INAM - A Scalable InfiniBand Network Analysis and Monitoring Tool *

N. Dandapanthula[1], H. Subramoni[1], J. Vienne[1], K. Kandalla[1], S. Sur[1], D. K. Panda[1], and R. Brightwell[2]

[1] Department of Computer Science and Engineering,
The Ohio State University
{dandapan, subramon, viennej, kandalla, surs, panda} @cse.ohio-state.edu
[2] Sandia National Laboratories
rbbrigh@sandia.gov

Abstract. InfiniBand's popularity in the field of cluster and high performance computing can be attributed to its open standard and high performance. As InfiniBand (IB) clusters grow in size and scale, predicting the behavior of the IB network in terms of link usage and performance becomes an increasingly challenging task. Although the IB specification proposes a detailed subnet management infrastructure to handle various aspects of the network, there currently exists no open source tool that allows users to dynamically analyze and visualize the communication pattern and link usage in the IB network. In this context, we design and develop a scalable InfiniBand Network Analysis and Monitoring tool - *INAM*. INAM monitors IB clusters in real time and queries the various subnet management entities in the IB network to gather the various performance counters specified by the IB standard. We provide an easy to use web-based interface to visualize performance counters and subnet management attributes of a cluster in an on-demand basis. It is also capable of capturing the communication characteristics of a subset of links in the network, thereby allowing users to visualize and analyze the network communication characteristics of a job in a high performance computing environment. Our experimental results show that INAM is able to accurately visualize the link utilization as well as the communication pattern of target applications.

1 Introduction

Across various enterprise and scientific domains, users are constantly looking to push the envelope of achievable performance. The need to achieve high resolution results with smaller turn around times has been driving the evolution of enterprise and supercomputing systems over the last decade. Interconnection networks have also rapidly evolved to offer low latencies and high bandwidths to meet the communication requirements of distributed computing applications. InfiniBand has emerged as a popular high performance network interconnect and is being increasingly used to deploy some of the top supercomputing installations around the world. According to the Top500 [13] ratings of supercomputers done in June'11, 41.20% of the top 500 most powerful supercomputers in the world

* This research is supported in part by Sandia Laboratories grant #1024384, U.S. Department of Energy grants #DE-FC02-06ER25749, #DE-FC02-06ER25755 and contract #DE-AC02-06CH11357; National Science Foundation grants #CCF-0621484, #CCF-0702675, #CCF-0833169, #CCF-0916302 and #OCI-0926691; grant from Wright Center for Innovation #WCI04-010-OSU-0; grants from Intel, Mellanox, Cisco, QLogic, and Sun Microsystems; Equipment donations from Intel, Mellanox, AMD, Obsidian, Advanced Clustering, Appro, QLogic, and Sun Microsystems.

are based on the InfiniBand interconnects. Recently, InfiniBand has also started to make in-roads into the world of enterprise computing.

Different factors can affect the performance of applications utilizing IB clusters. One of these factors is the routing of packets or messages. Due to static routing, it is important to ensure that the routing table is correctly programmed. Hoefler et al. showed, in [4], the possible degradation in performance if multiple messages traverse the same link at the same time. Unfortunately, there do not exist any open-source tools that can provide information such as the communication matrix of a given target application or the link usage in the various links in the network, in a user friendly way.

Most of the contemporary network monitoring tools for IB clusters have an overhead attached to them which is caused by the execution of their respective daemons which needs to run every monitored device on the subnet. The purpose of these daemons is to gather relevant data from their respective hosts and transmit it to a central daemon manager which renders this information to the user. Furthermore, the task of profiling an application at the IB level is difficult considering the issue that most of the network monitoring tools are not highly responsive to the events occurring on the network. For example, to reduce the overhead caused by constant gathering of information at the node by the daemons, a common solution is to gather the information at some time intervals which could be anywhere between 30 seconds to 5 minutes. This is called the sampling frequency. Thus, the higher the sampling frequency, the higher the overhead created by the daemons. This causes a tradeoff with the responsiveness of the network monitoring tool. This method has an additional disadvantage in that, it does not allow us to monitor network devices such as switches and routers where we will not be able to launch user specified daemon processes.

As IB clusters grow in size and scale, it becomes critical to understand the behavior of the InfiniBand network fabric at scale. While the Ethernet ecosystem has a wide variety of matured tools to monitor, analyze and visualize various elements of the Ethernet network, the InfiniBand network management tools are still in their infancy. To the best of our knowledge, none of the available open source IB network management tools allow users to visualize and analyze the communication pattern and link usage in an IB network. These lead us to the following broad challenge - *Can a low overhead network monitoring tool be designed for IB clusters that is capable of depicting the communication matrix of target applications and the link usage of various links in the InfiniBand network?*

In this paper we address this challenge by designing a scalable InfiniBand Network Analysis and Monitoring tool - *INAM*. INAM monitors IB clusters in real time and queries the various subnet management entities in the IB network to gather the various performance counters specified by the IB standard. We provide an easy to use web interface to visualize the performance counters and subnet management attributes of the entire cluster or a subset of it on the fly. It is also capable of capturing the communication characteristics of a subset of links in the network, thereby allowing users to visualize and analyze the network communication characteristics of a job in a high performance computing environment. Our experimental results show that INAM is able to accurately visualize the link usage within a network as well as the communication pattern of target applications.

The remainder of this paper is organized as follows. Section 2 gives a brief overview of InfiniBand and the InfiniBand subnet management infrastructure. In Section 3 we present the framework and design of INAM. We evaluate and analyze the correctness and performance of INAM in various scenarios in Section 4, describe the currently available related tools in Section 5, and summarize the conclusions and possible future work in Section 6.

2 Background

2.1 InfiniBand

InfiniBand is a very popular switched interconnect standard being used by almost 41% of the Top500 Supercomputing systems [13]. InfiniBand Architecture (IBA) [5] 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.

2.2 OFED

OFED, short for OpenFabrics Enterprise Distribution, is an open source software for RDMA and kernel bypass applications. It is needed by the HPC community for applications which need low latency and high efficiency and fast I/O. A detailed overview of OFED can be found in [11]. OFED provides performance monitoring utilities which present the port counters and subnet management attributes for all the device ports within the subnet. Some of the attributes which can be obtained from these utilities are shown in Table 1.

Utility	Attribute	Description
perfquery	XmtData	The number of 32 bit data words sent out through that port since last reset
perfquery	RcvData	The number of 32 bit data words received through that port since last reset
perfquery	XmtWait	The number of units of time a packet waits to be transmitted from a port
smpquery	LinkActiveSpeed	The current speed of a link
smpquery	NeighborMTU	Active maximum transmission unit enabled on this port for transmit

Table 1. Sample of attributes provided by utilities inside OFED

OFED includes an InfiniBand subnet manager called OpenSM which configures an InfiniBand subnet. It comprises of the subnet manager (SM) which scans the fabric, initiates the subnet and then monitors it. The subnet management agents (SMA) are deployed on every device port of the subnet to monitor their respective hosts. All management traffic including the communication between the SMAs and the SM is done using subnet management packets (SMP). IBA allocates the Virtual lane (VL) 15 for subnet management traffic. The general purpose traffic can use any of the other virtual lanes from 1 to 14 but the traffic on VL 15 is independent of the general purpose traffic.

3 Design And Implementation of INAM

We describe the design and implementation details of our InfiniBand Network Analysis and Monitoring tool (INAM) in this section. For modularity and ease of portability, we separate the functionality of INAM into two distinct modules - the InfiniBand Network Querying Service (INQS) and the Web-based Visualization Interface (WVI). INQS acts as a network data acquisition service. It retrieves the requested information regarding ports on all the devices of the subnet to obtain the performance counters and subnet management attributes. This information is then stored in a database using MySQL methods [9]. The WVI module then communicates with the database to obtain the

data pertaining to any user requested port(s) in an on-demand basis. The WVI is designed as a standard web application which can be accessed using any contemporary web browser. The two modes of operation of the WVI include the live observation of the individual port counters of a particular device and the long term storage of all the port counters of a subnet. This information can be queried by the user in the future. INQS can be ported to any platform, independent of the cluster size and the Linux distribution being used. INAM is initiated by the administrator and there exists a connection thread pool through which individual users are served. As soon as a user exits the application, the connection is returned to the pool. If all the connections are taken up, then the user has to wait. Currently the size of this connection pool is 50 and can be increased.

As we saw in Section 1, a major challenge for contemporary IB network monitoring tools is the necessity to deploy daemon processes on every monitored device on the subnet. The overhead in terms of CPU utilization and network bandwidth caused by these daemons often cause considerable perturbations in the performance of real user applications that use these clusters. INAM overcomes this by utilizing the Subnet Management Agents (SMA) which are required to be present on each IB enabled device on the subnet. The primary role of an SMA is to monitor and regulate all IB network related activity on their respective host nodes. The INQS queries these SMAs to obtain the performance counters and subnet management attributes of the IB device(s) on a particular host. The INQS uses Management Datagram (MAD) packets to query the SMAs. As MAD packets use a separate Virtual Lane (VL 15), they will not compete with application traffic for network bandwidth. Thus, compared to contemporary InfiniBand network management tools, INAM is more responsive and and causes less overhead.

INAM is also capable of monitoring and visualizing the utilization of a link within a subnet. To obtain the link utilization, the *XmtWait* attribute alone or *XmtData* / *RcvData* and *LinkActiveSpeed* attributes in combination are used. The *XmtWait* attribute corresponds to the period of time a packet was waiting to be sent, but could not be sent due to lack of network resources. In short it is an indication of how congested a link is. The *LinkActiveSpeed* attribute indicates the speed of the link. This can be used in combination with the change in *XmtData* or *RcvData* attribute to see whether the link is being over utilized or not. In either case, we update a variable called the link utilization factor to depict the amount of traffic in the link. There is also an option to use just the INQS as a stand alone system to save the device port information and the link usage information over a period of time (time can be varied depending on the memory available) to analyze the traffic patterns over an InfiniBand subnet. INQS initially creates a dynamic *MySQL* database of all the available LID-Port combinations, along with the physical links interconnecting these ports. The LID-Port combination signifies all combinations of the device LIDs in the subnet and their respective ports. This information is updated periodically and thus adapts to any changes in the network topology. The frequency at which the data is collected from the subnet and the frequency at which the data is displayed on the WVI can both be modified as per the requirement of the user. The overhead here would be associated with the WVI module. The display frequency can be reduced to 1 second and this would serve the users for all practical purposes. If this display frequency is less then 1 second, then we see a drop in the responsiveness of the dynamic graphs generated.

The WVI interacts with the database and displays the information requested by the user in the form of a graphical chart. Dynamic graphs are generated by using *HighCharts Js* [2]. This model currently does not support dynamic multiple data series displayed in the same diagram with a common y-axis. Thus the comparision is done using multiple graphs as shown in Figure 1. We use a push model instead of a pull model to update the data in the WVI. The connection between the MySQL

database and the WVI is kept open and hosting server pushes data to the browser as soon as the database is updated by INQS. This technique removes the overhead on the web server caused by the browser constantly polling the database for new data. This is implemented using a methodology called Comet [3]. This makes the web server stable and provides high availability even when deployed on large InfiniBand clusters with heavy data flow. The rest of the functionalities of the web server are implemented using Java 2 Platform Enterprise Edition (J2EE) [12]. The communication pattern of an MPI job is created by WVI by querying the database and then by using the canvas element of HTML5 [15] to chart out the physical topology and connections between the ports on a subnet.

3.1 Features of INAM

INAM can monitor an InfiniBand cluster in real time by using the functionalities provided by Open Fabrics Enterprise Distribution (OFED) stack. It can also monitor the link utilization on the fly and provide a post mortem analysis of the communication pattern of any job running on the IB cluster.

The user can select the device he wants to monitor through a dynamically updated list of all the currently active devices on the subnet. An option to provide a list of all the port counters which need to be compared in real time, is given to the user. Only the counters of one particular port can be monitored at a time. INAM shows the first derivative of the counter values. A detailed overview of all the subnet management attributes of a particular port in a subnet can also be obtained. The attributes are divided into four main categories which are *Link Attributes*, *Virtual Lane Attributes*, *MTU Attributes* and *Errors and Violations*. INAM also provides dynamic updates regarding the status of the master Subnet Manager (SM) instance to the user. If there is a change in the priority of SM or if the Master SM instance fails or if a new slave SM takes over as a Master SM instance, the status is updated and the user is notified. This can help to understand the fail-over properties of OpenSM. Further more, a user can ask INAM to monitor the network for the time period of an MPI job and then it helps the user understand the communication pattern of that job using a color coded link utilization diagram.

4 Experimental Results

4.1 Experimental Setup

The experimental setup is a cluster of 71 nodes (8 cores per node with a total of 568 cores) which are all dual Intel Xeons E5345 connected to an InfiniBand Switch which has an internal topology of a Fat Tree. We use a part of this cluster to show the functionality of INAM. This set up comprises of 6 leaf switches and 6 spine switches with 24 ports each and a total of 35 leaf nodes equipped with ConnectX cards. The functioning of INAM is presented using a series of benchmarks in varied scenarios. The first set of results are obtained using a *bandwidth sharing benchmark* to create traffic patterns which are verified by visualizing the link usage using INAM. The second set of benchmarks shows similar network communication patterns with *MPI_Bcast* configured for diverse scenarios. The third set of experiments verifies the usage of INAM using the LU benchmark from the SpecMPI suite.

4.2 Visualizing Port Counters

The user can select the device they want to monitor through a dynamically updated list of all the currently active devices on the subnet. The user can also provide a list of all the port counters they

want to compare in real time. Figure 1 depicts how INAM allows users to visually compare multiple attributes of a single port. In this example, we show how two attributes - transmitted packets and received packets, of user selected port can be compared.

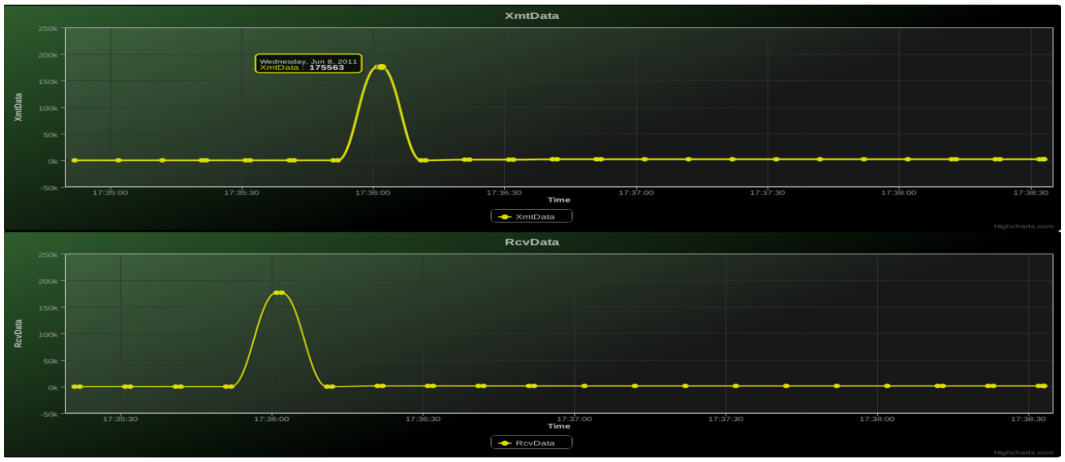

Fig. 1. Monitoring the XmtData and RcvData of a port

4.3 Point to Point Visualization: Synthetic Communication Pattern

We create custom communication patterns using the bandwidth sharing benchmark mentioned in [14] to verify the functioning of INAM. The benchmark in question enables us to mention the number of processes transmitting messages and the number of processes receiving messages at leaf switch level and thus creating a blocking point to point communication pattern. We created various test patterns, each incrementally more communication intensive then the previous pattern, to help us notice a difference in the pattern using INAM. Two of those patterns are mentioned in detail in the consequent sections.

Test Pattern 1 The first test pattern is visualized in Figure 2. The process arrangement in this pattern is such that 8 processes, one per each of the 8 leaf nodes connected to leaf switch 84, communicate with one process, on each of the four leaf nodes connected to the each of the two switches 78 and 66. The thick green line indicates that multiple processes are using that link. In this case, it can be observed that the thick green line originating from switch 84 splits into 2 at switch 110. The normal green links symbolize that the links are not being over utilized, for this specific case.

Test Pattern 2 Figure 3 presents the network communication for test pattern 2. The process arrangement in this pattern is such that 32 processes, four per each of the 8 leaf nodes connected to leaf switch 84, communicate with two processes, on each of the eight leaf nodes connected to the each of the two switches 78 and 66. 32 processes send out messages from switch 84 and 16 processes on each of the switches 78 and 66 receive these messages. This increase in the number of processes per leaf node explains the exorbitant increase in the number of links being overly utilized. Figure 3 also shows that all of the inter switch links are marked in thick lines, thus showing that each link is being used by more then one process. The links depicted in red indicate that the link is over utilized. Since each leaf node on switch 84 has four processes and each leaf node on the other switches have two processes, the links connecting the leaf nodes to the switch are depicted as thick red lines.

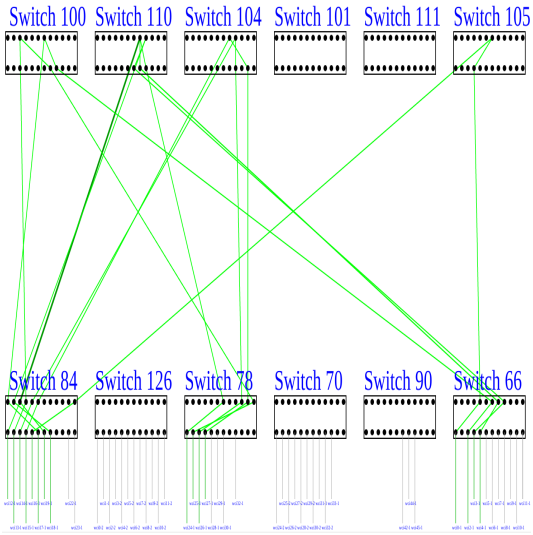

Fig. 2. INAM depiction of network traffic pattern for 16 processes

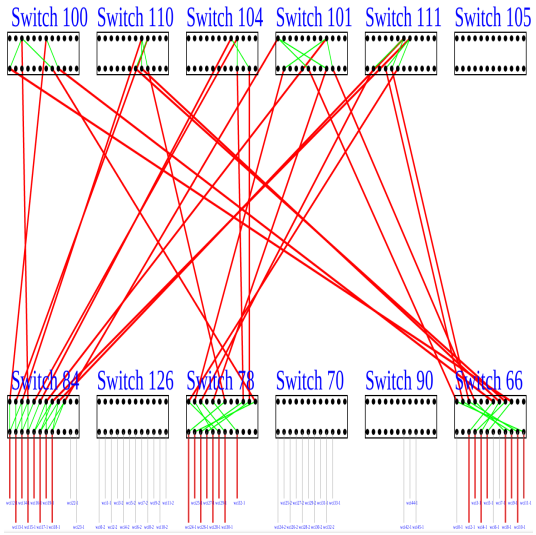

Fig. 3. INAM depiction of network traffic pattern for 64 processes

4.4 Link Utilization of Collective Operations: Case Study with MPI_Bcast Operation

In this set of experiments, we evaluate the visualization of the *One-to-All* broadcast algorithms typically used in MPI libraries, using INAM. MVAPICH2 [8] uses the tree-based algorithms for small and medium sized messages, and the scatter-allgather algorithm for larger messages. The tree-based algorithms are designed to achieve lower latency by minimizing the number of communication steps. However, due to the costs associated with the intermediate copy operations, the tree-based algorithms are not suitable for larger messages and the scatter-allgather algorithm is used for such cases. The scatter-allgather algorithm comprises of two steps. In the first step, the root of the broadcast operation divides the data buffer and scatters it across all the processes using the binomial algorithm. In the next step, all the processes participate in an allgather operation which can either be implemented using the recursive doubling or the ring algorithms.

We designed a simple benchmark to study the link utilization pattern of the MPI_Bcast operation with different message lengths. For brevity, we compare the link utilization pattern with the binomial algorithm with 16KB message length and we study the scatter-allgather (ring) algorithm with a data buffer of size 1MB. We used six processes for these experiments, such that we have one process on each of the leaf switches, as shown in Figure 4. In our controlled experiments, we assign the process on switch 84 to be the root (rank 0) of the MPI_Bcast operation, switch 126 be rank 1 and so on until the process on switch 66 is rank 5. Figure 4 shows a binomial traffic pattern for a broadcast communication on 6 processes using a 16KB message size. The binomial communication pattern with 6 processes is as follows:

- Step1: Rank0 → Rank3
- Step2: Rank0 → Rank1 and Rank3 → Rank4
- Step3: Rank1 → Rank2 and Rank4 → Rank5

In Figure 4, a darker color is used to represent a link that has been used more than once during the broadcast operation. We can see that processes with ranks 0 through 4, the link connecting

the compute nodes to their immediate leaf-level switches are used more than once, because these processes participate in more than one send/recv operation. However, process P5 receives only one message and INAM demonstrates this by choosing a lighter shade. We can also understand the routing algorithm used between the leaf and the spine switches by observing the link utilization pattern generated by INAM. We also observe that the process with rank4, uses the same link between switches 90 and 110 for both its send and receive operations. Such a routing scheme is probably more prone to contention, particularly at scale when multiple data streams are competing for the same network link.

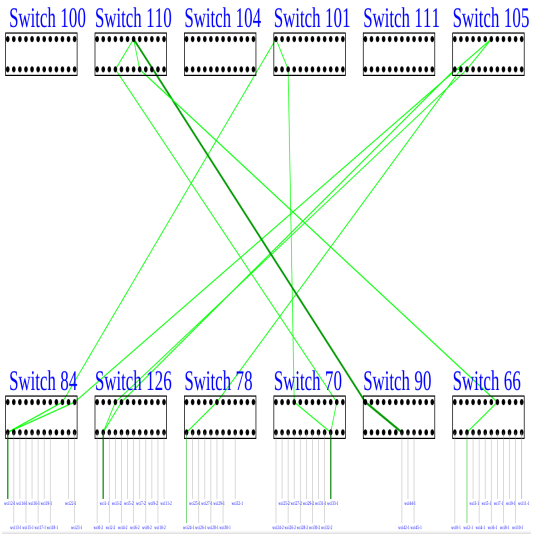

Fig. 4. Link utilization of binomial algorithm

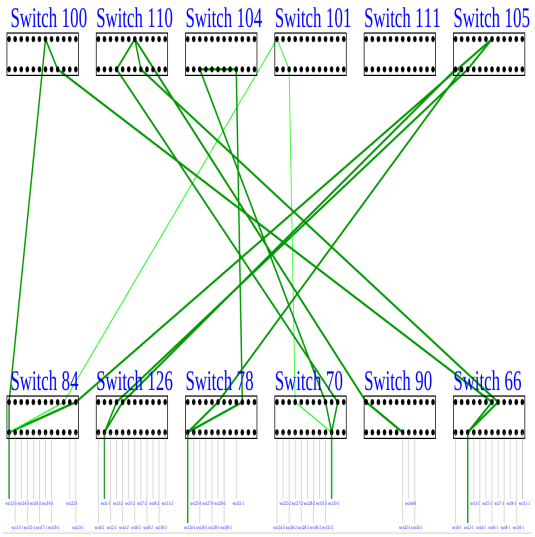

Fig. 5. Link utilization of scatter-allgather algorithm

Figure 5 presents the link utilization pattern for the scatter-allgather (ring) algorithm with 6 processes. We can see that the effective link utilization for this algorithm is considerably higher when compared to the binomial exchange. This is because the scatter-allgather (ring) algorithm involves a higher number of communication steps than the binomial exchange algorithm. With 6 processes, the ring algorithm comprises of 6 communication steps. In each step, process Pi communicates with its immediate logical neighbors processes $P(i-1)$ and $P(i+1)$. This implies that each link between the neighboring processes are utilized exactly 6 times during the allgather phase.

4.5 Application Visualization: SpecMPI - LU

In this experiment, we ran the *LU* benchmark (137.lu - medium size - mref) from the SpecMPI suite [10] on a system size of 128 processes using 16 leaf nodes with 8 nodes on each of the two leaf switches. The prominent communication used by LU comprises of MPI_Send and MPI_Recv. The communication pattern is such that each process communicates with its nearest neighbors in either directions (p2 communicates with p1 and p3). In the next step, p0 communicates with p15, p1 communicates with p16 and so on. This pattern is visualized by INAM and is shown in Figure 6. It can be seen that a majority of the communication is occurring on an intra-switch level.

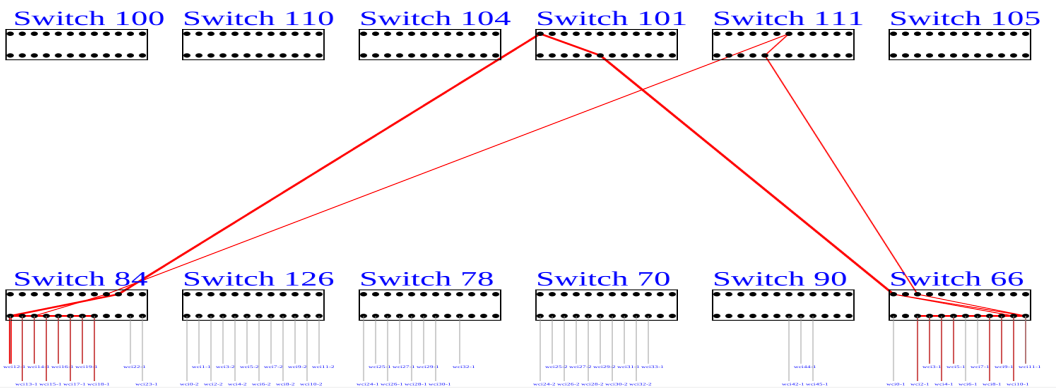

Fig. 6. INAM depicting the communication pattern using LU benchmark

4.6 Overhead of Running INAM

Since we use the subnet management agent (SMA), which acts like daemons monitoring all the devices of a subnet, we do not need to use any additional daemons installed on every device to obtain this data. This is a major advantage as it avoids the overhead in the contemporary approach caused by the daemons which are installed on every device. The user just needs to have the service *opensmd* started on the subnet. Since the queries used communicate through Virtual Lane 15 for the purpose of data acquisition, there is no interference with the generic cluster traffic. For the verification of this, we compared the performance of an IMB alltoall benchmark while toggling the data collection service on and off by using messages of size 16 KB and 512 KB for a system size varying from 16 cores to 512 cores. The results obtained are shown in Figure 7 which shows that the overhead is very minimal even though the service is on and there is not much difference even though the message size is increased.

System Size	16 cores	32 cores	64 cores	128 cores	256 cores	512 cores
Message size 16 KB	0.13%	0.11%	0.15%	0.09%	0.07%	0.14%
Message size 512 KB	0.19%	0.21%	0.16%	0.08%	0.21%	0.15%

Fig. 7. Overhead caused by running INAM

5 Related Tools

There is a plethora of free or commercial network monitoring tools that provide different kinds of information to the system administrators or the users. But only a few of them provide specific information related to IB network. We focus here on three popular network monitoring tools: *Ganglia* [6], *Nagios*[1] and *FabricIT* [7].

Ganglia is a widely used open-source scalable distributed monitoring system for high-performance computing systems developed by the University of California inside the Berkeley Millennium Project. One of the best features of Ganglia is to offer an overview of certain characteristics within all the nodes of a cluster, like memory, CPU, disk and network utilization. At the IB level, Ganglia can provide information through perfquery and smpquery. Nevertheless, Ganglia can't show any information related to the network topology or link usage. Furthermore, to get all the data, Ganglia needs to run a daemon, called *gmond*, on each node, adding an additional overhead.

Nagios is another common open-source network monitoring tool. Nagios offers almost the same information as Ganglia through a plug-in called "InfiniBand Performance Counters Check". But, as Ganglia, Nagios can't provide any information related to the topology.

FabricIT is a proprietary network monitoring tool developed by Mellanox. Like INAM, FabricIT is able to provide more information than Ganglia or Nagios, but the free version of the tool does not give a graphical representation of the link usage or the congestion.

INAM is different from the other existing tools by the richness of the given information and also its unique link usage information, giving all the required elements to users to understand the performance of applications at the IB level.

6 Conclusions and Future Work

In this paper, we have presented INAM - a scalable network monitoring and visualization tool for InfiniBand networks which renders a global view of the subnet through a web-based interface (WVI) to the user. INAM depends on many services provided by the open-source OFED stack to retrieve necessary information from the IB network. INAM also has an online data collection module (INQS) which runs in the background while a job is in progress. After the completion of the job, INAM presents the communication pattern of the job in a graphical format. The overhead caused by this tool is very minimal and it does not require the user to launch any special processes on the target nodes. Instead, it queries on the IB devices directly through the network and gathers data.

In future, we would like to extend this work to do an online analysis of the traffic patterns on a cluster. If next generation InfiniBand devices offer performance counters for each virtual lane, we could leverage it to study link utilization and network contention patterns in a more scalable fashion. Another dimension would be to create a time line graphical pattern to depict the exact amount of data being communicated in the subnet during a particular interval. We would also like to extend the functionality of INAM such that the user can monitor and compare various counters from different ports. We would also like to show if the links are used multiple times simultaneously when the communication matrix is generated.

References

1. Barth, W.: Nagios. System and Network Monitoring. No Starch Press, U.S. Ed edn. (2006)
2. Charts, H.: HighCharts JS - Interactive JavaScript Charting. http://www.highcharts.com/
3. DWR: DWR - Direct Web Remoting. http://directwebremoting.org/dwr/
4. Hoefler, T., Schneider, T., Lumsdaine, A.: Multistage Switches are not Crossbars: Effects of Static Routing in High-Performance Networks. In: Proceedings of the 2008 IEEE Cluster Conference (Sep 2008)
5. InfiniBand Trade Association: http://www.infinibandta.org/
6. Massie, M.L., Chun, B.N., Culler, D.E.: The Ganglia Distributed Monitoring System: Design, Implementation, and Experience. Parallel Computing 30(7) (Jul 2004)
7. Mellanox: Fabric-it. http://www.mellanox.com/pdf/prod_ib_switch_systems/pb_FabricIT_EFM.pdf
8. MVAPICH2: http://mvapich.cse.ohio-state.edu/
9. MySQL: MySQL. http://www.mysql.com/
10. Mller, M.S., van Waveren, G.M., Lieberman, R., Whitney, B., Saito, H., Kumaran, K., Baron, J., Brantley, W.C., Parrott, C., Elken, T., Feng, H., Ponder, C.: Spec mpi2007 - an application benchmark suite for parallel systems using mpi. Concurrency and Computation: Practice and Experience pp. 191–205 (2010)
11. Open Fabrics Alliance: http://www.openfabrics.org/
12. SUN: Java 2 platform, enterprise edition (j2ee) overview. http://java.sun.com/j2ee
13. Top500: Top500 Supercomputing systems. http://www.top500.org (November 2010)
14. Vienne, J., Martinasso, M., Vincent, J.M., Méhaut, J.F.: Predictive models for bandwidth sharing in high performance clusters. In: Proceedings of the 2008 IEEE Cluster Conference (Sep 2008)
15. W3C: HTML5 - Canvas Element. https://developer.mozilla.org/en/HTML/Canvas