nodes associated with a particular Aries router. The goal of this division is to provide a full set of RTR metrics while only incurring a quarter of the overhead per sampling host. This configuration provides no redundancy and downed nodes will result in missing data. In this simple configuration, blades with service nodes will be missing half the associated Aries RTR data, since there are only two service nodes on a service blade. For production purposes we will provide two way redundancy by collecting half of the RTR metrics on each of the four associated compute nodes and all of the RTR metrics on each of the service nodes.

Metrics can be combined in a post processing step or by use of the LDMS *function store* plugin, described in Section VI-D.

B. Impact Assessment

On Trinity, we collect and transport HSN performance counter data at 1 second intervals. The collection duration of a *gpcd* sample set of 200 metrics is ~135 usec. In order to assess the impact, in terms of OS noise, of collecting the Aries performance counter data, we utilized PSNAP [15]. PSNAP is an OS and network noise profiling tool which performs multiple iterations of a loop calibrated to run for a given amount of time t. On an unloaded system, variation from the ideal amount of time can be attributed to system noise at a lower frequency than $1/t$. We ran PSNAP both with and without monitoring in order to determine the additional impact of monitoring. PSNAP was run with 100,000 loops of 1000 microseconds (*t=1ms*). With this setting, noise occurring at a frequency of less than 1KHz will be evidenced by loop times greater than 1ms whether due to normal OS related processes or our monitoring processes which are running at 1Hz. The number of loops being run (100,000 in this case) times the loop time will bound the lowest frequency events that can be observed more than once. Thus if our network performance counter sampling incurs significant noise above that of the normal OS operations, we would expect to observe approximately 100 of these contributions due to the 100 occurrences of sampling over the 100 second PSNAP measurement window. Experiments were run both with and without barrier mode. When running with barriers the barriers were set to occur every 100 loops. When running in barrier mode, the workload corresponding to the calibration on the noisiest node is used across all nodes. Two runs were performed for each configuration on 100 nodes with 32 tasks (one per core) per node. Combined results for the two runs for the NIC sampler and the RTR sampler are shown in Figure 5 for the no barrier base and in Figure 6 for the barrier base. No substantive difference was observed.

The impact assessment data was collected on the Trinity Application Readiness Testbed (ART), Mutrino, sited at Sandia. While Mutrino's hardware is identical to that of Trinity, Mutrino is not currently running the Rhine/Redwood software stack being run on Trinity. Impact testing of our full set of monitoring samplers (e.g., Aries RTR and NIC, power, Lustre) on applications on Trinity, also showed no substantive impact on performance [16].



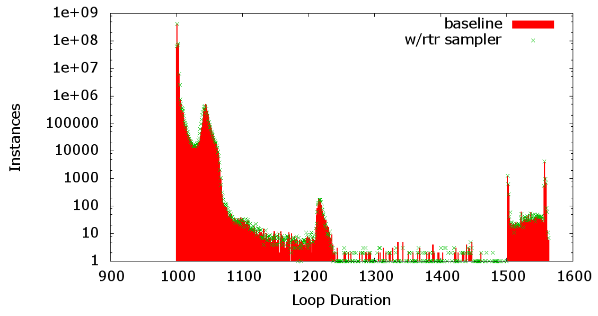
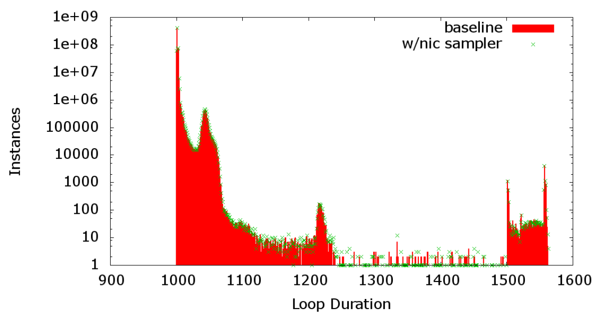


Figure 5: PSNAP results for the no barrier runs: Histogram of occurrences vs. loop time (μsec) with 1 second sampling (green) vs. no sampling (red). NIC sampler (top); RTR sampler (bottom). No substantive impact is seen.

C. Data Transport

LDMS *samplers* overwrite their metric set data at each sample interval. Thus transport of the sampled data off the nodes at each sample interval is required in order to preserve the data. LDMS *aggregators* are LDMS daemons that pull and aggregate metric sets from other LDMS daemons (whether *samplers*, other *aggregators*, or both). LDMS



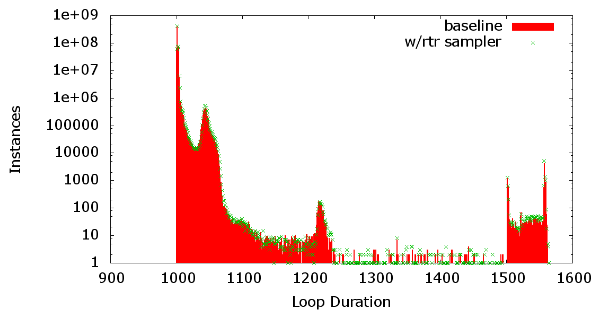


Figure 6: PSNAP results for the barrier runs: Histogram of occurrences vs. loop time (μsec) with 1 second sampling of the NIC sampler (green) vs. no sampling (red). NIC sampler (top); RTR sampler (bottom). No substantive impact is seen.

supports both RDMA and socket protocols over a variety of network technologies including Ethernet, Infiniband, Gemini, and Aries.

In order to minimize computational overhead on the compute nodes we utilize RDMA to periodically (at one second intervals in this case) pull Aries performance counter data from the LDMS sampler daemons. LDMS daemons performing the pull and aggregation are referred to simply as *aggregators* and those sampling host metrics as *samplers* in the remainder of this text. While two *aggregator* hosts would be sufficient to accommodate a Trinity sized platform from a transport perspective, for purposes of redundancy, storage, and analysis bandwidth we utilize four. For Trinity we have also implemented a Linux based monitoring cluster to perform storage and analysis of all system data, including logs, LDMS sampled metrics, SEDC data, and facilities power and cooling related data. (More information on the Trinity monitoring cluster can be found in [16]). Because the monitoring cluster is not connected to the Aries network fabric, we utilize a more complex LDMS aggregation topology using three levels of aggregation.

Aggregators performing RDMA reads from the compute nodes must be located on the Aries fabric. In order to transport data from the HSN to our external monitoring cluster we utilize service nodes which have connections to both the HSN and the monitoring cluster's 10 Gigabit Ethernet network fabric. On Trinity ther were only two service nodes available for this purpose. Thus, in order to support redundancy, we repurposed four compute nodes to act as first level aggregators, pulling LDMS data from the *samplers* using RDMA. The two service nodes run second level *aggregators* and pull LDMS data from the first level *aggregators* using RDMA. For redundancy purposes, *aggregators* can be configured for both primary and failover collection targets. A third level of *aggregators* runs on the monitoring cluster. The third level *aggregators* pull data from the second level *aggregators* using the socket transport, and can also be configured for redundancy and failover. The monitoring cluster is currently comprised of five Linux hosts of which three are configured as *aggregators* with redundant fail over configurations.

D. In-Transit Derived Data

While there are 852 metrics currently being collected for each Aries ASIC, desired quantities of interest are currently all functional combinations of these. For instance, from [4], a high ratio of stalls to flits over a time interval can be an indication of level of congestion, evidenced by backpressure, at the point and direction measured. Thus, we have identified a reduced (currently 169) set of *derived* metrics of interest. In order to minimize computational overhead on the compute nodes we do no processing of sampled data on the hosts being sampled. All data is transported in raw form to *aggregator* hosts and stored for historical comparison and post processing. Additionally, we perform *in-transit* processing of the raw data on *aggregator* hosts to produce our *derived* metrics of interest. Since the aggregator hosts are not utilized for user applications, the additional processing on them has no effect on application run times. This provides low latency access to *derived* metrics of interest while incurring no overhead on the compute nodes for the derivation. Additionally this enables immediate use of this more descriptive data for further analysis, visualization, and as a basis for triggering a system or application response to changing/changed conditions.

For this work, we have implemented a general *function store* plugin for LDMS. The functions, input, and output variables are defined in a configuration file. The current set of functional forms supported includes multivariate sums, minimums, maximums, and averages; bivariate multiplications, divisions, and subtractions; univariate deltas and rates from the previous timestep; and univariate thresholding comparisons. All computations include an optional scaling factor to address issues of resolution in the computations with integer outputs. Outputs of one function can be specified to be inputs to another function. The function store does not seek to be a general computational engine, but rather addresses the main types of identified functional forms of interest. *Function store* outputs can be optionally written out to CSV files and/or written to named pipes where they can be further utilized for analysis and possibly converted to different storage formats. On Trinity, the monitoring cluster LDMS daemons currently store both raw and derived data to CSV files; visualization and analysis of this data is described in Section VII.

The derived metrics that we currently calculate and store are a) ratios of STALLs to FLITs over the last time interval at any interface (an indicator of congestion) b) ratios of FLITS to PACKETS over the last time interval (an indicator of packet size) and c) change in FLITS over the last time interval (an indicator of traffic volume). For example, for each `(r)ow` and `(c)olumn`, we calculate the ratio of changes in `AR_RTR_r_c_INQ_PRF_ROWBUS_STALL_CNT` to `AR_RTR_r_c_INQ_PRF_INCOMING_FLIT_VCv`, summed over all `(v)` Virtual Channels (this is the quantity described in Section V) and store it as a new variable currently chosen to be named `AR_RTR_r_c_STALL_FLIT_RATIO`. Similarly, we calculate and store variables for ratios of changes among `AR_NIC_NETMON_ORB_EVENT_CNTR_REQ_PKTS`, `AR_NIC_NETMON_ORB_EVENT_CNTR_REQ_FLITS`, and `AR_NIC_NETMON_ORB_EVENT_CNTR_REQ_STALLED`.

VII. Analysis and Visualization

This section describes our current investigations into how the derived metrics of interest described in Section VI-D can be utilized through both numerical analysis and visualization to gain an understanding of the traffic and congestion characteristics of the Trinity platform's HSN.

Data in this section is system data collected during our initial LDMS installation on Trinity. As described in Section VI-A, in our initial *sampler* configuration, downed nodes and uncollected datasets on the service blades result in some missing data. This will be resolved via the redundant collection, also described in Section VI-A, in the production configuration; this accounted for missing data on about 5 percent of the links, generally evenly distributed among the different link types.

A. Visualization

We have developed a prototype web-based visualization upon which network-related data can be overlaid to provide connection based information about congestion between two components. Information can be visualized based on single link or aggregate information.

As described in Section II there are 27 *electrical groups* on Trinity. In our visualization shown in Figure 7 a cell at the intersection of a row and column is colored according to a chosen network related metric (e.g., traffic) between those two *groups*; thus the diagonal has no data. For example, we might color a cell by the maximum traffic between two *groups* over all the links between the two *groups*. From any *group* on Trinity, there are 8 links to each of the other *groups*. Hovering over a cell launches a window with a description of each of the eight composite links in a tooltip along with each corresponding value.

Selecting a cell displays sub-matricies related to the two groups, for example the two 6 x 16 Black link and Green link connected elements within the group shown in Figure 8. These may be colored by Green link, Black link, or aggregate information.

Our near-term goal is to have the visualization tool reading dynamically from a database backend based on user interaction. Longer term we would like to include the option to visualize run-time data streams. At the time of this writing, the web-based visualization reads from files defining the visualization heirarchy and associated data. These are built from queries to a database backend in which are stored the network topology specifications; the network data, bulk loaded from the CSV output of the *function store*; and the node-to-data-subset mappings from the LDMS RTR sampler configuration.

B. Router Tile Counters

In this section, we present an examination of backpressure received at routers from their Blue and Green connections using visualization tools we are currently developing specifically for exploration of Aries network related information.

From [4], an aggregate view of the backpressure incurred by the traffic received at the network tile is expressed by our derived variable `AR_RTR_r_c_STALL_FLIT_RATIO` in the *function store* output. Note that `r` and `c` refer to the individual tile row and column, shown in Figure 2, and are *not* the row and column of the matrix display of Figure 7. In this section we will refer to this quantity for any `r` and `c` as `STALL/FLIT`.

Determination of the appropriate tile counter for any given connection and extraction of its data is performed as follows. The command `rtr --interconnect` provides system specific network connectivity information. For example, the Blue connections between groups 17 and 21 include:

```
c11-2c2s12a0l05(17:5:12) blue -> c7-3c2s10a0l05(21:5:10)
c11-2c2s12a0l06(17:5:12) blue -> c7-3c2s10a0l06(21:5:10)

c7-3c2s10a0l05(21:5:10) blue -> c11-2c2s12a0l05(17:5:12)
c7-3c2s10a0l06(21:5:10) blue -> c11-2c2s12a0l06(17:5:12)
```

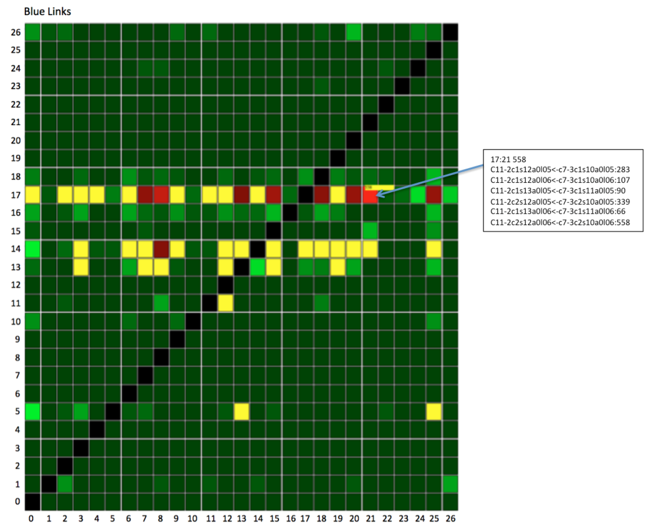
Each *group* has 8 connections of this type to each of the other *groups*. For `c11-2c2s12a0l06`, `r` and `c` are 0 and 6, respectively. The slot of the relevant Aries, (e.g., `c11-2c2s12`), and the RTR sampler configuration identify which node (or nodes, in the case of redundant collection) has collected the data. The topology information from `rtr --interconnect` for all connections and types and the RTR sampler configuration are stored in a database to enable fast lookups of any connection's variable's data.

In Figure 7 each cell shows, for a particular time, the maximum value over all of the Blue links between any two *groups* of the `STALL/FLIT` value over the last time interval (1 sec). For the connection between *groups* 17 and 21, the maximum value is on the incoming connection to `c11-2c2s12a0l06` from `c7-3c2s10a0l06`. This determines the value for the matrix cell row=17 column=21. Note that data for 2 of these 8 connections is missing (incoming into `c11-2c2s13a0l05` from `c7-3c2s11a0l05` and `c11-2c2s13a0l06` from `c7-3c2s11a0l06`). This is also the highest value in the figure. Since this data describes *incoming* traffic, this value indicates backpressure on traffic coming *into group* 17 from *group* 21. From the figure it is evident that, over this time interval, backpressure on traffic coming *into groups* 17, 14, and 13 is generally higher than for others.

The color scale in this figure uses black for the non-existent diagonal values. Existent values are colored on a manually selected green-yellow-red scale, with green being the lower values and red being the higher. The color range was selected in order to make the higher values stand out; it is not yet known what absolute values we may be interested in. If visualized quantities (e.g., traffic and backpressure) were reasonably symmetric, with about the same going in each direction between pairs of *groups*, there would be a symmetrical color pattern about the black diagonal.

The backpressure within the groups can be investigated via a drill down visualization. Selecting a cell in the Blue *group* visualization displays the 6 x 16 topology of both *groups* involved, as shown in Figure 8. In this figure we are investigating the backpressure on the traffic flowing *into* each Aries *from* `c11-2c2s12` via their Green links to that Aries router, so only cells in the row pertaining to chassis `c11-2c2` will be colored. The highest value of the `STALL/FLIT` along these Green links is that coming into `c11-2c2s3`. Black colored cells in the relevant row represent either invalid (i.e., `c11-2c2s12` to itself) or missing data (i.e., `s2`, `s10`, and `s13`).

We are also interested in understanding the evolution and attribution of congestion. Figure 9, is a time history plot of the `STALL/FLIT` ratio for the link of highest value in Figure 7. Time = 0 corresponds to the time associated with the snapshot of data displayed in Figure 7. While we have not yet performed any numeric analysis on this data it appears, from visual inspection of the Figure 9 time history plot, that there is periodic and symmetric structure showing building congestion that peaks and abates at roughly one minute intervals. Also we observe, over the time of this plot that the peaks of these structures seem to also have periodic envelopes whose peaks appear to increase through time until soon after our time slice of interest and then start falling off. However, the impact, if any, of any of this on an applications overall performance is not currently known and will be the focus of future work.



```
17:21 558
C11-2c1s12a0l05<-c7-3c1s10a0l05:283
C11-2c1s12a0l06<-c7-3c1s10a0l06:107
C11-2c1s13a0l05<-c7-3c1s11a0l05:90
C11-2c2s12a0l05<-c7-3c2s10a0l05:339
C11-2c1s13a0l06<-c7-3c1s11a0l06:66
C11-2c2s12a0l06<-c7-3c2s10a0l06:558
```

Figure 7: Web-based visualization of network quantities between Blue *groups*. Visualization is in the form of a matrix of cells where any (row, column) location is used to indicate quantities between the those two *groups*. Hovering over any location brings up a tooltip of the value and text representation of the data involved. In this example, the `STALL/FLIT` ratio indicating back pressure on *incoming* traffic on Blue *groups* over a particular one second interval on Trinity is shown. Locations are colored by the maximum value over the Blue links connecting the two associated *groups*. A text display shows the individual link data. Here backpressure on traffic *into groups* 17, 14, and 13 is generally higher. In the hovered-over location's text, 2 of the 8 Blue links are missing data.

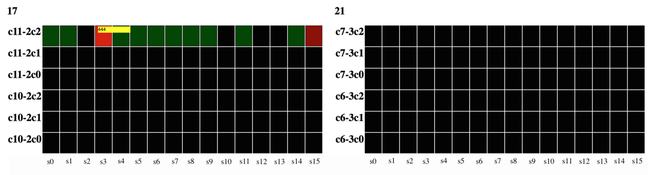


Figure 8: Web-based visualization for network quantities on Black/Green links. Selecting a location in the Blue *group* visualization 7 brings up two 6 x 16 possible Black-Green link visualizations for the *groups* involved. Rows are chassis of the two cabinets in a *group*; a given cell is an Aries router in that chassis. *Groups* in this figure are those of the selected cell (*groups* 17 and 21) in the previous figure. The `STALL/FLIT` counts are shown for Green links from `c11-2c2s12` (Aries router with highest `STALL/FLIT` counts on Blue inter group links). The highest value is seen to be *into* `c11-2c2s3`.

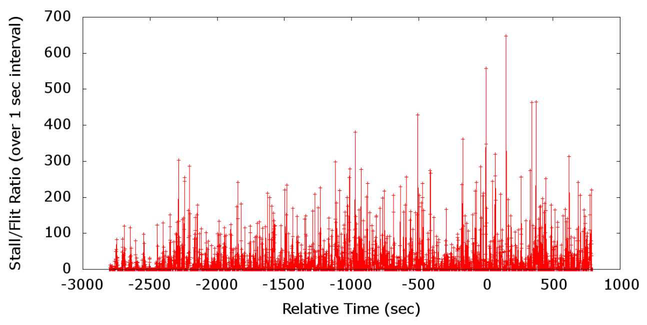


Figure 9: Time history of values for the link of the highest value in the visualization of Figure 7. Time = 0 corresponds to the time of that figure.

C. *Traffic Between NIC and HSN*

From [4], the quantities AR_NIC_NETMON_ORB_EVENT_CNTR_REQ_PKTS, AR_NIC_NETMON_ORB_EVENT_CNTR_REQ_FLITS, and AR_NIC_NETMON_ORB_EVENT_CNTR_REQ_STALLED represent the aggregate request traffic being injected by the NIC into the HSN. Here, we examine ratios of changes in counts of stalls to flits. These ratios are calculated in and stored by the *function store*. We will refer to them here, independent of location, as FLIT/PKT and STALL/FLIT.

Table I shows instances of STALL/FLIT over a randomly selected approximately 3 hour period, for approx 3/4 of the nodes binned in buckets of 50. It is interesting to note that the vast majority are small but that there are a couple of peaks that are one to two orders of magnitude larger.

Bin Range	Instances
[0-50)	75719905
[50-100)	24
[100-150)	10
[150-200)	4
[200-250)	2
[250-300)	1
[300-350)	2
[350-400)	0
[400-450)	2
[450-500)	0
[500-550)	0
[550-600)	0
[600-650)	1
[650-700)	0
[700-750)	0
[750-800)	0
[800-850)	0
[850-900)	0
[900-950)	2
[950-1000)	0

Table I: STALL/FLIT instances for traffic being injected by NICs into the HSN over an arbitrary 3 hour period

A time history plot of the values, shown in a binned view in Table I, for a particular NIC is shown in Figure 10 (red). This view reveals a periodic structure that is much more pronounced than that for the RTR STALL/FLIT ratio shown in Figure 7 and lacks the apparent buildup that we saw in the RTR case. In this case the feature periods appear to repeat every 90-100 seconds with feature widths of ~50 seconds. The zeros here are not due to missing data points. The time between samples is shown in green, with only a few missed samples around $t = 1275$. Values of missed points are of order 2, indicating that only single points have been misssed (i.e., there were no consecutive missed samples).

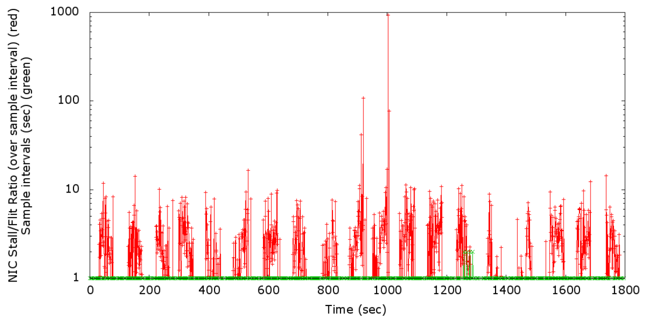


Figure 10: STALL/FLIT ratio for traffic being injected by a particular NIC into the HSN over an arbitrary half hour period (red). Time between samples is shown in green.

VIII. Related Work

There is substantial interest in the investigation and mitigation of the impact of congestion on application performance. Some examples targeting Cray Gemini and Aries systems are given here.

For Gemini systems, Bhatele et. al. [5] have done substantial work in assessing the variability in runtimes and messaging rates on NERSC's Hopper. Albing et al. [17] have investigated orderings for application placement in a 3D torus. NCSA has instituted topologically-aware scheduling [18] on its Blue Waters platform in order to reduce application interference.

Work involving reading of the Gemini network performance counters, outside of CrayPat, includes that of Pedretti and collaborators [13], [19] examining variations in congestion-related quantities as a result of different application and task mappings. Data in those works was limited to that accessible from within an application's allocation. Brandt et. al. [20] have demonstrated, in small-scale experiments, that remapping tasks based on dynamic system-wide network information in a congested environment can recover more time lost to congestion than using static network measures (e.g., hop counts). Network performance counters have been continuously collected system-wide on Blue Waters for over two years, via LDMS [11], in order to investigate network traffic and contention in the Gemini HSN.

For the Dragonfly toplolgy, Jain et. al. [21] are among those having performed modeling studies investigating throughput. Relatedly, Bhatele and collaborators [22], [23] have developed visualizations for the dragonfly topology which they have applied to traffic simulations on the architecture of NERSC's Edison.

Differentiating elements of our work are the development of continuous, system-wide collection, transport, and analysis of the Aries network performance counters including the use of production system data in the analysis.

IX. Future Work

In this paper, we have presented our work on collecting, analyzing, and visualizing HSN performance counters on an Aries based network. We have discussed challenges and solutions in support of this work. Establishing continuous system wide monitoring of Trinity's High Speed Network components is the first foundational step in gaining an understanding of how these resources are being utilized. With this in place we will be turning our attention to further development of the analysis and visualization work presented in this paper.

With respect to data analytics we will be developing tools to process the data, both *in-transit* and post processing. Since the XC platform management services do not natively provide facilities for continuous system wide collection and display of HSN performance counter data, it has not previously been possible to investigate what values should be used as indicators of performance-impacting contention on the system. Our intent is to quantify normal operation as well as to build tools for both run-time and post processing identification of problems in the fabric that stand out as anomalous and performance impacting.

Visualization is key to gaining the understanding necessary to drive the analytics as well as providing a useful production tool for status and diagnosis. With respect to visualizations we plan to further develop and expand the drill down visualizations presented in Figures 7 and 8 to be more interactive with: additional information (raw and derived) available in hover boxes, the ability to scroll-through-time, the ability to change metrics in a particular focus via a pull-down menu, and on demand time-history plots such as those in Figures 9 and 10 for particular tiles, metrics, and times of interest.

Acknowledgment

The authors would like to thank Joe Greenseid (Cray) for useful discussions; Adam DeConinck and Amanda Bonnie (LANL) for their participation in the LDMS install, testing, and operation on Trinity; and Jason Repik (Cray/SNL) for his assistance in testing and debugging LDMS and associated software on Mutrino.

References

[1] B. Alverson, E. Froese, L. Kaplan, and D. Roweth, "Cray XC Series Network," WP-Aries01-1112, 2012. [Online]. Available: http://www.cray.com/sites/default/files/resources/CrayXCNetwork.pdf

[2] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," *SIGARCH Comput. Archit. News*, vol. 36, no. 3, pp. 77–88, Jun. 2008.

[3] C. Di Martino, Z. Kalbarczyk, R. Iyer, F. Baccanico, J. Fullop, and W. Kramer, "Lessons Learned from the Failure Analysis of a Petascale System: the Case of Blue Waters," in *Proc. IEEE/IFIP Int'l Conference on Dependable Systems and Networks (DSN14)*, 2014.

[4] Cray Inc., "Aries Hardware Counters," Cray Doc S-0045-20, 2015.

[5] A. Bhatele, K. Mohror, S. H. Langer, and K. E. Isaacs, "There Goes the Neighborhood: Performance Degradation Due to Nearby Jobs," in *Proc. Int'l Conference on High Performance Computing, Networking, Storage and Analysis (SC13)*, 2013.

[6] G. Faanes, A. Bataineh, D. Roweth, T. Court, E. Froese, B. Alverson, T. Johnson, J. Kopnick, M. Higgins, and J. Reinhard, "Cray Cascade: A Scalable HPC System Based on a Dragonfly Network," in *Proc. Int'l Conference on High Performance Computing, Networking, Storage and Analysis (SC12)*, 2012.

[7] "Brief Blue Waters System Overview," https://bluewaters.ncsa.illinois.edu/user-guide.

[8] D. Greenseid and D. Roweth, private communication, Feb 2016, Cray, Inc.

[9] R. Alverson, D. Roweth, and L. Kaplan, "The Gemini System Interconnect," in *Proc. 2010 IEEE 18th Annual Symposium on High Performance Interconnects (HOTI)*, 2010.

[10] Cray Inc., "Managing System Software for the Cray Linux Environment," Cray Doc S-2393-5202axx, 2014.

[11] 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, "Lightweight Distributed Metric Service: A Scalable Infrastructure for Continuous Monitoring of Large Scale Computing Systems and Applications," in *Proc. Int'l Conference for High Performance Storage, Networking, and Analysis (SC14)*, 2014.

[12] Cray Inc., "Using the Cray Gemini Hardware Counters," Cray Doc S-0025-10, 2010.

[13] K. Pedretti, C. Vaughan, R. Barrett, K. Devine, and S. Hemmert, "Using the Cray Gemini Performance Counters," in *Proc. Cray User's Group*, 2013.

[14] "Ovis," http://github.com/ovis-hpc/ovis.

[15] "PAL System Noise Activity Program," Los Alamos National Laboratory Performance and Architecture Laboratory (PAL), March 2014. [Online]. Available: http://www.c3.lanl.gov/pal/

[16] A. DeConinck *et al.*, “Design and Implementation of a Scalable Monitoring System on Trinity,” in *Proc. Cray User’s Group*, 2016.

[17] C. Albing, N. Troullier, S. Whalen, R. Olson, and J. Glensk, “Topology, bandwidth and performance: A new approach in linear orderings for application placement in a 3D torus,” in *Proc Cray User Group (CUG)*, 2011.

[18] J. Enos, “Application Runtime Consistency and Performance Challenges on a Shared 3D Torus.” MoabCon 2014, 2014. [Online]. Available: http://www.youtube.com/watch?v=FR274JitRq8

[19] M. Deveci, S. Rajamanickam, V. Leung, K. Pedretti, S. Olivier, D. Bunde, U. V. Catalyurek, and K. Devine, “Exploiting Geometric Partitioning in Task Mapping for Parallel Computers,” in *Proc. 28th Int’l IEEE Parallel and Distributed Processing Symposium*, 2014.

[20] J. Brandt, K. Devine, A. Gentile, and K. Pedretti, “Demonstrating Improved Application Performance Using Dynamic Monitoring and Task Mapping,” in *1st Workshop on Monitoring and Analysis for High Performance Computing Systems Plus Applications (HPCMASPA 2014) Proc. IEEE International Conference on Cluster Computing (CLUSTER)*, 2014.

[21] N. Jain, A. Bhatele, X. Ni, N. J. Wright, and L. V. Kalè, “Maximizing Throughput on A Dragonfly Network,” in *Proc. Int’l Conference on High Performance Computing, Networking, Storage and Analysis (SC14)*, 2014.

[22] A. Bhatele, N. Jain, Y. Livnat, V. Pascucci, and P.-T. Bremer, “Simulating and Visualizing Traffic on the Dragonfly Network,” 2015, poster and extended abstract from Int’l Conference on High Performance Computing, Networking, Storage and Analysis (SC15). [Online]. Available: http://sc15.supercomputing.org/sites/all/themes/SC15images/tech_poster/tech_poster_pages/post109.html

[23] Y. Livnat, P.-T. Bremer *et al.*, “DragonView.” [Online]. Available: https://github.com/LLNL/DragonView