A Scalable Network-Based Performance Analysis Tool for MPI on Large-Scale HPC Systems*

Hari Subramoni, Xiaoyi Lu, and Dhabaleswar K. Panda
Department of Computer Science and Engineering,
The Ohio State University
{subramon, luxi, panda}@cse.ohio-state.edu

***Abstract*—Studying the interaction among applications, MPI runtimes, and the fabric they run on is critical to understanding application performance. There exists no high-performance and scalable tool that enables understanding this interplay on modern multi-petaflop systems. Designing such a tool is non-trivial and involves multiple components including 1) data profiling/collection from network/MPI library, 2) storing and, 3) rendering the data. Furthermore, achieving this with minimal overhead and scalability is a challenging task. We take up this challenge and propose a high-performance and scalable network-based performance analysis tool for MPI libraries operating on modern networks like InfiniBand and Omni-Path. Our designs facilitate caching and pre-rendering, allowing a cluster with 6,541 nodes, 764 switches and, 16,893 network links renders in just 30 seconds — a 44X speed up over non-prerendered solutions. The proposed lock-free and optimized memory-backed storage design enables the tool to handle over a quarter million inserts into the database every 45 seconds (data from 27,504 switch ports and 104,656 MPI processes). The tool has been successfully deployed and validated on HPC systems at OSC and on Comet at SDSC.**

I. Introduction

Currently, administrators of HPC systems and developers of HPC applications/middleware rely on a plethora of tools to aid them in this interplay. There exists a variety of MPI level profiling tools (TAU [1], HPCToolkit [2], Intel VTune [3], IPM [4], mpiP [5]) that give insights into the MPI communication behavior of applications. However, they are unable to profile what happens in the communication fabric. On the other hand, several network level profiling and analysis tools exist that allow system administrators to analyze and inspect the network fabric (Nagios [6], Ganglia [7], Mellanox Fabric IT [8], BoxFish [9], Lightweight Distributed Metric Service (LDMS) [10]). However, they are unable to relate network activity to their triggering events in the MPI library. Thus, a fundamental gap exists in the current class of monitoring tools making the correlation of MPI events to network activity cumbersome at best.

The MPI forum [11] has been actively working on bridging this gap with the MPI_T [12] interface. The MPI_T interface allows tools to interact with the MPI library through control and performance variables. Researchers have already begun to take advantage of this interface to provide optimization and tuning hints to the users [13]. However, these tools have no knowledge about the underlying network fabrics and thus suffer from the same drawbacks as other existing MPI tools. Recently, by designing InfiniBand Network Analysis and Monitoring with MPI ($INAM^2$) [14, 15], we have made attempts to bridge this gap by developing a tool that is capable of allowing users to correlate MPI events and network activity. While $INAM^2$ is able to visualize and profile clusters of smaller sizes, several challenges must be addressed to enable scaling it to larger supercomputing systems. Further, $INAM^2$ is not capable of monitoring and visualizing intra-node and GPU-based communications.

*This research is supported in part by National Science Foundation grants CCF #1565414, CNS #1513120, ACI #1450440, and ACI #1664137.

A. Motivation and Contributions

As described in Section I, there is a clear need and unfortunate lack of a high-performance and scalable tool that is capable of correlating the MPI and network behavior for existing/emerging large HPC systems. Such a tool must: 1) be portable, easy to use/understand, 2) have high-performance and scalable visualization and storage techniques and, 3) be applicable to the different communication fabrics and high-performance MPI libraries that are likely to be used on existing/emerging large HPC systems.

This paper addresses the above broad challenge through the design of a high-performance and scalable network-based performance analysis tool for MPI that is capable of effectively visualizing and profiling networks of large scale supercomputing systems on top of OSU INAM [14, 15]. It explores intelligent designs that move the cost of rendering large networks out of the critical path and presents high performance designs and optimizations for database access that increase the data storage performance many fold. Novel abstraction interfaces are also proposed that enhances the applicability of the tool to multiple high-performance networks as well as MPI libraries. The major contributions of this paper are:

- Analyze and profile intra-node and GPU-based communication activities with many metrics at user specified granularity
- Design caching/pre-rendering based techniques significantly enhance network rendering performance
- Propose, design and develop lock-free designs and optimized memory-backed storage to enhance data storage performance
- Propose modular designs to generalize data gathering across multiple communication fabrics (PCIe, InfiniBand and Omni-Path)
- Enable fine-grained introspection of particular regions of application through "Points-of-Interest" feature

2168-9253/17 $31.00 © 2017 IEEE
DOI 10.1109/CLUSTER.2017.78

- Create a simple load generator to stress the various components of the tool — the data collection daemon, the database and the UI

Note that many features and capabilities described here are already present in OSU INAM available for free download at [14]. At the time of submission, over 600 downloads of OSU INAM has taken place from the project site. While we chose MVAPICH2 for implementing our designs, any MPI runtime can be enhanced to perform similar data collection and transmission.

II. Challenges and Proposed Design

Any tool that aims to visualize and profile large scale HPC systems must address some fundamental challenges — 1) it must be portable, easy to use / understand, 2) have high-performance and scalable rendering and storage techniques and 3) be applicable to the different intra-node and inter-node fabrics and high-performance MPI libraries that are likely to be used on large HPC systems. This section highlights the designs proposed to address these challenges.

Figure 1 depicts the overall architecture of the proposed design. It consists of two major design components: 1) Designing High-Performance and Scalable Data Storage and Visualization Techniques for Large Scale HPC Clusters and 2) Design of Portable and High-Performance Data Collection Substrate. Component #1 has two sub-components — A) Enhancing Data Storage Performance through Lock-Free Designs and Scalable Schema and B) Optimizing Network Rendering through Caching/Pre-rendering based Designs. Component #2, on the other hand, has three sub-components — A) Designing MPI Library Agnostic Plugin, B) Design of Network and MPI Load Generator and, C) Designing a Generalized Network Abstraction Interface for Multiple Communication Fabrics. The dotted lines between the different sub-components indicate that the entities are logically distinct and can be run on the same or on different physical hosts.

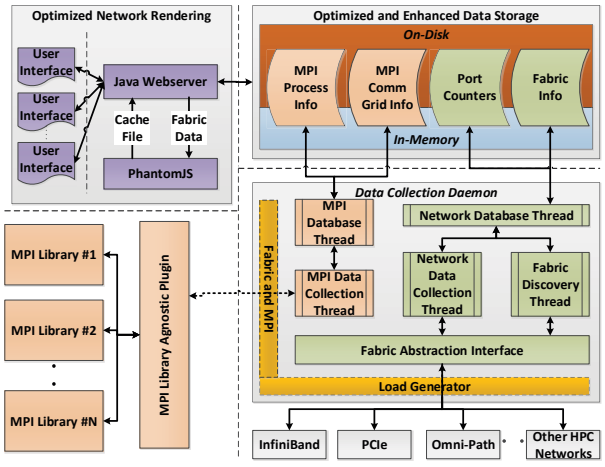


Fig. 1. Overall framework of proposed design

We redesign the database scheme so that the information is stored in a single text field instead of as separate entries. This eliminates the multiplicative effect during insertions thereby significantly improving the insertion performance. Figure 2 depicts the optimizations made to database schema in this regard. We also optimize MySQL performance and explored memory backed storage techniques to significantly enhance the performance of database operations.

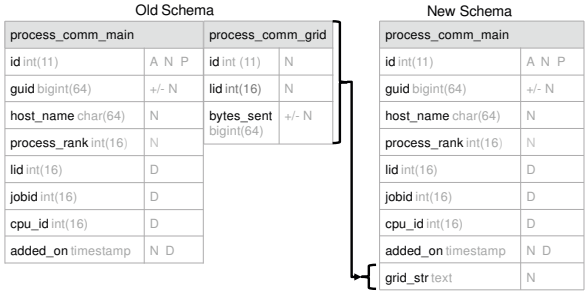
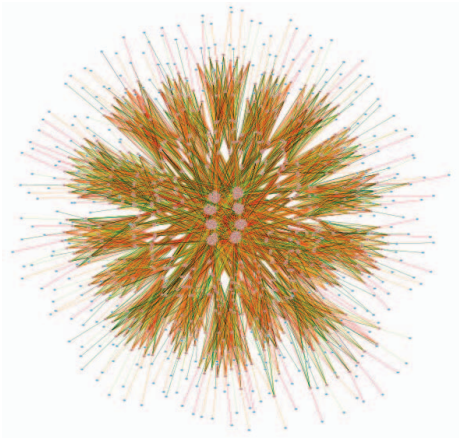


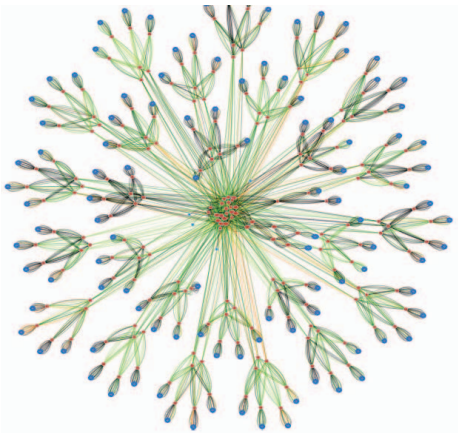
Fig. 2. Optimizations to database schema

We utilize PhantomJS [16], a headless browser that can be run on the server side, to pre-render the network when the web server is launched for the first time. The result is stored in a "cache file" that can be directly rendered on the client-side browser. Although PhantomJS execution adds to the web application deployment time, it removes the overhead of stabilization from the critical path thereby significantly improving the user experience. To avoid showing a stale view due to the changes in the network topology or individual elements, the pre-rendered view is updated in the background each time the fabric is scanned. Figure 3 shows the depiction of the full network views of Stampede@TACC, Comet@SDSC, Gordon@SDSC and various HPC systems at the Ohio Supercomputing Center (OSC) by the proposed tool. Note that the number of nodes and links in parenthesis represents the actual number of nodes that were active when the results were taken and can thus vary slightly from the advertised numbers on the websites of these systems.

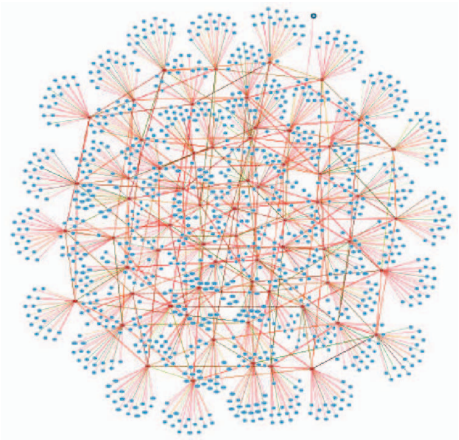
A high-performance and portable data collection substrate is required for any tool that targets large scale HPC clusters which can use multiple MPI libraries as well as different intra-/inter-node communication fabrics. We envision that the plugin will query the MPI library for the list of supported MPI_T PVARS to determine if an MPI library is which it already knows the mapping. Another alternative is for the system administrator to define a configuration file that provides the mapping of PVARs for libraries that the plugin currently does not support. We design a network and MPI load generator module that allows users of their tool to stress test and evaluate it in a contained, single node environment. The module feeds data about the MPI processes, fabric (topology and counters) to the data collection daemon so that it can visualize the network with simulated performance counter packets for all switch ports. A group of general abstractions and interfaces is proposed to express the common properties of different networks. Network-specific data collection functions are implemented under the general interface. The collected data include static properties of nodes and links (GUID, type, bandwidth, etc.), as



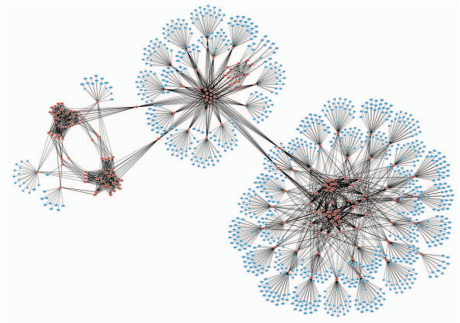
(a) Stampede@TACC — Clustered (6,541 nodes, 764 switches, 16,893 network links)



(b) Comet@SDSC — Clustered (1,879 nodes, 212 switches, 4,377 network links)



(c) Gordon@SDSC — Fully expanded (1,081 nodes, 64 switches, 1,657 network links)



(d) OSC Clusters — Fully expanded (1,400 nodes, 184 switches, 3,027 network links)

Fig. 3. Full network view of various supercomputing systems as depicted by proposed tool

well as dynamic runtime indices of each port (transmit/receive counter, error counter, etc.), fabric connectivity information and information about routes between different hosts.

III. Discussion on Features and its Impact

We highlight how the new features introduced in Section II enables understanding the interaction between the application and the communication fabric in this section. These features are not available in $INAM^2$ [15]. Note that the performance and scalability improvements gained as a result of design introduced in this paper makes all existing features of OSU INAM applicable to large scale HPC systems (a full list of features can be found at [14]).

A. Addressing Security and Trustworthiness in the Webserver

On typical HPC systems, a regular user is only allowed to access his/her own job. However, the HPC center staff need to have elevated privileges to access information pertaining to all users as well as low-level details of the network. We propose to incorporate such "user-based filtering" capabilities in the Java web server. The web server will use the accounting facilities in state-of-the-art job schedulers like SLURM [17] to distinguish between different users and present each user with an appropriate view of jobs/system.

B. Analyzing and Understanding Intra-Node Communication

The advent of multi-/many-core processes has led to very complex intra-node topologies with varying communication costs. Understanding the communication happening inside a node can help application developers immensely. For instance, the capability to monitor and visualize intra-node communication enables understanding the communication pattern of processes and allows application developers to place frequently communicating processes on the same socket / NUMA node using the process to core binding features that most modern MPI libraries provide.

C. Analyzing and Understanding GPU-Based Communication

The emergence of accelerators such as NVIDIA Graphics Processing Units (GPUs) is changing the landscape of supercomputing systems. GPUs, being PCI Express (PCIe) devices, have their own memory space and require data to be transferred to their memory through specific mechanisms prior computation. There are multiple communication paths (up to 11 [18]) available with these accelerator based architectures

(host-to-host, host-to-device, host-to-remote-device etc). This variety, in possible communication paths, makes the jobs of identifying and analyzing these challenging for normal users. However, with the proposed new features, users can easily visualize and identify which are the more frequently used communication paths. This, combined with the fact that the tool can correlate and overlay MPI level information on top of low level information makes this unique in its capability.

IV. Experimental Results

The load time of the user interface and the database performance heavily dependent on the characteristics of the underlying hardware and software. As different supercomputing systems use different generations of hardware/software (like CPUs, SSDs, HDDs, versions OS etc.) with varying performance characteristics, it will be unfair to compare the results obtained on one system with those obtained on another. Thus, in the interest of achieving a fair comparison, all experiments were run on the hardware platform described in Section IV-A. The proposed network and MPI load generator is used to simulate the load that four different HPC systems can potentially put on the proposed tool. Note that the tool has been deployed on the HPC systems at OSC and Comet at SDSC. It has also been verified that the results seen on the actual systems are in line with the performance observed locally using the load generator.

A. Experimental Setup

Each node in our 184 experimental cluster is equipped with Intel Westmere series of processors using Xeon dual socket, quad-core processors operating at 2.67 GHz with 12 GB RAM, MT26428 QDR ConnectX-2 HCAs (32 Gbps data rate) with PCI-Ex Gen2 interfaces. The operating system used is Red Hat Enterprise Linux Server release 6.7 (Santiago), with kernel version 2.6.32-431.el6 and Mellanox OFED version 2.2-1.0.1. We used MySQL v5.1.73 and MemSQL v5.0.3 for the evaluations. The version of PhantomJS used was 2.0.0. Table I highlights the salient points of the different HPC systems used as testbeds for evaluation. As mentioned above, we used a single node on this cluster to simulate the MPI and network load generated when the HPC systems described in Table I are "fully-loaded".

TABLE I
DETAILS OF VARIOUS HPC SYSTEMS

Cluster	Number of Nodes	Number of Switches	Number of Ports	Number of Links	Network Topology
Gordon	1,081	64	2,304	1,657	3D Torus
OSC	1,400	184	5,628	3,027	Composite Fat-Tree
Comet	1,879	212	7,636	4,377	Hybrid Fat-Tree
Stampede	6,541	764	27,504	16,893	Partial Fat-Tree

Table II indicates the number of records of each type and the frequency of insertion of the records for different HPC systems. Three different types of records are inserted — a) performance counters obtained from port counters on the switch (Port Counters), b) process specific information being sent by each MPI process (MPI Process Info) and, c) process specific communication grid information being sent by each MPI process (MPI Comm Grid Info). Note that the number of "Port Counters" records inserted is a function of the number of switch ports on the system while the number of "MPI Process Info" and "MPI Comm Grid Info" records are a function of the number of processes (which can loosely be linked to the number of cores assuming no over subscription) on the system.

TABLE II
NUMBER OF RECORDS INSERTED OF DIFFERENT CATEGORIES FOR VARIOUS HPC SYSTEMS RUNNING AT FULL LOAD

HPC Cluster	Number of Records Inserted / Frequency of Insertion (seconds)		
	Port Counters	MPI Process Info	MPI Comm Grid Info
Gordon	4,608 / 30	17,296 / 30	17,296 / 30
OSC	11,256 / 30	22,400 / 30	22,400 / 30
Comet	15,272 / 30	30,064 / 30	30,064 / 30
Stampede	55,008 / 30	104,656 / 45	104,656 / 45

B. Impact of Profiling on Applications Kernels

Figure 4 compares the performance of the version of the MPI runtime with support for MPI level data collection with one which does not have the support. As we can see, at the application level, there is little to no impact on the performance due to the addition of the data collection and reporting. These are encouraging trends which positively advocate the use of such tools for end applications on modern HPC systems.

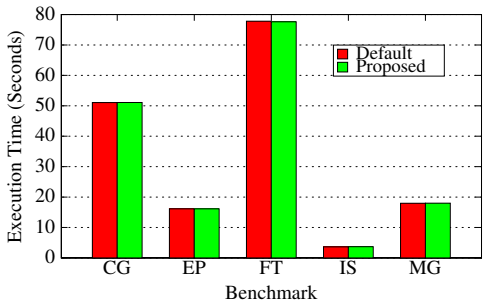


Fig. 4. Performance of class D NAS parallel benchmarks at 512 processes

V. Conclusion and Future Work

The design of a high-performance and scalable network-based performance analysis tool for MPI built on top of OSU INAM that is able to visualize and profile networks of large scale supercomputing systems with high-performance and scalability was presented in this paper. The proposed designs enabled the ability to monitor and visualize intra-node, inter-node and GPU-based MPI communication for multiple communication fabrics with fine granularity on large HPC clusters for multiple MPI libraries. The proposed caching/pre-rendering based designs were able to deliver a speedup of 44X over the over non pre-rendered / caching enabled solution in depicting a cluster with 6,541 nodes, 764 switches and, 16,893 network links. The lock-free and optimized memory-backed storage design allowed OSU INAM to to handle the data generated by a) querying 27,504 switch ports at a frequency of once every 30 seconds and b) 104,656 MPI processes at a frequency of once every 45 seconds — over quarter million inserts into the database every 45 seconds. The tool has been deployed successfully on HPC systems at OSC and on Comet at SDSC.

REFERENCES

[1] A. D. Malony and S. Shende, "Performance Technology for Complex Parallel and Distributed Systems," in *Proc. DAPSYS 2000, G. Kotsis and P. Kacsuk (Eds)*, 2000, pp. 37–46.

[2] HPCToolkit, http://hpctoolkit.org/.

[3] Intel Corporation, "Intel VTune Amplifier," https://software.intel.com/en-us/intel-vtune-amplifier-xe.

[4] "Integrated Performance Monitoring (IPM)," http://ipm-hpc.sourceforge.net/.

[5] "mpiP: Lightweight, Scalable MPI Profiling," http://www.llnl.gov/CASC/mpip/.

[6] "Nagios," http://www.nagios.org/.

[7] "Ganglia Cluster Management System," http://ganglia.sourceforge.net/.

[8] Mellanox Integrated Switch Management Solution, http://www.mellanox.com/page/ib_fabricit_efm_management.

[9] Lawrence Livermore National Laboratory, "PAVE: Performance Analysis and Visualization at Exascale," https://computation.llnl.gov/project/performance-analysis-through-visualization/software.php.

[10] 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, "The Lightweight Distributed Metric Service: A Scalable Infrastructure for Continuous Monitoring of Large Scale Computing Systems and Applications," ser. SC '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 154–165. [Online]. Available: http://dx.doi.org/10.1109/SC.2014.18

[11] MPI Working Group, "Message Passing Interface Forum," http://www.mpi-forum.org/.

[12] Martin Schulz, "MPIT: A New Interface for Performance Tools in MPI 3," `http://cscads.rice.edu/workshops/summer-2010/slides/performance-tools/2010-08-cscads-mpit.pdf`.

[13] E. Gallardo, J. Vienne, L. Fialho, P. Teller and J. Browne, "MPI Advisor: A Minimal Overhead MPI Performance Tuning Tool," in *EuroMPI 2015*, 2015.

[14] OSU InfiniBand Network Analysis and Monitoring Tool, http://mvapich.cse.ohio-state.edu/tools/osu-inam/.

[15] H. Subramoni, A. M. Augustine, M. Arnold, J. Perkins, X. Lu, K. Hamidouche, and D. K. Panda, "INAM$\hat{2}$: InfiniBand Network Analysis and Monitoring with MPI," 2016.

[16] Ariya Hidayat, "PhantomJS," 2010. [Online]. Available: http://phantomjs.org/

[17] Simple Linux Utility for Resource Management (SLURM), http://www.llnl.gov/linux/slurm/.

[18] S. Potluri, D. Bureddy, K. Hamidouche, A. Venkatesh, K. Kandalla, H. Subramoni, and D. K. D. Panda, "MVAPICH-PRISM: A Proxy-based Communication Framework Using InfiniBand and SCIF for Intel MIC Clusters," ser. SC '13. New York, NY, USA: ACM, 2013, pp. 54:1–54:11. [Online]. Available: http://doi.acm.org/10.1145/2503210.2503288