A Scalable InfiniBand Network Topology-Aware Performance Analysis Tool for MPI*

Hari Subramoni, Jerome Vienne, and Dhabaleswar K. (DK) Panda

Department of Computer Science and Engineering,
The Ohio State University
{subramon,viennej,panda}@cse.ohio-state.edu

Abstract. Over the last decade, InfiniBand (IB) has become an increasingly popular interconnect for deploying modern supercomputing systems. As supercomputing systems grow in size and scale, the impact of IB network topology on the performance of high performance computing (HPC) applications also increase. Depending on the kind of network (FAT Tree, Tori, or Mesh), the number of network hops involved in data transfer varies. No tool currently exists that allows users of such large-scale clusters to analyze and visualize the communication pattern of HPC applications in a network topology-aware manner. In this paper, we take up this challenge and design a scalable, low-overhead InfiniBand Network Topology-Aware Performance Analysis Tool for MPI - *INTAP-MPI*. INTAP-MPI allows users to analyze and visualize the communication pattern of HPC applications on any IB network (FAT Tree, Tori, or Mesh). We integrate INTAP-MPI into the MVAPICH2 MPI library, allowing users of HPC clusters to seamlessly use it for analyzing their applications. Our experimental analysis shows that the INTAP-MPI is able to profile and visualize the communication pattern of applications with very low memory and performance overhead at scale.

1 Introduction

Across scientific domains, application scientists are constantly looking to push the envelope by running large-scale, parallel jobs on supercomputing systems. Message Passing Interface (MPI) [12] is a very popular programming model being used to write such large-scale parallel programs. Supercomputing systems are currently comprised of thousands of compute nodes based on modern multi-core architectures. Interconnection networks have also rapidly evolved to offer low latencies and high bandwidths to meet the communication requirements of parallel applications. InfiniBand has emerged as a popular high performance network interconnect and is being used increasingly to deploy some of the top supercomputing installations around the world. Most supercomputing systems consist of racks of compute nodes and use complex network architectures comprised of many levels of leaf and spine switches. Different factors can affect the performance of applications utilizing such IB clusters. One of these factors is the number of hops the packet has to traverse in the IB network. In [9], Hoefler et al. showed

* This research is supported in part by U.S. Department of Energy grant #DE-FC02-06ER25755; National Science Foundation grants #CCF-0916302, #OCI-0926691 and #CCF-0937842.

I. Caragiannis et al. (Eds.): Euro-Par 2012 Workshops, LNCS 7640, pp. 439–450, 2013.

network topology can have a significant impact on the performance of scientific parallel applications. In our previous work, we designed *INAM* [13] to understand the network communication pattern of InfiniBand. However, it still lacks the ability to profile and visualize the MPI communication pattern in a network topology-aware manner. As IB clusters grow in size and scale, it becomes critical for the users of HPC installations to clearly understand how an HPC application interacts with the underlying IB network and the impact it can have on the performance of the application.

No tool currently exists that allows users of such large scale clusters to analyze and to visualize the communication pattern of their MPI based HPC applications in a network topology-aware manner. Most contemporary MPI profiling tools for IB clusters also have an overhead attached to them. This prevents users from deploying such profilers on large-scale production runs intent on achieving the best possible performance from their codes. These lead us to the following broad challenge - ***Can a network topology-aware, scalable, low-overhead profiler be designed for IB clusters that is capable of depicting the communication pattern of high performance MPI applications?***

In this paper we take up this challenge and design *INTAP-MPI* - a network topology-aware, scalable, low-overhead MPI profiler that allows users to analyze and visualize the communication pattern of MPI based HPC applications. INTAP-MPI gives the flexibility to profile the MPI application for either the entire duration of application execution (by means of environment variables) or for specific sections of the application (by means of Unix signals). We integrate INTAP-MPI into the MVAPICH2 MPI library [2], allowing users to seamlessly use it for analyzing their applications. Our experimental analysis shows that INTAP-MPI is able to profile and visualize the communication pattern of applications with very low memory and performance overhead at scale.

We organize the remainder of this paper as follows. We present the broad vision and possible use cases for INTAP-MPI in Section 2. Section 3 gives a brief overview of InfiniBand, IB network topology discovery and MPI over IB. In Section 4 we present the framework and design of INTAP-MPI. We evaluate and analyze the correctness and performance of INTAP-MPI in various scenarios in Section 5, describe the currently available related tools in Section 6, and summarize the conclusions and possible future work in Section 7.

2 Vision

As seen in Section 1, it is critical for users of HPC installations to clearly understand the impact IB network topology can have on the performance of HPC applications. Figure 1 illustrates how INTAP-MPI can be utilized by different users of a supercomputing installation. For instance, an MPI developer, can utilize INTAP-MPI to analyze how various MPI level collective algorithms are impacted by network topology and devise new and more network topology-aware algorithms that might prove more tolerant to network congestion. A scheduler developer / system administrator might attempt to classify the applications based on their communication pattern and make modifications to the job launching infrastructure so as to place the various ranks in a network topology-aware fashion. INTAP-MPI can also be used by scientists / application developers to re-write their applications with an eye on the underlying network topology.

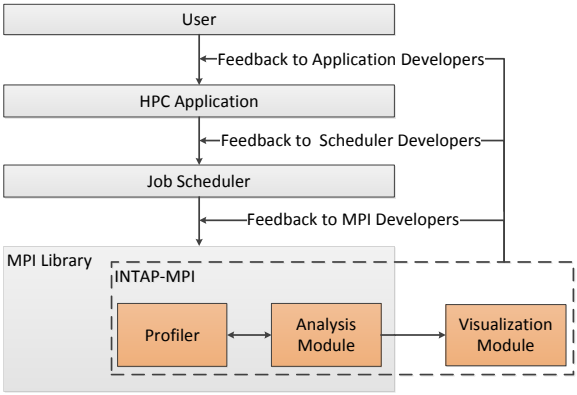


Fig. 1. Envisioned use cases for INTAP-MPI

3 Background

3.1 InfiniBand

InfiniBand is a very popular switched interconnect standard being used by almost 41% of the Top500 Supercomputing systems [18]. InfiniBand Architecture (IBA) [10] defines a switched network fabric for interconnecting processing nodes and I/O nodes, using a queue-based model. InfiniBand standard does not define a specific network topology or routing algorithm and provides the users with an option to choose as per their requirements. IB also proposes link layer Virtual Lanes (VL) that allows the physical link to be split into several virtual links, each with their specific buffers and flow control mechanisms. This possibility allows the creation of virtual networks over the physical topology. However, current generation InfiniBand interfaces do not offer performance counters for different virtual lanes.

3.2 InfiniBand Network Topology Detection Service

InfiniBand network topology data is not available in a mode that is easily used by other programs. The physical connections between entities in the fabric are discoverable with the `ibnetdiscover` utility from the OFED distribution [3]. The Logical routing data is available by querying each switch in turn and dumping its Linear Forwarding Table (LFT) with the `ibroute` utility from OFED. We leverage the basic topology detection service proposed in [8], but have since extended the method to query routing information directly within the OpenSM [15] subnet manager using a plugin interface.

3.3 MPI

Message Passing Interface (MPI) [12], is one of the most popular programming models for writing parallel applications in cluster computing area. MPI libraries provide

basic communication support for a parallel computing job. In particular, several convenient point-to-point and collective communication operations are provided. High performance MPI implementations are closely tied to the underlying network dynamics and try to leverage the best communication performance on the given interconnect. In this paper, we use modified MVAPICH2 [2] based on the 1.8 release for our evaluations. However, our observations in this context are quite general and they should be applicable to other high performance MPI libraries as well.

4 Design of Network Topology-Aware Performance Analysis Tool for MPI

The overall architecture of InfiniBand Network Topology-Aware Performance Analysis Tool for MPI (INTAP-MPI) is presented in Figure 2. It consists of two major design components: (1) light weight profiling interface and (2) topology-aware analysis module. We will go into the details of each in the following sections.

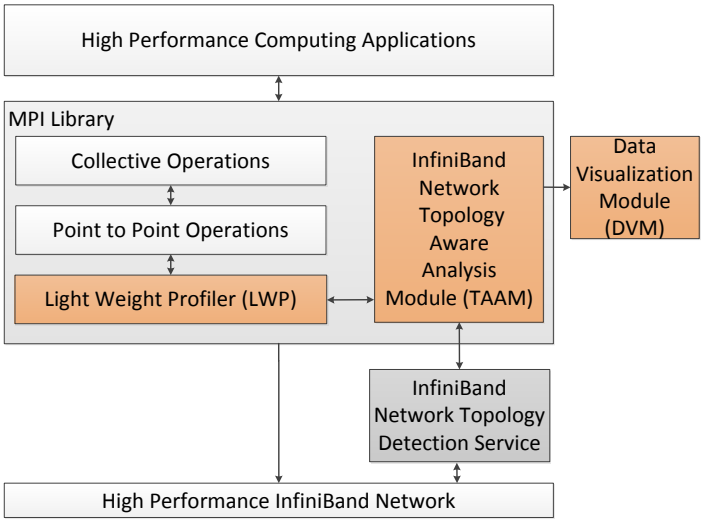


Fig. 2. Overall framework

4.1 Design of Topology-Aware Analysis Module

The Topology-Aware Analysis Module (TAAM) forms the core of INTAP-MPI. TAAM initiates and coordinates all profiling and analysis in INTAP-MPI. The user is responsible for initiating all profiling activities. This can be done either for the entire duration of application execution (by means of environment variables) or for specific sections of the application (by means of Unix signals). TAAM performs the following activities on receiving the user request to initialize the profiling: (1) informs the Light Weight

Profiling interface (LWP) (described in Section 4.2) to initiate logging the intra-node and inter-node communication inside the MPI library and, (2) queries the InfiniBand Network Topology Detection service and identifies the layout of the various processes on the InfiniBand network. Depending on the users choice (through environment variables), TAAM can either request LWP to profile either the number of messages or the volume of messages.

Once TAAM performs these activities, it remains idle until either the application terminates (calls MPI_Finalize) or receives a signal from the user requesting INTAP-MPI to finalize the profiling. On receiving the request from the user to finalize the profiling, TAAM informs the LWP to stop logging the messages. Once the LWP has stopped logging messages and handed it over to the TAAM on the local process, TAAM in *rank 0* of the MPI job gathers the communication profile from each rank. It then uses the IB network topology information obtained earlier to classify the messages logged by the LWP based on the number of hops they had to traverse. TAAM depends on the InfiniBand Network Topology Detection Service described in Section 3.2 to perform this classification. Currently, the service is only capable of distinguishing between processes based on the number of inter-node network hops. As part of future work, we plan to add capabilities to the service that will enable it to account for intra-node NUMA topologies as well. TAAM can perform this classification various granularities - process, compute node and switch blade, based on the users choice. This information is stored in a file which is later parsed by the data visualization module described in the next section.

Data Visualization Module: As the name suggests, the Data Visualization Module (DVM) visualizes the network topology-aware communication matrix generated by TAAM using standard Unix tools like *gnuplot* [1]. It generates two kinds of graphs - (1) a stacked histogram showing the splitup of physical communication based on the number of network hops and (2) a heatmap depicting the relative volume of various types of message transfers (intra-node, inter-node-1-hop, inter-node-3-hops, inter-node-5-hops, etc). Depending on the granularity of the data generated by TAAM (process, compute node or switch blade level), the graphs generated by the DVM will also depict different patterns. This information can then be used by HPC application developers, MPI library developers as well as system administrators to decide upon the best rank / task layout for a given job. DVM also provides scripts that allow users to perform post-run comparison and analysis of the communication pattern of multiple jobs. The scripts summarize the overall communication pattern of the jobs into a single graph allowing users to reason about the performance of various jobs from a network perspective.

4.2 Design of Light Weight Profiler

The Light Weight Profiler (LWP) is responsible for logging all intra-node and inter-node communication performed by the MPI library. We integrate the LWP into the lowest communication layers of the MVAPICH2 MPI library allowing it to capture the actual communication behavior, including any fragmentation done by the MPI library for load balancing purposes. On receiving the signal from TAAM to start message logging, LWP at each process allocates an array whose size is determined by the level of profiling granularity chosen by the user. If the user wants to view the communication pattern

between each pair of processes, LWP allocates an array of size $\mathcal{O}(N_{\text{processes}})$ to log the various messages issued by the process to its peers in the MPI job. If the user desires to view the communication pattern at a compute node level granularity, LWP allocates an array of size $\mathcal{O}(N_{\text{nodes}})$. LWP continues to log the messages until it receives the signal from TAAM to stop profiling. On receiving this message, it stops profiling and transfers the collected data to the TAAM on the local process which will then be gathered by the TAAM on *rank 0* of the MPI job, as described in Section 4.1.

5 Experimental Results

We describe the results of the various experiments carried out for this paper in this section.

5.1 Experimental Setup

Multiple high performance computing systems were used to obtain the results for this paper:

Ranger: Ranger is comprised of 3,936 16-way SMP compute nodes providing a total of 62,976 compute cores. Each core operates at 2.3 GHz and has 32 GB of memory with an independent memory controller per socket. Ranger has two 3,456 port SDR Sun InfiniBand Datacenter switches at its core. The interconnect topology is a 7-stage, full-CLOS fat tree.

Hyperion: This is a 1,400-core testbed where each node has eight Intel Xeon (E5640) cores (Nehalem) running at 2.53 Ghz with 12 MB L3 cache. Each node has 12 GB of memory and an MT26428 QDR ConnectX-2 HCA. It has a 171-port Mellanox QDR switch, with 11 leafs, each having 16 ports. The switches form a partial FAT tree. Each node is connected to the switch using one QDR link. Although Hyperion does have additional compute cores, they use the Harpertown range of CPU's. Hence we restrict ourselves to the nodes described above.

The experiments described in Sections 5.2 and 5.3 were obtained on the Ranger supercomputing system. However, due to lack of system resources, we ran the larger scale jobs studying the scalability of INTAP-MPI described in Sections 5.4 and 5.5 on Hyperion.

In Sections 5.2 and 5.3, we analyze the performance of two types of communication patterns, collective and point-to-point respectively, using INTAP-MPI. We use Alltoall, a common collective communication pattern, to study the impact of network topology on performance of collective operations. We analyze the performance of two different runs of the MPI_Alltoall operation from a network topology-aware point of view to study and analyze the impact network topology can have on the performance of collective operations. As the NAS MG benchmark uses mostly point-to-point communications, we use that to study the impact network topology can have on such operations. The MPI_Alltoall operation as well as the NAS MG benchmark were run on the Ranger supercomputer on separate allocations of 16 compute nodes (256 processes, 16 processes per node).

In the following visualizations of communication patterns, *Red* color represents intra-node communication, *Green* color represents 1-hop communication, *Blue* color represents 3-hop communication, *Pink* color represents 5-hop communication and *Cyan* color represents 7-hop communication. In the various heat maps to follow, the intensities of these colors represent the amount of communication between the respective entities with respect to the maximum amount of communication that any two entities in the MPI job perform. The stacked histogram representation allow us to understand how the communication pattern of individual processes change over multiple runs. The heatmaps, on the other hand, help us understand which process pairs end up doing more long distance communication over the network.

5.2 Visualizing Network Characteristics of Collective Communication

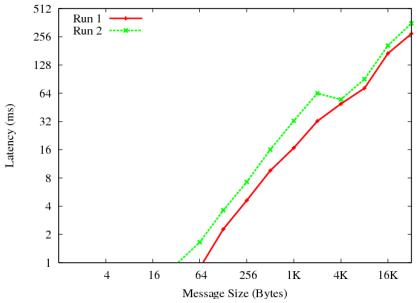
In this section, we analyze the performance of two different runs of MPI_Alltoall operation using INTAP-MPI. Figure 3(a) compares the average performance of the MPI_Alltoall operation over the two runs. Inside each run, the MPI_Alltoall operation was executed multiple times to remove possible variations in performance due to system noise. We can see that, on average, run #1 is seen to perform better than run #2. Below, we describe how an user of INTAP-MPI can use network topology-aware communication information to reason about the difference in performance of the MPI_Alltoall operation seen between the two runs.

Figure 3(b) shows the summary of the network topology-aware communication patterns of both jobs. As we can see, the number of long distance network communication (7-hops and 5-hops) is lesser in run #1 than run #2. Figures 3(c) and 3(d) depict the network topology-aware communication pattern of the MPI_Alltoall operations as stacked histograms at node level granularity. Figures 3(e) and 3(f) illustrate the same communication pattern as heat maps at node level granularity.

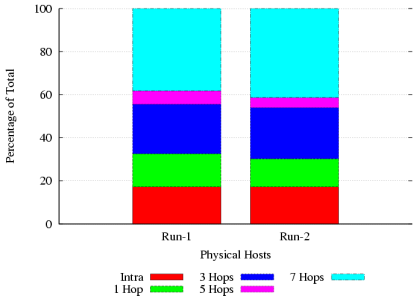
5.3 Visualizing Network Characteristics of NAS Application Benchmarks

In this section, we analyze the performance of two different runs of a class C NAS MG benchmark operation using INTAP-MPI. Figure 4(a) compares the average performance of the NAS-MG benchmark performed over the two runs. Inside each run, the NAS-MG benchmark was executed multiple times to remove possible variations in performance due to system noise. We can see that, on average, run #2 is seen to perform better than run #1. Below, we describe how an user of INTAP-MPI can use network topology-aware communication information to reason about the difference in performance of the class C NAS-MG benchmark seen between the two runs.

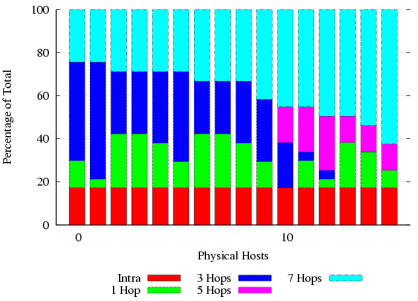
Figure 4(b) shows the summary of the network topology-aware communication patterns of both jobs. As we can see, the number of long distance network communication (7-hops) is lesser in run #2 than run #1. Figures 4(c) and 4(d) illustrate the network topology-aware communication pattern of the NAS-MG benchmark as stacked histograms at node level granularity. Figures 4(e) and 4(f) depict the same communication pattern as heat maps at node level granularity.



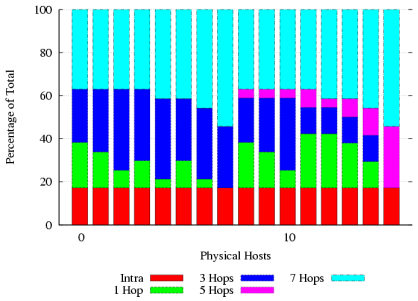
(a) Performance comparison of MPI_Alltoall between run #1 and run #2.



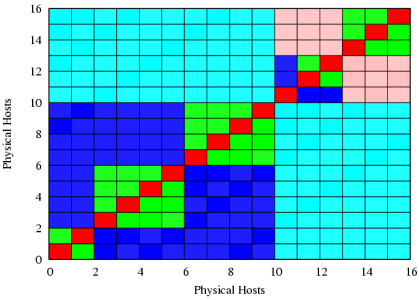
(b) Summary graph comparing communication patterns of run #1 and run #2.



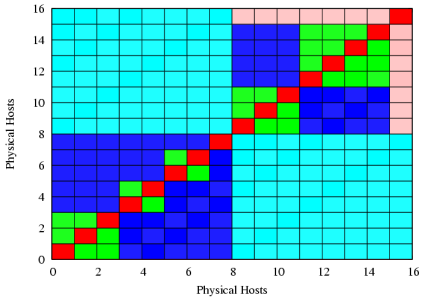
(c) Stacked histogram depicting network topology-aware communication pattern for run #1 at node level granularity.



(d) Stacked histogram depicting network topology-aware communication pattern for run #2 at node level granularity.



(e) Heatmap depicting network topology-aware communication pattern for run #1 at node level granularity.



(f) Heatmap depicting network topology-aware communication pattern for run #2 at node level granularity.

Fig. 3. Analysis of 256 process MPI_Alltoall operation on Ranger using INTAP-MPI

5.4 Impact of INTAP-MPI on Memory Consumption

Table 1 shows the impact INTAP-MPI has on the total memory consumption of the MPI job. As we saw in Section 4, only *rank 0* of the MPI job needs to allocate large memory data structures like the $\mathcal{O}(N^2_{\text{nodes}})$ matrix required to store the communication matrix of the job at various granularities (process, node, switch blade). The memory consumption at each individual rank is only of the order of $\mathcal{O}(N_{\text{nodes}})$ bytes. As we can see, the overhead imposed by INTAP-MPI on the memory consumption of the entire application is very minimal at scale. As number of nodes reaches $\mathcal{O}(10^5)$, we can start looking at switch blade level granularity to save memory.

Table 1. Overhead (in MegaBytes) of INTAP-MPI on memory consumption of MPI_Alltoall on Hyperion

Job Size (#Processes)	64	128	256	512	1,024
Memory Overhead	0.04	0.16	0.58	2.19	8.61

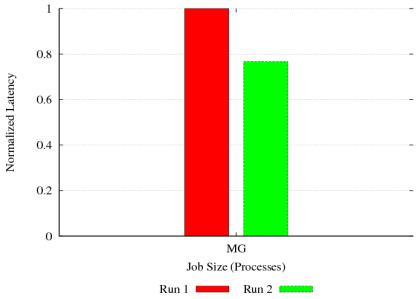
5.5 Impact of INTAP-MPI on Performance of MPI Jobs

Figure 5 depicts the performance of the MPI_Alltoall collective operation for various message sizes and system sizes. Figure 5(a) compares the communication performance of the collective operation with and without profiling at a system size of 1,024 processes. As we can see, there is very little impact on the communication performance due to INTAP-MPI. Figure 5(b) compares the performance of the MPI_Alltoall collective operation at various system sizes for a message size of 64 KB. As seen in Figure 5(a), INTAP-MPI has very little overhead on the communication performance.

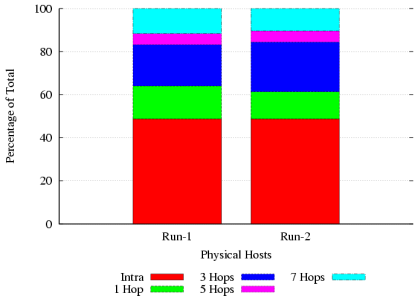
6 Related Tools

The benefit of data visualization and performance analysis tools have been demonstrated since Prism [16]. A classical MPI profiling tool, mpiP [19], is able to display details of the performance of MPI calls through MpiPView but cannot provide any topology information. Some tools like IPM [6] or Intel Trace Analyzer and Collector (ITAC), based on Vampir [14], are able to generate the communication matrix of the messages sent and received between each task, but this information is independent of the process mapping.

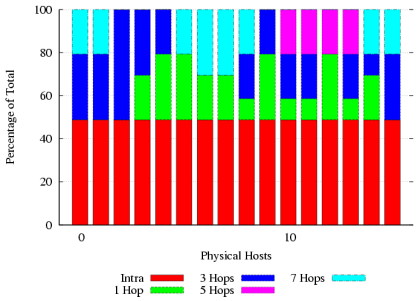
For IBM Blue Gene and Cray clusters, Bhathele [4] developed a Topology Manager API allowing an easy access to the topology information. This knowledge of the topology was possible through system calls. It allowed the development of pattern language for optimizing communication [5] or heuristics [4] included inside CHARM++ [11] to provide a topology-aware process mapping. Using similar techniques, some profiling tools, like TAU [17] or the Scalasca performance toolkit [7], are able to display the topology mapping of the processes on these systems. Nevertheless, none of the current tools is able to to profile and display the mapping information of MPI processes on InfiniBand clusters in a network topology-aware manner.



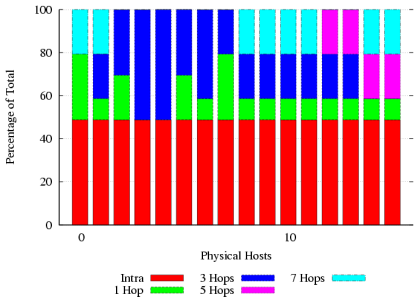
(a) Performance comparison of class C NAS MG benchmark between run #1 and run #2.



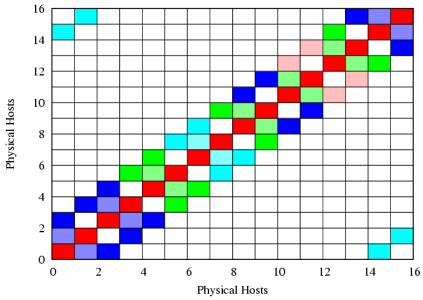
(b) Summary graph comparing communication patterns of run #1 and run #2.



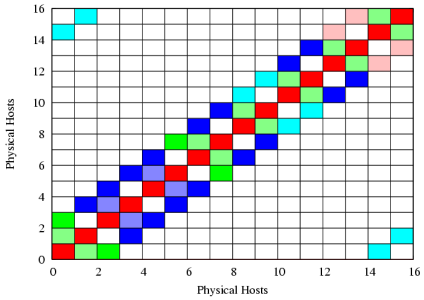
(c) Stacked histogram depicting network topology-aware communication pattern for run #1 at node level granularity.



(d) Stacked histogram depicting network topology-aware communication pattern for run #1 at node level granularity.

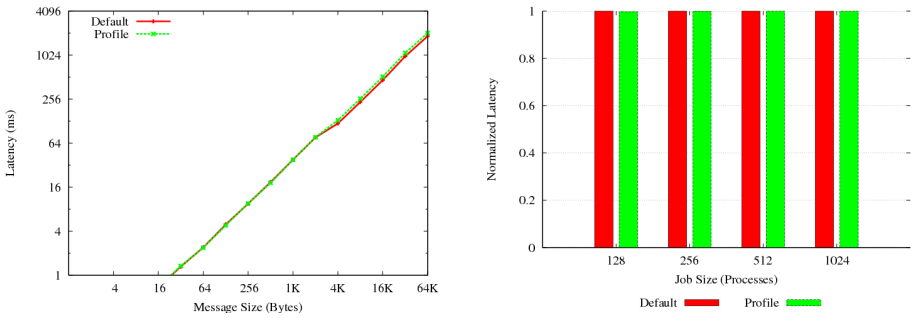


(e) Heatmap depicting network topology-aware communication pattern for run #1 at node level granularity.



(f) Heatmap depicting network topology-aware communication pattern for run #1 at node level granularity.

Fig. 4. Analysis of 256 process class C NAS MG benchmark on Ranger using INTAP-MPI



(a) Performance on 1,024 processes (b) Performance for 64 KB message size

Fig. 5. Impact of INTAP-MPI on performance MPI_Alltoall on Hyperion

7 Conclusions and Future Work

In this paper, we presented the design, evaluation and use case driven analysis of INTAP-MPI - a network topology-aware, scalable, low-overhead MPI profiler that allows users to analyze and visualize the communication pattern of MPI based HPC applications for any IB network architecture (FAT Tree, Tori, or Mesh). INTAP-MPI gives the flexibility to profile the MPI application either for the entire duration of application execution (by means of environment variables) or for specific sections of the application (by means of Unix signals). We integrated INTAP-MPI into the MVAPICH2 MPI library, allowing users to seamlessly use it for analyzing their applications. Our experimental analysis showed that INTAP-MPI is able to profile and visualize the communication pattern of applications with very low memory and performance overhead at scale. In future, we would like to extend this work to make it easily usable for other MPI libraries and software stacks.

References

1. Gnuplot, `http://www.gnuplot.info/`
2. MVAPICH2: High Performance MPI over InfiniBand, 10GigE/iWARP and RoCE, `http://mvapichcse.ohio-state.edu/`
3. Open fabrics enterprise distribution, `http://www.openfabrics.org/`
4. Bhatele, A.: Automating Topology Aware Mapping for Supercomputers. Ph.D. thesis, Dept. of Computer Science, University of Illinois (August 2010)
5. Bhatele, A., Kale, L.V., Chen, N., Johnson, R.E.: A Pattern Language for Topology Aware Mapping (June 2009)
6. Fürlinger, K., Wright, N.J., Skinner, D.: Performance Analysis and Workload Characterization with IPM. In: Parallel Tools Workshop, pp. 31–38 (2009)
7. Geimer, M., Wolf, F., Wylie, B., Ábrahám, E., Becker, D., Mohr, B.: The Scalasca Performance Toolset Architecture. Concurr. Comput.: Pract. Exper. 22(6), 702–719 (2010)
8. Subramoni, H., Kandalla, K., Vienne, J., Sur, S., Barth, B., Tomko, K., Mclay, R., Schulz, K., Panda, D.K.: Design and Evaluation of Network Topology-/Speed- Aware Broadcast Algorithms for InfiniBand Clusters. In: CLUSTER (2011)

9. Hoefler, T., Snir, M.: Generic Topology Mapping Strategies for Large-scale Parallel Architectures. In: Proceedings of the 2011 ACM International Conference on Supercomputing, ICS 2011, pp. 75–85. ACM (June 2011)
10. InfiniBand Trade Association, http://www.infinibandta.org/
11. Kalé, L., Krishnan, S.: CHARM++: A Portable Concurrent Object Oriented System Based on C++. In: Proceedings of OOPSLA 1993, pp. 91–108. ACM Press (September 1993)
12. MPI Forum: MPI: A Message Passing Interface. In: Proceedings of Supercomputing (1993)
13. Dandapanthula, N., Subramoni, H., Vienne, J., Kandalla, K., Sur, S., Panda, D.K., Brightwell, R.: INAM - A Scalable InfiniBand Network Analysis and Monitoring Tool. In: Alexander, M., D'Ambra, P., Belloum, A., Bosilca, G., Cannataro, M., Danelutto, M., Di Martino, B., Gerndt, M., Jeannot, E., Namyst, R., Roman, J., Scott, S.L., Traff, J.L., Vallée, G., Weidendorfer, J. (eds.) Euro-Par 2011 Workshops, Part II. LNCS, vol. 7156, pp. 166–177. Springer, Heidelberg (2012)
14. Nagel, W.E., Arnold, A., Weber, M., Hoppe, H.C., Solchenbach, K.: VAMPIR: Visualization and Analysis of MPI Resources. Supercomputer 12, 69–80 (1996)
15. OFED: Open Subnet Manager, http://www.openfabrics.org/downloads/management/README
16. Sistare, S., Allen, D., Bowker, R., Jourdenais, K., Simons, J., Title, R.: A Scalable Debugger for Massively Parallel Message-Passing Programs. IEEE Parallel Distrib. Technol. 2(2), 50–56 (1994)
17. Spear, W., Malony, A.D., Lee, C.W., Biersdorff, S., Shende, S.: An Approach to Creating Performance Visualizations in a Parallel Profile Analysis Tool. In: Alexander, M., D'Ambra, P., Belloum, A., Bosilca, G., Cannataro, M., Danelutto, M., Di Martino, B., Gerndt, M., Jeannot, E., Namyst, R., Roman, J., Scott, S.L., Traff, J.L., Vallée, G., Weidendorfer, J. (eds.) Euro-Par 2011 Workshops, Part II. LNCS, vol. 7156, pp. 156–165. Springer, Heidelberg (2012)
18. Top500: Top500 Supercomputing systems (November 2010), http://www.top500.org
19. Vetter, J.S., McCracken, M.O.: Statistical Scalability Analysis of Communication Operations in Distributed Applications. In: Proceedings of the Eighth ACM SIGPLAN Symposium on Principles and Practices of Parallel Programming, PPoPP 2001, pp. 123–132 (2001)