SAND2020-9600PE

LDMS Monitoring of EDR InfiniBand Networks

Benjamin A. Allan[1], Michael Aguilar[1], Benjamin Schwaller[1], Steven Langer[2],
Sandia National Laboratories[1] — Lawrence Livermore National Laboratory[2]
Albuquerque, New Mexico[1] — Livermore, California[2]
baallan|mjaguil|bschwal@sandia.gov[1] — langer1@llnl.gov[2]

We introduce a new HPC system high-speed network fabric production monitoring tool, the *ibnet* sampler plugin for LDMS version 4. Large-scale testing of this tool is our work in progress.When deployed appropriately, the ibnet sampler plugin can provide extensive counter data, at frequencies up to 1 Hz. This allows the LDMS monitoring system to be useful for tracking the impact of new network features on production systems. We present preliminary results concerning reliability, performance impact, and usability of the sampler.

***Index Terms*—monitoring; high performance computing, networks, file servers**

I. Introduction

Being able to measure and understand how an internal high-speed network fabric (HSN) is behaving has potential benefits in improved cluster performance. It can become clear to system stakeholders when networks are suffering from abnormally high levels of traffic congestion, when a network is suffering from uneven and highly localized traffic, and how well user programs are taking advantage of available hardware resources such as hardware collective MPI acceleration.

To provide these insights, we have been working toward comprehensive, high frequency (1 Hz) collection and analysis of HPC system metrics and application metrics, including HSN traffic metrics from InfiniBand and OmniPath Management Datagram MAD) packets [1].

As in previous versions of our network fabric monitoring tools, we continue to utilize the RDMA stack tools Management Datagram packets (MAD) and User Management Datagram packets (UMAD). MAD packets are now able to provide metric information per Quality of Service service level, adaptive routing, and congestion information. We seek to provide system administrators and users with insights into systems and applications using these switch features.

LDMS monitoring samplers and monitoring daemons are required to be low overhead and integrate well into the overall HPC support ecosystem. The MAD calls allow us to maintain visibility into individual switch port and node performance even where the nodes and switches do not have an ldmsd daemon running. Other network tools typically sweep an entire network for information from a single server and only complete this operation over an interval of several seconds to minutes.

To extend our current monitoring ecosystem capabilities, we are developing a distributed tool, the *ibnet* LDMS sampler, which can sweep multiple interconnected InfiniBand networks, when appropriately deployed, at 1 Hz and provide global HSN metrics. This tool is an extension of our work [2], [3] to support a wider range of metrics, to lower the administrative burden of configuring the samplers on complex networks, and to update our previous recommendations derived for QDR InfiniBand .

II. Approach

We seek to gather synchronous data at a high enough frequency (1-10 Hz) that we can point to specific network hardware events and match them with application and file system events. Our new sampling tool is intended to enable us to analyze the impact of new hardware features such as quality-of-service (QoS), adaptive routing (AR), and the frame relay metrics forward and backward explicit congestion notification. To this end, we have created the ibnet sampler and administrative aids to simplify deployment. We are testing the sampler with applications and benchmarks on a medium scale system.

A. A new LDMS sampler

Our previous *opa2* (OmniPath) sampler and *fabric* (InfiniBand) samplers made single MAD calls per sample to individual components within a fabric.

We designed the new *ibnet* sampler to:

- Take samples following an administrator-supplied list of assigned LIDs and ports.
- Create one LDMS metric set per sampled port.
- Label the resulting data with host and interface names following an administrator-supplied list.
- Provide (optionally) any of the per-port data sets available via the perfquery utility.
- Provide (optionally) the time taken to collect data about each port and to decode the data about each port.
- Provide (optionally) the total time taken to sweep all the assigned ports.

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.
SAND No. INPUT SAND NUMBER HERE.
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344.

The new ibnet sampler can be obtained from the current OVIS-4 release of LDMS on github [4].

B. Administrative usability aids

To balance *ibnet* MAD call work loads among the data gathering nodes, the list of ports must be divided following (as much as possible) the guidelines suggested in our previous work [3] without substantial time-consuming work by an administrator. We created the program *ldms-gen-lidfile.sh* to generate a sampler configuration file for a single sampling node gathering data from the entire network at low frequencies. For high sampling frequencies, we created the *ldms-ibnet-sampler-gen* program to heuristically assign subsets of ports in a three level fat tree to members of a user-specified list of sampling nodes. The heuristics applied are:

1) All data from any single switch is handled serially by a single sampler instance, thus avoiding contention induced by multiple samplers for any embedded switch processor.
2) The switch(es) and adapters directly attached to a sampler's host are assigned to that sampler instance.
3) The switches in higher tiers of the fabric are assigned round-robin fashion among the least-loaded sampler instances.
4) The switches not yet assigned are distributed round-robin to the least-loaded sampler instances.
5) The adapters not yet assigned are assigned to the sampler instance which also gathers from the switch to which the adapter is attached.
6) When computing the hierarchical relationships among switches, links connected to SHArP ports are ignored.

These heuristics attempt to minimize the number of network hops needed to gather the data. Both programs consume the output of *ibnetdiscover -p*. Administrators may choose to give the same configuration to two samplers, providing controlled redundant data collection in the event of single node failure. The other information needed to configure the sampler is a *node name map*, a text file linking host and interface names to InfiniBand GUIDs. This mapping file already exists somewhere in most InfiniBand deployments.

C. Benchmark software and hardware features

We are updating our previous work on sampling large-scale InfiniBand networks. We seek to quantify the impact of Adaptive Routing, QoS, and hardware collectives using SHArP. We have added PF3D, HPL, ember, and IOR benchmarks. PF3D is a laser beam-plasma interaction simulation code. For the tests reported here, PF3D is configured to maximize inter-node message passing during its computational phase, thus increasing the chance of network resource competition among simultaneous runs making checkpoints or computations. The ember incast benchmark [5] tests the fabric's handling of high-volume many-to-one communications.

D. Benchmark system Stria

Our tests use Stria, a production 288 node ARM64 cluster with 1005 active ports that is a small version of the 2.3 petaflop Astra supercomputer [6]. The Stria cluster InfiniBand network design is a that of an Up/Down fat tree. The compute node rack switches are connected to six core switches through a 2x over-subscribed design while the IO server switches are connected through a full fat tree. Each compute node is designed with a Socket Direct configuration so that each socket can communicate directly to the InfiniBand network through the PCI bus. The Lustre 2.12 file system consists of two metadata servers and four object storage servers. Each server has an InfiniBand link to two separate switches. This Lustre instance is not shared with other clusters.

III. Preliminary Results

We monitored the Stria cluster network at 0.2 Hz during four days of production using a single compute node as the ibnet sampler host, and at one Hz during a non-production testing time using four or twelve reserved compute nodes (one or three per rack) as the sampler hosts. The production work load cannot be documented here, but includes many jobs that regularly stress the Lustre storage system. We planned the testing time to establish preliminary wall-time benchmarks of the impact of advanced network features and LDMS on a compute-heavy and check-point heavy application, PF3D, and on micro-benchmarks, ember, IOR and HPL.

A. Production days

During the four production days, we saw data points missed from the 0.2 Hz data only when nodes were down or during short periods when a job was accidentally scheduled on the sampling node. We timed the period to sweep the ports (which includes failed query time for down nodes). We conclude our sampler collects at reliable time cost based on Table I.

statistic	Sweep	MAD decode	Total(seconds)
minimum	1.10	0.02	1.11
average	1.20	0.02	1.22
maximum	1.75	0.02	1.77
std. dev.	0.072	0.0004	0.072

TABLE I
Duration of 64230 sampling operations using one instance.

B. Testing time

During our dedicated testing time, we ran various work load combinations to discover what we can see in the collected data and to gather data for planning future repetitive experiments to establish statistical significance of overhead measurements.

1) PF3D: We submitted four PF3D jobs at approximately the same time to an otherwise unoccupied system. The simulations were the same model with weak scaling applied to consume 8, 16, 32, or 128 nodes. They started within two minutes of each other. To eliminate any other possible effects of the four LDMS daemons used to sweep the fabric, the ldmsd service was not running on the any other nodes; thus we have no contemporaneous data from LDMS lustre client or server samplers. The three smaller jobs are configured to be check-point heavy, and the large job is configured to be compute heavy with some check pointing.

Figure 1 shows the total network bandwidth used by MPI and file output, as measured from the job node adapter ports. Figure 2 shows the write bandwidth achieved per core and the time of the writes. Figures 1 and 2 show the write rates for 8, 16, and 32 node jobs are heavily depressed in the window of 15-20 minutes due to the check-point of the 128 node job, but interestingly Figure 3 does not show very high average bandwidth in use for the same time span. Figure 3 shows the rates of data flowing into Lustre object store nodes, as measured from the switch ports attached to these nodes.

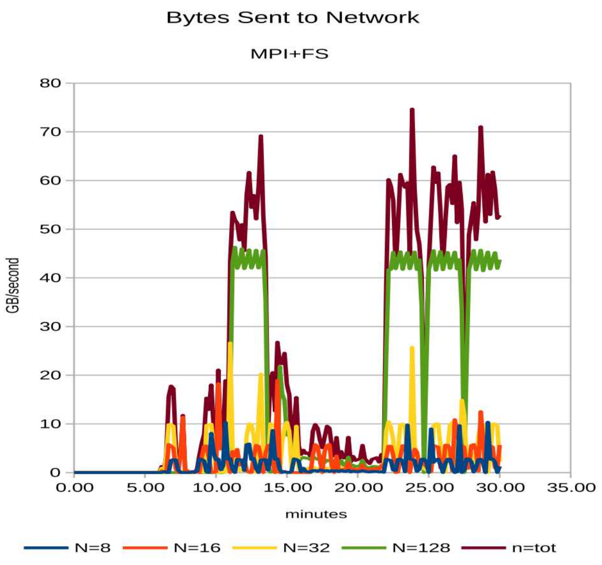

Fig. 1. Data injection rates summed by job with data binned to 10 second intervals. The top line is total injection rate.

Port query time: To support estimates of the time to complete a sweep of many nodes on a loaded EDR network, we collected the duration of the queries per port, where each query is composed of eight MAD calls for the subsets *"extended, xmtdisc, rcverr, flowctlcounters, vlxmitcounters, xmitcc, smplctl"*. Figure 4 shows the frequency of wall-clock duration for the combined query. The tail of the graph contains two data points at 0.0044, but the average query is approximately 0.0004 seconds per port.

2) *Port sweep time and ember:* The ember incast benchmark provides a heavy many-to-one network load that is not throttled by storage system limitations. We configured a 64 node job with all cores in use so half the messages must traverse the entire switch hierarchy and so congestion is expected. We configured twelve ldmsd samplers (three nodes per rack, two or less switches per sampler) for 1 Hz sampling, to verify we can collect reliably under load. The distribution of sweep times is shown in Figure 5. The compound MAD query used in this test is the subsets *extended, xmtdisc, rcverr, vlxmitcounters*. The benchmark is execution time is four hours

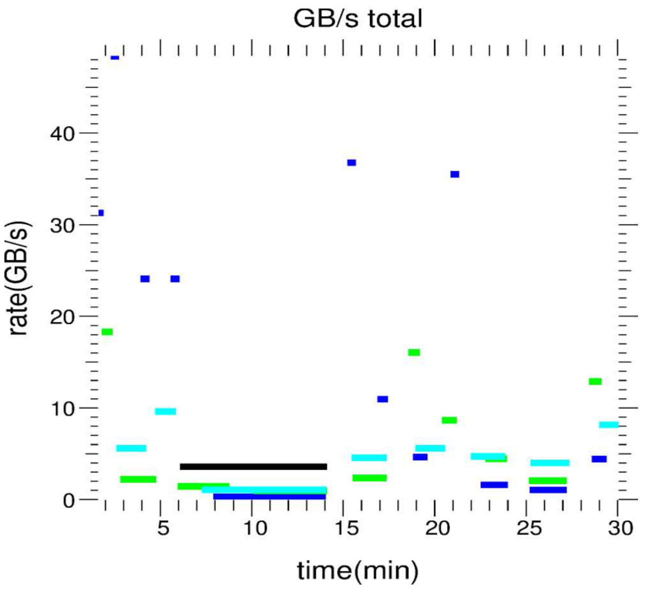

Fig. 2. Per-job write rates and intervals recorded by PF3D, colored by job node count (blue: 8, green: 16, cyan: 32), where each node hosts 56 MPI processes.

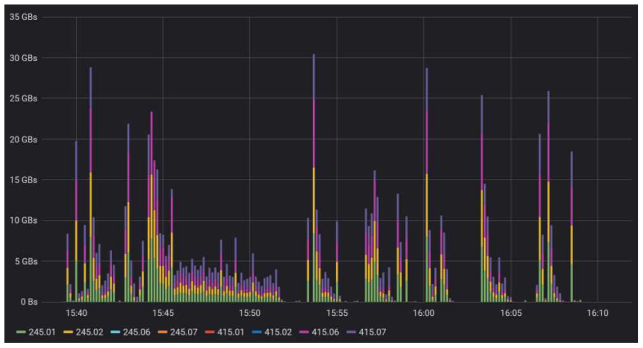

Fig. 3. Check-point data flows into Lustre, binned to 10 second intervals and colored by associated switch port. Total bar height is average total bandwidth for each bin.

in a seven hour measurement window; all the high outliers occurred while the benchmark is running.

3) *HPL and IOR:* We seek to show not only useful data but also that sampling does not contribute significant network overhead. We gathered data about the effects of enabling fabric sampling with the ibnet sampler during overlapping HPL benchmarks and IOR benchmarks using Lustre. Table II results suggest there is no obvious degradation due to LDMS sampling, though many further experiments are needed to provide statistical significance.

LDMS (0.2 Hz) on	HPL GFLOPS	IOR xfer read MB/s	IOR xfer write MB/s
no	734.1	47313	50658
yes	733.7	47272	52663

TABLE II
HPL AND IOR SIMULTANEOUS RUNS.

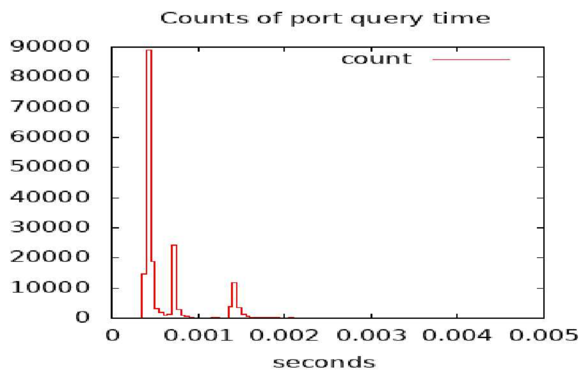

Fig. 4. 182,000 compound MAD queries in bins of width 0.00005 seconds

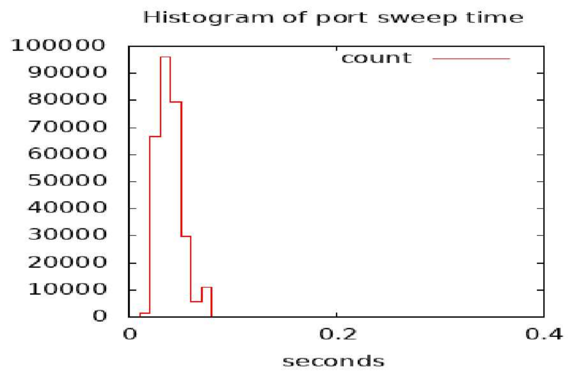

Fig. 5. 289642 port sweep durations in bins of width 0.01 second

IV. Lessons Learned

Network data collection appears easier than it is. Here we share a few of our early lessons learned.

1) To achieve 1 Hz data on EDR equipment, limit a single sampler to querying no more than 150 ports.
2) Store data with human decodable identifiers, not just LID and port designation, to aid downstream user interface operations.
3) When making a sequence of MAD calls to gather data about a single port and the first of them fails, skip the rest. Also limit the pma_query_via timeout to 5 milliseconds. The total of the timeouts can stretch the gathering time beyond the sampling interval.
4) Do not run the ibnet sampler on a node with heavy core usage. This may stretch the time to gather sampled port information beyond the length of the desired sampling interval. Conversely, an ibnet sampler handling many ports may interfere with an application on the same node unless the processes are restricted to separate cores.

V. Related Work

Metric gathering from high-speed network and correlation to running jobs is widely studied. However, none of the latest approaches is suitable to HPC centers such as ours, since we want a non-proprietary source of data at high frequency and sample times correlated with other data being gathered. Caldwell [7] recognized the scalability issues of fabric monitoring and developed a distributed collection solution with awareness of switch locality using the perfquery utility. The balanced-perfquery source code [8] based on OFED 3 has not been updated since 2015.

Currently, the INAM project of MVAPICH2 is remotely targeting compute node and switch ports and correlating events with MPI traffic and IO [9]. The INAM project is targeted to the InfiniBand fabric and GPUs. Unlike previous versions of INAM, information outside of a current MPI run can be gathered, as well. This is important because outside traffic congestion and events can contribute to a specific MPI batch job computational performance. Unfortunately, it appears that the metrics gathered only consists of the overall traffic performance data and not of traffic at different Service Levels. The MVAPICH-based approach is both intrusive and unfeasible in our constrained HPC environment, but may provide much more detailed data about applications by default.

At Los Alamos National Laboratory, changes have been made to their existing HSNMON monitoring suite to provide daemons to gather metrics on network performance and traffic. [10], [11] However, their current design requires access to underlying layers of vendor provided programs that make MAD calls at lower levels of the software stack.

In previous work performed at Sandia, LDMS InfiniBand fabric samplers were designed for older generations of InfiniBand . While LDMS gathered switch port metrics efficiently on large systems, metrics related to QoS and other recent developments were not collected, being unavailable on the fabric [1].

VI. Conclusions and Future Work

We have shown that 1 Hz sampling of all the port counters in an EDR fabric is technically possible with LDMS. We have shown and published tools to make configuring such collection administratively practical. We have evidence, not yet complete, that there is no significant interference in either direction between applications and the appropriately configured ibnet sampler. We have demonstrated that the sampler allows us to form basic insights into the work loads on various network regions even when we have no access to load source or load sink operating systems and application codes.

Our next steps include:

1) Scaling the work to clusters of approximately 2000 nodes.
2) Thorough quantification of the impact of ibnet sampling on CPU intensive and latency sensitive workloads in the same network.
3) To enable 10 Hz sampling, thorough quantification of the impact of ibnet sampling with no more than 72 ports sampled per instance when sharing a node with a CPU intensive or latency sensitive workload.
4) Validation of more MAD metric subsets, when we upgrade to a subnet manager that enables them.

References

[1] 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.

[2] M. Aguilar, J. Brandt, B. Allan, and D. Pase, "Host Based Infiniband Network Monitoring Performance." OpenFabrics Workshop 2017, 2017, OpenFabrics Workshop. [Online]. Available: https://www.openfabrics.org/images/eventpresos/2017presentations/306_InfiniBandMonitoring_MAguilar.pdf

[3] M. Aguilar, B. A. Allan, and S. Polevitzky, "Measuring minimum switch port metric retrieval time and impact for multi-layer infiniband fabrics," in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, 2017, pp. 805–808.

[4] (2020) Ovis version 4 source code. [Online]. Available: https://github.com/ovis-hpc/ovis/tree/OVIS-4

[5] (2020) READEME.mpi.incast. [Online]. Available: https://github.com/sstsimulator/ember/blob/master/README.MPI.incast

[6] M. Aguilar, K. Pedretti, S. Hammond, J. Laros, and A. Younge, "Evaluation of Hardware-Based MPI Acceleration on Astra." OpenFabrics Workshop 2017, 2019. [Online]. Available: https://www.openfabrics.org/wp-content/uploads/104_MAguilar.pdf

[7] B. Caldwell. (2015) Scripts for InfiniBand Monitoring. [Online]. Available: https://downloads.openfabrics.org/ofv/tools/Infinibandmonitoringtools.pdf

[8] (2015) balanced_perfquery repository. [Online]. Available: https://github.com/bacaldwell/balanced-perfquery

[9] H. Subramoni, P. Kousha, K. Ganesh, and D. K. Panda, "Visualize and Analyze Your Network Activities Using OSU INAM." OpenFabrics Alliance Workshop 2020, 2020. [Online]. Available: https://www.openfabrics.org/wp-content/uploads/2020-workshop-presentations/305.-OFA-Virtual-Workshop-2020-PPT-Template.pdf

[10] J. Martinez, "HPC Networking in the Real World." OpenFabrics Alliance Workshop 2017, 2019. [Online]. Available: https://www.openfabrics.org/wp-content/uploads/108_JMartinez.pdf

[11] B. Holman and J. Martinez, "High Speed Network Monitoring Enhancements." OpenFabrics Alliance Workshop 2017, 2019. [Online]. Available: https://www.openfabrics.org/wp-content/uploads/111_BHolman.pdf