Fan-In Communications On A Cray Gemini Interconnect

Terry Jones[1], Bradley W. Settlemyer[1]

[1] Oak Ridge National Laboratory
Oak Ridge, TN 37831, USA
{trj, settlemyerbw }@ornl.gov

Abstract

Using the Cray Gemini interconnect as our platform, we present a study of an important class of communication operations—the fan-in communication pattern. By its nature, fan-in communications form 'hot spots' that present significant challenges for any interconnect fabric and communication software stack. Yet despite the inherent challenges, these communication patterns are common in both applications (which often perform reductions and other collective operations that include fan-in communication such as barriers) and system software (where they assume an important role within parallel file systems and other components requiring high-bandwidth or low-latency I/O). Our study determines the effectiveness of differing client-server fan-in strategies. We describe fan-in performance in terms of aggregate bandwidth in the presence of varying degrees of congestion, as well as several other key attributes. Comparison numbers are presented for the Cray Aries interconnect. Finally, we provide recommended communication strategies based on our findings.

Keywords: Time service, clock synchronization, MPI, supercomputing, system software, programming tools

1. Introduction

Fan-in communication patterns are collections of data transfer between two groups of distributed nodes in which one group is larger in node count than the other (*M to N* communications where *M* > *N*). Many times, the smaller group of nodes consists of a single node forming an *N-ary tree* graph. Fan-in communications may be characterized in terms of scalability (incorporating both bandwidth and latency aspects), fairness, and performance variability. Because of their pervasiveness and their capacity to be an impediment to overall performance, fan-in communications are important to both applications and system software.

Our interest stems from research of user-level reservation schemes. Fan-in patterns intrinsically create hot-spots; congestion caused by hot-spot traffic can significantly degrade the performance of a computer network therefore requiring specialized techniques for optimizing one or more fan-in attributes. A reservation scheme coordinates message traffic flow by establishing a multiplexing capability. In our previous work, we presented several techniques to improve the fine-grained communication of a sophisticated application (NAMD) using the uGNI library for Gemini [1], fan-in based time synchronization protocols [2], the performance and scalability measurements of key parallel file system components [3] and reservation-based quality of service schemes for parallel storage systems [4]. Our current work is focused at user-level communication strategies and is being explored with a tool we have developed on top of MPI that we call *reservation* [5]. Using the *reservation* tool, we are able to arbitrarily establish the scale of the fan-in tree as well as the other pertinent communication factors including the number of large message-size clients, the number of small message-size clients, the associated buffer sizes, and details about request queues and fairness properties. We have chosen to implement our strategies over MPI which carries certain benefits and liabilities. Unlike our earlier work, MPI provides less control than programming the uGNI layer directly. On the positive side, all techniques that we have employed are available to uGNI-based implementations (the reverse is not necessarily true), our methods are available to a wide audience including MPI-based applications, and we were able to fully explore the strategies we wished to investigate over an MPI layer. Finally, our work seeks to address general solutions and we do not require pre-determined coordination as in some collective reductions or gather operation strategies.

Our contributions therefore are: *reservation*, a performance analysis tool designed specifically to explore fan-in communication patterns; an assessment of the fan-in capabilities of Gemini 3D-torus interconnect; a comparison to a different topology (Cray Aries, a Dragonfly topology); and a collection of recommended communication strategies based on our findings.

The rest of this paper is organized as follows. In section 2, we introduce our testing environment and methodologies. Section 3 provides a series of relevant data measurements, followed by a discussion of these measures in Section 4. In Section 5, we describe work related to our own. We close with our future plans and a conclusion.

2. Testing Environment and Methodologies

Frequently in parallel machines, it is useful to have a client-server arrangement. While multiple client scenarios are common, it is often desirable to limit the number of servers to avoid inconsistencies in data state. Such an arrangement naturally results in fan-in communications. Fan-in communication patterns are collections of data transfer between two groups of distributed nodes in which one group is larger in node count than the other (*M to N* communications where $M > N$), see Figure 1. Many times, the smaller group of nodes consists of a single node forming an *N-ary tree* graph.

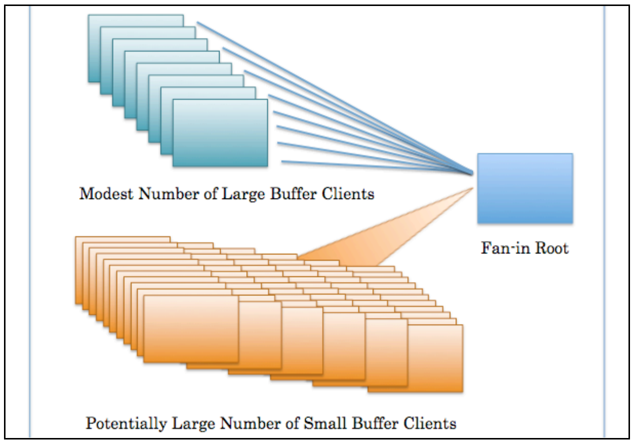
Modest Number of Large Buffer Clients
Fan-in Root
Potentially Large Number of Small Buffer Clients

Figure 1. Fan-In communication: A fan-in root exchanges messages with both large buffer clients (in teal) and small buffer clients (in orange).

Notice that Fan-in communications may vary along several attributes. The number of large-buffer clients can range from 0 to the available nodes of the environment. (Here, we define a "large-buffer client" as a communicating client that is transferring messages sizes above some minimum threshold, 64Kbytes in our testing). Likewise, the number of "small-buffer" clients can vary from 0 to the number of available nodes. The message sizes associated with both large and small provides another basic attribute. Our primary interest is in maximizing the large-buffer bandwidth; we refer to the interfering small-buffer bandwidth as ***chatter*** traffic.

The communication interface and underlying network protocol can influence the behavior of fan-in communications. We have chosen to use MPI, the Message Passing Interface, in our studies [6]. MPI is the de facto standard for inter-node communication within our test environment, a DOE managed supercomputer center charged with yielding scientific insight in the nation's interest through computer applications. While our results are presented in MPI-specific terms, non-MPI interfaces (including uGNI and UNIX sockets) usually expose similar API choices. MPI provides collective topology functions for many Cartesian communication patters (rings, meshes, etc.). However, fan-in communication patterns are not typically provided as a pre-defined topology. As with many APIs, MPI incorporates a receive queue to buffer incoming messages until they are matched via source tags and user defined message tags. This design yields several possible design choices for the MPI programmer: the depth of the receive queue; the number of receive queues; and the communication fairness which may be managed through how strict or open the policy is defined for accepting arriving messages (i.e., arriving messages may be included or excluded based on policies ranging from the very restrictive exact match of both message-source and message-tag, to the completely unrestricted case of using special match-any flags for both message-sources and message-tags). These design choices yield a rich environment for performance variation. [7,8,9].

To help us understand the impact of these various attributes and parameters, we developed a performance measure tool that we call *Reservation*.

2.1. A Tool For Evaluating Fan-Ins: Reservation

The Reservation tool is an MPI program capable of measuring the performance for various fan-in configurations. It measures performance for a single-queue design as well as a dual-queue design. The dual-queue measurements result from large-buffers being handled separately than small buffers and control messages. In addition, it permits adjustable sizes for both "small" messages and "large" messages. Finally, messages may be either matched with wildcard match-any tags, or with specific matching. The basic logic for the application is presented in Figure 2.

```
ROOT:
    (1)  First, the single root node posts Irecvs for the
         clients;
    (2)  then the root primes a set of handshake
         communications from each client by cycling
         through sending a message to each client and
         enters a response loop (small or large
         message depending on client);
    (3)  in the root response loop, the root receives a
         message from any client, then it returns it back
         to the same client.
CLIENTS:
    (1)  Meanwhile, the client posts a set of Irecvs for
         the root.
    (2)  then the client enters its response loop: first
         receiving a message from root, then returning
         the message (small or large message
         depending or client).
```

Figure 2. "Reservation" is a tool developed at ORNL to perform performance measurements for fan-in communication. Various aspects of the communication can be tailored.

2.2. The Gemini and Aries Network

The Gemini computer interconnect is based on a three-dimensional torus topology (see Figure 3). An n-dimensional torus is a mesh with the processors on the end of each dimension connected together. This reduces the diameter of the network by half. The diameter of an X x Y x Z torus is (X + Y + Z)/2. Each Gemini chip is connected to 6 of its nearest neighbors: X+, X-, Y+, Y-, Z+, and Z-. While this maps well to nearest-neighbor exchanges, fan-in communications can pose challenges. In Gemini, near-neighbors according to the routing scheme are given more bandwidth than farther clients, which receive geometrically less a share of bandwidth according to fan-in/distance (the network is locally fair, but globally unfair). This means that for large machines, unless each link has adequate bandwidth for the traffic pattern, the chances of contention increase as messages travel farther and farther; large fan-in communications therefore can pose significant challenges for such topologies. Link speeds for Gemini are presented below in Table 1 [10].

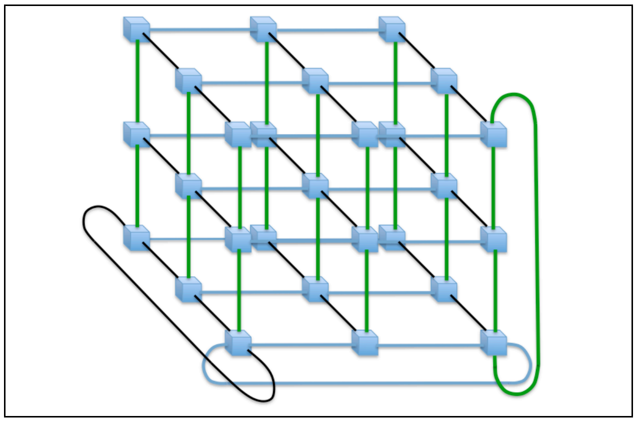

Figure 3. The Gemini interconnect connects nodes via a 3D torus network.

Each Gemini chip supports two nodes, and those nodes communicate via "lanes". Three lanes comprise a "link". The link speed depends on the link type, and protocol overheads are about 35% for large messages [11]. The expected bandwidths are shown in Table 1. Note that Gemini bandwidth is asymmetrical among X, Y, and Z with Y bandwidth equal to ½ X bandwidth and ½ Z bandwidth.

Table 1. Gemini Speeds by Link Type.

GEMINI Speeds gbps = Giga bits per second GBytes/s = Giga bytes per second				
Link Type	Data Rate	# Links	Bitrate	Data Rate
Y-Mezzanine	6.25 gbps	12	9.375 GB/s	~6 GByte/s
Z-Backplane	5.0 gbps	24	15 GB/s	~9.75 GByte/s
X,Z Cable	3.125 gbps	24	9.375 GB/s	~6 GByte/s
Y Cable	3.125 gbps	12	4.687 GB/s	~3 GByte/s

In Contrast, Aries is based on a Dragonfly topology (see Figure 4). Systems can be configured to meet bandwidth requirements by varying the number of optical connections. Bidirectional bandwidth for two Aries nodes at 4K message size is approximately 14.3 GBytes/s. Peak global bandwidth is 11.7 GBytes/s per node for a full network; with a payload efficiency of 64 percent this equates to 7.5 GBytes/s per direction [12].

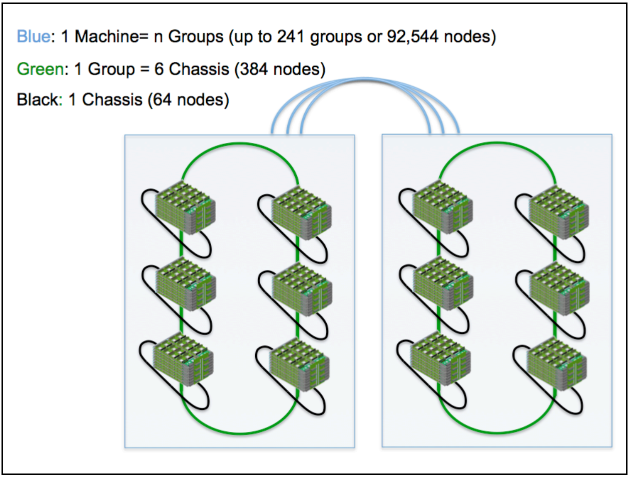
Blue: 1 Machine= n Groups (up to 241 groups or 92,544 nodes)
Green: 1 Group = 6 Chassis (384 nodes)
Black: 1 Chassis (64 nodes)

Figure 4. The Aries interconnect connects nodes via a Dragonfly network.

Table 2. Aries Speeds by Link Type.

ARIES Speeds gbps = Giga bits per second		
Aries Component	Link Type	Data Rate
Black	PCI Express Gen3 x16	16 gbps per lane
Green	Electrical cables	14 gbps per lane
Blue	Optical Cable	12.5 gbps per lane

2.3. The OLCF Environment: Titan and EOS

Titan, currently the world's largest machine for open science, is located at Oak Ridge National Laboratory at the Oak Ridge Leadership Computing Facility (OLCF) [13]. A Cray XK7, Titan is based on the Gemini network and features a hybrid-architecture with a theoretical peak performance exceeding 27,000 trillion calculations per second (27 petaflops). It contains both advanced 16-core AMD Opteron central processing units (CPUs) and unconventional NIVIDIA Kepler graphics processing units (GPUs) [14]. Titan incorporates 18,688 compute nodes, a total system memory of 710 terabytes, and Cray's high-performance Gemini network. Its 299,008 CPU cores guide simulations while the accompanying GPUs that can handle hundreds of calculations simultaneously. The 3D-torus for Titan has the dimensions of Z=24, X=25, Y=16 (that is, 24 blades per cabinet, 25 cabinets, 2*8 rows).

Eos is a 744-node Cray XC30 cluster with a total of 47.6 TB of memory [15]. The processor is the Intel® Xeon® E5-2670 (10-core Ivy Bridge). Eos uses Cray's Aries interconnect. Aires provides a higher bandwidth and lower latency interconnect than Gemini. In total, the Eos compute partition contains 11,904 traditional processor cores (23,808 logical cores with Intel Hyper-Threading enabled), and 47.6 TB of memory. The Dragonfly topology interconnect of Eos is configured with 240 Blue links using 60 optical cables, 4 links per cable.

3. Experimentation and Data Measurements

3.2. Measurements

We began our investigation by conducting multiple measurements of a basic fan-in scenario in which a variable number of large buffer clients exchange messages with one root node; the average results are presented in Figure 5. The graph (and the subsequent graphs) show aggregate bandwidth as measured at the fan-in root on the Y-axis versus the number of large-buffer clients on the X-axis (higher numbers on Y-axis is better). In each test, *large-buffer* is defined as 4MB and *small buffer* is defined as 4K. The four series of Figure 5 present results for: (i) Aries interconnect with only large-buffer clients; (ii) Aries interconnect with both large-buffer clients and small-buffer clients where the number of small-buffer and large-buffer clients are arranged in a 4:1 ratio; (iii) Germini interconnect with only large-buffer clients, (iv) Gemini interconnect with both large-buffer clients and small-buffer clients – also with small-buffer and large-buffer clients in a 4:1 ratio. The "DualQ" series utilize separate queues for small & large messages; "SingleQ" series use a single queue; this is explained in more detail below.

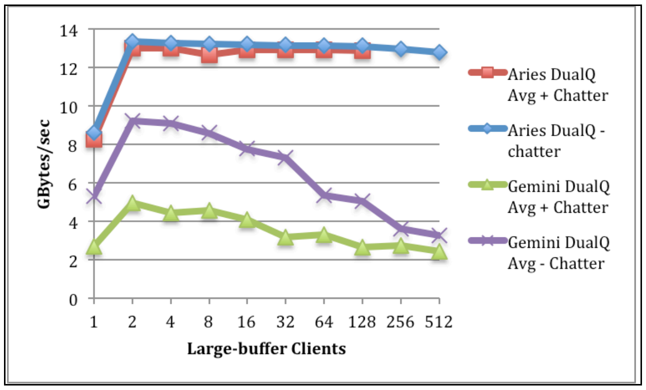

Figure 5. Performance measurements for Aries and Gemini; bandwidth is given for chatter clients with a 4:1 client to chatter client ratio, and in the absence of chatter clients. The "+ chatter" means with chatter present, "- chatter" means without chatter present.

Since Figure 5 presents averages, it's interesting to look a little deeper into the data and check for variability between runs. Figures 6 and 7 present individual runs on both Gemini and Aries. The remarkable consistency of the Aries interconnect stands out in Figure 7. In fact, the multiple runs appear to be a single series as each indivdual series lies almost exactly on its predecessor.

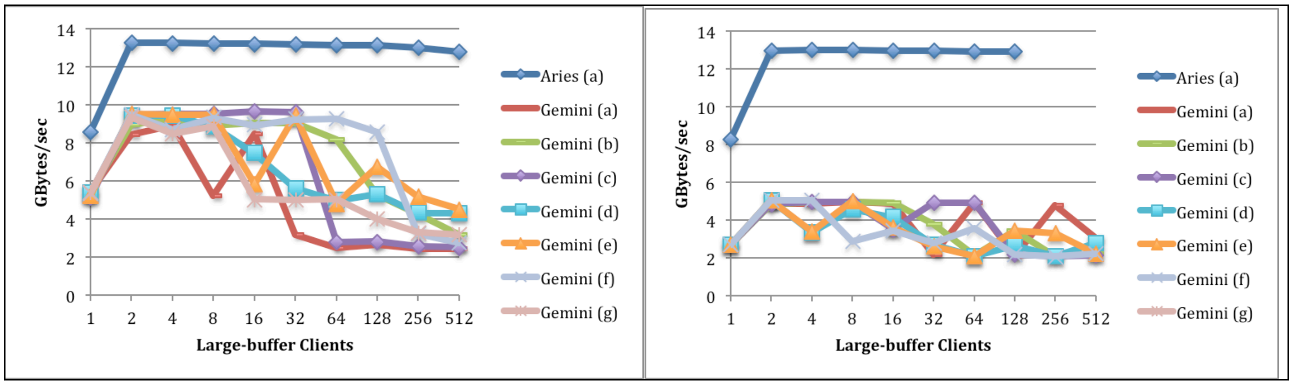

Figure 6. Variability: The figure at left shows bandwidth variance when no chatter is present; the right figure shows bandwidth with a 4:1 client to chatter client ratio.

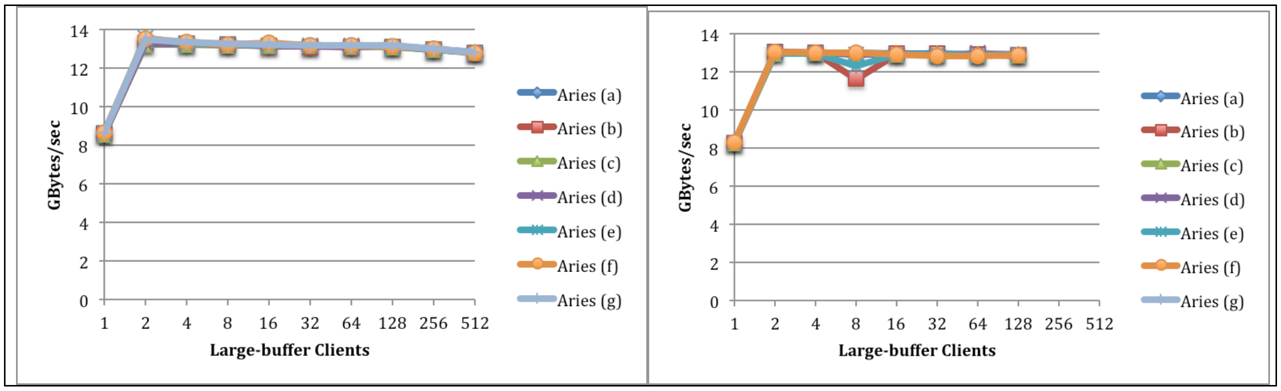

Figure 7. Aries Variability in Detail: Aries exhibits very little variability. The figure at left shows bandwidth variance when no chatter is present; the right figure shows bandwidth with a 4:1 client to chatter client ratio.

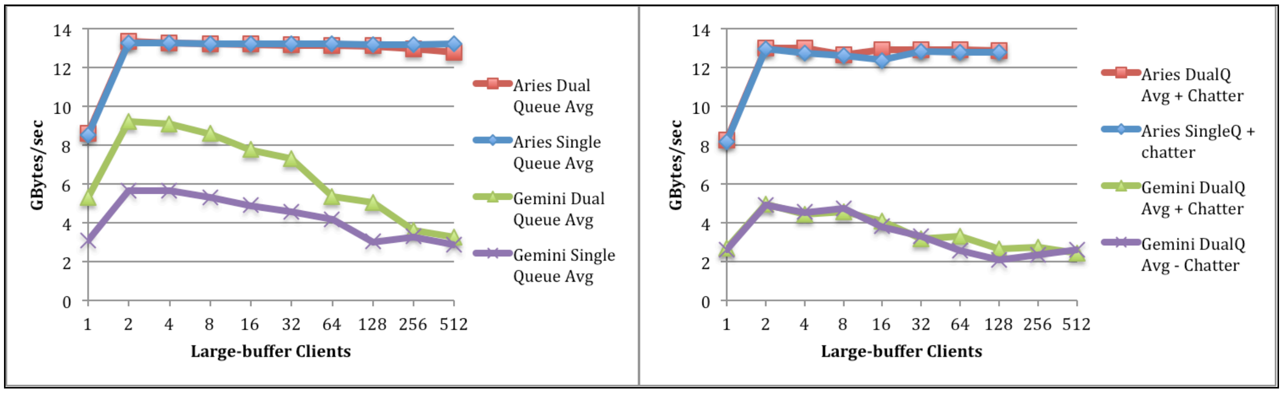

Figure 8. Separate Queues and Single Queues: These graphs depict performance when using multiple queues for different message types (red) and the performance for all messages being routed to the same queue (blue). As above, the figure at left shows bandwidth variance when no chatter is present; the right figure shows bandwidth with a 4:1 client to chatter client ratio. The "+ chatter" means with chatter present, "- chatter" means without chatter present

In Figure 8 we provide information on the effectiveness of using separate MPI queues for different message types. Non-blocking communications use opaque request objects to identify communication operations and match the operation that initiates the communication with the operation that terminates it. These are system objects that are accessed via a handle. A request object identifies various properties of a communication operation, such as the send mode, the communication buffer that is associated with it, its context, the tag and destination arguments to be used for a send, or the tag and source arguments to be used for a receive. In addition, this object stores information about the status of the pending communication operation [6]. The "DualQ" series utilize separate queues for small & large messages; "SingleQ" series use a single queue.

Figure 9 returns interest to another MPI programming choice. MPI programmers must decide how deep the queue should be for these request objects. Figure 8 presents results for deep queues (1 queue entry for each client) and shallow clients (fixed queues of 20 entries).

Figure 10 shows the impact of message-matching decisions upon fan-in communications. The selection of a message by a receive operation is governed by the value of the message envelope. A message can be received by a receive operation if its envelope matches the source, tag and communicator values specified by the receive operation. The receiver may specify a wildcard MPI_ANY_SOURCE value for source, and/or a wildcard MPI_ANY_TAG value for tag, indicating that any source and/or tag are acceptable. Thus, a message can be received by a receive operation only if it is addressed to the receiving process, has a matching communicator, has matching source unless source=MPI_ANY_SOURCE in the pattern, and has a matching tag unless tag=MPI_ANY_TAG in the pattern. The message tag is specified by the tag argument of the receive operation. The argument source, if different from MPI_ANY_SOURCE, is specified as a rank within the process group associated with that same communicator [6]. Figure 10 presents series with wildcard match-any tags, and with specific matching.

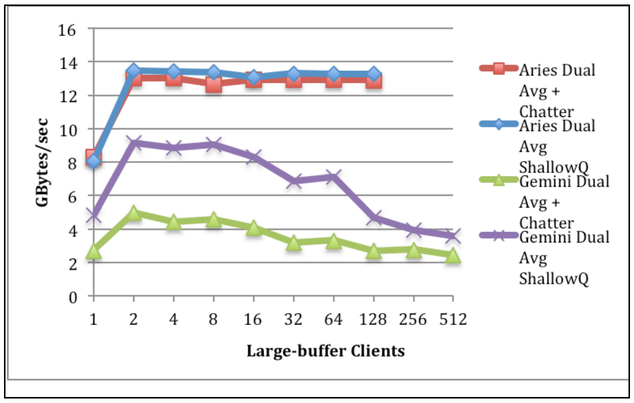

Figure 9. Affect of strategies fixed MPI queue depth versus equal to clients. All series include chatter (small-buffer clients). The "+ chatter" means with chatter present, "- chatter" means without chatter present

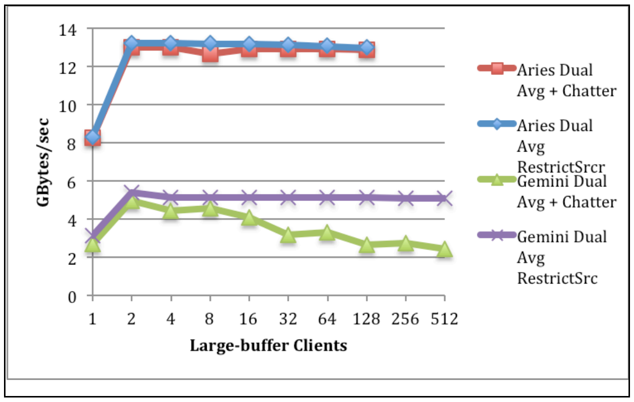

Figure 10. Affect of strategies MPI_SOURCE_ANY versus specific client. All series include chatter (small-buffer clients). The "+ chatter" means with chatter present, "- chatter" means without chatter present.

4. Discussion of Measurements

4.1. General Findings

Several initial observations stand out from our data. First, not only does Aries exhibit significantly higher bandwidths (which is to be expected from a later generation interconnect), it also maintains its performance over a much broader spectrum of client loads and programming choices. This was true for every graph, but is perhaps most pronounced in Figure 6. Meanwhile, Gemini bandwidth is stronly influenced by the particular placement of communicating nodes as seen in run-to-run variability of Gemini data series in Figure 6 — likely a consequence of the routing scheme which gives more bandwidth to closer clients (recall farther Gemini clients receive geometrically less a share of bandwidth according to fan-in/distance resulting in locally fair, but globally unfair routing). Hence, a favorable Gemini mapping of ranks to nodes in one run can have significantly higher performance than an unfavorable mapping, and a collection of runs with different mappings may exhibit large variability. We did test this hypothesis by running the same tests that exhibited large variability in Figure 6, but all with the same mappings (multiple jobs within the same batch script), and Gemini exhibited very little variation under this scenario (see Figure 11).

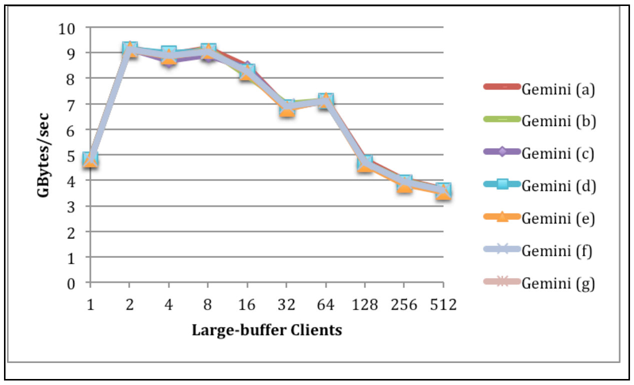

Figure 11. Variation of Gemini across 7 series when *both* program parameters & node-to-rank mappings are held constant.

Neither Aries nor Gemini showed strong reaction to the depth of the non-blocking receive queue (Figure 9), but Gemini was heavily influenced by the absence or presence of wildcard matching on the message source and/or message tag (Figure 10): when the receiver employs restrictive measures (that is, matching on a specific source and message tag), the achieved bandwidth remains constant at about 5.1 GB/sec — a significant win for large numbers of clients under load from small-buffer clients, but a significant loss when there is less than 128 large-buffer clients and no small-buffer clients.

These observations lead to the following generalizations:

- Neither Aries or Gemini are able to reach their peak bandwidth with only one large-buffer client (i.e., two or more clients are required to saturate a servers bandwidth capacity).
- Aries is able to achieve slightly more than 13 GB/sec bandwidth at the root; Gemini is able to achieve slightly more than 9 GB/s.
- Aries bandwidth does not vary considerably from run to run, nor does it drop considerably under increased contention (more client nodes), the depth of the non-blocking receive queue, the rank-to-node mapping, or the presence or absence of wildcard matching for either message source or message tag.
- Gemini drops between 50% and 20% under contention from small-buffer clients, and exhibits a constant bandwidth of around 5.1 GB/sec regardless of the number of clients when the message source and tag are restricted.

4.2. Recommendations

Our results indicate several policies will help to ensure maximum performance for fan-in communication scenarios.

For Aries, adding additional code complexity or constraints upon batch submissions is probably unwarranted: the Aries interconnect simply does an impressive job of delivering its best performance across a very wide range of scenarios. In particular, we observed little benefit for managing the rank-to-node mappings, whether one or more queues are employed, adding restrictive matching for non-blocking communications, adding deep message queues. You should be able to achieve around 13 GB/sec

For Gemini, performance may be tuned for different scenarios. Fan-in communications should benefit from several choices when possible: limit concurrent small-buffer traffic, choose a dual-queue architecture if small amounts of chatter (small-buffer traffic) are anticipated; choose restrictive message matching policies for large-buffer client counts above 128, and choose rank-to-node mappings which minimize the number of hops for the most performance sensitive communications.

5. Related Work

Reservation schemes [16, 17, 18] have been a popular mechanism for delivering quality of service (QoS) guarantees to long-haul networking. These protocols have typically not been leveraged in the system software running on the high performance interconnection networks common to modern high performance computing systems. Our study measures the interference costs associated with the fan-in communication patterns anticipated for a storage system QoS reservation scheme.

Benchmarking and evaluation of high performance interconnection networks, while a popular area of study [19, 20, 21, 22, 23, 24], has been primarily focused on exploring the performance of popular scientific application patterns, such as collectives and bulk transfers. Although small message performance is typically included in the evaluation, the focus has been on the aggregate performance rather than the interference patterns generated by competing clients.

Bhatelé and Kalé examined the effects of contention in high performance interconnection networks [25]. A benchmark was constructed to have all pairs of processes send messages at the same time with the number of hops between each sender fixed. The results indicated that large message sizes and high-hop counts could severely reduce the performance of the entire interconnection network. The confidence toolkit [26] also examined the performance impacts of interference workloads with point-to-point messages by constructing empirical distributions of message latencies that described interference-based delay. The work presented here builds on these efforts by including recent high performance interconnection networks and fan-in communication patterns.

6. Future Work and Conclusion

This paper has described *reservation*, a performance analysis tool designed specifically to explore fan-in communication patterns and an assessment of the fan-in capabilities of both the Cray Gemini 3D-torus and Cray Aries Dragonfly interconnect. Aries is able to maintain its maximum bandwidth under a wide range of settings and code simplicity should probably influence design choices; for Gemini, limit concurrent small-buffer traffic, choose restrictive message matching policies for large-buffer client counts above 128, and choose rank-to-node mappings which minimize the number of hops for the most performance sensitive communications.

While these results meet our immediate needs and objectives, this line of inquiry has led us to consider still further related lines of inquiry. We have plans to pursue data collection for additional machines. Finally, we intend to investigate methods of scheduling client activity to realize better overall bandwidth or better bandwidth from a predefined client.

Acknowledgements

The authors thankfully acknowledge the computer resources and assistance provided by the National Center for Computational Sciences at Oak Ridge National Laboratory. These Oak Ridge National Laboratory resources are supported by the Office of Science of the Department of Energy and were made available by a Director's Discretion award. Research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U. S. Department of Energy. Finally, we would like to express our gratitude to the anonymous reviewers for their helpful comments and suggestions.

The submitted manuscript has been authored by a contractor of the U.S. Government under Contract No. DE-AC05-00OR22725. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

7. References

[1] Yanhua Sun, Gengbin Zheng, Chao Mei, Eric Bohm, James Phillips, Laxmikant Kale, and Terry Jones. Optimizing Fine-grained Communication in a Biomolecular Simulation Application on Cray XK6. *The 25th International Conference for High Performance Computing, Networking, Storage and Analysis (SC'12)*. Salt Lake City, UT. November 2012.

[2] Terry Jones, and Gregory Koenig. "Clock Synchronization in High-end Computing Environments: A Strategy for Minimizing Clock Variance at Runtime." *Journal of Concurrency and Computation: Practice & Experience*, Volume 25, Issue 6, doi: 10.1002/cpe.2868, pages 881-897, April 25, 2013.

[3] Feiyi Wang, Mark Nelson, Sarp Oral, Scott Atchley, Sage Weil, Bradley W. Settlemyer, Blake Caldwell, and Jason Hill. 2013. Performance and scalability evaluation of the Ceph parallel file system. In *Proceedings of the 8th Parallel Data Storage Workshop* (PDSW '13). ACM, New York, NY, USA, 14-19. DOI=10.1145/2538542.2538562 http://doi.acm.org/10.1145/2538542.2538562.

[4] Michael J. Brim, David A. Dillow, Sarp Oral, Bradley W. Settlemyer, and Feiyi Wang. 2013. Asynchronous object storage with QoS for scientific and commercial big data. In *Proceedings of the 8th Parallel Data Storage Workshop* (PDSW '13). ACM, New York, NY, USA, 7-13. DOI=10.1145/2538542.2538565 http://doi.acm.org/10.1145/2538542.2538565.

[5] ORNL LDRD Project: Towards a Resilient and Scalable Infrastructure for Big Data. Oak Ridge National Laboratory.

[6] MPI Forum. MPI: A Message-Passing Interface Standard. Version 3.0, September 21st 2012. available at: http://www.mpi-forum.org (Apr. 2014)."

[7] Sur, S., Chai, L., Jin, H. W., & Panda, D. K. (2006, April). Shared receive queue based scalable MPI design for InfiniBand clusters. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International* (pp. 10-pp). IEEE.

[8] Underwood, K. D., & Brightwell, R. (2004, August). The impact of MPI queue usage on message latency. In *Parallel Processing, 2004. ICPP 2004. International Conference on*, pp. 152-160. IEEE.

[9] Brightwell, Ron, Sue Goudy, and Keith Underwood. "A preliminary analysis of the mpi queue characterisitics of several applications." In *Parallel Processing, 2005. ICPP 2005. International Conference on*, pp. 175-183. IEEE, 2005.

[10] R. Alverson, D. Roweth, and L. Kaplan, "The Gemini system interconnect." *In High Perofmance Interconnects (HOTI), 2010 IEEE 18th Annual Symposium on*, Aug., pp. 83-87.

[11] Matt Ezell. Understanding the Impact of Interconnect Failures on System Operation. Napa Valley, CA. CUG 2013. May, 2013.

[12] Greg Faanes, Abdulla Bataineh, Duncan Roweth, Tom Court, Edwin Froese, Bob Alverson, Tim Johnson, Joe Kopnick, Mike Higgins, and James Reinhard. 2012. Cray cascade: a scalable HPC system based on a Dragonfly network. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (SC '12). IEEE Computer Society Press, Los Alamitos, CA, USA, , Article 103 , 9 pages.

[13] OLCF. The Oak Ridge Leadership Computing Facility. https://www.olcf.ornl.gov

[14] Cray XK7 Data Sheet, http://www.cray.com/Products/Computing/XK7/Specifications.aspx.

[15] Cray XC Data Sheet, http://www.cray.com/Products/Computing/XC/Specs/Spcifications-XC30.aspx.

[16] Zhang, L., Deering, S., Estrin, D., Shenker, S., & Zappala, D. (1993). RSVP: A new resource reservation protocol. *Network, IEEE*, *7*(5), 8-18.

[17] Le Faucheur, F., Wu, L., Davie, B., Davari, S., Vaananen, P., Krishnan, R., ... & Heinanen, J. (2002). *Multi-protocol label switching (MPLS) support of differentiated services*. RFC 3270, May.

[18] Eriksson, A., & Gehrmann, C. (1998, May). Robust and secure light-weight resource reservation for unicast IP traffic. In *Quality of Service, 1998.(IWQoS 98) 1998 Sixth International Workshop on* (pp. 168-170). IEEE.

[19] Fabrizio Petrini, Eitan Frachtenberg, Adolfy Hoisie, and Salvador Coll, Performance evaluation of the quadrics interconnection network, Cluster Computing 6 (2003), no. 2, 125–142

[20] Kei Davis, Adolfy Hoisie, Greg Johnson, Darren J. Kerbyson, Mike Lang, Scott Pakin, and Fabrizio Petrini, A performance and scalability analysis of the BlueGene/L architecture, SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing (Washington, DC, USA), IEEE Computer Society, 2004, p. 41.

[21] Sadaf R. Alam, Jeffery A. Kuehn, Richard F. Barrett, Jeff M. Larkin, Mark R. Fahey, Ramanan Sankaran, and Patrick H. Worley, Cray XT4: an early evaluation for petascale scientific simulation, nov. 2007, pp. 1–12.

[22] S. Alam, R. Barrett, M. Bast, M. R. Fahey, J. Kuehn, C. McCurdy, J. Rogers, P. Roth, R. Sankaran, J. S. Vetter, P. Worley, and W. Yu, Early evaluation of IBM BlueGene/P, SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing (Piscataway, NJ, USA), IEEE Press, 2008, pp. 1–12

[23] Kevin J. Barker, Kei Davis, Adolfy Hoisie, Darren J. Kerbyson, Mike Lang, Scott Pakin, and Jose C. Sancho, Entering the petaflop era: the architecture and performance of Roadrunner, SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing (Piscataway, NJ, USA), IEEE Press, 2008, pp. 1–11.

[24] Kerbyson, D.K.; Barker, K.J. "Analyzing the Performance Bottlenecks of the POWER7-IH Network", *Cluster Computing (CLUSTER), 2011 IEEE International Conference on,* On page(s): 244 - 252

[25] Abhinav Bhatele and V. Laxmikant Kale. An evaluative study on the effect of contention on message latencies in large supercomputers, IPDPS '09: Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing (Washington, DC, USA), IEEE Computer Society, 2009, pp. 1–8

[26] Settlemyer, B. W., Hodson, S. W., Kuehn, J. A., & Poole, S. W. (2010, September). Confidence: Analyzing performance with empirical probabilities. In *Cluster Computing Workshops and Posters (CLUSTER WORKSHOPS), 2010 IEEE International Conference on* (pp. 1-8). IEEE.

The submitted manuscript has been authored by a contractor of the U.S. Government under Contract No. DE-AC05-00OR22725. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.