SAND2017-8817C

Measuring Minimum Switch Port Metric Retrieval Time and Impact for Multi-Layer InfiniBand Fabrics

Michael Aguilar, Benjamin A. Allan, and Sergei Polevitzky
Sandia National Laboratories, Albuquerque, New Mexico 87123
Email: {mjaguil,baallan,sipolev}@sandia.gov

***Abstract*—In this work, we seek to gain an understanding of the InfiniBand network processing limitations that might exist in gathering performance metric information from InfiniBand switches using our new LDMS ibfabric sampler. The limitations studied consist of delays in gathering InfiniBand metric information from a specific switch device due to the switch's processor response delays or RDMA contention for network bandwidth.**

***Keywords*-InfiniBand, Switch Port, Metric, Retrieval Time**

I. Introduction

In HPC systems, tightly coupled parallel threads communicate information across InfiniBand (IB) networks using RDMA data exchanges [1]. IB network RDMA carries data for both applications and network performance metrics when the Lightweight Distributed Metric Service (LDMS) is used [3]. IB is provided with virtual traffic lanes to implement a type of quality of service for RDMA network traffic [5] [6]. By design, the highest priority traffic is given to both IB network infrastructure commands and performance metric information from within the IB subnet [7]. In this way, an IB network Subnet Manager can be notified when either specific switches or HCAs are unavailable for RDMA communications. There are 16 Virtual Lanes in an IB network. Virtual Lane 15 (VL15) is designated to be the virtual traffic lane for network performance metric data.

Large scale HPC IB network fabrics are hierarchical with several levels of switches. Congestion can degrade the performance of any or all applications whose traffic is transiting affected ports. Monitoring systems with detailed recording of network performance metrics can provide important insights into when, for how long, and possible reasons for IB network congestion. It is difficult to identify, with high confidence, correlations between events occurring within an IB fabric and unsynchronized low time resolution network performance data. It has been shown that synchronized collection of network performance data at intervals of 1 second can provide more useful information [2] [4] than the kinds of coarse grained unsynchronized collection performed by most other HPC monitoring systems.

The LDMS based *sysclassib* sampler collects detailed information about compute node HCA metrics. However, this sampler provides only compute node's views of the periphery of the fabric and still leaves the fabric internals opaque. To provide insight into the IB fabric's internal characteristics, we created the *ibfabric* sampler. Our goal is to acquire much more detailed performance data from IB network fabrics.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energys National Nuclear Security Administration under contract DE-NA0003525.

II. Approach

We created the LDMS based *ibfabric* sampler to gather performance metrics from each operational switch port within an InfiniBand fabric. We additionally wrote a *Port Delegator* (or *Delegator*) to automate scalable configuration of samplers for distributed monitoring of disjoint portions of an InfiniBand fabric while maintaining complete fabric port. The *Delegator* program reads the IB network fabric configuration information using the OFED *libnetdisc* library. It builds a list of available LIDs and ports to query. Once a list of LIDs and ports is assembled, the *Delegator* acts as the central task information service for the *ibfabric* samplers (Figure 1). The Delegator defines, and communicates a subset of the list for each *ibfabric* sampler, the set of IB switch ports and metrics for each sampler to monitor.

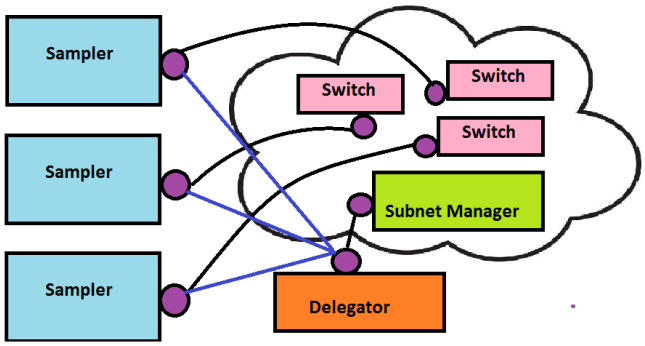

Fig. 1. LDMS ibfabric Sampler

For a sampler on a single node serially querying a list of IB ports, we define *spread* as the time interval between the start of the first port query made and the end of the last port query made. Synchronous acquisition of port counter data is subject to spread, to clock skew across nodes, and to jitter due to thread processing on the sampler host. Higher sampling frequencies, along with low time skews and spreads, of network performance counters enable higher resolution exposure of traffic and congestion details within an IB network fabric.

We seek to understand the limits and overheads of collecting system-wide IB port performance snapshots, including:

- the interference of applications with IB data collection.
- the interference of IB queries with applications.
- the worst case effects of redundant port data collection.
- the minimum spread time of port data collection.
- the impact of topology (hops, multiple targets) on spread.

Experimental details and results are provided in Sections IV and V respectively.

III. Related Works

Previous research has been performed, using simulations, to try to predict how often IB network congestion occurs and the severity of the congestion when computations are performed [8]. Current hardware and firmware for reducing contention on IB networks involves checking network drain interfaces for congestion and then reporting back to the network source interface. Delay injection is performed in some implementations of IB hardware [17]. Note that source initiated delay injection reduces the overall flow rate of data and can reduces computational performance.

Experimental work showing use of synchronized fine grained network performance counter data by user applications for making run-time partitioning decisions has shown promise in Cray Gemini based networks [9]. While this example uses a different network technology, similar methodologies could provide run-time insight, and potentially improved computational speed, in IB based networks.

The ability to scan a complete InfiniBand fabric, including switches, was made possible in the Ohio State University INAM monitoring system and in the INAM^2 monitoring system [10] [11]. Note that for both INAM and INAM^2, network performance metrics are obtained from Open Subnet Manager sweeps and the data is neither time synchronized nor fine-grained.

Ganglia monitoring systems enable end-point analysis of real-time IB network traffic [13]. However, Ganglia doesn't gather performance data from IB switch ports.

Nagios, another common monitoring system used in HPC systems, can be programmed to collect congestion metrics [14]. However, Nagios has a very wide and unpredictable window of sampling time and does not provide time stamps to help in later analysis of the metric data that was gathered.

The IB Subnet Manager has the ability to query for network congestion metrics during an interval fabric connection integrity sweep [12]. Each IB switch utilizes an internal processor to respond to performance metric requests made to it. During a Subnet Manager sweep, a Perf Manager program is tasked with retrieving a subset of performance metrics from each IB switch port. Depending upon the version of Perf Manager used, transmit wait counter data, which can be used as an indicator of congestion, may or may not be recorded. In addition, due to the dynamic and iterative procedure that the Subnet Manager utilizes to traverse the network fabric and associated time variation in performance metric data gathering, find-grained time stamping of IB network performance data is not performed.

The LDMS *ibfabric* sampler, that we created for gathering IB switch data, has the ability to gather and aggregate large amounts of synchronized high-fidelity IB network performance data. This sampler also provides a time-stamp to aid in IB network traffic analysis [15]. We could not find any previous studies that assess the impact that large volumes of IB performance counter queries might have on user application performance.

Our expectation was that application run-times might increase with large volumes of performance metric queries because the highest suality of service priority is given to performance metric data requests on IB. These port metric query requests are served by virtual lane VL15. The goal of our current work is to quantify any adverse impact that IB performance counter sampling might have on user applications and understand the limitations of sampling frequency in terms of switch capabilities.

IV. Experiments

A Sandia National Labs HPC testbed system with 16 compute nodes was used for our overhead analysis work. The IB fabric consists of a core switch and 2 leaf switches (See Figure 2) each with 36 QDR ports. Eight compute nodes are connected to each leaf switch. Each leaf switch is connected to the core switch with 2 IB links. For our experiments, samplers were available to be run from each compute node. Each query experiment utilized either 1 sampler or all 16 samplers. In both cases, every target port was queried by each active sampler.

Each data sample included 24 port performance data counters obtained through *MAD performance_query_via*. To gather spread time, we wrapped the MAD calls with *gettimeofday*. Each experiment was carried out eleven times to generate results. Our samplers were configured to request counters once every second.

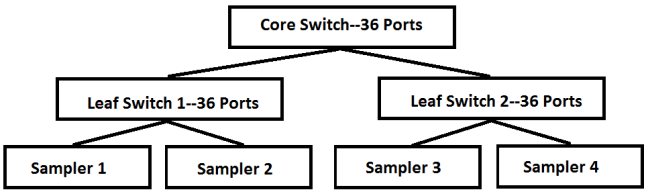

Fig. 2. Testbed Cluster Switch Layout

A. Application Loads

In order to provide long enough traffic data generation periods for each test performance metric query, very large iterations were chosen for both the InfiniBand bandwidth tests and the MPI RDMA InfiniBand traffic tests. For spread time and application interference experiments, our benchmarks include *ib_send_bw*, *ib_write_bw*, and *ib_read_bw* consisting of 1,000,000 64K messages and MPI tests.

MPI tests included *MPI All-Gather*, *MPI All-Reduce*, and *MPI Scatter*. The MPI tests were generated using SLURM,

OpenMPI, and Sandia National Labs MPIPerf [16]. MPI traffic was generated using *long long* C variable types, and direct RDMA messaging was chosen as the thread data exchange format. The matrix size used in our MPIPerf experiments was 1000 elements by 1000 elements. We exchanged matrix values 1 billion times.

B. Measuring Query Times

IB performance metric sets are collected from switches that are adjacent to the samplers (local), from samplers where queries require a traversal to the core switch (1 hop), and where queries that require a 2-level traversal to the leaf switch that is located the largest distance from the sampler (2 hops). For example, in Figure 3, ibfabric Sampler 1 is configured to request port performance metrics from Leaf Switch 2.

We wanted to measure how network topology and application load affect spread time. We configured samplers to gather performance metrics from 1, 2, or 3 switches. Each sampler would gather metrics for either 36 ports, 72 ports, or for 108 ports (3 switches by 36 ports each).

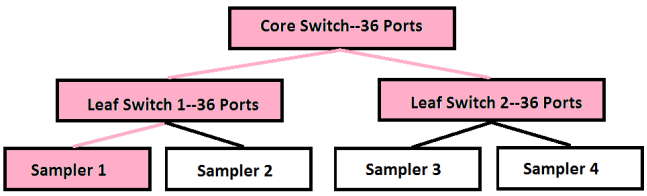

Fig. 3. Sampler Retrieving Metric Data From 2 Hops

C. Measuring Application Impact

We expected that there would be some impact on applications when the ibfabric samplers are making performance metric requests. We varied the sampler topology and the network switch sampling frequency to see if there was an effect on application network bandwidth. Data was taken at once a second (1 Hz), at 10 Hz, and at 100 Hz. We performed impact test experiments on application network bandwidth using the case of 1 sampler requesting performance metrics for all 3 switches and for 16 samplers for all 3 switches.

V. Results

For each observation set, we computed the mean, standard deviation, maximum, and minimum. For deployment purposes, worst cases are often more important than the standard deviation, so the error bars plotted in Figures 4 and 5 are extreme observations rather than the standard deviation.

Metric spread time: As seen in Figure 4, the largest delays in gathering IB switch performance metrics occured when all 16 ibfabric samplers queried the same 36 ports on the same core switch at the same time. The core switch is located 1 hop from the samplers over their leaf switches. Our worst result shows the spread time reaches just over 1 millisecond.

Our spread time experiments showed that the shortest performance counter query times are found for a single ibfabric sampler reaching its locally connected switch.

After adding application loads, we found small increases in the performance metric access times when we were accessing local ports. As might be expected, a single sampler accessing portions of the fabric has lower performance metric acquisition times than all samplers accessing the same switches in the fabric simultaneously.

Interestingly, the minimum observed sample spread times are around 50 microseconds independent of topology, and the minimum observed spread time is significantly higher for the ib_read_bw test at around 150 microseconds.

Data collection impact on applications: As seen in Figure 5, a single sampler gathering metrics from all 3 switches (108 ports) has minimal impact on the IB (IP-over-IB) bandwidth benchmarks. When we ran our experimental case with all the samplers simultaneously making requests to all 3 switches, the entire switch hierarchy, we found that sampling reduces bandwidth performance significantly at 100 Hz for both read and write benchmarks, but insignificantly at 1 Hz.

A naive t-test suggests the bandwidth increase seen with the 1 Hz and 10 Hz non-redundant samplings of the write benchmark is due to the sampling with a 99% confidence level.

VI. Conclusions & Preliminary Recommendations

Analysis conducted thus far suggests the following:

1) We can measure IB switch performance metrics with negligible bandwidth overhead and at frequencies up to 100Hz if the data collection is organized carefully.
2) Local and remote access have similar spread time when there is no redundancy.
3) Figures 4 and 5 both suggest that the presence of average metric collection activity may slightly improve performance of both the sampling process and co-located applications; explaining this effect requires further study.
4) Collecting with complete redundancy slows response to performance metric queries.

Recommendations Include:

1) Following conclusions 1 and 2, sampling of the switch hierarchy in a production cluster can be done from a few service nodes rather than spread across compute nodes.
2) Prefer sampling local switches over remote switches, when possible.

VII. Further Work

To maximize the amount of IB performance metric data available, we will:

1) Repeat this work on a larger fabric where core switch port counts number in the hundreds.
2) Perform experiments where switch ports are distributed to multiple samplers, non-redundantly.
3) Quantify how the ibfabric sampler affects MPI application and RDMA benchmark runtimes and latencies.
4) Seek a method for determining near-optimum organization of sampling for various switch topologies, under application interference and sampler process failures.
5) Publish our LDMS ibfabric plugin code.

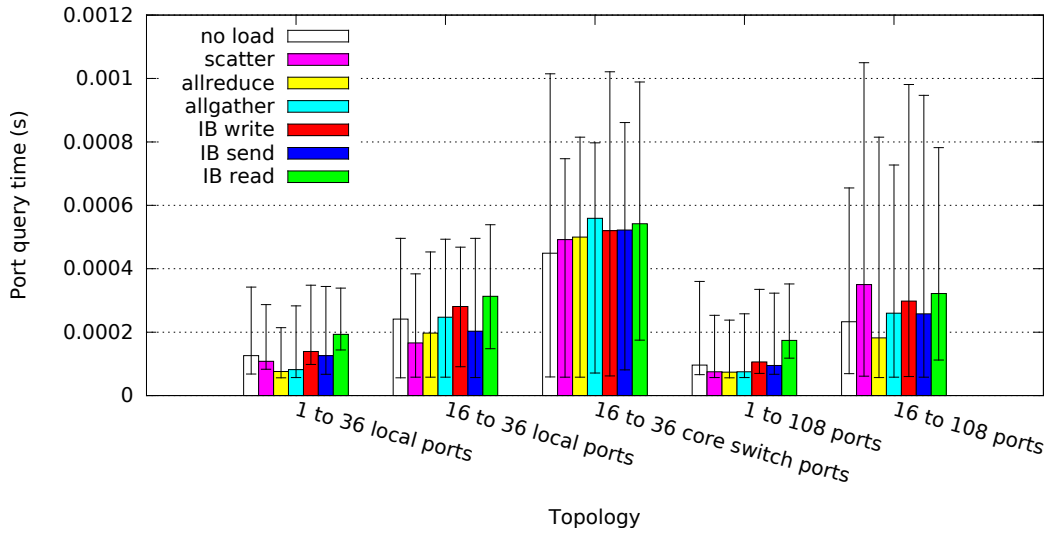

Fig. 4. Comparison of Average Sample Spread Time vs Load and Topology with Error Bars Indicating Extreme Spread Times Observed

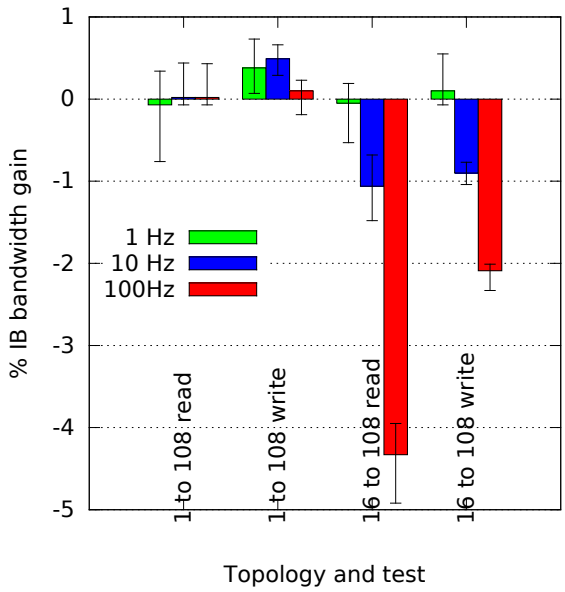

Fig. 5. IPoIB Overhead as Percentage of Average Unsampled Bandwidth vs Sampling frequency and Redundancy with Error Bars Indicating Extreme Individual Change Observed

REFERENCES

[1] Web article about InfiniBand use in HPC systems. [Online]. Available: *https://www.hpcwire.com/off-the-wire/infiniband-continues-its-position-as-preferred-top500-supercomputing-interconnect* .

[2] A. Agelastos, B. Allan, J. Brandt, A. Gentile, S. Lefantzi, S. Monk, J. Ogden, M. Rajan, and J. Stevenson *Continuous whole-system monitoring toward rapid understanding of production HPC applications and systems*, Journal of Parallel Computing, October, 2016.

[3] 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, T. Tucker *The Lightweight Distributed Metric Service: A Scalable Infrastructure for Continuous Monitoring of Large Scale Computing Systems and Applications*, SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, November 21, 2014.

[4] A. Deconinck, A. Bonnie, K. Kelly, S. Sanchez, C. Martin, M. Mason, J. Brandt, A. Gentile, B. Allan, A. Agelastos, M. Davis, and M. Berry, *Design and implementation of a scalable monitoring system for Trinity*, CUG2016 Proceedings–Cray User Group, June 13, 2016.

[5] J. Pelissier, *Providing Quality of Service over InfiniBand Architecture Fabrics*. Proceedings of the Eighth Symposium of Hot Interconnects, August, 2000.

[6] F. Alfaro, S. Sanchez, and D. Duato, *QoS in InfiniBand Subnetworks*. IEEE Transactions on Parallel and Distributed Systems, September, 2004.

[7] D, Crupnicoff, S. Das, and E. Zahavi, *Deploying Quality of Service and Congestion Control in InfiniBand-based Data Center Networks*, . Mellanox White Paper, October, 2005.

[8] G. Pfister, M. Gusat, W. Denzel, D. Craddock, N. Wi, W. Rooney, T. Engbersen, R. Luijten, R. Krishnamurthy, and J. Duato, *Solving Hot Spot Contention Using InfiniBand Architecture Congestion Control*, . Proceedings of the 11th International Symposium on High-Performance Computer Architecture, February, 2005.

[9] J. Brandt, K. Devine, and A. Gentile, *Demonstrating Improved Application Performance Using Dynamic Monitoring and Task Mapping*, . 2014 IEEE International Conference on Cluster Computing, September, 2014.

[10] N. Dandapanthula, H. Subramoni, J. Vienne, K. Kandalla, S. Sur, S. K. Panda, R. Brightwell, *INAM - A Scalable InfiniBand Network Analysis and Monitoring Tool*, . , 4th International Workshop on Productivity and Performance, August, 2011.

[11] H. Subramoni, A. M. Augustine, M. Arnold, J. Perkins, X. Lu, K. Hamidouche, D. K. Panda, *INAM2: InfiniBand Network Analysis and Monitoring with MPI*, 31st International Conference ISC High Performance, May, 2016.

[12] Open Fabrics Alliance (2017). [Online]. Available: *The OpenFabrics Enterprise Download*, . , https://www.openfabrics.org/downloads/OFED/ofed-4.8-daily/

[13] M. Massie, V. Vuksan, B. Nicholes, B. Li, R. Alexander, J. Buchbinder, F. Costa, A. Josephsen, P. Phaal, and D. Pocock, *Monitoring with Ganglia–Tracking Dynamic Host and Application Metrics at Scale*. Newton, Massachusetts: O'Reilly Media, 2012.

[14] D. Josephsen, *Building a Monitoring Infrastructure with Nagios*. Boston, Massachusetts: Pearson Education, Inc., 2007.

[15] M. Aguilar, J. Brandt, B. Allan, and D. Pase *Host Base Inifniband Network Fabric Monitoring*. presented at Open Fabrics Alliance 13th Annual Workshop, March 30, 2017.

[16] D. Pase *MPI Performance Benchmark*. private communication, MPI performance benchmark, August 19, 2016. dmpase@sandia.gov.

[17] Userguide explaining Delay Injection. [Online]. *Mellanox OFED for Linux User Manual*, Rev 2.1-1.0.0 . Mellanox Technologies, February 18, 2014, pp. 216-217.