SANDIA REPORT

SAND2016-9525
Unlimited Release
Printed September 23, 2016

High Performance Computing Metrics to Enable Application-Platform Communication

Anthony M. Agelastos, James M. Brandt, Ann C. Gentile, Justin M. Lamb, Kevin P. Ruggirello, Joel O. Stevenson

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories

SAND2016-9525
Unlimited Release
Printed September 23, 2016

High Performance Computing Metrics to Enable Application-Platform Communication

Anthony M. Agelastos
amagela@sandia.gov

James M. Brandt
brandt@sandia.gov

Ann C. Gentile
gentile@sandia.gov

Justin M. Lamb
jmlamb@sandia.gov

Kevin P. Ruggirello
kruggir@sandia.gov

Joel O. Stevenson
josteve@sandia.gov

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185

Abstract

Sandia has invested heavily in scientific/engineering application development and in the research, development, and deployment of large scale HPC platforms to support the computational needs of these applications. As application developers continually expand the capabilities of their software and spend more time on performance tuning of applications for these platforms, HPC platform resources are at a premium as they are a heavily shared resource serving the varied needs of many users. To ensure that the HPC platform resources are being used efficiently and perform as designed, it is necessary to obtain reliable data on resource utilization that will allow us to investigate the occurrence, severity, and causes of performance-affecting contention between applications. The work presented in this paper was an initial step to determine if resource contention can be understood and minimized through monitoring, modeling, planning and infrastructure. This paper describes the set of metric definitions, identified in this research, that can be used as meaningful and potentially actionable indicators of performance-affecting contention between applications. These metrics were verified using the observed slowdown of IOR, IMB, and CTH in operating scenarios that forced contention. This paper also describes system/application monitoring activities that are critical to distilling vast amounts of data into quantities that hold the key to understanding for an application's performance under production conditions and that will ultimately aid in Sandia's efforts to succeed in extreme-scale computing.

Acknowledgment

The authors of this document would like to acknowledge the contributions from the following individuals: Michael E. Davis, Jason J. Repik, and Lauren Hund. Mike was instrumental with tuning the IMB test suite to be as efficient as possible. Jason's hard work with setting up and supporting the dedicated application time (DAT) for our experiments was critical to the success of our experiments. Lauren's statistical operator recommendations helped fast track model development which enabled additional project time for model parameter studies.

Contents

List of Figures

List of Tables

Nomenclature

DOE	U.S. Department of Energy
DoD	U.S. Department of Defense
NNSA	DOE's National Nuclear Security Administration
SNL	Sandia National Laboratories
LLNL	Lawrence Livermore National Laboratory
LBNL	Lawrence Berkeley National Laboratory
NERSC	National Energy Research Scientific Computing Center
ASC	NNSA's Advanced Simulation and Computing Program
ACES	NNSA's New Mexico Alliance for Computing at Extreme Scale
ATS	Advanced Technology System
DAT	Dedicated Application Time
PE	Processing Element
CLE	Cray Linux Environment
ALPS	CLE's Application Level Placement Scheduler
HSW	Intel Haswell CPU
OST	Object Storage Target
HSN	High Speed Network
CADE	Cray Application Developer's Environment
KB	Kilobyte, 1024^1 bytes
MB	Megabyte, 1024^2 bytes
GB	Gigabyte, 1024^3 bytes
μ_g	Geometric mean
σ_g	Geometric standard deviation

δ	Confidence interval
HPC	High-performance computing
MPI	Message Passing Interface
TPL	Third-party library
IOR	Interleaved Or Random (Benchmark)
IMB	Intel MPI Benchmarks
CHARTD	**C**omputational **H**ydrodynamics and **R**adiative **T**hermal **D**iffusion
CSQ	**C**HARTD **SQ**uared
CTH	**C**SQ to the **T**hree **H**alves
AMR	Adaptive Mesh Refinement
LDMS	**L**ightweight **D**istributed **M**etric **S**ervice
AMM	**A**ctionable **M**etric **M**odel

Chapter 1

Introduction

High Performance Compute (HPC) platforms leverage parallelism in application execution models in order to achieve performance scaling. The typical use case for resource sharing among applications is time sharing of computational and memory resources at a *node* granularity and designing networks and file systems with sufficient bandwidth and capacity to enable space sharing with minimal adverse affect due to contention.

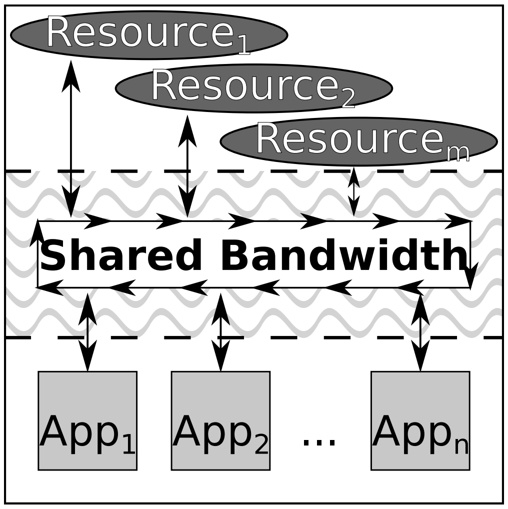


Figure 1.1. HPC platform topology illustrating shared bandwidth contention from intra-application bandwidth (e.g., an application's MPI ranks communicating amongst themselves via shared bandwidth that other applications are using) and application-to-generic-resource usage (e.g., applications writing checkpoint/restart files to the file system).

There are several major problems with these assumptions with respect to the resources being space shared: 1) different applications may have different resource needs which may vary over the the time and phases of each applications execution time, 2) the temporal characteristics of these needs with respect to the various resources being used have not been well studied even on a per-application basis, and 3) resource allocation and scheduling policy with respect to resource allocation are typically based on *compute node* availability and do not consider effects on other shared resources such as network and file system as these have presumably been engineered to accommodate the target shared workloads.

While application performance degradation due to resource congestion has been observed [1], approaches to mitigation have generally been to increase capacity in future platform acquisitions and incorporate better load sharing mechanisms. These approaches have fallen short because while the cycle for a new platform is around five years, developers are continuously tuning algorithms and methods to utilize all platform resources and maximize performance. Thus, even with a substantial increase in processing elements and interconnect and file system bandwidth, a new system's shared resources are rapidly taken advantage of and easily oversubscribed. Though a particular application's constituent parts may operate in harmony with respect to available resources, there are currently no mechanisms to enable competing applications to load balance. Performance-crippling contention often occurs which can lead to platform instability if competing applications' cumulative needs exceed the platform's capabilities.

The goal of the work presented in this paper is to enable balanced platform resource usage while minimizing/bounding resource contention. We take the novel approach of using whole system high fidelity monitoring to identify metric definitions that can be used as meaningful actionable indicators of performance-affecting contention. Use of these metrics by applications, schedulers, and resource managers can potentially have a large impact on efficient use of platforms and minimization of application run-times. We have seen in small-scale experiments for particular architectures that dynamic adaptation to contention scenarios can recover significant time lost due to contention [2], however the identification of actionable indicators and the quantification of performance impact based on such indicators is largely unknown.

Our resources of interest for this project are the High Speed Network interconnect (HSN) and the Lustre parallel file system. Metrics investigated in this work correspond to usage of these resources and, for network resources, backpressure in the network due to oversubscription of these resources. Time-dependent statistical operations on this data are performed and analyzed to determine values that can indicate the presence and impact of contention. We define and verify a set of metrics pertaining to network and file system resources that quantify the existence and impact of the resources' utilization/congestion.

In this work we examine performance and resource utilization/contention for several applications (IOR, IMB, and CTH) in operating scenarios that force contention. We utilize Sandia's Lightweight Distributed Metric Service (LDMS) to monitor raw resource utilization data that are used as the basis for the metrics. We discuss methods to distill LDMS data into actionable metrics that can be provided to application developers and can be used for run-time resource management.

This work targets the current ACES supercomputer, Trinity. Trinity is a 20,000 compute node Cray XC40 based system with an Aries based high speed network interconnect using a dragonfly topology and adaptive routing. The work presented is performed on the 200 node (2 cabinet) Cray XC40 Application Readiness Testbed system, Mutrino, which utilizes the same interconnect and file system technologies.

Chapter 2

Network Contention

Details of the Aries interconnect, the network performance counters and our collection of such, and the quantities used in the assessment of contention are described in our previous work [3]. In this chapter we present details necessary for understanding the network and congestion in this work.

2.1 Aries Interconnect

Trinity is a Cray XC40 computer utilizing a modified Dragonfly network topology on top of Cray's high performance Aries [4] distributed router technology. The predecessor to Trinity, Cielo, was a Cray XE/XK. The XE/XK architecture utilized the Cray's Gemini router chip [5, 6] in a 3D torus topology for its High Speed Network (HSN). In the Gemini network, 2 nodes share a Gemini router and the routing algorithm is deterministic. Generally, traffic is routed in X, Y, and Z directions, in that order. Multiple applications may have their traffic routed through the same routers, and an application can be impacted by the traffic of other applications. Network performance counter information is accessible to a node for that node's local router chip, but not from remote router chips. Thus an application, in general, cannot get direct insight into the level of contention experienced at every hop along its communication paths.

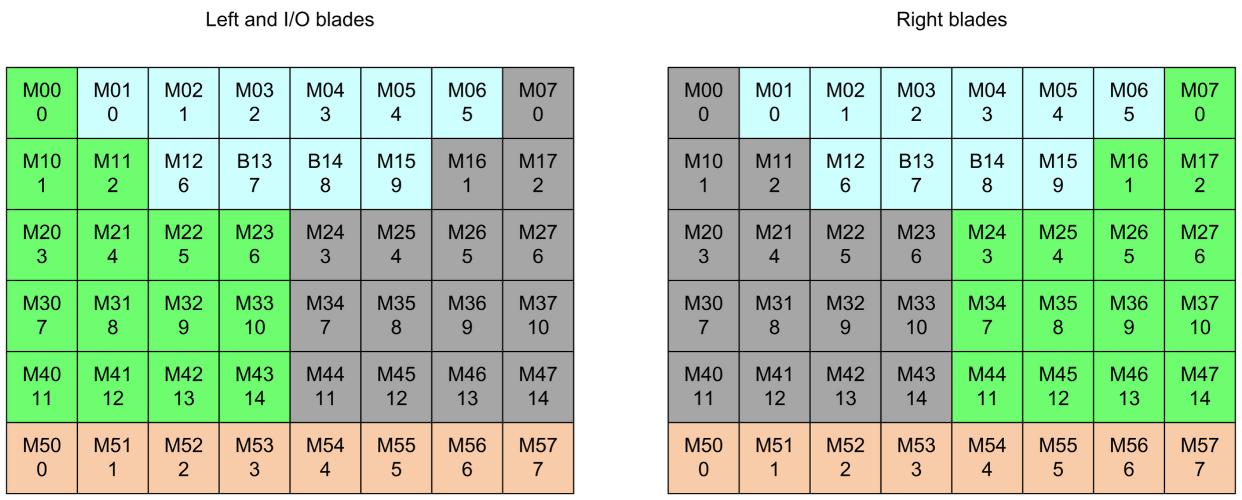
The Aries technology coupled with the Dragonfly topology provides a much richer platform for adaptive routing and traffic spreading and theoretically should drastically decrease the level of contention experienced by applications running at equivalent scale on an equivalent scale machine as compared with the Gemini based 3D torus. The Aries utilizes adaptive routing algorithms which have dependencies on a forwarding router's view of congestion at its possible targets. Nevertheless, significant congestion has been seen in the Aries networks on moderate sized systems. Because of the number of variables used in making each routing decision and the variability of system state, analysis and attribution is much more difficult than in the deterministic Gemini based 3D torus case. As in the Gemini case, network performance counters are only available to the nodes for an Aries chip located on their blade. This results in the same restriction with respect to the nodes of an application not being able to get insight into network contention at every hop along its traffics paths.

In the Aries based network, at each router along a packet's path, there are multiple options for which port a packet can be forwarded to move it towards its destination. Routing rules determine the legal set of ports for the packet's next hop. Options can include hops along both minimal path and non-minimal hop paths. The decision of which port is actually selected, 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.

The Aries Dragonfly topology thus uses three different levels of interconnectivity, with specific Aries router tiles being designated for each (see Figure 2.1). At the chassis level there are 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 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.

Black and Green connectivities are shown in Figs 2.2. Blue connectivities are shown in 2.3.

The Aries router, 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 2.1. Aries tiles with color denoting one of Orange (NIC electrical), Green (intra-chassis electrical), Black (intra-group electrical), and Blue (inter-group optical) (From [7]).

Network congestion concerns in large-scale Aries base systems have been largely focused on applications in *groups* affecting applications in remote *groups*. This can occur because

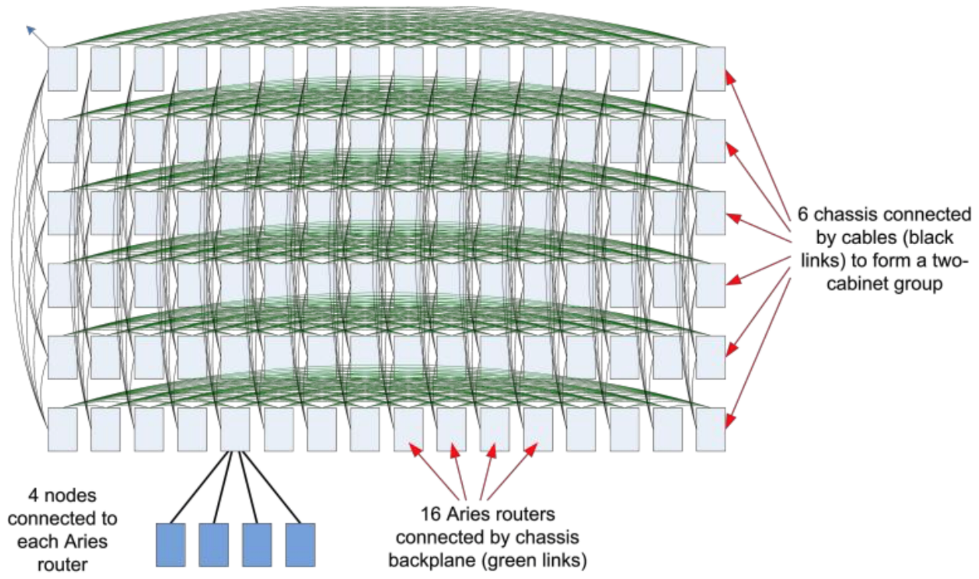

6 chassis connected by cables (black links) to form a two-cabinet group
4 nodes connected to each Aries router
16 Aries routers connected by chassis backplane (green links)

Figure 2.2. 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 [4])

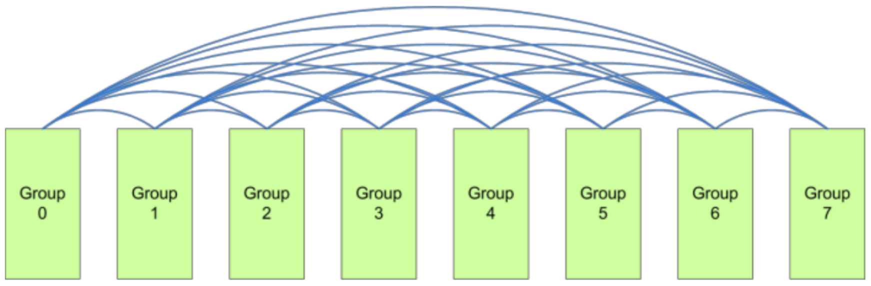


Figure 2.3. The global (Blue) optical links connect Dragonfly groups together. (from [4])

communication between *groups* can involve intermediate routers along the path in other *groups*. Thus the potential for wide-spread contention exists. However, note that the routing algorithm also supports non-minimal routing even for communications within a *group*.

2.2 Assessing Contention in the Aries Network

There are nearly two thousand Aries network 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. These counters are used by the system to identify areas, occurrences, and longevity of high levels of congestion. In this work, we continuously monitor all Aries routers on the system for such information in order to relate traffic levels and contention in the HSN for correlation with application performance variation. Counters (metrics) of interest 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.

NIC and RTR metrics relevant to this work are given below. A more extensive list, including detailed descriptions, can be found in [8].

- `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.

The term congestion is used 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` over 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.

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. Severity of impact will depend on the both the severity and longevity of the congestion.

The behavior of these quantities in regular operation is largely unknown since Cray does not provide a mechanism for regular collection and exposure of this data to the user. In this work, we provide such a facility and use it to analyze measures of contention.

Chapter 3

Experiment Description

This section will discuss the motivation and implementation for the experiment to measure and quantify resource contention on HPC systems. The discussion will begin with an overview discussing the chosen system and applications along with brief descriptions of the experiment components (Section 3.1). Following this overview will be in depth discussions on Mutrino (Section 3.2), data collection (Section 3.3), IOR (Section 3.4), IMB (Section 3.5), and CTH (Section 3.5).

3.1 Overview

The motivation for this effort comes from the desire to minimize application run time and platform instability. There is a need for communicable and actionable metrics that will quantify applications' impacts on HPC systems. This motivation informs the experimental needs and requirements. A relevant platform is needed in order for the metrics to be widely applicable, and the applications chosen must adequately induce platform instability and represent real application behavior.

We chose SNL's Mutrino system as our experimental platform because it was the same architecture and software environment as our target, Trinity, had exclusive access to a Lustre file system and we could easily schedule Dedicated Application Time (DAT) where we had exclusive access to the system.

Mutrino is an unclassified testbed system with 100 Intel Haswell nodes with 32 cores each, 100 Intel KNL nodes, 64 GB of memory per node. The Haswell nodes were used exclusively due to their familiarity and ease of use. Mutrino has a captive Lustre file system that is only accessible to Mutrino users. This provides a high level of confidence that the system will provide consistent metrics during a scheduled DAT. Mutrino was available for exclusive use during the desired project timeline. Mutrino will have relevance to the broader HPC community by virtue of being the Sandia test bed platform for the latest DOE Advanced Technology System (ATS) deployed at LANL, Trinity.

The chosen applications were selected for their relevance in creating network and file system resource degradation as well as being representative of typical application behavior. The chosen applications were IOR, IMB, and CTH. IOR was chosen to introduce contention to the file system hardware. IMB was chosen to introduce contention to the network infrastructure, and CTH was chosen to represent typical application behavior. Each subsequent sub-section will discuss in further detail each application's role in the overall experiment.

3.2 Mutrino

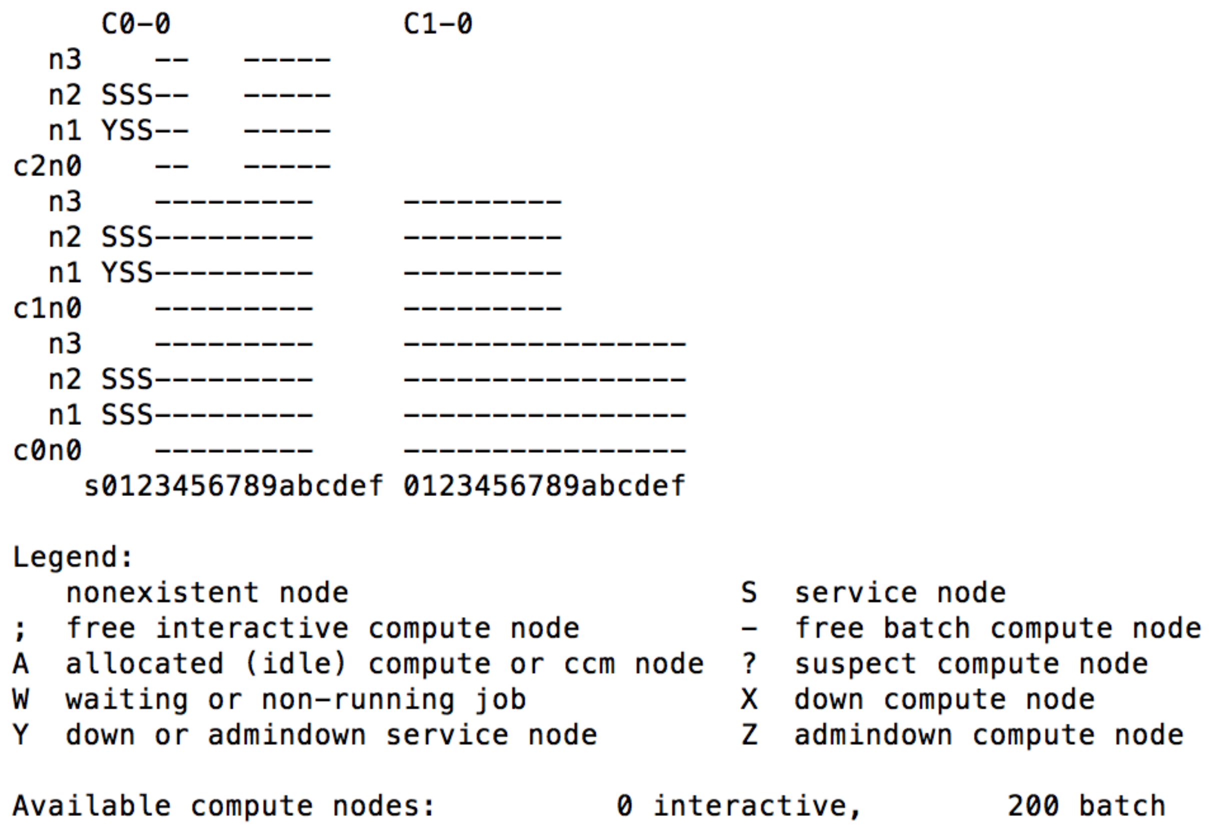
Trinity consists of approximately 110 cabinets of Intel based Haswell and Intel based KNL compute node cabinets and utilizes the same Aries interconnect for all cabinets. Thus Trinity has ~55 *groups*. In contrast, Mutrino consists of 100 Intel Haswell and 100 Intel KNL nodes in each of two incompletely populated cabinets, comprising only a single *group*. Due to the incomplete population, some of the Aries routers and thus the related green and black links may not exist.

Among the existing Aries routers, all green and black connectivity is in place and therefore traffic from any node can reach any other node in at least one and at most two hops (one black and one green in either order). The routing algorithm allows for non-minimal hop routes, however, and one item of investigation is the actual usage of the network with respect to job placement on the system.

The actual system layout, as provided by the output of the `xtnodestat` command, is shown in Figure 3.1 from which the existing Aries routers can be inferred. (The figure is not to scale, nor is it meant to be exactly representative of the actual physical layout of the machine.) The Haswell nodes, which were used in this experiment, are in cabinet `c0-0`. The KNL nodes, which were not used by applications, are in cabinet `c1-0`. The cabinets may appear unevenly populated because of the additional service nodes in the Haswell cabinet only. The chassis are numbered bottom to top (e.g., `c0-0c0`). The 16 blades are labeled along the bottom. Compute nodes are indicated by the `-`. Service nodes are designated by `S` or `Y`. Nodes in the same chassis and slot share a common Aries router. For example, the 2 service nodes `c0-0c0s0n1` and `c0-0c0s0n2` share the common Aries router, `c0-0c0s0a0`; the 4 compute nodes `c0-0c1s6n0` through `c0-0c0s1n3` share the common Aries router `c0-0c0s1a0`. In the rest of this document, since there is only a single Aries router per slot, we will suppress the the `a0` in the Aries router name and use the slot designator (e.g., `c0-0c0s1`) to refer to the Aries router.

Nodes can be designated by the *cname*, as above, or by a *nidname*, or by a *nidnumber*. For example, `c0-0c0s0n1` is also `nid00001` and nid number `1`. Functionally, the *cname* to *nid* mapping can be easily found from the host table or in the *proc* file system on a node. The *nidname* and *nidnumber* are predominantly used in job descriptions in this paper.

The Lustre parallel file system on Mutrino is not cross-mounted on any other HPC platforms or systems. Ergo, it is possible to limit the system-wide file system usage during a controlled experiment. Mutrino is a small-scale counterpart to Trinity and, as a result, its file system is much smaller than the one on Trinity. The Mutrino Lustre filesystem contains 4 object storage targets, OSTs, and its filesystem is set to have a default striping of 4. This



```
     C0-0             C1-0
  n3    --   -----
  n2 SSS--   -----
  n1 YSS--   -----
c2n0    --   -----
  n3    ---------     ---------
  n2 SSS---------     ---------
  n1 YSS---------     ---------
c1n0    ---------     ---------
  n3    ---------     ----------------
  n2 SSS---------     ----------------
  n1 SSS---------     ----------------
c0n0    ---------     ----------------
    s0123456789abcdef 0123456789abcdef

Legend:
   nonexistent node                      S  service node
;  free interactive compute node         -  free batch compute node
A  allocated (idle) compute or ccm node  ?  suspect compute node
W  waiting or non-running job            X  down compute node
Y  down or admindown service node        Z  admindown compute node

Available compute nodes:          0 interactive,        200 batch
```

Figure 3.1. Output of xtnodestat showing the existing nodes. From this the existing Aries routers and their connectivities can be inferred.

means that any file written to the file system from any users will be split into 4 separate pieces with each one stored on a separate OST. Because of this small number of OSTs, forcing contention is a matter of saturating the write/read requests to these 4 OSTs, which is easily accomplished given the node count of Mutrino. These attributes make Mutrino an ideal platform for performing file system contention experiments.

3.3 Data Collection via LDMS

Since Cray does not provide a facility for the regular collection, transport, and presentation of the type of system data we are interested in, we utilize the Lightweight Distributed Metric Service [9] for our data collection, transport, and storage infrastructure. LDMS a) is highly scalable, b) has been shown to have low impact both for large number of nodes and metrics collected per node, and c) supports synchronized sampling (with respect to each node's clock) which is necessary for comparing data coherently across the entire system. In this work, we collect data on all nodes at one second intervals and transport it off the system as it is being collected.

Details for Aries router network counter collection on Trinity are given in [3]. The Mutrino configuration is similar, but simpler. On Mutrino, we collect data on all compute and service nodes. The data is pulled to an aggregator on the service node `c0-0c0s1n1`. Since it is not a compute node, no application processes will be run on this node. Note that Aries routers that are one green or one black hop away are potentially on the route taken by traffic going from data samplers to the aggregator. This is discussed in more detail in Section 4.1.1.

The data is subsequently pulled from the first-level to a second-level aggregator external to Mutrino for processing and storage. New to this work is enhanced redundant collection on the Service nodes. All data for an Aries router is accessible to all nodes that share the same blade as the Aries router. Collection of all available Aries router counter data at each node would result in 4x redundancy of monitored data and traffic for the compute nodes and in most cases is unnecessary, however some level of redundancy is desired in case of node failure or network loss. Thus we collect with 2x redundancy - collecting all the Aries router data available to each of the service nodes (2 service nodes per Aries router) and half of the Aries router data available to each of the compute nodes (4 compute nodes per Aries router).

As described in [3], raw network counters on node are accessed via Cray's Generic Performance Counter Daemon (*gpcd*) interface. In order to facilitate analysis, *in transit* computation of derived quantities are calculated and stored at the last level aggregator.

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 2.2) and store it as a new variable currently chosen to be named `AR_RTR_r_c_STALL_FLIT_RATIO`.

Note that we collect data on the nodes, but the quantities to be assessed are per-Ariesrouter. Thus, our collected data includes both node and Aries router identifiers in order to facilitate making such associations across different nodes' data.

Client side Lustre data is accessed on the node via `/proc/fs/lustre/llite/snx*/stats` where `snx*` is the name of the Sonexion Lustre mount point. In this work, the metric of interest from this set of data is Lustre bytes written.

3.4 IOR

Our goal for this portion of the study is to run a widely used and well characterized Input/Output (I/O) performance "benchmark" application that can generate large amounts of file system I/O traffic. We will vary the number of running instances of the application and the individual I/O load of each instance to demonstrate contention for file system resources between multiple running instances of the application. We selected the Interleaved or Random (IOR) [10] benchmark that was developed at Lawrence Livermore National Laboratory (LLNL) to evaluate file system performance for the Purple platform procurement. IOR is commonly used within the High Performance Computing (HPC) community [11][12][13] and we are using the 2.10.3 version of the benchmark obtained from Lawrence Berkeley National Laboratory (LBNL) National Energy Research Scientific Computing Center (NERSC) [14].

IOR uses the Message Passing Interface (MPI) for process synchronization and to execute the I/O tests across multiple Lustre/Compute Clients and therefore requires that the HPC platform have an implementation of MPI. IOR performs file writes/reads to the parallel file system, measures the read/write performance under different file size, I/O transaction size, and concurrency, and reports the resulting I/O throughput rates. Input arguments to IOR allow control of the overall I/O size, individual transfer size, file access mode (file-per-process or single shared file), and whether the data is sequentially or randomly accessed. File-per-process creates a unique file per task and is the most common way to measure peak throughput of a Lustre Parallel Filesystem. Single shared file mode creates a single file across all tasks running on all clients. IOR can be configured to test parallel file system I/O performance using the traditional POSIX I/O interface or more advanced parallel I/O interfaces (via middleware libraries), including MPI-IO, HDF5, and Parallel-NetCDF. IOR can therefore be used to simulate the I/O patterns of "real" HPC applications.

An IOR test plan was developed to adequately measure and quantify resource contention on HPC file system hardware. The test would need to provide data from multiple IOR cases that could be used to determine application slowdown. It was determined that two classes of cases would be required in order to obtain this data: continuous contention cases and varied contention cases. The continuous contention cases would have IOR processes start simultaneously in order to provide expected run times for varying levels of contention. IOR processes would be staggered in the varied contention cases. The continuous contention cases would be used as a benchmark for specific IOR processes in the varied contention cases in order to measure the increased run time.

With the chosen platform, IOR scripts and settings had to be adjusted to properly induce scheduled contention on the file system hardware. Mutrino has four OSTs that handle the I/O load of the system. In order to fully load all OSTs, IOR runs were adjusted to have a striping level of four for each file that was written out. To create the contention cases, major tweaks made to the run scripts included an algorithm to spawn multiple single node IOR runs simultaneously, input values to stagger specific IOR runs, and input values to specify the file output size for each IOR run. These modifications allowed for varied levels of contention during the same run and for longer runs that could be put under varied levels of contention.

Continuous contention cases of 1, 2, 3, 4, 8, and 16 nodes were run in order to provide an adequate benchmark for the varied contention cases. Table 3.1 shows all varied contention cases with the input values that determined the run schedule, and the expected level of contention throughout the case. A file size of 64 GB was set as the default file size. The DAT results section will discuss the success of determining application slowdown using these cases.

Table 3.1. Varied contention cases with sleep times and file size multipliers for each node. A file size of 64 GB was set as the default file size. For each case, the input values were set in order for a nearly symmetric contention schedule to be produced. For cases 19-21, one IOR run was started at time 0 with a file size of 256 GB. Three IOR processes were then started 120 seconds later with file sizes of 64 GB. These delayed processes would then finish before the initial process. This description parallels the contention schedule seen in the fifth column (1-4-1). There was initially a single IOR process. Once the other three processes were started, there were four IOR processes running simultaneously. Once these three processes ended, the initial process kept running until the end of the case. This gave periods of no contention under the same IOR process before and after a period of contention.

Cases	Nodes	Sleep Time (s)	File Size Multiplier	Contention Schedule
19-21	4	0, 120(x3)	4, 1(x3)	1-4-1
22-24	8	0, 120(x7)	4, 1(x7)	1-8-1
25-27	16	0, 120(x15)	4, 1(x15)	1-16-1
28-30	8	0(x2), 135(x6)	4(x2), 1(x6)	2-8-2
31-33	16	0(x2), 135(x14)	4(x2), 1(x14)	2-16-2
34-36	8	0(x4), 150(x4)	4(x4), 1(x4)	4-8-4
37-39	16	0(x4), 150(x12)	4(x4), 1(x12)	4-16-4
40-42	16	0, 120(x3), 200(x4), 260(x8)	8, 4(x3), 2(x4), 1(x8)	1-4-8-16-8-4-1

3.5 IMB

Our goal for this portion of the study is to run a widely used and well characterized Message Passing Interface (MPI) performance "benchmark" application that can generate large amounts of network traffic. We will vary the number of running instances of the application and the individual network load of each instance to demonstrate contention for network resources between multiple running instances of the application. We selected the Intel MPI Benchmarks (IMB) [15] that was developed to provide MPI performance measurements for point-to-point and global communication operations for a range of message sizes. IMB (formerly the Pallas MPI Benchmarks) is commonly used within the High Performance Computing (HPC) community [16][17] and we are using the version 4.1 of the benchmark obtained from Intel.

IMB requires that the HPC platform have an implementation of MPI, including a startup mechanism for parallel MPI programs. Benchmark data generated by IMB characterizes the performance of a cluster system, including node performance, network latency, throughput, and the efficiency of the MPI implementation used.

The IMB package consists of the following five components: IMB-MPI1 (benchmarks for MPI-1 functionality), IMB-EXT (one-sided communications benchmarks for MPI-2 functionality), IMB-IO (input/output benchmarks for MPI-2 functionality), IMB-NBC (non-blocking collectives benchmarks that provide measurements of the computation/communication overlap and of the pure communication time for MPI-3 functionality), and IMB-RMA (Remote Memory Access benchmarks that use passive target communication to measure one-sided communication for MPI-3 functionality). Each component corresponds to a separate executable file.

The IMB-MPI1 component was selected and the `MPI_Alltoall` test was focused upon since this routine requires each MPI rank to send data to and receive data from all other MPI ranks. This simple test creates a lot of network traffic and forces all ranks to communicate with each other which makes it sensitive to shared resource contention.

IMB was run in several different configurations based upon the system hardware layout. Several node groupings were created to facilitate the creation of these run configurations. The details for these node groupings can be seen in Tables 3.2 and 3.3 . Groupings were split by node numbering and chassis number. This was done in order maximize the hardware utilization for each IMB run and capture a more significant level of contention with multiple IMB processes running. Using these node groupings, three main configurations of contention runs were created as follows: run all even or odd numbered nodes, run all nodes in a chassis, or run all nodes. The contention configurations are detailed in Table 3.4.

Table 3.2. Node groupings for IMB contention cases.

Case	Node List
Even Nodes 1	12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46
Even Nodes 2	76,78,80,82,84,86,88,90,92,94,96,98,100,102,104,106,108,110
Even Nodes 3	140,142,144,146,160,162,164,166,168,170,172,174,176,178
Odd Nodes 1	13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47
Odd Nodes 2	77,79,81,83,85,87,89,91,93,95,97,99,101,103,105,107,109,111
Odd Nodes 3	141,143,145,147,161,163,165,167,169,171,173,175,177,179

Table 3.3. MPI Ranks and Chassis for IMB contention cases.

Case	Chassis	MPI Ranks
Even Nodes 1	0	576
Even Nodes 2	1	576
Even Nodes 3	2	448
Odd Nodes 1	0	576
Odd Nodes 2	1	576
Odd Nodes 3	2	448

A message length of 524,288 bytes was used since that was the maximum amount that IMB allows at the MPI rank counts required to use 32 ranks per node on half of a chassis of Mutrino. A sample invocation of IMB is provided below along with a CrayPat[18] profile of this `Alltoall` test. The profile shows that almost 99% of the total runtime is spent

Table 3.4. List of contention cases for the IMB experiment.

Case	Node Grouping Run	Contention Schedule
25, 28, 31	Even Nodes 1, Odd Nodes 1	Start 180s apart
26, 29, 32	Even Nodes 2, Odd Nodes 2	Start 180s apart
27, 30, 33	Even Nodes 3, Odd Nodes 3	Start 180s apart
34, 35, 36	All Even Nodes, All Odd Nodes	Start 180s apart

performing the `MPI_Alltoall` function a total of 202 times.

```
aprun -n 576 -N 32 \
-L "12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46" \
./IMB-MPI1 -npmin 576 -iter_policy off -iter 100 \
-msglen msglen.txt -time 3600 Alltoall
```

Invocation of IMB for “Even Nodes 1” MPI-1 `Alltoall`, 100 iteration case with message lengths of 524,288 bytes.

```
Table 1:  Profile by Function Group and Function

  Time% |       Time |     Imb. |  Imb. | Calls |Group
        |            |     Time | Time% |       | Function
        |            |          |       |       |  PE=HIDE

 100.0% | 305.975193 |       -- |    -- | 896.5 |Total
|-------------------------------------------------------------
|  98.7% | 301.995471 |       -- |    -- | 641.0 |MPI
||------------------------------------------------------------
||  98.7% | 301.976727 | 0.290653 |  0.1% | 202.0 |MPI_Alltoall
||============================================================
|   1.1% |   3.464625 |       -- |    -- | 218.0 |MPI_SYNC
|=============================================================
```

CrayPat profile of “Even Nodes 1” MPI-1 `Alltoall`, 100 iteration case with message lengths of 524,288 bytes. This IMB invocation performs 202 total calls to `MPI_Alltoall` and these calls make up the vast majority of the overall runtime.

3.6 CTH

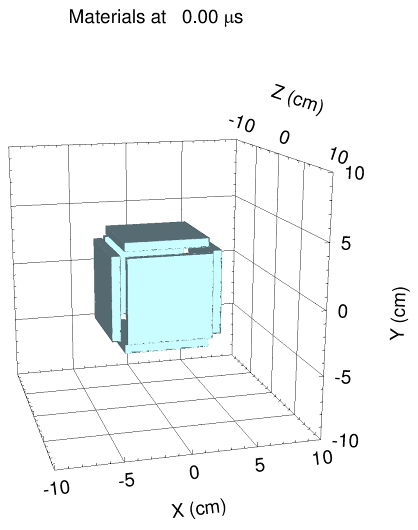
To examine the potential for and effectiveness of identifying contentious behavior in a production environment, the CTH [19] hydrocode was used. CTH is a massively parallel Eulerian finite volume code for solving large deformation and strong shock problems, and is widely used throughout the DOE and DoD. It uses a structured Cartesian mesh and has block based Adaptive Mesh Refinement (AMR). The parallelization is done through a standard halo cell exchange method using MPI or memory copies within the same MPI task. Profiling of the code has shown that it is memory bandwidth bound, and the algorithms used have a nearest neighbors communication pattern.

For all the CTH runs in this study, an AMR problem representative of a typical use case is used. The problem consists of 6 spheres impacting 6 different targets at 1 km/s. Plots of the problem at $t = 0$ and at $t = 10\mu s$ are shown in Fig. 3.2a-3.2d. Several variants of this problem were run to identify a case which was the most sensitive to system contention, while still being representative of a typical workload a user would run. A problem with minimal IO, and 10^3 cells per AMR block is used, leading to message sizes of approximately 100-225 mb per cycle.

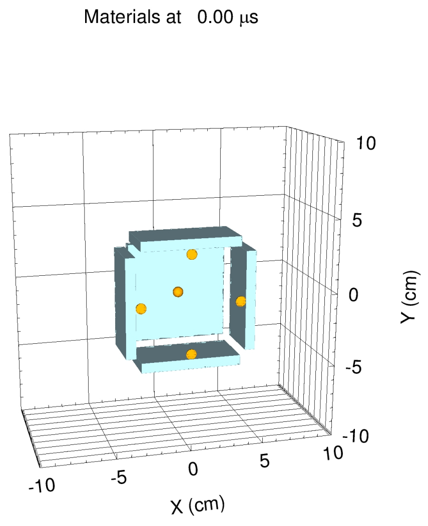
Each simulation is run using 512 MPI tasks (16 nodes), spread as evenly as possible across all the chassis in the cabinet. The node lists used for each of the CTH runs are listed in Table 3.5, and shown visually in Fig. 3.3. First, a baseline case is run using the same node list without any other jobs present. The case was run 3 times to make sure the results were repeatable. Next, 2 of the same cases are run using 512 MPI tasks each, spread across the cabinet while 3 IMB jobs are launched approximately 58 seconds after the start of the CTH cases on all remaining nodes to simulate a contentious condition for the system. Running multiple CTH cases to fill the entire cabinet is not able to generate substantial contention between themselves. The overall runtimes are found to be within 1-2% of the baseline for these cases because the code tends to send MPI messages in bursts instead of continuously, so the chances of the messages being contentious with each other is lower.

Table 3.5. Node lists used for 2 simultaneous CTH runs during the contention experiment.

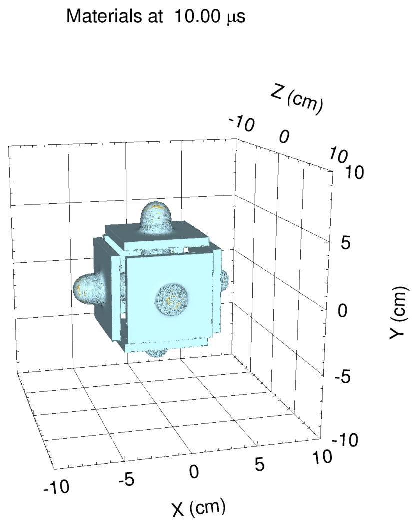
Case	Node List
a	162,23,87,163,24,88,164,25,89,165,26,90,166,27,91,167
b	12,76,140,13,77,141,14,78,142,15,79,143,16,80,144,17



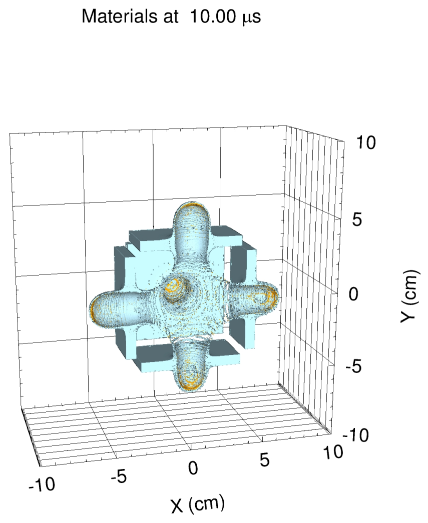
(a) Initial condition.



(b) Initial condition, cut at $z = 0$ plane.



(c) Simulation at t=10μs.



(d) Simulation at t=10μs, cut at $z = 0$ plane.

Figure 3.2. Problem used for CTH cases.

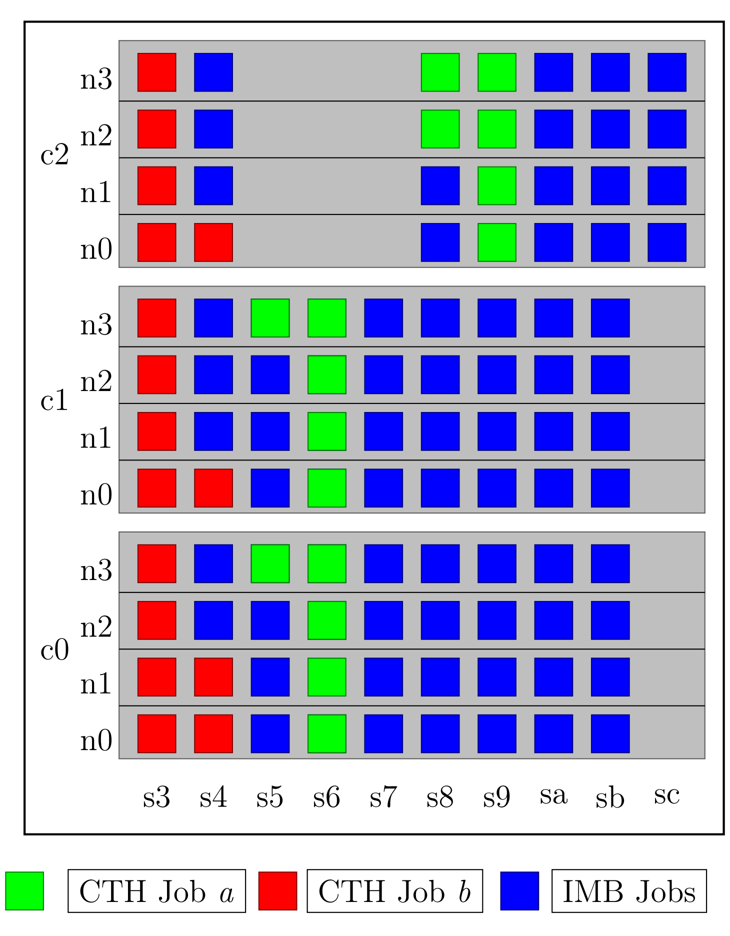


Figure 3.3. Visualization of node list used for CTH contention cases. Note that the x-axis starts with the compute nodes, the first 3 service nodes are not shown.

Chapter 4

Experiment Results

This section will discuss the results from the Mutrino DAT starting with a brief synopsis of the DAT and the experimental results collected during the DAT (Section 4.9). In depth discussions related to DAT results follow from this for background traffic (Section 4.1.1), network utilization (Section 4.1.2), IOR (Section 4.1.3), IMB (Section 4.1.4), and CTH (Section 4.1.5).

4.1 DAT Results

To gather results for the relevant applications, an exclusive period of use was scheduled on Mutrino. The Mutrino DAT was scheduled for a period of less than 24 hours due to the needs of the applications. Each application would need only several hours to collect the relevant data.

The Mutrino DAT was successful and contention was observed for all applications that were run. The IOR runs measured considerable slowdown on the file systems for continuous and varied contention cases. The largest slowdown reported was 213 seconds for a continuous IOR contention run and 370 seconds for a varied IOR contention run. IMB contention cases had up to 80% increased run time due to contention. The CTH cases displayed increased run times of around 7.8% and 1.5%.

4.1.1 LDMS and System Traffic

In order to derive actionable metrics from the data, we must first be assured that our data collection system is not significantly and/or irregularly affecting the measurement. Thus we examined the LDMS traffic data over a period of time with no application running. The data collected was stored by the second level aggregator to a local disk and thus had no impact the Luster filesystem.

In the Mutrino configuration, the LDMS aggregator was run on a node associated with `c0-0c0s1`. Therefore, we expected that system LDMS traffic will be routed through routers with green link connectivity to that Aries router: these are `c0-0c0sX` for `X` in slots 0 and

2-15 (if populated), and also Aries routers with black link connectivity to that Aries router: these are are `cY-0cZs1` which are Aries routers in all blade 1 positions in all other chassis in the two cabinets (if populated). These Aries router links would be expected to show higher network traffic than others.

A comprehensive study of the values of the individual Aries router link traffic values will be the subject of future work. In this section we present enough general data to show that the LDMS and System traffic measures are insignificant with respect to the application traffic.

Note that some Aries routers in figures in this Section may show less than the potential full compliment of links (e.g., less than 15 green or black links for an Aries router). This is reflective of the incomplete population of the Mutrino cabinets as described previously. Note that the network metric of interest represents *incoming* traffic into the Aries router. The Figure labeling, for example, for Figure 4.1 indicates incoming traffic into Aries router `c0-0c01s3`, and each line's title triad $(metricname, color, Ariesrouter)$ indicates the color of the link and the Aries router from which the traffic is coming.

Traffic The LDMS and other System traffic, measured by `AR_RTR_r_c_INQ_PRF_INCOMING_FLIT_VC_SUM` (shortened in this Section to `FLITS`) for a generally located Aries router in the system is generally of order of $1000 - 10000$ flits/s and is fairly regular. An example of green and black link traffic for two such Aries routers is given in Figure 4.1.

Since all traffic flows into the aggregator, the highest values are seen there. Figure 4.2 shows the green and black links traffic for the aggregator. The highest values are for the black links connected to `c1-0c0s1`.

Of interest are the Aries routers which are one hop in either direction, black or green, from the aggregator since traffic flows to the aggregator via these links. In general, traffic values on green links of the chassis in which the aggregator node resides, are not noteworthy. However, values of green link traffic in chassis one black hop away e.g., cabinet `c1-0` can be higher than that of the green links local to the chassis of the first level aggregator. These are shown in Figure 4.3.

STALL/FLIT Ratio The `AR_RTR_r_c_STALL_FLIT_RATIO`, (which will be shortened in this Section to `STALL/FLIT`) is typically zero, and small (< 10) when non-zero, on all Aries routers' green and black links regardless of location (unshown) and including the Aries router associated with the aggregator. The exceptions to this occurred on the black links of Aries routers `c1-0c0s0` through `c1-0c0s3`. This exception case is shown for a representative Aries router connected, via a black link, to the aggregator in Figure 4.4. This level of `STALL/FLIT` non-zero occurrences is still far below that seen during application runs and in large scale testing on Trinity has been shown to have no impact on the run times of large scale applications [?]. There was no evidence of network throttles or quiesce events in the logs over this period of time.

Comparison of magnitude Figure 4.5 is the `FLITS` for a CTH run with no contention. This is the run discussed on Figure 4.16. Comparison with the LDMS traffic plots above shows that the CTH traffic is orders of magnitude greater than the LDMS traffic.

This discrepancy is even greater for a communication-heavy scenario such as the IMB Case 25 in Section 3.5, where multiple concurrent IMB all-to-all were run. Traffic plots for a random Aries router with a node involved in the job allocation in this case are given in Figure 4.6.

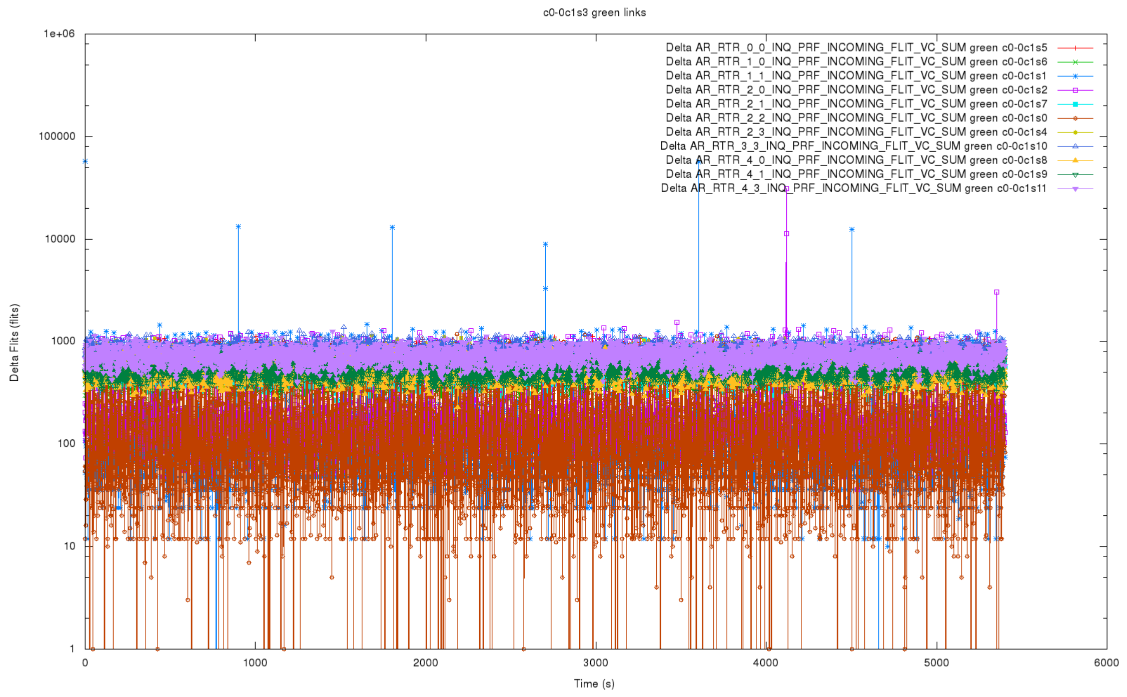
Finally, for this same case we consider the `STALL/FLIT` ratio. This measure of possible contention is markedly higher than the same value applied to the LDMS traffic, as show in Figure 4.7.

4.1.2 Network Utilization

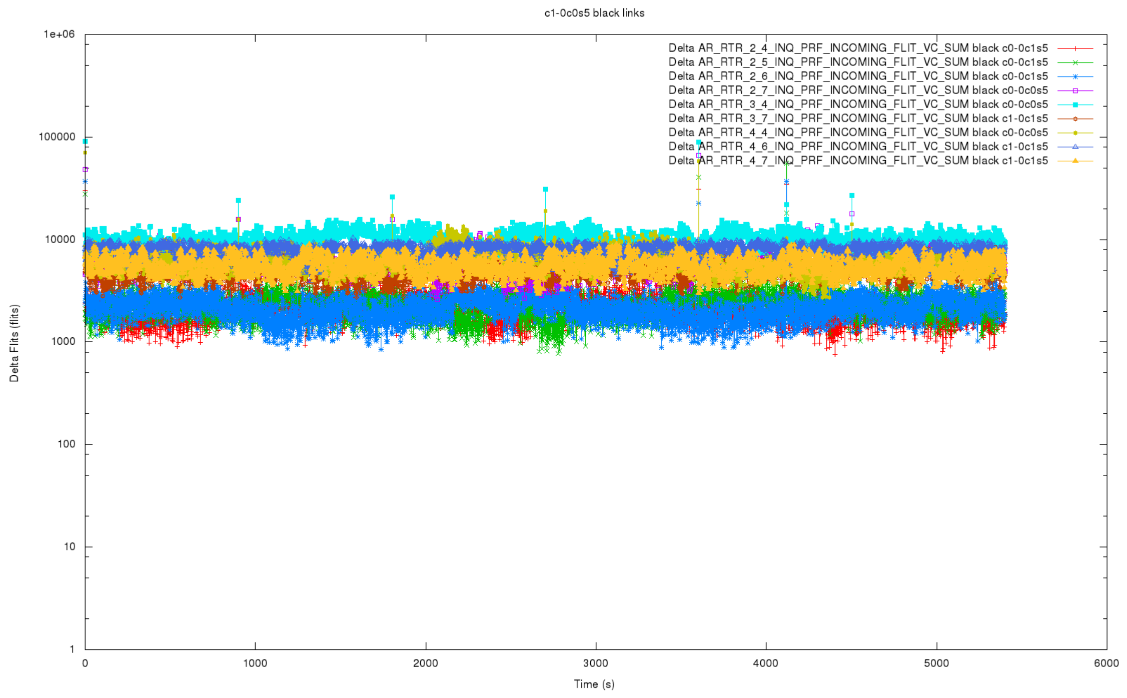
Continuous global monitoring enables us to investigate the Aries network usage which results from the routing rules. Note that if every route taken were the minimal route, everything would be reachable in at most 2 hops. Also that none of the routers in `c1-0` would be used at all (other than for LDMS) since all jobs were in the Haswell `c0-0` cabinet in which every node is reachable from every other node by the green and black links in that cabinet alone.

Figure 4.8 shows network traffic for an Aries router of the system in the IMB case 25 with contention discussed at the end of the last section. We choose this case, since it is communication heavy, running multiple IMB in all-to-all mode. This Aries router is in the `c1-0` cabinet which does not include the nodes allocated to the job. We see that even within a *group*, non-minimal routing occurs and applications can impact others as a result.

Understanding global state for the purpose of attributing contention will be pursued in follow-on work.

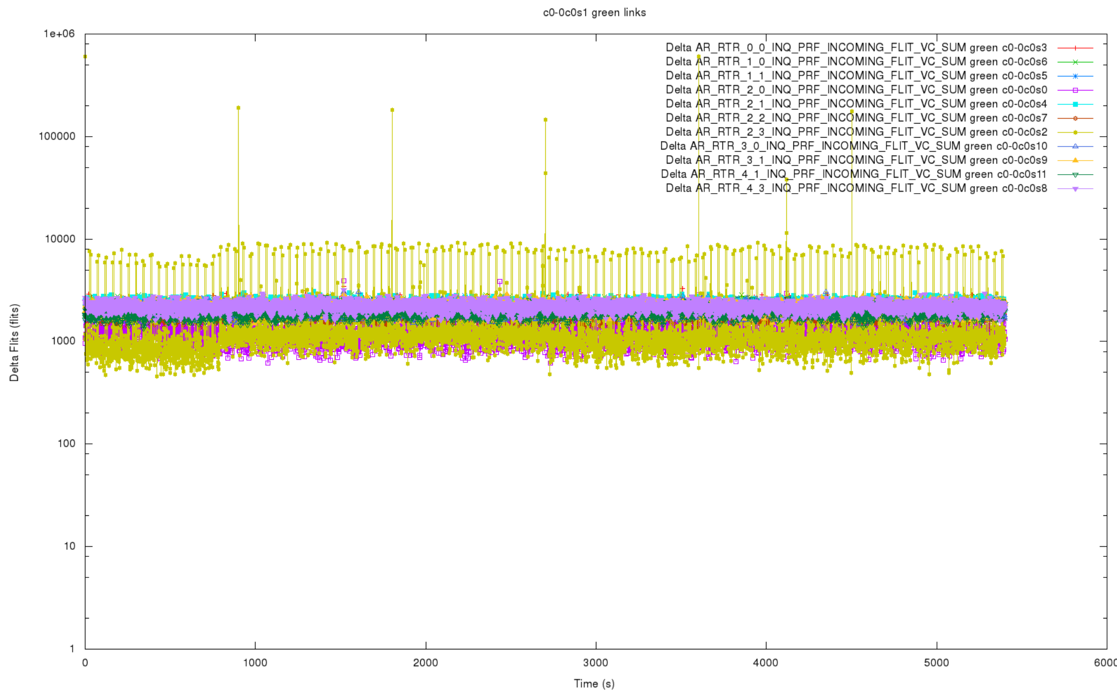


(a) Traffic on green links for a generally located Aries router.

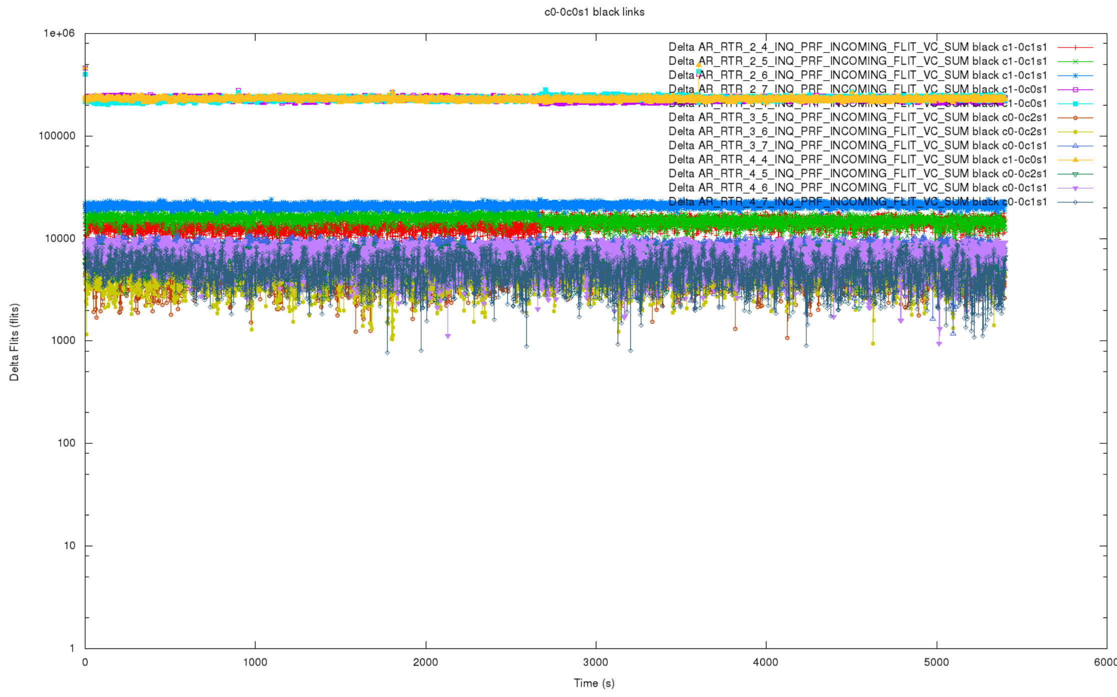


(b) Traffic on black links for a generally located Aries router.

Figure 4.1. LDMS and system traffic for a generally located Aries router.



(a) Traffic on green links for the Aries router of the node running the aggregator.



(b) Traffic on black links for the Aries router of the node running the aggregator.

Figure 4.2. Traffic for the Aries router of the node running the aggregator.

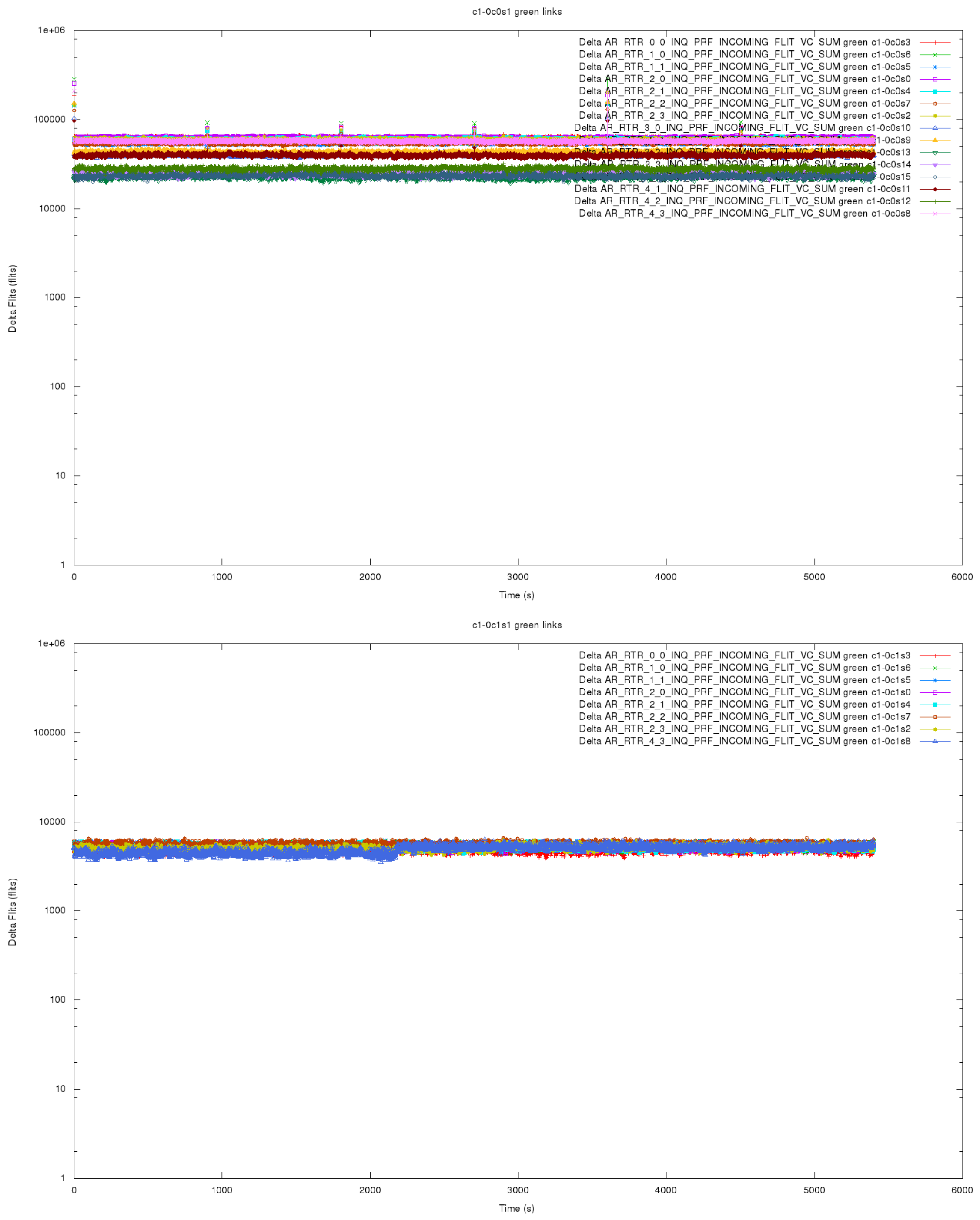


Figure 4.3. Background Traffic across the individual green links for the Aries routers in cabinet `c1-0` 1 black link hop from the aggregator.

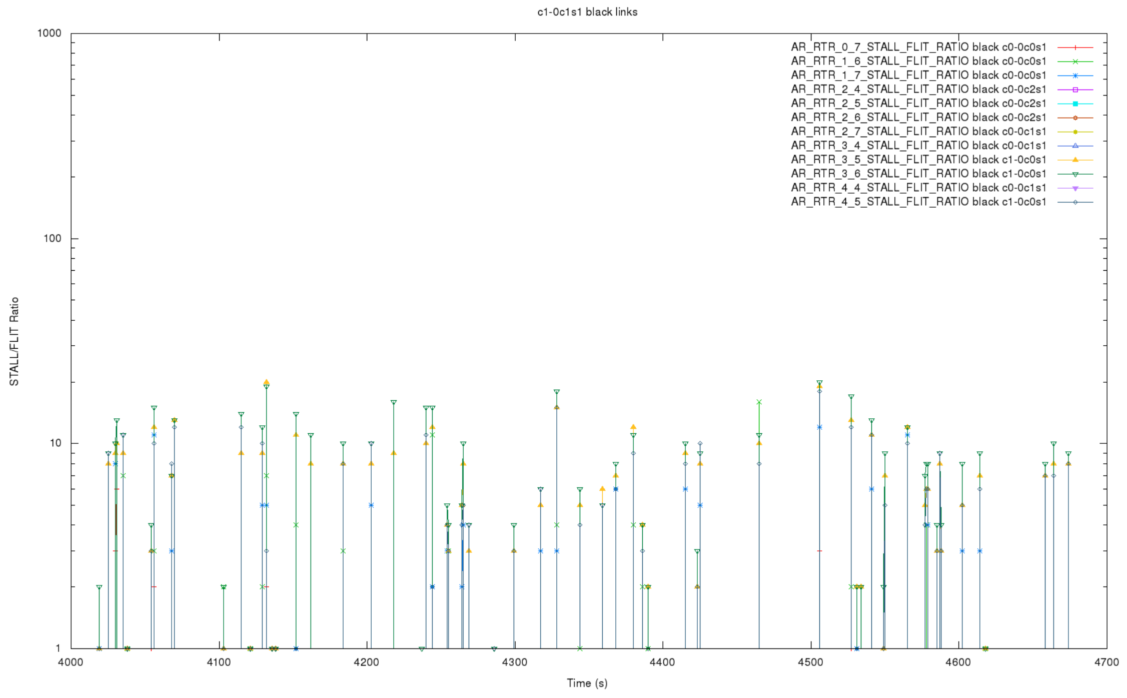
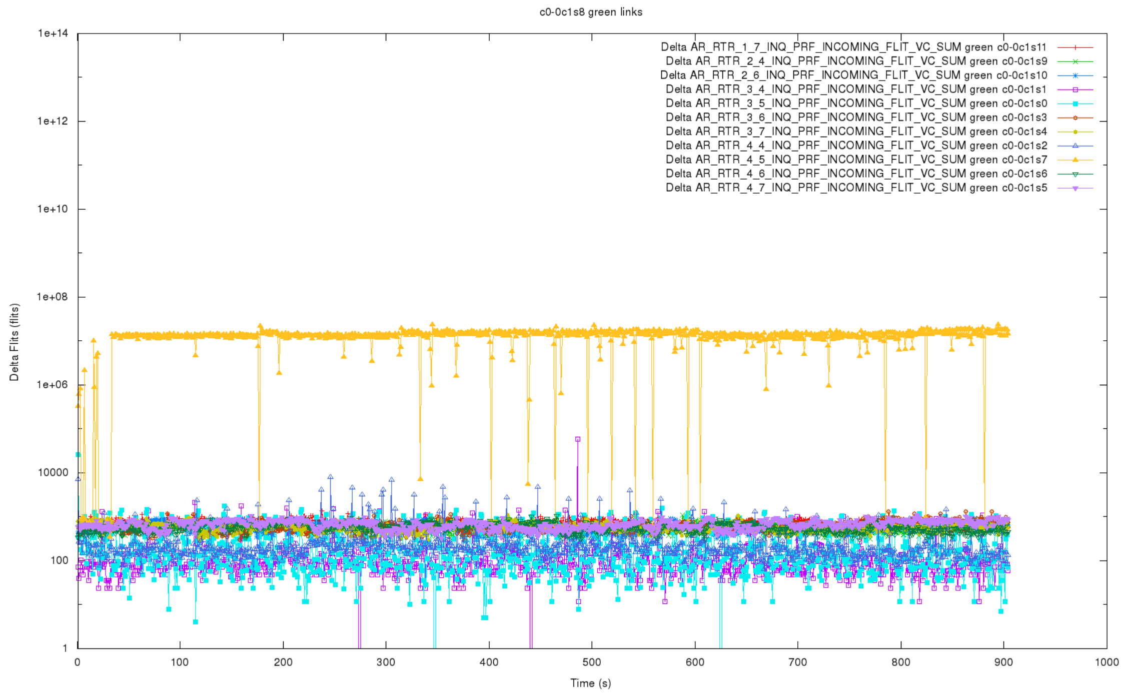
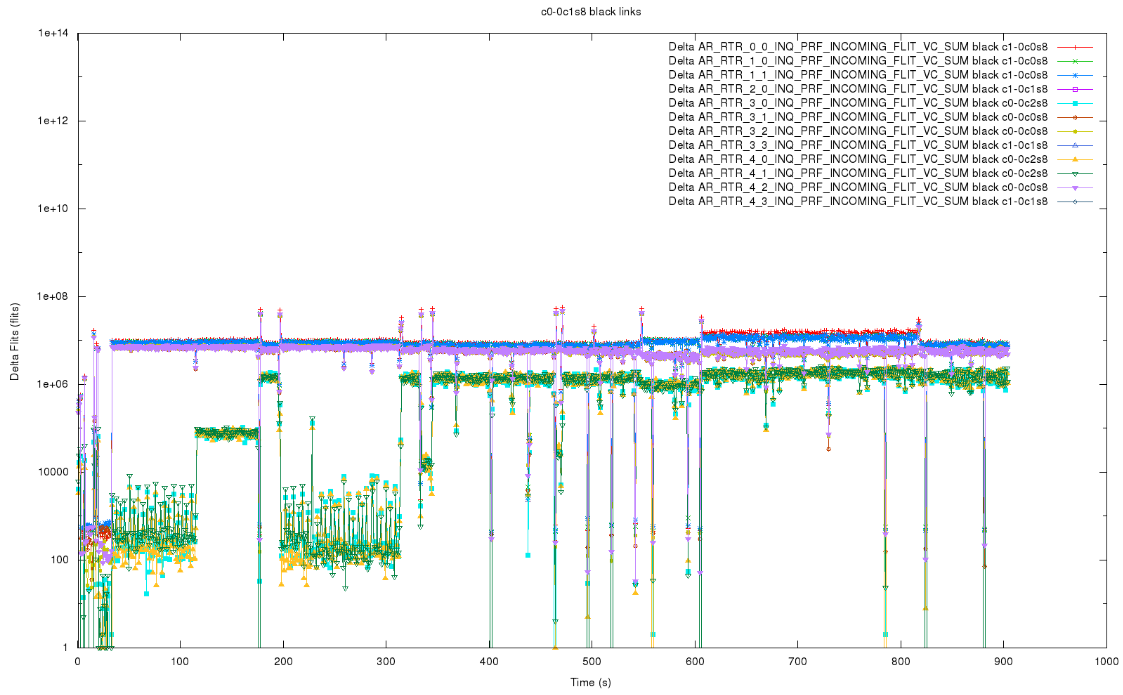


Figure 4.4. `STALL/FLIT` for Aries router `c1-0c1s1`.

.



(a) Traffic on the green links for the Aries router examined in the CTH baseline case. The values around 1000 Flits correspond to the LDMS traffic.

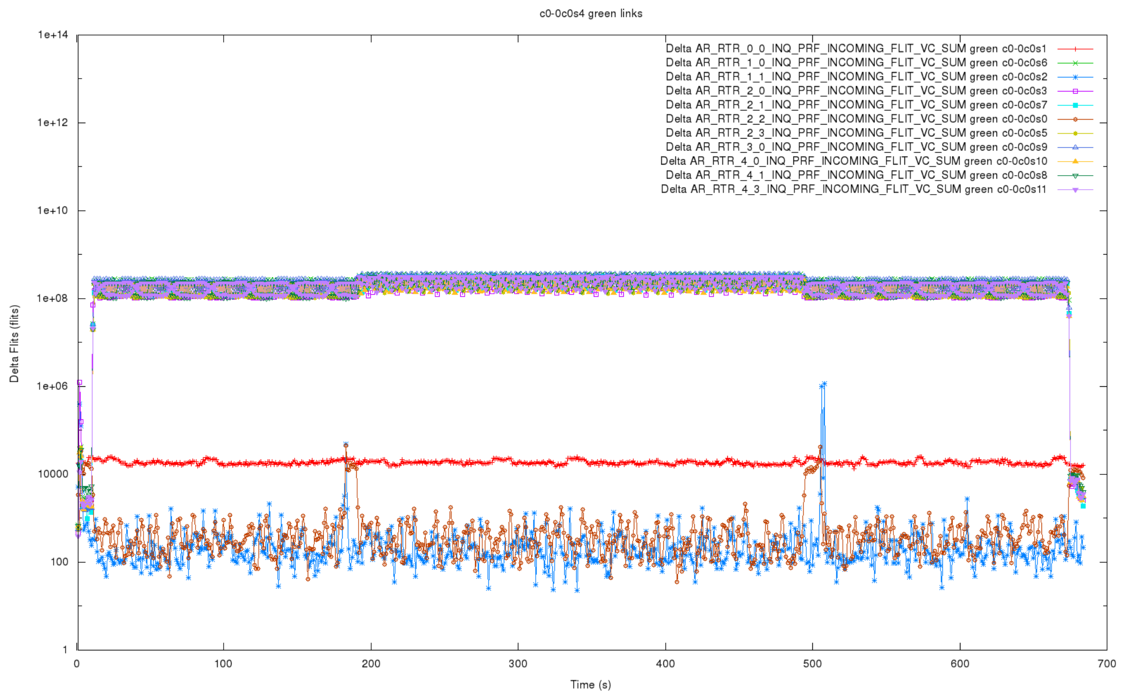


(b) Traffic on the black links for the Aries router examined in the CTH baseline case. LDMS traffic is swamped by the CTH traffic

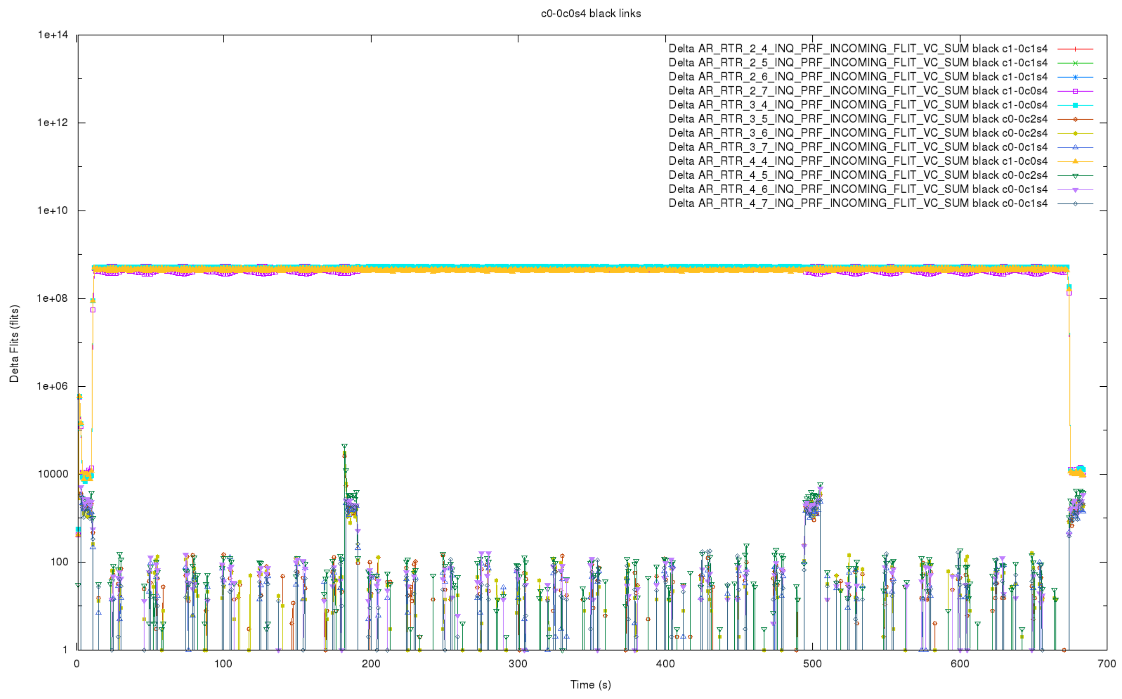
.

Figure 4.5. Traffic on the Aries examined in the CTH baseline case is orders of magnitude greater than that of the LDMS traffic

.



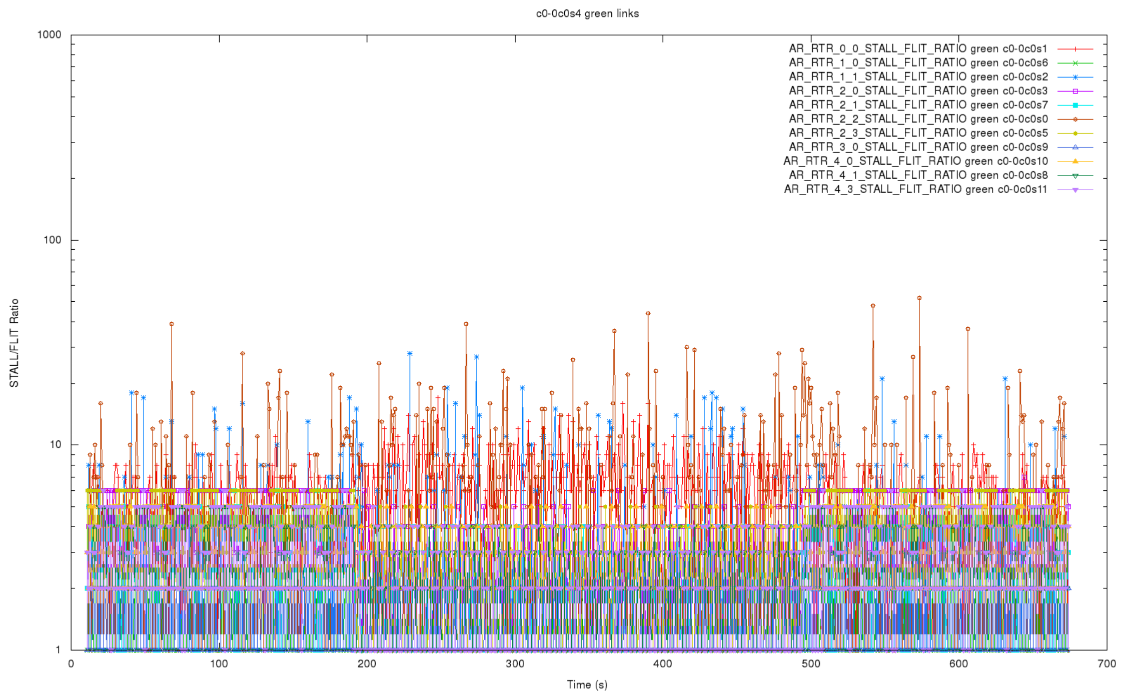
(a) Traffic on the green links of an Aries router in a contentious IMB case.



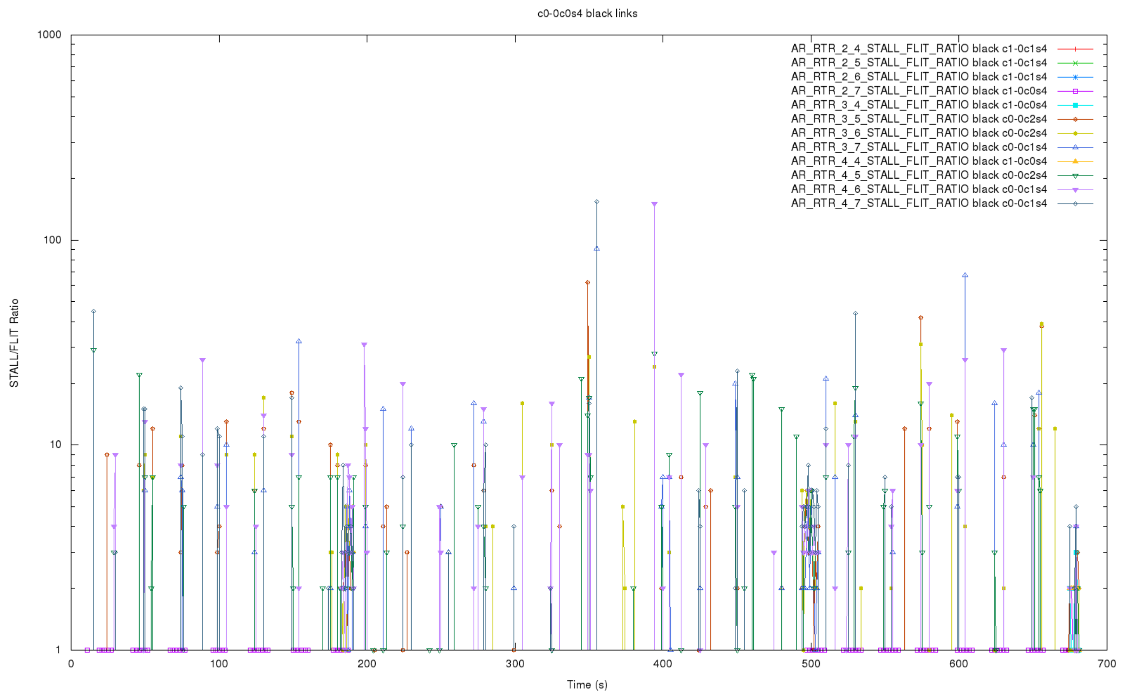
(b) Traffic on the black links of an Aries router in a contentious IMB case.

Figure 4.6. Traffic on the Aries router in a contentious IMB case is orders of magnitude greater than that of the LDMS traffic

.



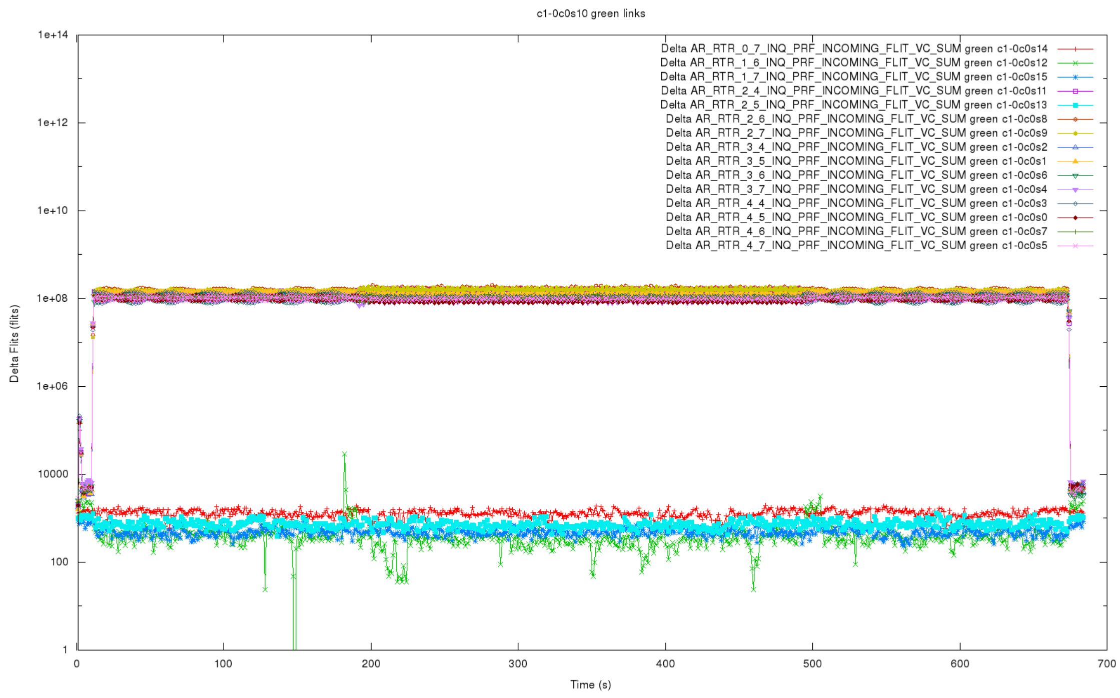
(a) STALL/FLIT ratio on the green links of an Aries router in a contentious IMB case.



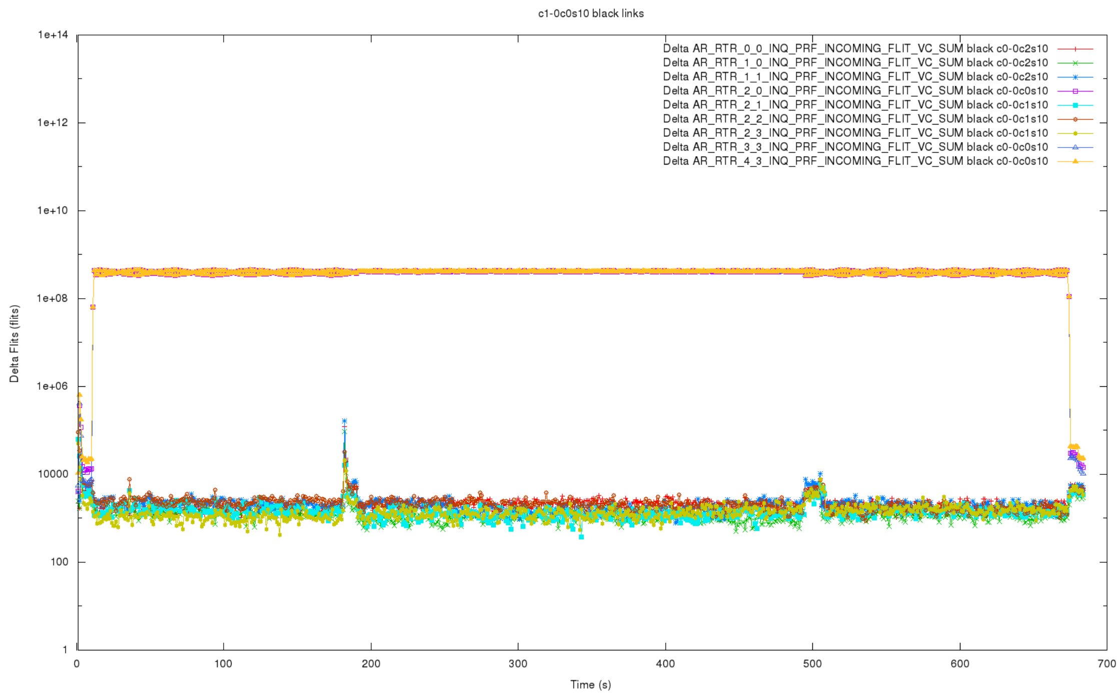
(b) STALL/FLIT ratio on the black links of an Aries router in a contentious IMB case.

Figure 4.7. STALL/FLIT ratio on an Aries router in a contentious IMB case is markedly higher than that due to LDMS traffic alone

.



(a) Traffic on the green links of an Aries router with no job allocation during a contentious IMB case.



(b) Traffic on the black links of an Aries router with no job allocation during a contentious IMB case.

Figure 4.8. Traffic occurs on an Aries router that is not part of the job allocations. Global system state is required for understanding contention.

4.1.3 IOR

Application slowdown was quantified by comparing continuous contention benchmark cases to the varied contention cases in Table 3.1. For all cases in Table 3.1, the relevant IOR runs for the comparison had file size multiplier values greater than one. This was done to increase the run time for these IOR runs while running shorter duration IOR runs for segments of the overall run time. To make a proper comparison, the benchmark cases would have to be scaled by the file size multiplier value, and the scaling would have to be verified by checking the consistency of IOR metrics. Figures 4.9 and 4.10 show the consistency of the calculated write speed for the duration of an IOR process on a specific node. The write speed remained stable throughout most of the run; therefore, it was determined that scaling would be valid. The repeatability of IOR runs was demonstrated by running each contention case three times, and the run times for these repeat runs can be seen in Table 4.1. These repeat runs were all within at least 4% of each other. With these factors, there was a high level of confidence that contention metrics would be valid and repeatable.

Contention was observed with the IOR cases described in Sec. 3.4, and a summary of the timing data for the cases can be seen in Table 4.1 . Contention was observed for both classes of cases: continuous and varied. With more than one IOR process running on the system, there was observable contention seen in every case. The level of contention increased with a greater number of simultaneous IOR processes. Cases 19 and 25 showed this with increases in average run times of around 35 and 213 seconds, respectively. Case 19 had up to four IOR processes running simultaneously while Case 25 had up to 16 IOR processes running simultaneously.

The slowdown times for the continuous contention cases were calculated by comparing each case run time to the case 1-3 average run time. The slowdown times for the varied contention cases corresponded to the total run time of the initial IOR processes that were started and ran for that entire case. This run time would be compared to the appropriate base line case with the same number of IOR processes. For example, in cases 37-39, the total runtime of the four initial IOR processes would be compared to the average scaled run time of the four IOR process baseline case. The baseline case run time was scaled by a factor of four due to the 256 GB to 64 GB file size difference.

The contention metrics for several of these cases and their application to a contention model will be discussed further in Sec. 4.2.2.

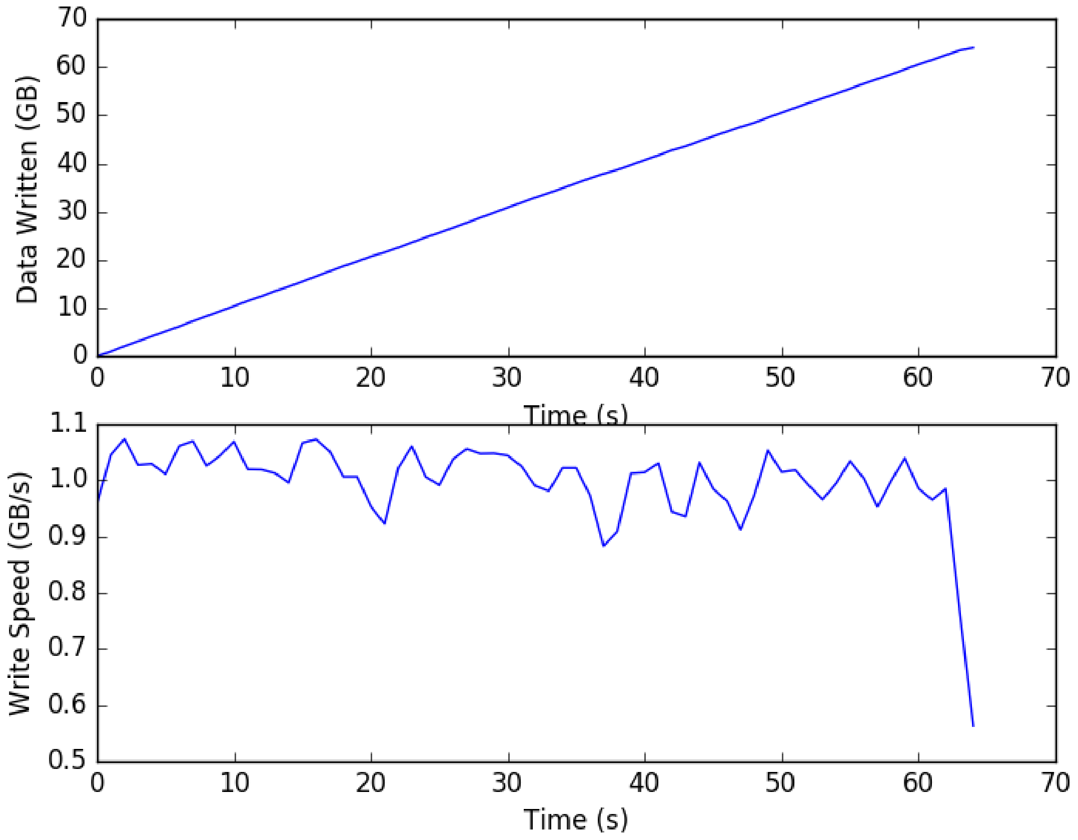


Figure 4.9. LDMS data for Case 1. The top plot shows the amount of data that is written by a specific IOR run as a function of time. The file size for Case 1 is specified as the default 64 GB. The bottom plot shows the derivative of the top plot which is the write speed of the IOR run.

Table 4.1. Timing data for all 42 contention cases that were run during the DAT. The timing data includes run times and slowdown times for a node or grouping of nodes in each case. The file size in the second column corresponds to the initial IOR processes in each case. With cases 31-33, the total run time of the initial IOR processes will be compared to the average scaled run time of cases 4-6. The average run time scaled by a factor of four due to the file size difference gives an average scaled run time of 308 seconds. Subtracting this value from the run times for cases 31-33 gives the appropriate slowdown times in the third column.

Cases	File Size (GB)	Run Times (s)	Slowdown Times (s)	Contention Schedule
1-3	64	63, 64, 65	0, 0, 0	Continuous, 1
4-6	64	76, 77, 79	12, 13, 15	Continuous, 2
7-9	64	85, 86, 87	21, 22, 23	Continuous, 3
10-12	64	94, 96, 97	30, 32, 33	Continuous, 4
13-15	64	150, 151, 151	86, 87, 87	Continuous, 8
16-18	64	276, 277, 277	212, 213, 213	Continuous, 16
19-21	256	286, 290, 295	30, 34, 39	1-4-1
22-24	256	341, 345, 352	85, 89, 96	1-8-1
25-27	256	460, 473, 473	204, 217, 217	1-16-1
28-30	256	375, 384, 387	67, 76, 79	2-8-2
31-33	256	509, 511, 512	201, 203, 204	2-16-2
34-36	256	438, 439, 442	58, 59, 62	4-8-4
37-39	256	565, 570, 572	185, 190, 192	4-16-4
40-42	512	862, 865, 882	350, 353, 370	1-4-8-16-8-4-1

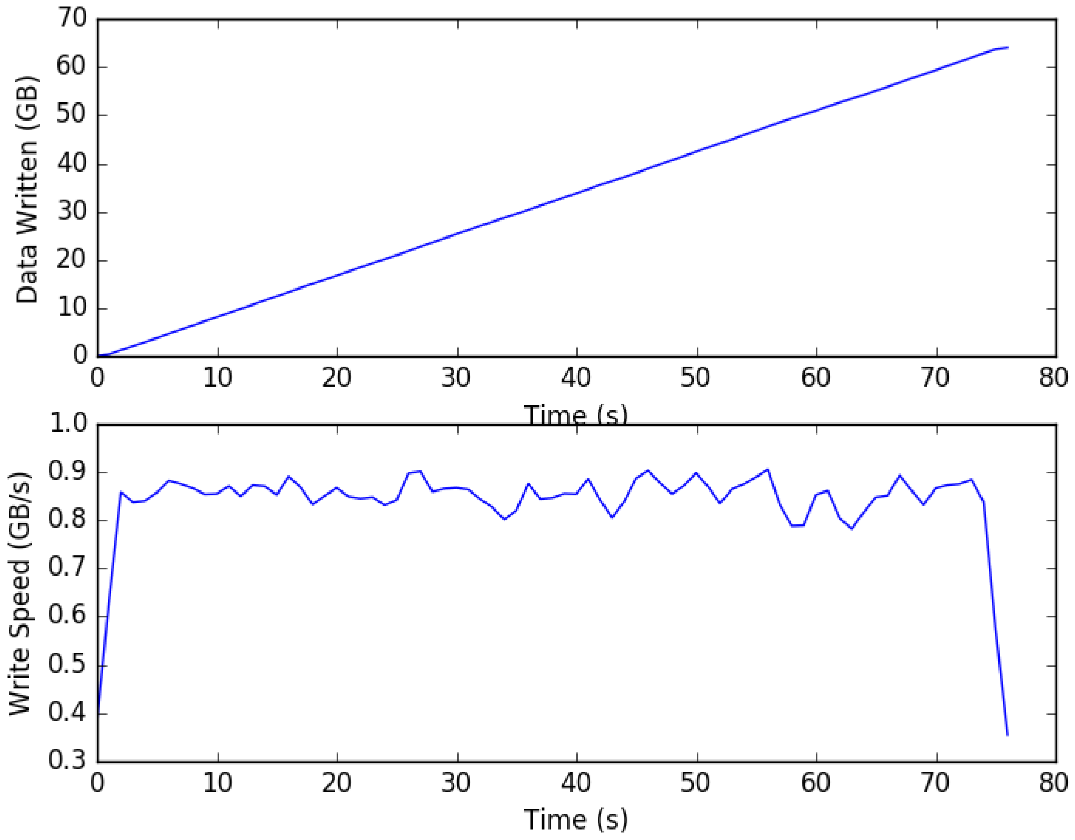


Figure 4.10. LDMS data for Case 5. The top plot shows the amount of data that is written by a specific IOR run as a function of time. The file size for Case 1 is specified as the default 64 GB. The bottom plot shows the derivative of the top plot which is the write speed of the IOR run. Note how the write speed is reduced when compared to 4.9.

4.1.4 IMB

A comparison of the baseline and contentious IMB cases described in Sec. 3.5 was conducted to quantify resource contention. See Table 4.2 for the relevant timing data. Repeatability in the IMB cases was seen with different node lists in separate non-contentious cases. The run times for each case were all within 0.7% of each other. Run times for cases 19 and 20 exhibited the least amount of contention. These were cases that used all even or odd nodes across all chassis. Run times for cases 25, 26, and 27 exhibited a moderate level of contention. These were cases that used all of the nodes in a particular chassis with staggered odd and even numbered node runs. The run time for case 34 exhibited the most contention. This case utilized all of the resources in a single cabinet. These timings show that contention was observed on the system hardware, and it is repeatable for various configurations.

Table 4.2. Base line and contention run times for IMB cases. Note that the baseline times listed are based on the even node baseline cases. Odd cases could be used as baseline values if desired.

Cases	Baseline Times (s)	Cont. Time (s)	% Difference
25, 28, 31	411.21	684.32, 685.51, 686.2	66.42, 66.71, 66.87
26, 29, 32	415.44	703.43, 702.01, 700.26	69.32, 68.98, 68.56
27, 30, 33	329.70	580.37, 578.19, 595.22	76.03, 75.37, 80.53
34, 35, 36	463.42	797.76, 795.57, 793.61	72.15, 71.67, 71.25

4.1.5 CTH

For the CTH contention cases described in Sec. 3.6, the total run times are summarized in Table 4.3. The averaged baseline times are within 0.04% of each other, demonstrating repeatability with the different nodes lists for non-contentious runs. With contention on the system, the run times increase by 7.8% and 1.5% for cases *a* and *b*, respectively. This shows that contention had a measurable effect, and case *a* exhibited significantly higher effects from contention than case *b*. The LDMS counters for both of these cases and their baselines will be examined in Sec. 4.2.3.

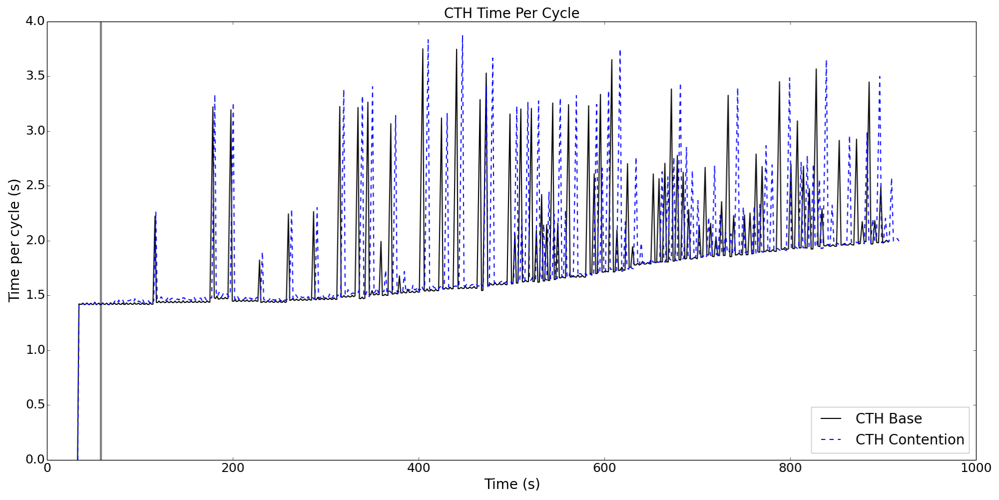
Figure 4.11 shows the time per computational cycle compared to the baseline cases. At 58 seconds the vertical line designates the start time of the IMB cases. The first approximately 31 seconds of each case is spent initializing the computational blocks, therefore the cycle time is 0 during this period. The spikes in cycle times which occur periodically throughout the simulations correspond to AMR load balances which transfer computational blocks between processors in order to balance the load. The real slow down observed is simply defined as

$$\mathrm{S}_n = \sum_{i=1}^{n} \Delta t_{cyc,n} \tag{4.1.1}$$

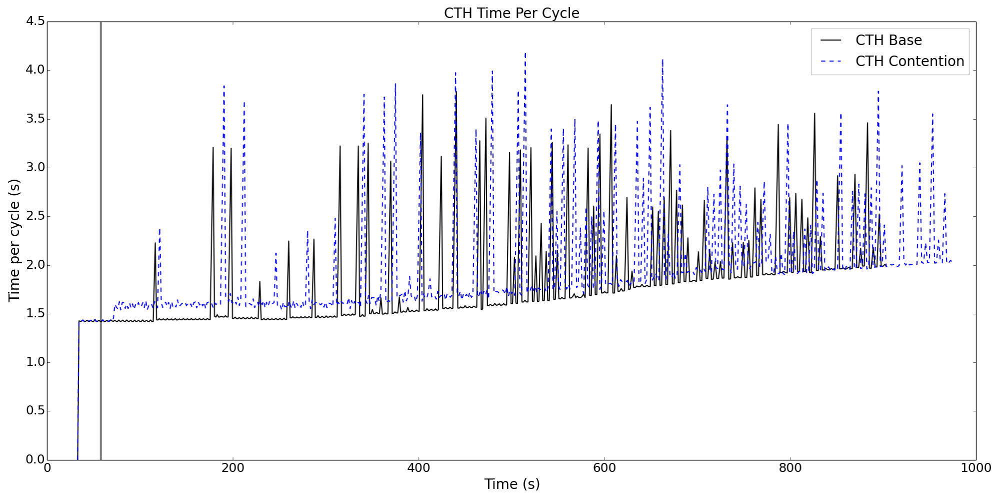
where subscript n is the cycle, and Δt_{cyc} is the change in computational time for that cycle.

Table 4.3. Base line and contention run times for CTH cases.

Case	Base Line Time (s)	Cont. Time (s)	% Difference
a	903.698	974.485	7.83304
b	903.357	917	1.51025



(a) Time per cycle for case *a* and baseline.



(b) Time per cycle for case *b* and baseline.

Figure 4.11. Time per cycle for contentious and baseline CTH cases.

4.2 Model Results

This section will define the current Actionable Metric Model, AMM, in Section 4.2.1 and discuss how well it characterizes the observations on a subset of the data from the DAT on Mutrino with IOR (Section 4.2.2) and CTH (Section 4.2.3). Further discussion of AMM occurs within Chapter 5.

4.2.1 Algorithm Description

AMM leverages common statistical operators and methodologies; please refer to Appendix A for a brief introduction to and definition of these quantities.

HPC platforms produce a plethora of metrics (e.g., hardware counters) and the optimal ones to detect and quantify contention can vary across platforms. On Mutrino, the metric `client.lstats.write_bytes#llite.snx11137`, referred hereafter as `lustre_write`, was utilized for analyzing Lustre file system contention. This metric is a monotonically increasing counter on each node that provides a measure of the number of bytes that node has written to the Lustre file system. The metrics `AR_RTR_r_c_STALL_FLIT_RATIO` and `AR_RTR_r_c_INQ_PRF_INCOMING_FLIT_VC_SUM` were utilized for analyzing and quantifying Aries Interconnect contention. All of these counters are described in more detail within Section 3.3.

LDMS provides metric data, $m_i(n)$, at a specified frequency, f, where i is the spatial component of the metric (e.g., which node `lustre_write` is reporting) and n is the reference time, t. On Mutrino, $f = 1$ Hz however this parameter could vary across platforms.

For each metric, the various statistical quantities are computed over 2 different windows, both of size ω. The first window, Ω_A, encompasses all of the most recent ω data points, i.e.,

$$\Omega_\text{A} = \{t \mid n - \omega < t \leq n\} \tag{4.2.1}$$

The second window, Ω_NC, encompasses the most recent ω data points that are determined to be non-contentious, i.e.,

$$\Omega_\text{NC} = \{t_\text{NC} \mid n_\text{NC} - \omega < t_\text{NC} \leq n_\text{NC}\} \tag{4.2.2}$$

The primary three quantities computed are the geometric mean of m_i within Ω_A, the geometric mean of m_i within Ω_NC, and the geometric standard deviation of m_i within Ω_NC, i.e.,

$$\mu_{g,i}\left(t \mid t \in \Omega_\text{A}\right) = \mu_{g\text{A},i} \tag{4.2.3}$$

$$\mu_{g,i}\left(t \mid t \in \Omega_\text{NC}\right) = \mu_{g\text{NC},i} \tag{4.2.4}$$

$$\sigma_{g,i}\left(t \mid t \in \Omega_\text{NC}\right) = \sigma_{g\text{NC},i} \tag{4.2.5}$$

Contention, for the purposes of this study, occurs when a reference application experiences slowdown due to other applications using the same shared resource(s). Therefore, if there is only a single application using a shared resource, then that resource is not experiencing contention. However, if there are two or more applications using the same shared resource, then it is possible contention is occurring.

If m_i increases during contention, then its windowed geometric mean is denoted as $\mu^{+}_{g,i}$, whereas if m_i decreases during contention, then its windowed geometric mean is denoted as $\mu^{-}_{g,i}$. Contention is determined to occur if the following criteria are met.

$$\mu^{+}_{g\mathrm{A},i}\left(n\right) > \delta^{+}_{g\mathrm{NC}}\left(n-1\right) \tag{4.2.6a}$$

$$\mu^{-}_{g\mathrm{A},i}\left(n\right) > \delta^{-}_{g\mathrm{NC}}\left(n-1\right) \tag{4.2.6b}$$

Please refer to Equation Set A.0.3 within Appendix A for more information regarding how these factors are computed.

If contention occurs, then the non-contentious confidence intervals are held fixed, i.e.,

$$\delta_{g\mathrm{NC}}\left(n\right) = \delta_{g\mathrm{NC}}\left(n-1\right) \tag{4.2.7}$$

However, if contention is not present, then these non-contentious confidence intervals are recomputed with the new time step's data.

A simple slowdown model, ζ_i, is to compare the windowed geometric mean of the metric with its non-contentious counterpart when contention occurs, i.e.,

$$\zeta^{+}_{i}\left(n\right) = \zeta^{+}_{i}\left(n-1\right) + \left(\frac{\mu^{+}_{g\mathrm{NC},i}\left(n\right)}{\mu^{+}_{g\mathrm{A},i}\left(n\right)}\right) \times \left(n-\left(n-1\right)\right) \tag{4.2.8a}$$

$$\zeta^{-}_{i}\left(n\right) = \zeta^{-}_{i}\left(n-1\right) + \left(1-\frac{\mu^{-}_{g\mathrm{A},i}\left(n\right)}{\mu^{-}_{g\mathrm{NC},i}\left(n\right)}\right) \times \left(n-\left(n-1\right)\right) \tag{4.2.8b}$$

This slowdown model, if the metric is a time rate, is the number of seconds that the application is estimated to have been impacted.

4.2.2 IOR

The LDMS metric focused upon for IOR was the node-based `lustre_write`. This metric is monotonically increasing and difficult to directly apply the AMM to. However, the time rate of change of `lustre_write` provides a performance-based metric that is ideal for AMM. This rate was computed with the backward difference formulation, i.e.,

$$\mathrm{Rate} = \frac{\mathrm{d}}{\mathrm{d}t}\big(m_i(t)\big) \approx \frac{\nabla\left[m_i\right]\left(t\right)}{t_n - t_{n-1}} = \frac{m_i\left(t_n\right) - m_i\left(t_{n-1}\right)}{t_n - t_{n-1}} \tag{4.2.9}$$

AMM was applied to the longest-running IOR process for cases 19, 22, 25, and 40 utilizing the `lustre_write` rate metric. Table 4.4 provides a comparison between the actual and modeled slowdown and contentious times. Figures 4.12-4.15 provide the same comparison along with some of the statistical quantities AMM used for its decision making.

Table 4.4. This table compares the actual and modeled slowdown and contentious times for IOR cases 19, 22, 25, and 40; the model used a windowing size, ω, of 8 and a single geometric standard deviation offset, η, for its decision-making strategy.

Case No.	Actual Slowdown (s)	Model Slowdown (s)	% Diff.	Actual Time in Contention (s)	Model Time in Contention (s)	% Diff.
19	39.0	28.6	26.8	94.0	94.0	0
22	85.0	87.8	3.3	150.0	150.0	0
25	217.0	213.0	1.8	274.0	274.0	0
40	353.0	349.5	1	629.0	628.0	0.2

For each of these cases, the number of simultaneous IOR processes changes according to the following list.

Case 19: $1 \rightarrow 4 \rightarrow 1$

Case 22: $1 \rightarrow 8 \rightarrow 1$

Case 25: $1 \rightarrow 16 \rightarrow 1$

Case 40: $1 \rightarrow 4 \rightarrow 8 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 1$

These changes are observed in the bottom sub-figure within Figures 4.12-4.15. Table 4.1 shows that more than a single simultaneous IOR instance will create contention. Therefore, contention with IOR for this experiment is present whenever more than a single IOR is operating at a time; this trend is also observed within the bottom sub-figure. AMM's contention time is also within this sub-figure and for 3 out of the 4 cases it is directly on top of the actual trend except for Case 40 where it was delayed by a single time step, which suggests that a window size, ω, of 8 and the number of standard deviations for decision making, η, of 1 are sufficient.

During this contentious time, the non-contentious windowed geometric mean and standard deviation parameters, $\mu_{g\mathrm{NC}}^{-}$ and $\delta_{g\mathrm{NC}}^{-}$ respectively, remain fixed; this trend is observed within the middle sub-figure within Figures 4.12-4.15. It is also observed that $\mu_{g\mathrm{A}}$ continues to adequately track the performance of the `lustre_write` rate, which suggests that a window size, ω, of 8 is sufficient.

During contention, the slowdown model in the top sub-figure within Figures 4.12-4.15 appears to increase linearly for cases 19, 22, and 25. This is expected since the contentious period for these cases involves the IOR processes constantly writing out the same type of data which is observed with the nearly constant `lustre_write` rate in the middle sub-figure and the linearly increasing `lustre_write` counter in the top sub-figure. Case 40 also linearly changes however since its schedule of IOR processes is more complicated than the other cases, its `lustre_write` counter looks like a smoothed version of case 25 and its slowdown model also exhibits a similar trend. The slowdown model for cases 22, 25, and 40 approach the actual slowdown quite well however case 19's slowdown model falls short.

For these cases, AMM shows excellent correlation between the actual and modeled time in contention. There are slightly higher differences between the actual and modeled slowdown times, though. For IOR, these differences are attributed to the sensitivity of the model to the degree of contention; case 19 had the largest difference and the lowest amount of contention. Tweaks to AMM parameters including ω and η can impact this accuracy and are further discussed within Chapter 5.

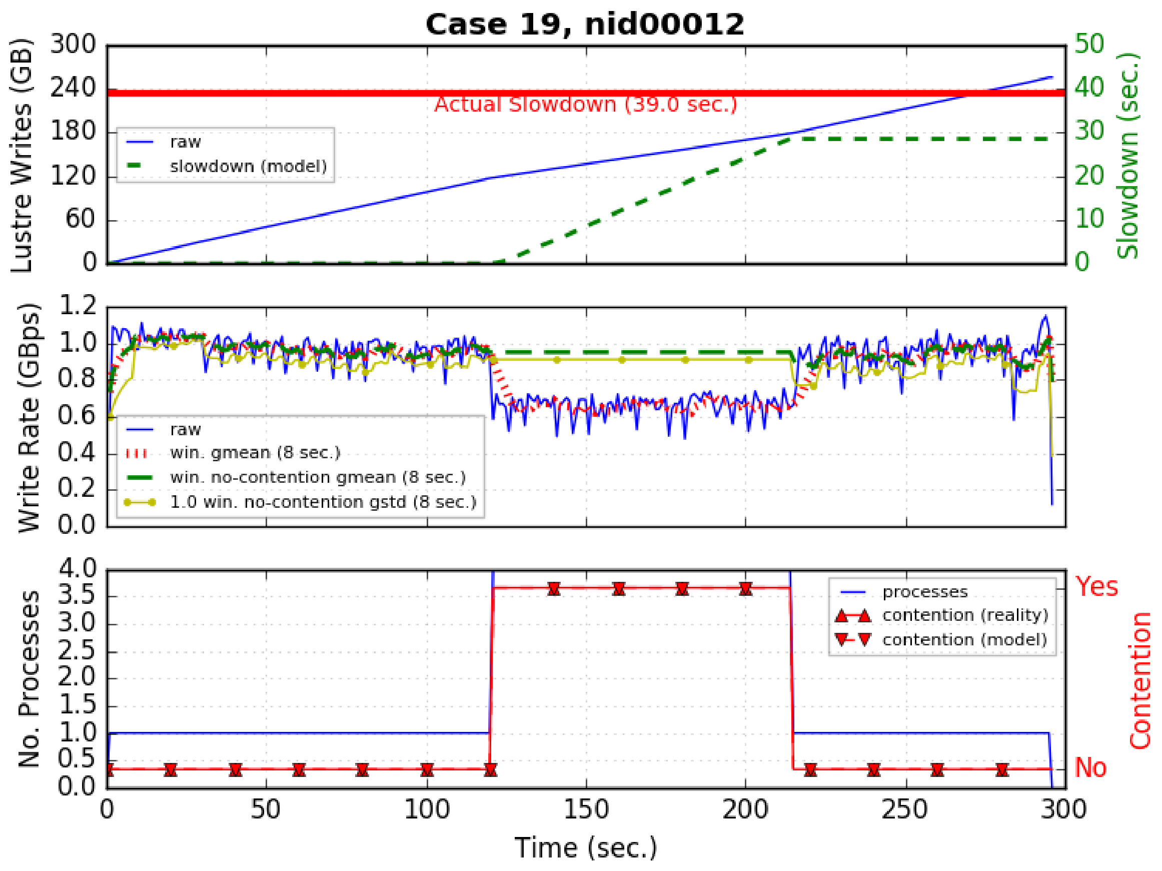


Figure 4.12. This figure contains 3 sub-figures that compare the actual performance and contention observed with their model counterparts for IOR Case 19 observed from node 12 (nid00012). The top sub-figure plots the "raw" `lustre_write` data from LDMS on the primary y-axis along with the actual and modeled slowdown on the secondary y-axis. The middle sub-figure plots the "raw" `lustre_write` rate along with this rate's 8 sec. windowed geometric mean, μ_g, the 8 sec. windowed geometric mean of the non-contentious portion of this rate, and the lower confidence interval from a single (η) 8 sec. windowed geometric standard deviation, σ_g^-. The bottom sub-figure plots the number of simultaneous IOR processes for this case along the primary y-axis and the actual and modeled contention characteristic along the secondary y-axis.

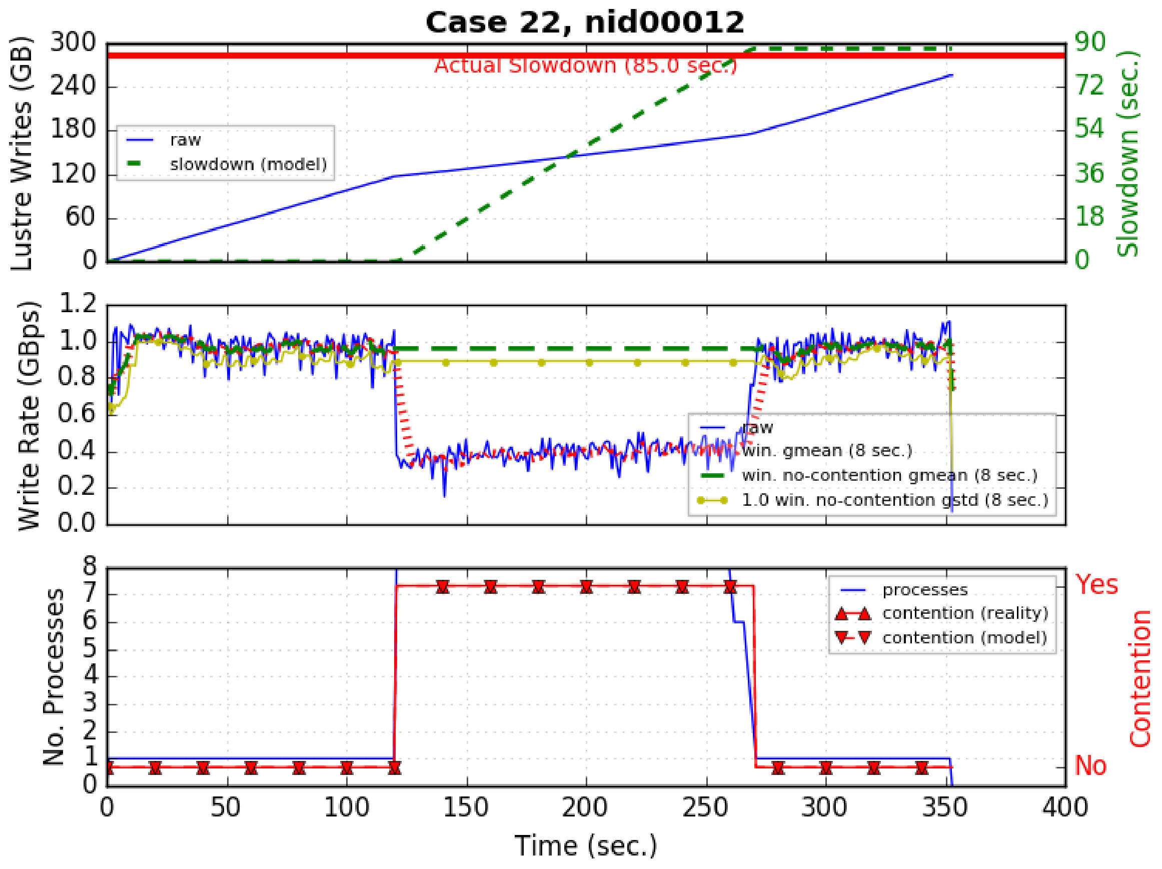


Figure 4.13. This figure contains 3 sub-figures that compare the actual performance and contention observed with their model counterparts for IOR Case 22 observed from node 12 (nid00012). The top sub-figure plots the "raw" `lustre_write` data from LDMS on the primary y-axis along with the actual and modeled slowdown on the secondary y-axis. The middle sub-figure plots the "raw" `lustre_write` rate along with this rate's 8 sec. windowed geometric mean, μ_g, the 8 sec. windowed geometric mean of the non-contentious portion of this rate, and the lower confidence interval from a single (η) 8 sec. windowed geometric standard deviation, σ_g^-. The bottom sub-figure plots the number of simultaneous IOR processes for this case along the primary y-axis and the actual and modeled contention characteristic along the secondary y-axis.

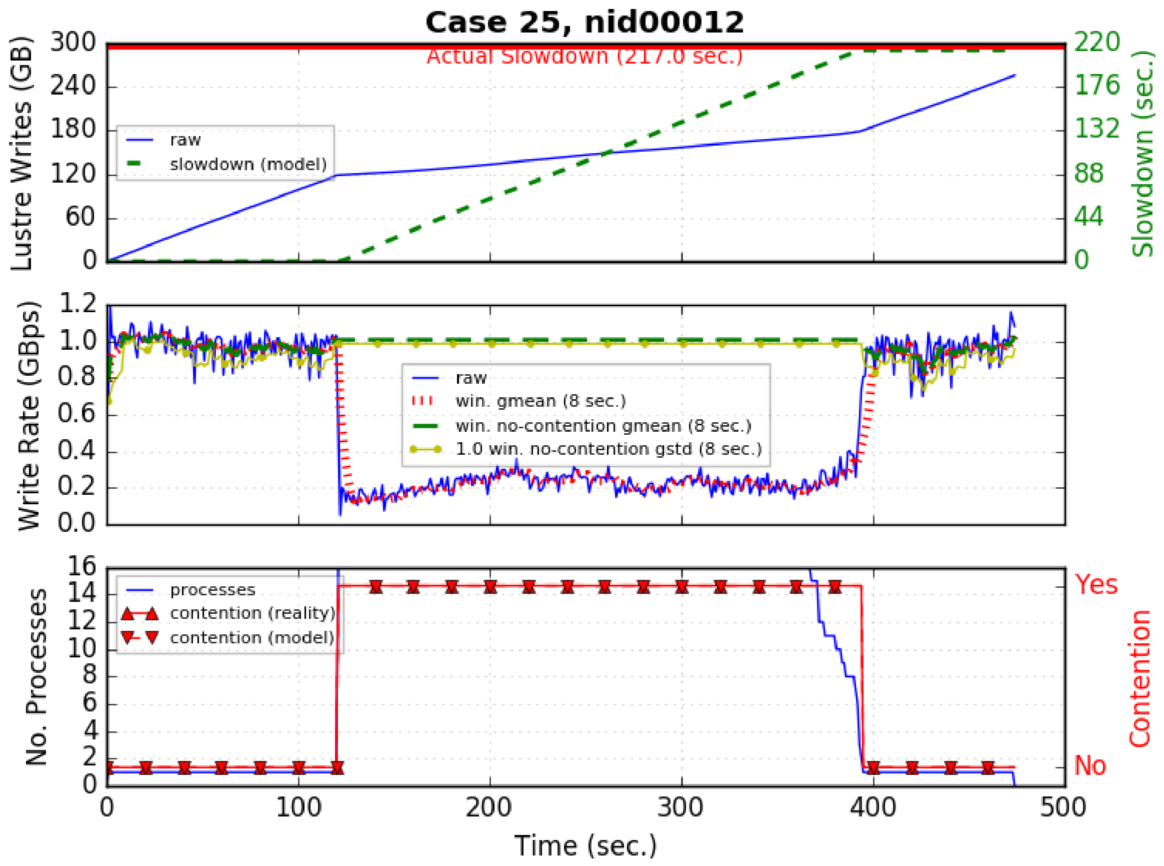


Figure 4.14. This figure contains 3 sub-figures that compare the actual performance and contention observed with their model counterparts for IOR Case 25 observed from node 12 (nid00012). The top sub-figure plots the "raw" `lustre_write` data from LDMS on the primary y-axis along with the actual and modeled slowdown on the secondary y-axis. The middle sub-figure plots the "raw" `lustre_write` rate along with this rate's 8 sec. windowed geometric mean, μ_g, the 8 sec. windowed geometric mean of the non-contentious portion of this rate, and the lower confidence interval from a single (η) 8 sec. windowed geometric standard deviation, σ_g^-. The bottom sub-figure plots the number of simultaneous IOR processes for this case along the primary y-axis and the actual and modeled contention characteristic along the secondary y-axis.

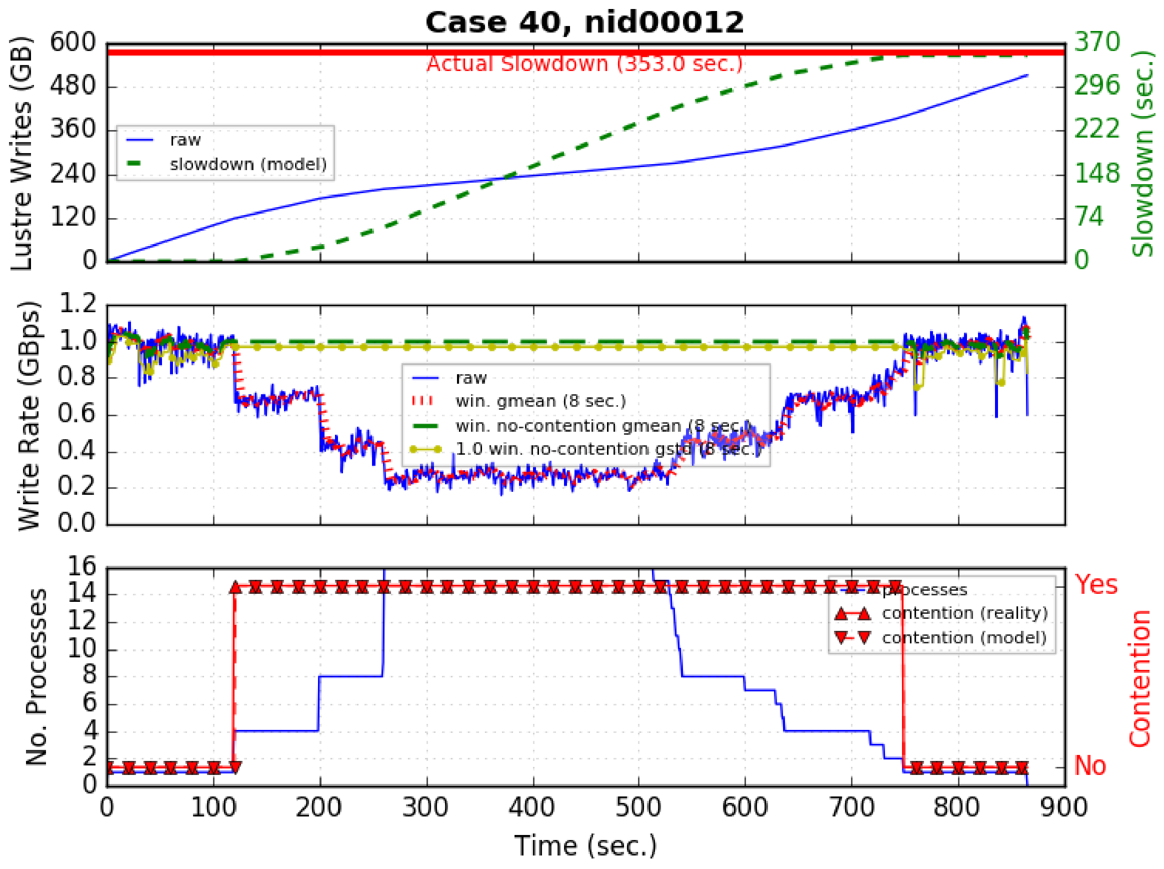


Figure 4.15. This figure contains 3 sub-figures that compare the actual performance and contention observed with their model counterparts for IOR Case 40 observed from node 12 (nid00012). The top sub-figure plots the "raw" `lustre_write` data from LDMS on the primary y-axis along with the actual and modeled slowdown on the secondary y-axis. The middle sub-figure plots the "raw" `lustre_write` rate along with this rate's 8 sec. windowed geometric mean, μ_g, the 8 sec. windowed geometric mean of the non-contentious portion of this rate, and the lower confidence interval from a single (η) 8 sec. windowed geometric standard deviation, σ_g^-. The bottom sub-figure plots the number of simultaneous IOR processes for this case along the primary y-axis and the actual and modeled contention characteristic along the secondary y-axis.

4.2.3 CTH

The LDMS interconnect performance counters for the contentious CTH cases and their baselines were examined and the contention model was applied to them on a per Aries router basis. Specifically the AR_RTR_r_c_STALL_FLIT_RATIO and AR_RTR_r_c_INQ_PRF_INCOMING_FLIT_VC_SUM counters described in Section 3.3 were used. These counters will be referred to as STALL_FLIT_RATIO and INCOMING_FLIT_VC_SUM for the remainder of this document. While additional counters might be able to provide supplementary information to the models, they were not considered for the current study.

For the AMM, the STALL_FLIT_RATIO counter was found to give the most reliable indication of contention on a given Aries router. The geometric mean for each performance counter over both the black and green connections used for each Aries router associated with the CTH cases were computed for each LDMS sample time. Connections which were not used for that case on each Aries router were not included in the statistics. For the contention cases, other jobs could share the same Aries router if all the nodes for that Aries router were not utilized in the node list for the case. Additionally a rolling windowed geometric mean was used to smooth the mean data with a window size of 10 seconds ($\omega = 10$). The upper confidence interval at each sample time (δ_g^+) was also computed using Eq. A.0.3a with $\eta = 1$ (*i.e.*, 1 standard deviation). These statistics were then used in the contention model as described in Section 4.2.1.

A slowdown model based on the approach described in Section 4.2.1 was attempted for the CTH cases, but a simple model which captured all the observed behavior could not be attained in the time frame of this project. Possible reasons for the deficiency and extensions to it are discussed in Chapter 5.

Figure 4.16 shows the INCOMING_FLIT_VC_SUM and STALL_FLIT_RATIO on the c0-0c1s8 Aries router, along with their statistical quantities for each LDMS sample time for the case a baseline. The results of the contention model described above are also shown. The INCOMING_FLIT_VC_SUM counter is divided by 1E7 for all cases to avoid floating point overflows at larger rolling window sizes. During the first approximately 30 seconds, minimal traffic is shown on the INCOMING_FLIT_VC_SUM metric as expected during the initialization phase of CTH, and the STALL_FLIT_RATIO oscillates between 0 and 1 indicating little to no contention. Once the computational cycles start a sharp jump is seen in the INCOMING_FLIT_VC_SUM, and the MPI traffic stays approximately constant for the rest of the simulation except for occasional dips which correspond to the AMR load balances. The STALL_FLIT_RATIO also shows spikes for some sample times when the code load balances since large messages ($\mathscr{O}(100 - 200\text{mb})$) can potentially be exchanged during this operation in addition to the normal per cycle messages. The contention model picks up on these spikes as signaling a contentious state since the upper confidence interval is relatively close to the windowed mean for the STALL_FLIT_RATIO. The model usually drops to non-contentious at the next sample after the spike.

The counters for the contentious run of case a are shown in Fig. 4.17 for Aries router c0-0c1s5. The vertical line at $t = 58s$ designates the start of the IMB cases on the system.

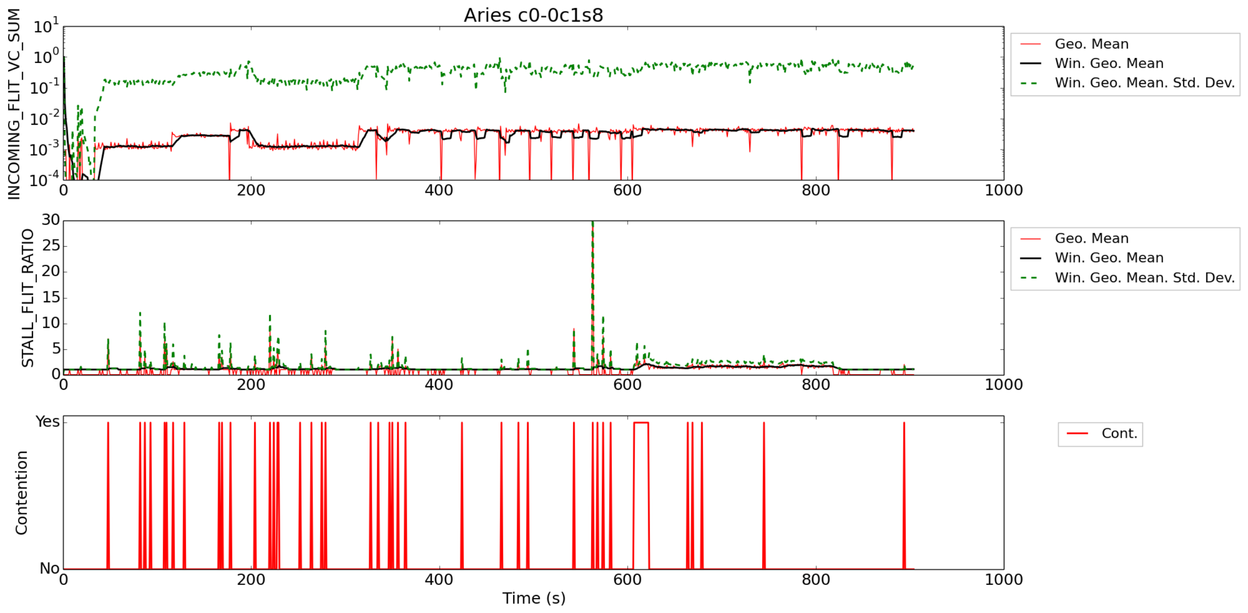


Figure 4.16. Mean LDMS counters and contention model on Aries router `c0-0c1s8` for CTH case *a* baseline.

Comparing to the baseline, both the `INCOMING_FLIT_VC_SUM` and `STALL_FLIT_RATIO` are significantly higher once IMB starts, indicating increased traffic and a contentious state for the Aries router. For this Aries router, it is shared between CTH and IMB jobs. For all the contentious results examined, there is an initial spike seen in both metrics at the time IMB is supposed to start but it drops back to the baseline for 10-11 seconds before sharply increasing for the rest of the case. The cause of this has not been identified yet. The contention model picks up the small initial spike, the drop down to the non-contentious baseline, and the subsequent contentious behavior at the correct times. The model then predicts contention for the rest of the simulation.

At approximately 780 seconds the `INCOMING_FLIT_VC_SUM` decrease by almost an order of magnitude while the `STALL_FLIT_RATIO` almost doubles. This change in behavior is reflected in the slow down by a change in slope which indicates that the case showed less of a slow down relative to the baseline past this point. The CTH log file was examined for anything which would have caused this, but no obvious cause was found. Increased `STALL_FLIT_RATIO` reflects a reduced ability to move traffic across an Aries router from an ingress to egress port due to buffer occupancy at the egress port. Since we are not able to look at buffer occupancy at egress ports or know the destination of the traffic that is stalled at the ingress port it is unclear whether this change was due to adaptive routing changes that increased egress port buffer occupancy or backpressure from a destination host. We will be working with the vendor to identify performance counters that might provide more insight into this situation in future work. The change in slow down shows that this change in interconnect behavior captured by the LDMS metrics had a real and measurable effect on the programs run time, which suggests it could be possible to come up with a slow down model which reflects this for CTH.

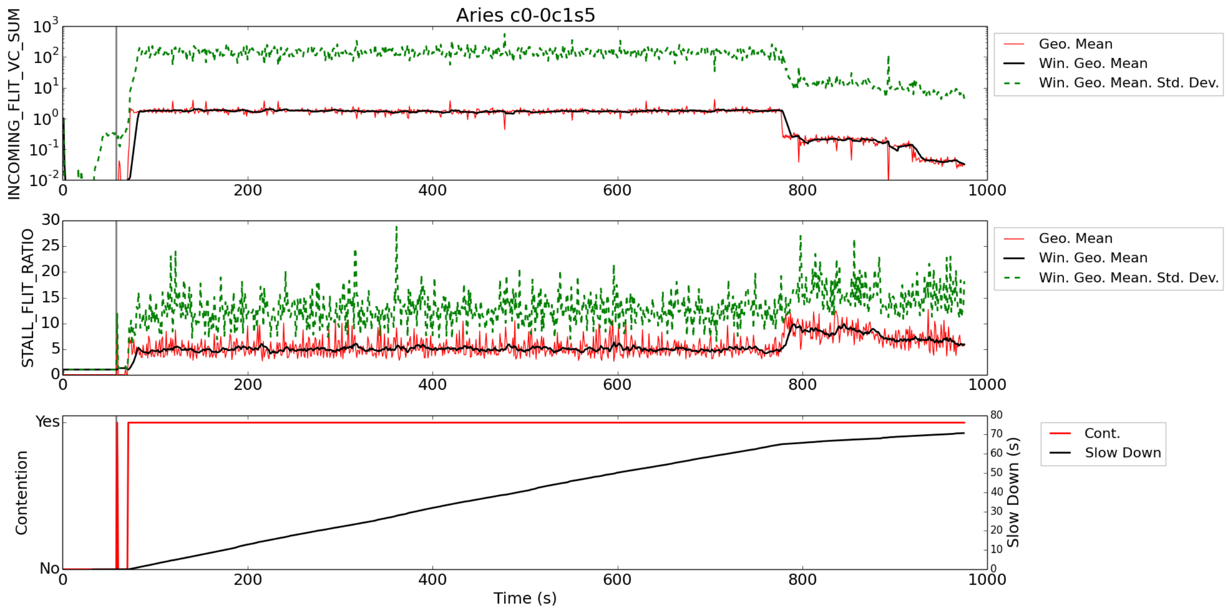


Figure 4.17. Mean LDMS counters and contention model on Aries router `c0-0c1s5` for CTH case a contention.

For the same contentions case a, Fig. 4.18 shows the counter behavior for Aries router`c0-0c2s8`. In contrast to the results in Fig. 4.17, this Aries router shows significantly less contention due to it not being shared with as many jobs. The `STALL_FLIT_RATIO` stays below 5 for the entire simulation and the `INCOMING_FLIT_VC_SUM` shows similar trends to the previous Aries router. The contention model predicts spikes of contention for a few of the sample times, but it returns to a non-contention state a few samples later in all cases.

For case b, the contention had much less of an effect as shown in Table 4.3. Figure 4.19 shows the interconnect counters for the `c0-0c1s4` Aries router for the baseline case. The behavior is similar to the baseline for case a. Figure 4.20 shows the counters for the same case with contention. In this case the contention model predicts a contentious state before IMB starts, but correctly predicts contention for the rest of the simulation. The slow down relative to the baseline is substantially lower than for case a, and seems to be leveling out towards the end of the simulation. The reason for the differences in the effect of contention between case a and b are not clear, but could be related to which Aries routers are being used by each case and if they are shared with other jobs.

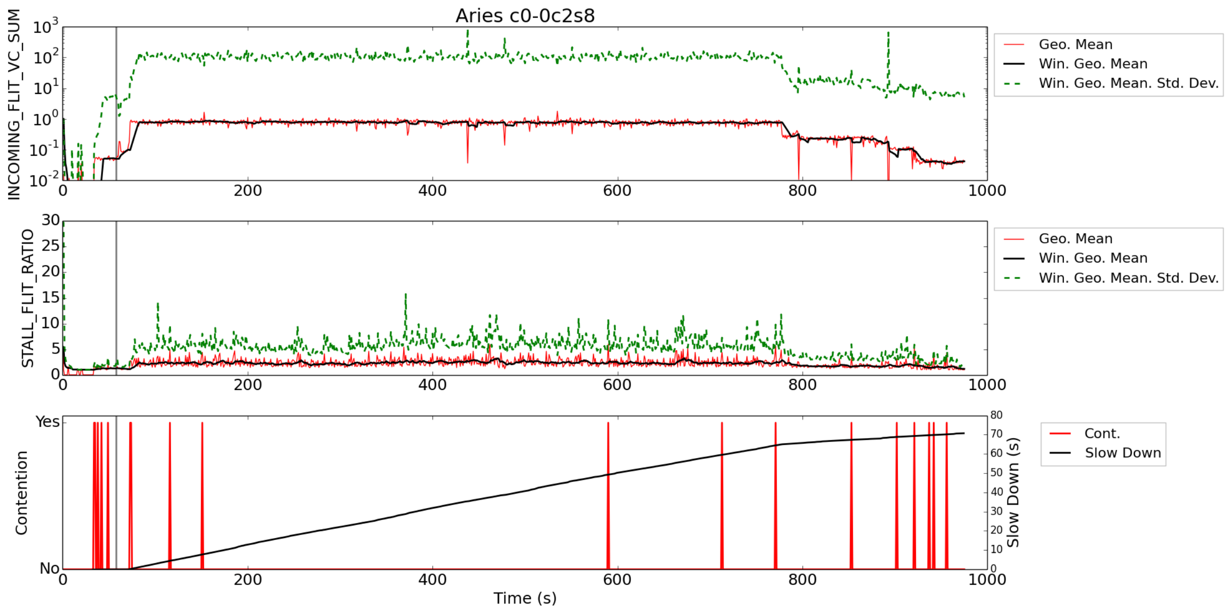


Figure 4.18. Mean LDMS counters and contention model on Aries router `c0-0c2s8` for CTH case *a* contention.

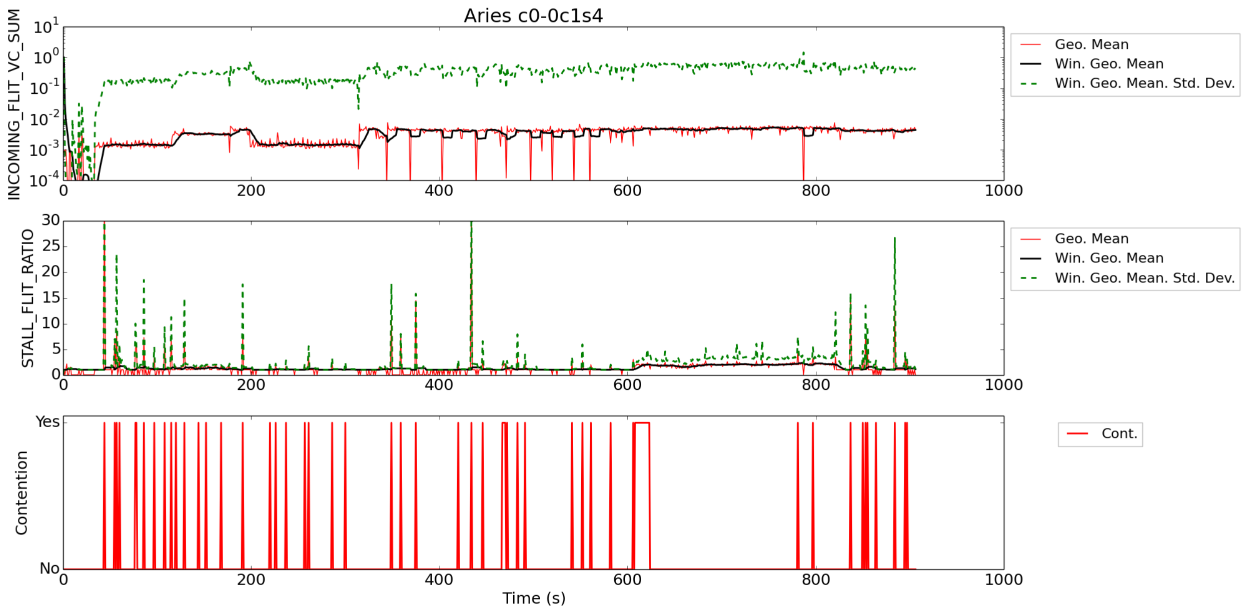


Figure 4.19. Mean LDMS counters and contention model on Aries router `c0-0c1s4` for CTH case *b* baseline.

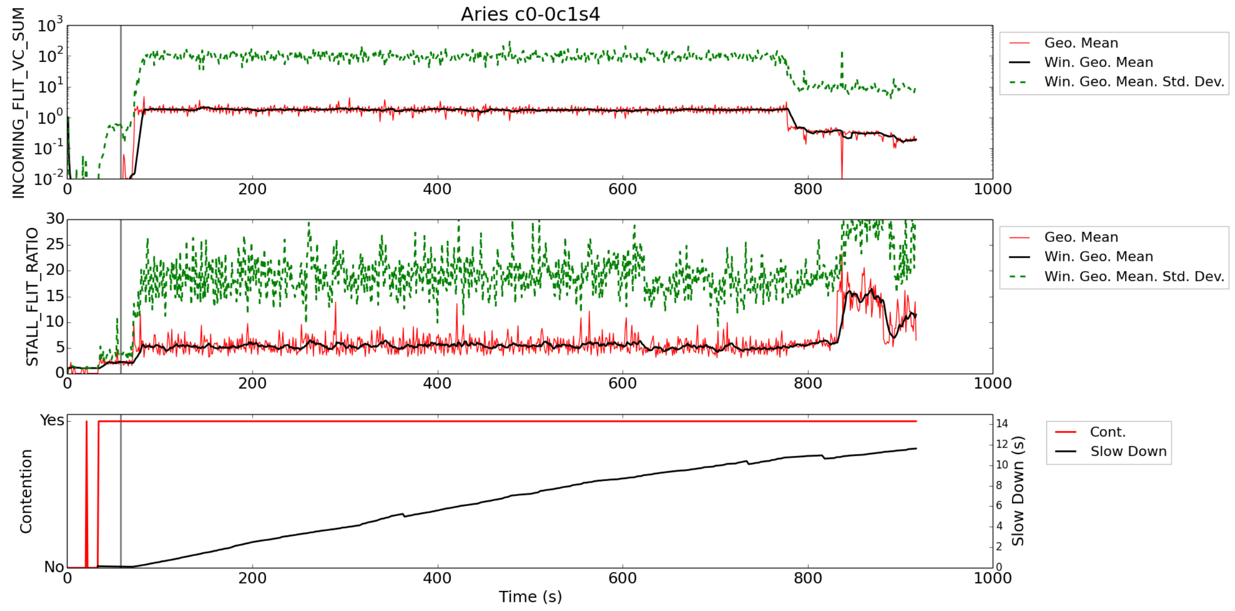


Figure 4.20. Mean LDMS counters and contention model on Aries router `c0-0c1s4` for CTH case *b* contention.

Chapter 5

Discussion

This section will discuss the limitations encountered for the contention and slowdown model discussed in Section 4.2.1 applied to the IOR and CTH cases, along with possible extensions to the models for further development. In addition to these extensions, larger systems involving multiple cabinet pairs (i.e., *groups*) should be investigated since it might be easier to characterize contention over these links.

5.1 Current Model Limitations

The contention and slowdown models applied to the single-node IOR cases correctly identified the time spent in contention and provided slowdown correlation within 30%. The error in slowdown estimation increases as the degree of contention decreases, thus implying a growing sensitivity to the windowing function parameters employed.

The contention model applied to the CTH cases was generally able to correctly predict contention on Aries router which shared nodes with the IMB cases. However for Aries routers which were not shared with IMB cases, and the baseline cases, the model could be overly sensitive to noise in the `STALL_FLIT_RATIO` since the upper confidence interval was relatively close to the mean. This could be remedied by incorporating a cut off value into the contention model if the behavior of the metric in a non-contentious state is known. A larger window size or higher confidence interval might also alleviate the issue.

5.2 Slowdown Model for Metrics with a Non-Linear Performance Impact

While the slowdown model applied to the IOR cases was able to accurately calculate the amount of slowdown experienced by the application due to contention, the same model applied to CTH did not capture the observed behavior. The slowdown model was applied to both the `STALL_FLIT_RATIO` and `INCOMING_FLIT_VC_SUM` counters without success. Since CTH only spends a fraction of it's time in MPI and the rest in computation/IO, a slowdown in interconnect throughput due to contention will not translate into an equal relative slowdown in overall run time. Similarly since the code makes use of asynchronous communications where possible, some of the increased latency due to contention can be absorbed due to the overlap with computation.

For a general application, the relationship between the actual slowdown and measure of degradation for a given metric (e.g., ratio of windowed average to non-contentious average) cannot be known a-priori. It is expected to depend on the degree to which asynchronous execution is used around the operations the given metrics impact, and could also depend on the problem and scale the application is being run at. As a first order extension to the slowdown model, a multiplicative factor as a function of time, $\beta(n)$, could be introduced,

$$\zeta_i^+(n) = \zeta_i^+(n-1) + \beta(n)\left(\frac{\mu^+_{g\mathrm{NC},i}(n)}{\mu^+_{g\mathrm{A},i}(n)}\right) \times (n-(n-1)) \tag{5.2.1a}$$

$$\zeta_i^-(n) = \zeta_i^-(n-1) + \beta(n)\left(1-\frac{\mu^-_{g\mathrm{A},i}(n)}{\mu^-_{g\mathrm{NC},i}(n)}\right) \times (n-(n-1)) \tag{5.2.1b}$$

The $\beta(n)$ factor would represent an estimate of how much the degradation of the metric of interest actually impacts the slowdown of the application. This function will most likely have to be evolved over the run-time of the simulation, or evaluated at the conclusion of the job to assess the impact on the performance from contention.

5.3 Initial Conditions

One deficiency for the contention model is determining an initial non-contentious state. All the IOR and CTH cases considered started in a non-contentious environment and subsequently various degrees of contention were introduced on the system. If the job is started in an environment with contention influencing it from the beginning, the current model would not detect contention unless the contention increased further to trigger it. Additionally the non-contentious mean state for the metric of interest would not be an estimate of the truly non-contentious state, and this would feedback into the slowdown model and cause it to underestimate the applications slow down.

If the behavior of a counter is known a-priori on the system (e.g., `STALL_FLIT_RATIO` counters below 1-2 generally indicate a non-contentious state for that Aries router, or an `lustre_write` above a large fraction of the maximum expected) then the contention model could incorporate this to detect starting in a contentious environment. If the model detects that it started in a contentious state it could then wait until the metrics drop to a non-contentious level and obtain the estimate of the non-contentious values at that point. Using the non-contentious values, the slowdown model could then be utilized at the conclusion of the job to estimate the overall impact if the LDMS data for the duration of the job is available. If the job never encounters a non-contentious state, default estimates of the metrics for no contention could be used to estimate the slowdown model.

5.4 Multi-resource Rollup

All of the AMM results herein have exhibited AMM's adherence to a single metric. HPC platforms are designed to run applications with intra- and inter-nodal paralellism. Accordingly, a single, multi-node simulation will utilize many different spatial components of the same metric, i.e., the i in m_i. This multi-dimensional aspect needs to be considered for both the contention and slowdown time models.

There are several strategies that may be investigated for addressing the multi-dimensional metric data as follow-on, future work; some of them are enumerated below.

1. Dimensionally reduce the metric data to a single quantity and then to apply AMM to this quantity.
2. Apply AMM to each individual m_i and then to dimensionally reduce the model's output. This could take the form of performing simple statistical operations on the data at each time step (e.g., the maximum).

Chapter 6

Anticipated Impact

The ultimate goal of the work presented in this paper is to maximize the Science throughput for both current and future HPC systems by enabling balanced platform resource utilization while minimizing congestion due to contention for shared resources. The impact of realizing this goal would be more Science per dollar spent on compute platforms.

The results of the exploratory work performed for this LDRD project and presented in this paper are validation that: 1) whole system monitoring, even at one second intervals, can provide valuable information that can be used to provide actionable insight into resource contention related application performance degradation, 2) while currently employed mechanisms for load spreading with respect to network traffic have an effect, they do not mitigate congestion related application performance degradation, and 3) the analytic approaches and techniques applied in this study can provide valid and time effective detection and prediction of an applications response to contention for resources it requires,

The target of this work is the Cray XC40 supercomputing platform because it is the platform currently being used for large scale simulation work at both SNL and LANL. Applications run on the Trinity system can experience significant slowdowns (50% and greater observed). While this performance variation appears to be due to congestion related phenomena, the tools do not yet exist to identify sources or to quantify impact other than observed run to run variation. The direct impact of this work is that we have identified metrics and analytic methods for using those metrics to identify when contention related performance degradation begins and to quantify its expected impact on performance. Because the LDMS data collection used in this study is being deployed on Trinity, our approaches analyzing this data will be drop-in compatible with the data being collected. With some additional validation and production hardening, these approaches can be used by application users and developers as well as Trinity system administrators to gain an understanding of which applications run simultaneously and across which node resources drive congestion related degradation. This understanding will ultimately enable automated use of follow-on tools for making run-time decisions on scheduling and resource allocation based on both historic information on an application and input deck's expected resource utilization, including but not limited to network and file system related resources, and how those resources are currently being utilized by running applications.

Near term we expect the results and insights from this work and its application to actual workloads being run on Trinity, to result in better adaptive routing techniques being employed in Trinity's High Speed Networks (HSN). The current adaptive routing techniques, employed in Cray's Aries based Dragonfly topology, have been shown, via simulation, to spread traffic and minimize network hot spots and congestion. In practice we see what appears to be HSN contention related performance degradation in these networks even when the applications experiencing the contention are placed such that we don't expect traffic paths to overlap. Understanding and mitigating this problem is an active area of investigation for both the vendor and system operators and research staff. Utilizing and expanding the analysis techniques developed in this work in conjunction with the HSN, file system, and application performance information currently being collected on Trinity can provide the insight needed to mitigate the adverse effects through better adaptive routing algorithms and feedback loops.

Longer term this work and results will form the basis for follow-on research in this area by both this team and other researchers. In particular we would like to identify other metrics of interest that could be used to refine our model's prediction accuracy. We also envision application of these techniques to other shared resources such as shared "Burst Buffers" and resources shared by processes of an individual application such as memory bus bandwidth, CPU, GPU, etc. While the work presented here targeted the Cray XC40 platform and associated metrics of interest, the techniques are generally applicable to HPC platforms which all are impacted by the same problems (i.e., application performance degradation due to contention for limited shared resources). Additionally these adverse effects limit application and system scalability and their mitigation will be required in order to continue to effectively scale systems, applications, and science.

DOE spends hundreds of millions of dollars per year on HPC platforms, facilities, support staff, applications developers, and analysts in order to solve problems of importance to national security. Research has identified contention for shared resources as a major cause of application performance degradation on large scale HPC systems where run time increases of 50% and more are common. Mitigating even a small fraction of these adverse effects will have an enormous impact on the efficiency of these costly resources and throughput in terms of work accomplished per-dollar and per-time.

Chapter 7

Conclusion

The goal of this LDRD project was to identify high-performance computing platform monitoring metrics that could be used to identify contention related performance degradation of simultaneously running CTH simulations that are adversely impacting each other due to their use of shared resources. Target resources of interest included the network interconnect and the file system. Our approach was to perform a preliminary assessment of related monitoring data and attempt to develop a model that could quantify the overall time the target applications spent in a contentious state along with an estimate of the overall slowdown experienced. We succeeded with every aspect of this LDRD goal, even with an expanded that added inclusion of Intel's IMB and LLNL's IOR codes for additional correlation data. Every project milestone was achieved. A list of the major accomplishments are listed below.

- Experiments with IOR, IMB, and CTH were crafted to exhibit network interconnect and file system contention.
- Dedicated Application Time was scheduled for Mutrino and, during this time, the experiments were performed and were successful in demonstrating interconnect and file system contention through total wall time comparisons between baseline and contentious cases.
- The *Actionable Metric Model*, AMM, was shown to properly identify time spent in contention for both IOR and CTH.
- AMM's preliminary slowdown model provided good estimates for file system performance-based metrics whereas it has room for improvement for network interconnect metrics.

Follow-on research to continue model development (e.g., improved initial conditions and multi-resource rollup) and refinement (e.g., improved slowdown model for metrics with a non-linear performance impact) is being actively planned.

References

[1] A. Bhatele, K. Mohror, S. Langer, and K Isaacs. There Goes the Neighborhood: Performance Degradation due to Nearby Jobs. In *Proc. Int'l Conf. on High Performance Computing, Networking, Storage and Analysis (SC)*, 2013.

[2] J. Brandt, K. Devine, A. Gentile, and K. Pedretti. Demonstrating Improved Application Performance Using Dynamic Monitoring and Task Mapping. In *1st Wrk. on Monitoring and Analysis for High Performance Computing Systems Plus Applications (HPCMASPA) Proc. IEEE Int'l Conf. on Cluster Computing (CLUSTER)*, 2014.

[3] J. Brandt, E. Froese, A. Gentile, L. Kaplan, B. Allan, and E. Walsh. Network Performance Counter Monitoring and Analysis on the Cray XC Platform. In *Proc. Cray User's Group*, 2016.

[4] Greg Faanes, Abdulla Bataineh, Duncan Roweth, Tom Court, Edwin Froese, Bob Alverson, Tim Johnson, Joe Kopnick, Mike Higgins, and James Reinhard. Cray Cascade: A Scalable HPC System Based on a Dragonfly Network. In *Proc. Int'l Conf. on High Performance Computing, Networking, Storage and Analysis (SC12)*, 2012.

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

[6] The Gemini Network, Aug 2010.

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

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

[9] 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 Conf. for High Performance Storage, Networking, and Analysis (SC)*, 2014.

[10] LLNL/ior: Parallel filesystem I/O benchmark. Lawrence Livermore National Laboratory, September 2016.

[11] Hongzhang Shan and John Shalf. Using IOR to Analyze the I/O Performance for HPC Platforms. In *Cray User Group Conference (CUG)*, 2007.

[12] Hongzhang Shan, Katie Antypas, and John Shalf. Characterizing and Predicting the I/O Performance of HPC Applications Using a Parameterized Synthetic Benchmark. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, SC '08, pages 42:1–42:12, Piscataway, NJ, USA, 2008. IEEE Press.

[13] ASC Sequoia Benchmark Codes. Lawrence Livermore National Laboratory, September 2016.

[14] IOR. National Energy Research Scientific Computing Center, September 2016.

[15] Getting Started with Intel MPI Benchmarks 2017, Intel Software. Intel Corporation, September 2016.

[16] Subhash Saini, Robert Ciotti, Brian T.N. Gunney, Thomas E. Spelce, Alice Koniges, Don Dossa, Panagiotis Adamidis, Rolf Rabenseifner, Sunil R. Tiyyagura, and Matthias Mueller. Performance evaluation of supercomputers using HPCC and IMB Benchmarks. *Journal of Computer and System Sciences*, 74(6):965 – 982, 2008.

[17] Ahmed Bukhamsin, Mohamad Sindi, and Jallal Al-Jallal. Using the Intel MPI Benchmarks (IMB) to Evaluate MPI Implementations on an Infiniband Nehalem Linux Cluster. In *Proceedings of the 2010 Spring Simulation Multiconference*, SpringSim '10, pages 240:1–240:4, San Diego, CA, USA, 2010. Society for Computer Simulation International.

[18] Cray Inc. Performance Measurement and Analysis Tools. Cray Doc S-2376-63, 2015.

[19] J. M. McGlaun and S. L. Thompson. CTH: A Three-Dimensional Shock Wave Physics Code. *International Journal of Impact Engineering*, 10:351–360, 1990.

Appendix A

Statistical Operators

The geometric mean, μ_g, of data set a_i with n entries is defined in equation A.0.1.

$$\mu_g = \left(\prod_{i=1}^{n} a_i \right)^{\frac{1}{n}} \tag{A.0.1}$$

The geometric standard deviation, σ_g, of the same data set is defined in equation A.0.2.

$$\sigma_g = \exp \left(\sqrt{\frac{\sum_{i=1}^{n} \left(\ln \frac{a_i}{\mu_g} \right)^2}{n}} \right) \tag{A.0.2}$$

The geometric confidence interval above (δ_g^+) and below (δ_g^-) μ_g are defined in equations A.0.3a A.0.3b, respectively.

$$\delta_g^+ = \mu_g \times \sigma_g^\eta \tag{A.0.3a}$$
$$\delta_g^- = \mu_g \div \sigma_g^\eta \tag{A.0.3b}$$

where η is the number of standard deviations to include in the confidence interval. The contention model typically computes these quantities for a data set that includes the metrics' current values within a rolling window of ω previous values. When this occurs, $n = \omega$.

DISTRIBUTION:

1	MS	0807	Anthony M. Agelastos, 9326
1	MS	0807	Joel O. Stevenson, 9326
1	MS	0823	James M. Brandt, 9328
1	MS	0823	Ann C. Gentile, 9328
1	MS	0823	Justin M. Lamb, 9326
1	MS	0840	Kevin P. Ruggirello, 1555
1	MS	0359	D. Chavez, LDRD Office, 1911
1	MS	0899	Technical Library, 9536 (electronic copy)

v1.40

Sandia National Laboratories