# Performance Benchmark and MPI Evaluation Using Westmere-based Infiniband HPC Cluster

Basem Madani, Raed Al-Shaikh
*Department of Computer Engineering*
King Fahd University of Petroleum and Minerals
Dhahran 31261, Saudi Arabia
{mbasem, g199607190}@kfupm.edu.sa

***Abstract*** **- In recent years, we have witnessed a growing interest in optimizing the high performance computing (HPC) solutions using advanced CPU and Interconnect technologies. These advances are driven by the fact that CPU manufacturers are facing extreme challenges in doubling the processors' speeds due to various reasons, such as the high temperatures and power consumptions. Therefore and as a continuation to Moore's law, recent trends in HPC systems have shown that future increases in performance can only be achieved through increases in system scale using a larger number of components, such as multi-cores and faster interconnects. In this paper, we evaluate a large-scale Infiniband cluster, equipped with Intel's latest Westmere processor using two MPI implementations. The paper presents the cluster configuration and evaluates its performance using various High Performance LINPACK (HPL) and Intel MPI (IMB) benchmarks. Our results show that system scalability can still be achieved with up to 87% efficiency when considering the right combination of MPI, interconnect and CPU technologies.**

*Keywords — **HPC, Linpack, Pallas, Infiniband**.*

## I. INTRODUCTION

In recent years, we have witnessed a growing interest in optimizing the high performance computing solutions using advanced CPU and Interconnect technologies. This interest is driven by the fact that single CPU-chips are reaching their physical limits in terms of heat dissipation and power consumption. Therefore and as a continuation to Moore's law, recent trends in HPC systems have shown that future increases in performance can only be achieved through increases in system scale using a larger number of components, such as multi-core CPUs and ultra-fast interconnects. Accordingly, scaled-out computing is clearly becoming the trend.

In terms of HPC interconnects, there are several network interconnects that provide ultra-low latency (less than 1 microsecond) and high bandwidth (several gigabytes per second). Some of these interconnects may provide flexibility by permitting user-level access to the network interface cards for performing communication, and also supporting access to remote processes' memory address spaces [1]. Examples of these interconnects are Myrinet from Myricom [6], Quadrics [5] and Infiniband [8]. The focus of this paper is on the Infiniband architecture, which is one of the latest industry standards, offering low latency and high bandwidth as well as many advanced features such as Remote Direct Memory Access (RDMA), atomic operations, multicast and QoS [10]. Currently, available Infiniband products can achieve a latency of 200 nanoseconds for small messages and a bandwidth of up to 3-4 GB/s [1]. As a result, it is becoming increasingly popular as a high-speed interconnect technology option for building high performance clusters.

In terms of compute power, Intel and AMD are still the leaders in the CPU industry, dominating the top500.org list of the most powerful supercomputers worldwide, and taking over 80% of HPC as of 2010 [17]. Nowadays, most of the high performance clusters use multi-core CPUs in their compute nodes, ranging from 2 to 4 cores per nodes, while 6-cores sockets will be become more common on clusters as Intel and AMD released their Westmere and Phenom II multi-core CPUs, respectively [15].

Our objective in this paper is to evaluate the performance of a large-scale Infiniband cluster using two of the most commonly used MPI implementations. As systematically described in the evaluation section, we first built a performance baseline for our HPC machine using LINPACK [1] benchmarking utility, to ensure that our HPC cluster is tuned properly. Subsequently, we evaluated two of the most commonly used MPI implementations in the HPC industry, which are MVAPICH2 and Intel MPI using the Intel MPI Benchmark utility (IMB). Our results show that HPC systems' scalability can still be achieved with up to 87% efficiency when considering the right combination of MPI, interconnect and CPU technologies. To the best of our knowledge, this is the first paper that evaluates the

performance of Westmere-based clusters using a large-scale Infiniband Quadratic-Data Rate (QDR) interconnect.

The rest of the paper is organized as follows: In section 2, we briefly shed some light on the Infiniband interconnect technology, the Intel Westmere CPU architecture, and the MPI implementations used to benchmark our HPC cluster. In section 3, we present our cluster design, while in section 4, we describe our methodology of measuring the compute power of the cluster using LINPACK. Section 5 describes our experimental evaluation and interprets the benchmark results. We state our conclusion and future work in the last section.

## II. BACKGROUND

In this section, we briefly describe the technologies used to benchmark our HPC cluster. These are: the Quad Data Rate (QDR) Infiniband interconnect technology, the Intel Westmere architecture, and the MPI implementations.

### *A. Infiniband Architecture*

Infiniband is a technology that provides a high bandwidth I/O communication over a high speed serial data bus. It uses a switched fabric topology, as opposed to a hierarchical switched network like Ethernet [3]. It is designed to directly route data from one point to another point through a switch, where all transmissions begin or end at a channel adapter (HCA). Each Infiniband processor contains a host channel adapter (HCA) and each peripheral has a target channel adapter (TCA).[3] The Infiniband serial connection signaling rate is 2.5 Gbit/s in single data rate (SDR) technology, 5.0 Gbit/s in double data rate (DDR) technology or 10 Gbit/s in quad data rate (QDR), in each direction per connection. Moreover, the links can be aggregated in units of 4 or 12, designated as 4X and 12X. However, Infiniband uses 8B/10B encoding, which implies four fifths of the traffic is useful, therefore DDR 4X link curries 20 Gbit/s raw, or 16 Gbit/s of useful data. Table-1 summarizes the different Infiniband technologies with their associated theoretical performance numbers.

Table 1: Performance numbers of different Infiniband technologies

| IB technology | SD IB Data Rate | DD IB Date Rate | QDR IB Data Rate |
|---|---|---|---|
| 1x | 2Gbps | 4Gbps | 8Gbps |
| 4x | 8Gbps | 16Gbps | 32Gbps |
| 12x | 24Gbps | 48Gbps | 96Gpbs |

Infiniband uses a hardware-offload protocol stack [3]. Extra memory copies that are sent from the application to an adapter can be avoided by the zero copy mechanism that optimizes the message transfer time. Moreover, Infiniband allows moving data from local memory to remote memory using RDMA (Remote Direct Memory Access), which allows the zero copy mechanism without involving the receiver host processor [2]. The number of user-kernel context switching and memory copies can be reduced by the direct access to the Infiniband HCA. Obviously, enabling communication between devices and hosts, without the traditional system resource overhead associated with network protocols, off-loads data movement from the server CPUs to the Infiniband HCA. Through virtual lanes (VLs), Infiniband offers traffic management, creating multiple virtual links within a single physical link that allows a pair of linked devices to isolate communication interference from other connected devices.

### *B. Intel Westmere Specifications*

Westmere is the code name for the latest in the series of multi-core processors by Intel. This is Intel's true hexa-core processor with L2 cache sharing and utilizing the revolutionary Quick Path Interconnect (QPI) architecture [15] that provides two separate lanes for the communication between the CPU and the chipset. The QPI technology allows the CPU to transmit and receive I/O data in parallel, as opposed to the traditional architecture using a single external bus where the external bus is used for both input and output operations reads and writes cannot be done at the same time. The latest version of the QPI works with a clock rate of 3.2 GHz, transferring two data per clock cycle (Double Data Rate), making the bus to work as if it was using a 6.4 GHz clock rate [15].

Further, Intel Westmere generation is equipped with Turbo Boost Technology [15] that automatically allows processor cores to run faster than the base operating frequency if it's operating below power, current, and temperature specification limits. This frequency change is dependent on the number of active cores, estimated current consumption, estimated power consumption and processor temperature. When the processor is operating below these limits and the user's workload demands additional performance, the processor frequency will dynamically increase by 133 MHz on short and regular intervals until the upper limit is met or the maximum possible upside for the number of active cores is reached.

### *C. MPI Implementations*

The Message Passing Interface (MPI) is the dominant programming model for parallel scientific applications. Given the role of the MPI library as the communication substrate for application communication, the library must ensure to provide scalability both in performance and in resource usage. In our experiments, we used two of the most commonly used MPI implementations in the HPC industry, which are MVAPICH2 and Intel MPI.

- Intel MPI

Based on MPI-2 specifications, Intel MPI Library focuses on making applications perform better on Intel architecture-based (AI) clusters. This MPI implementation has the ability to function on multiple HPC fabrics using an accelerated

universal, multi-fabric layer for fast interconnects via the Direct Access Programming Library (DAPL) methodology [15]. Thus, developers can deal with MPI codes, independent of the fabric, knowing that it will run on whatever fabric is chosen at runtime. In addition, Intel MPI supports various runtime environments and modules to integrate with other HPC job schedulers, such as Load Sharing Facility (LSF) by Platform Computing and Torque scheduler that is provided by Cluster Resources.

Further, the latest version of Intel MPI supports dispersive routing, which load-balances traffic among multiple pathways by using QLogic's prepraiaratory Performance Scaled Messaging (PSM) technology to automatically ensure that packets arrive at their destination for rapid processing. Dispersive Routing leverages the entire fabric for maximum communications performance for all jobs, even in the presence of other messaging-intensive applications.

- MVAPICH

Maintained by the Department of Computer Science and Engineering at Ohio State University, this implementation of MPI is based on MPICH and MVICH. MVAPICH [17] implementation is mainly known for its support for InfiniBand interconnect technologies as well as having high performance scalability support for clusters running thousands of cores. As for the Intel MPI, MVAPICH also supports various runtime environments such as SLURM and PBS.

## III. THE CLUSTER DESIGN

To perform benchmark evaluation, a DELL cluster of PowerEdge M610 Blade Servers was used. The cluster consisted of 512 nodes with dual sockets and Intel hexa-Core x5670 (Westmere) 2.93GHz processors. The operating system running on the nodes was RedHat Enterprise Linux Server 5.3 with the 2.6.18-128.el5 kernel. Each node was equipped with an Infiniband Host Channel Adapter (HCA) supporting 4x Quad Data Rate (QDR) connections with the speed of 32Gbps. Each node also had 24 GB (6 x 4GB) DDR3 1333Mhz of memory, thus the total amount of memory the system had was around 12 TB.

The physical layout of the cluster consisted of sixteen racks, each rack contains two chassis, and each chassis hosts up to 16 blade nodes. That is, each rack supports 32 nodes. From each node we had a 4x-QDR Infiniband connection going to a central 512-port Qlogic Infiniband switch. Figure 1 shows the Infiniband interconnection design as described. It is important to mention that this design is considered non-blocking as each node guarantees to have the full 4x QDR 32Gbps interconnect speed. This fast interconnect would drive the cluster to a higher utilization, which in theory, may affect the diskless concept.

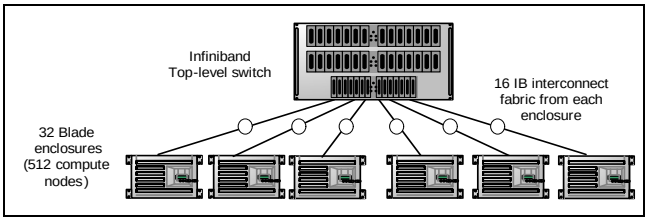


Figure 1. The DDR Infiniband interconnect topology of a 512 nodes cluster

Our Infiniband interconnect topology uses seven switches: A top-level switch and other 6 leaf switches. Under this configuration, IPC communication among nodes of 32 sub-clusters is localized to one leaf switch, but for the cluster of 512 nodes, the top-level switch is involved to support more nodes.

## IV. PERFORMANCE BASELINE USING LINPACK

In this section, we present our configuration to setup and perform (HPL) benchmark for our HPC cluster. We used "LINPACK" benchmark to measure and compare performance, while we used PALLAS benchmark to measure the interconnect performance of the cluster. The following two subsections describe these two packages in more details.

### *A. High Performance Linpack*

This is the first benchmarking tool used to make sure that the actual TFlops (trillion Floating Point Operations Per Second) of our HPC cluster is very close to the theoretical figure, which means that the cluster components (Infiniband interconnect, OS, firmware configuration …etc) are tuned perform at their best performance.

LINPACK is one of the standard benchmarking tools for HPC, is a collection of Fortran subroutines that analyze and solve linear equations and linear least-squares problems. The package solves linear systems whose matrices are general, banded, symmetric indefinite, symmetric positive definite, triangular, and tridiagonal square. In addition, the package computes the QR and singular value decompositions of rectangular matrices and applies them to least-squares problems. LINPACK uses column-oriented algorithms to increase efficiency by preserving locality of reference" [1].

The HPL benchmark uses Basic Linear Algebra Subprograms (BLAS), which is a collection of routines to perform basic vector and matrix operations. Therefore, the HPL benchmark performance heavily depends on the implementation of the BLAS package being used. In our evaluation, we used the version provided by Intel's Math Kernel Libraries (MKL) since it is the one recommended to be used with Intel's Westmere processor in order to make the most use of the processor's enhanced features.

Tuning the input file parameters for HPL can be a challenging task. For each cluster size that we were

evaluating, a different tuned HPL input file had to be generated. We describe next HPL's main input parameters that were of interest to us to tune. We also discuss our methods and criteria in choosing these input parameters. The four parameters of interest were *P, Q, N,* and *NB.*

The (*P* x *Q*) value represents the size of the computational grid that HPL resolves, which is equal to the number of processors the system has. We have noticed that the best performances were achieved when we chose the value of (*P* x *Q*) to be as "square" as possible as a grid shape, so we chose them to be approximately equal keeping in mind that *Q* needs to be slightly larger than *P*.

The next important parameter for HPL's input is "*N*", which is the size of the problem. Our goal was to find the largest problem size *N* that would fit in our system's memory and would gives us the tuned performance. We have chosen "*N*" to be close to our system's total memory size (in double precision 8 bytes), but keeping in mind not to make it equal to 100% of the memory size since some of the memory needs to be used by the system. It is important to notice that when we choose a small value for "*N*", this will result in not enough work performed on each CPU and will give us low performance results and low efficiency. While if we choose a value of "*N*" exceeding our memory's size, swapping will take place and the performance will go down. From our experiment with various values of *N*, the best performance was achieved when *N* was equal to 92% of the size of the system's total memory.

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of   Tennessee
   HPL.out      output file name (if any)
   6    device out (6=stdout,7=stderr,file)
   1    # of problems sizes (N)
   1179648 1179148      Ns
   1    # of NBs
   192  NBs
   0    PMAP process mapping (0=Row-,1=Column-major)
   1    # of process grids (P x Q)
   64   Ps
   96   Qs
   16.0 threshold
   1    # of panel fact
   1    PFACTs (0=left, 1=Crout, 2=Right)
   1    # of recursive stopping criterium
   8    NBMINs (>= 1)
   1    # of panels in recursion
   2    NDIVs
   1    # of recursive panel fact.
   1    RFACTs (0=left, 1=Crout, 2=Right)
   1    # of broadcast
   1  BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
   1    # of lookahead depth
   0    DEPTHs (>=0)
   2    SWAP (0=bin-exch,1=long,2=mix)
   64   swapping threshold
   0     L1 in (0=transposed,1=no-transposed) form
   0     U  in (0=transposed,1=no-transposed) form
   0    Equilibration (0=no,1=yes)
   8    memory alignment in double (> 0)
```

Figure2. The HPL file configuration with N value set to 92% of available memory

The last parameter that we discuss is "*NB*", which is the block size in our grid. Usually block sizes giving good results are within the (96, 104, 112, 120, 128, …, 256) range, and from our experimental runs, the value of 192 for *NB* has shown to give the best results compared to the various test runs we did with other values of *NB.* Figure 2 shows an example of the HPL input file that was used for our 512-node benchmark runs with the tuned input parameters.

To evaluate the performance of our system, the theoretical peak execution speed of the system had to be calculated in GFLOPS to know the maximum theoretical speed which cannot be exceeded. In the Top500 supercomputers terminology [17], the maximum theoretical system speed is referred to as "Rpeak", while "Rmax" is the actual speed obtained when HPL is run. The "Efficiency" of the system is the ratio of Rmax to Rpeak (Rmax/Rpeak). The efficiency can be affected by the underlying interconnect technology used for IPC communication among compute nodes, the amount of RAM available for individual compute nodes, as well as the MPI implementation used for communication among cluster nodes of the system.

For example, to calculate the theoretical Rpeak value for a system that consists of 512 nodes each with 12 Westmere cores (dual sockets per node) capable of 4 operations per cycle with a speed of 2.93GHz per core, the following formula is used:

$$\begin{aligned} Rpeak &= CPU\ Speed(GHz) \times Total\ Cores \times Ops/Cycles \\ &= 2.93 \times (12 \times 512) \times 4 \\ &= 72007\ \text{GFLOPS(theoretical)} \end{aligned}$$

Once we calculated the theoretical performance (Rpeak) for the system, we proceeded with running the HPL benchmark, using the corresponding HPL input file to get the actual performance (Rmax). The HPL binary was built over the Intel compilers version 11.1 and Intel's MKL libraries version 10.2.0.013 as this was the recommendation by Intel in order to make the most use of their Westmere processor's new features to enhance the performance. The HPL source code was used to build the HPL binary. The HPL Make file was pointing to the path of the Intel MPI binaries and libraries since it was the MPI implementation that was used for the evaluation. The Make file was also directed to use the Intel compilers and Intel MKL which has BLAS (Basic Linear Algebra Subprograms) as part of it. The use of Intel's MKL BLAS is an essential component for having successful HPL runs with high efficiencies on the Westmere processor. The standard Infiniband driver and DAPL that came with the RedHat EL 5.3 operating system were used.

### *B. The Intel MPI Benchmark (IMB)*

IMB 3.2 was used during this evaluation and it is the successor of PMB 2.2 from Pallas GmbH, Intel MPI Benchmarks 2.3, 3.0, and 3.1. This is a popular set of

benchmarks which provides an efficient way to measure the performance of some of the important MPI functions. It consists of three parts: IMB-MPI1, IMB-MPI2 and IMB-IO. We will focus on IMB-MPI1 which is used in our evaluation and it mainly replaces the formerly known Pallas benchmarks. The IMB-MPI1 benchmarks are classified into 3 groups, single transfer benchmarks, parallel transfer benchmarks, and collective benchmarks.

Single transfer benchmarks focus on measuring startup and throughput of a single message sent between two processes. For our evaluation we used the two benchmarks in this category, the ping-pong and ping-ping benchmarks. In ping-pong a process sends a single message to another process then the second process sends it back to the first process. As for the ping-ping benchmark, both processes send a message to each other at the same time.

Parallel Transfer benchmarks focus on calculating the throughput of concurrence messages sent or received by a particular process in a periodic chain. For our evaluation we used two benchmarks in this category, the sendrecv and exchange benchmarks. The sendrecv is based on the mpi_sendrecv function where each process in the communication chain sends to the process on its right and receives from the process on its left. In the exchange benchmark, each process exchanges data with both right and left process in the communication chain.

Collective benchmarks measure the time needed to communicate between a group of processes in different behaviors. There are several benchmarks of this category and the following is description of the collective benchmarks that was used in our evaluation:

- *Reduce:* each process sends a number to the root then the total number is calculated by the root.
- *Allreduce:* same as reduce but the final result is sent to all processes.
- *Scatter:* the root of the process sends a message to all processes. The size of the message equals to the chosen size * number of processes.
- *Reduce_scatter:* same as reduce but followed by scatter.
- *Gather:* all processes send the same message to the root.
- *Alltoall:* all processes send a message of a size equal to the chosen size * number of processes to all processes.
- *Bcast:* the root process broadcasts data to all processes.

## V. PERFORMANCE EVALUATION AND RESULTS

In this section, we discuss our measurement criteria and interpret the obtained IMB benchmark results. In order to evaluate the performance of the two implementations of the MPI, the benchmarks were run on the cluster starting with 8 and up to 6144 processes of the entire 512 nodes (remember that each node has 6x2 cores).

In figure 3, we used IMB Ping Pong test, which is the classical pattern for measuring startup and throughput of a single message sent between two processes. In this test, we compared the latency for the two different types of MPI; they are about the same (~200 ns). As the message size gets bigger (>128k), Intel MPI starts to pickup and match the performance of MVAPICH. Both are capable of delivering up to 3100MB/s with a message size of 16M. We notice a dip in performance when using Intel MPI at a message size of 64k due to caching effect.

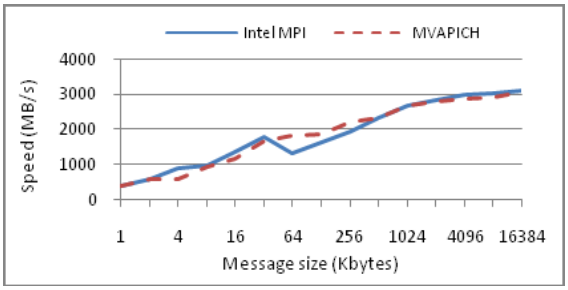


Figure 3. Ping Pong Test

In the Pallas send recev test, each process sends to the right and receives from the left neighbor in the chain. The turnover count is two messages sample (1 in, 1 out) for each process. It is observed that MVAPICH gets faster between message sizes 4 Kbytes to 256 Kbytes, as shown in figure 4. For larger message sizes, both MPI versions turn to be equal at around 1600MB/s.

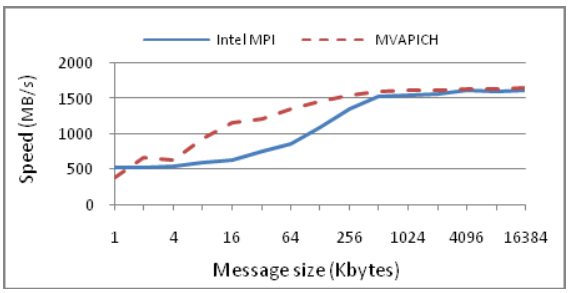


Figure 4. SendRecv Test

Pallas Exchange test is a communications pattern that often occurs in grid splitting algorithms. The group of processes is seen as a periodic chain, and each process exchanges data with both left and right neighbor in the chain. Figure 5 shows the Pallas Exchange test and it is observed that for large size message > 16MB, MVAPICH performance starts to decrease, matching the performance of Intel MPI. We can notice a dip in performance at message size of 16k. This is due to caching effect in exchange since each processor will have essentially 32k.

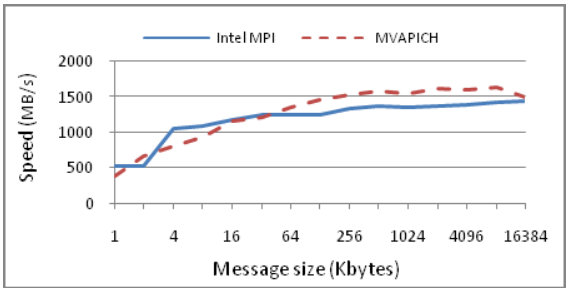


Figure 5. Exchange Test

The following set of tests measures the time needed to communicate between a group of processes in different behaviors. Figure 6 shows IMB Allreduce test. Allreduce reduces vectors of length L float items from every process to a single vector and distributes it to all processes. As shown in the figure, the time increases as we increase the message size for all type of interconnects. Intel MPI performs better when the message size exceeds 256KB.

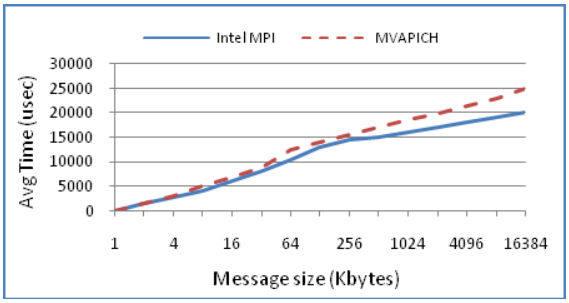


Figure 6. AllReduce Test

Reduce test, which also reduce vectors of length L float items from every process to a single vector but in the root process. The root of the operation is changed cyclically. Clearly, Intel MPI performs better when the message size exceeds 512KB, as shown in figure 8.

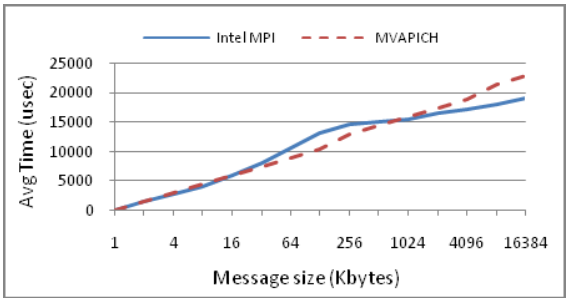


Figure 7. Reduce Test

Figure 8 shows the same case for Reduce Scatter test, which as well reduces vectors (of length float items) from every process to a single vector but, this time, the L items are split as evenly as possible between all processes.

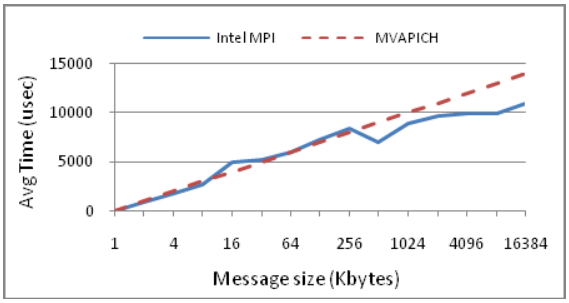


Figure 8. Reduce Scatter Test

On All Gather test, as in figure 9, every process sends X bytes and receives the gathered X*(#processes) bytes from the receivers.

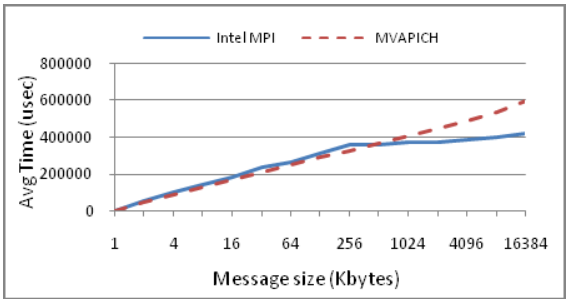


Figure 9. All Gather Test

In Pallas Bcast test, the root process broadcasts X bytes to all other processes, Intel MPI is still faster than MVAPICH, see figure 10. In particular, as we increase the message size, the number of performance difference increases. For large message size (16MB) we see the time to do bcast using Intel

MPI is about 15000 usec where the time using MVAPICH is about 18000 usec.

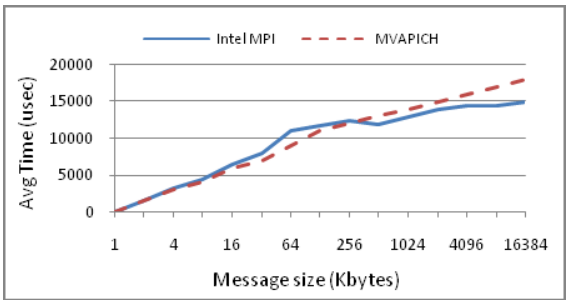


Figure 10. Bcast Test

## VI. CONCLUSION

Multi-core CPUs have been the path to continued reliance and benefits of Moore's Law while reining in the growth of power consumption and temperature rates. Specifically, recent trends in HPC systems have shown that future increases in performance can only be achieved through increases in system scale using a larger number of components, such as multi-cores and faster interconnects. In this paper, we evaluated a large-scale Infiniband cluster, equipped with Intel's latest Westmere processor using two MPI implementations. The paper presents the cluster configuration and evaluates its performance using High Performance LINPACK (HPL) and Intel MPI (IMB) benchmarks. Our results show that system scalability can still be achieved with up to 87% efficiency when considering the right combination of MPI, interconnect and CPU technologies. Further, our tests showed that in such a cluster, MVAPICH implementation excels in single-transfer communication where a single message is sent between two processes, while Intel MPI performs better in collective communication between groups of processes.

## ACKNOWLEDGMENT

The authors would like to thank King Fahd University for Petroleum and Minerals (KFUPM) and the EXPEC Computer Center (ECC) at Saudi Aramco for their invaluable support and contributions to conduct this research.

## REFERENCES

[1] R. AlShaikh, M. Ghuson, M. Baddourah, "Performance Evaluation of Myrinet and Cisco Infiniband Using Intel MPI Middleware", the 9th LCI International Conference on High Performance Computing, NCSA, Univerity of Illinois, USA, May 2008.

[2] V. Tipparaju, G. Santhanaraman, J. Nieplocha, and D. K. Panda, "Host-Assisted Zero-Copy Remote Memory Access Communication on InfiniBand", Int'l Parallel and Distributed Processing Symposium (IPDPS 04), April, 2004.

[3] R. Gupta, V. Sahasrabudhe, T. Sebastian and R. Ali, "An Introduction to DDR InfiniBand", Dell PowerSolutions, Aug 2007

[4] C. Bell, D. Bonachea, Y. Cote and et al. "An Evaluation of Current High-Performance Networks", Int'l Parallel and Distributed Processing Symposium (IPDPS'03), April 2003.

[5] J. Liu, B. Chandrasekaran, J. Wu and et al. "Performance Comparison of MPI Implementations over InfiniBand, Myrinet and Quadrics", Supercomputing, ACM/IEEE, pages 58- 58, Nov. 2003.

[6] Myrinet, Myricom. Available at: http://www.myri.com

[7] R. Fatoohi, K. Kardys, S. Koshy and el at. "Performance evaluation of high-speed interconnects using dense communication patterns", Parallel Computing Volume 32, Issue 11-12, pages 794-807, 2006.

[8] E. J. Kim, K. H. Yum, C. R. Das, M. Yousif, and J. Duato, "Performance Enhancement Techniques in InfiniBand Architecture," in Proceedings of the Ninth International Symposium on High-Performance Computer Architecture(HPCA-9), pp.253-262, February 2005.

[9] J. Liu, A. Mamidala, A. Vishnu and D. K. Panda. "Evaluating InfiniBand Performance with PCI-Express", IEEE Micro Volume 25, Issue 1, Pages 20-29, Feb 2005.

[10] E. J. Kim, K. H. Yum, and C. R. Das, "Performance Analysis of a QoS Capable Cluster Interconnect," Performance Evaluation, Volume 60, Issues 1-4, pp. 275-302, May 2005.

[11] T. Hoefler C. Viertel T. Mehlan F. Mietke W. Rehm "Assessing Single-Message and Multi-Node Communication Performance of InfiniBand", In Proceedings of IEEE International Conference on Parallel Computing in Electrical Engineering, PARELEC 2006, Bialystok, Poland, pages 227-232, IEEE Computer Society, Sep. 2006

[12] W. Yu, Q. Gao, and D. K. Panda. Adaptive Connection Management for Scalable MPI over Infiniband. In Proceedings of International Parallel and Distributed Processing Symposium (IPDPS) 2006, Rhodes Island, Greece, April 2006.

[13] M.Hussain, R. Gupta, T. Liu, "Using OpenFabrics InfiniBand for HPC Clusters", Dell PowerSolutions, Nov 2006

[14] Pallas GmbH. Available at: http://www.pallas.com

[15] Intel Inc. Available at: http://www.intel.com

[16] MVAPICH. Available at: http://mvapich.cse.ohio-state.edu/overview/mvapich

[17] The top500 supercomputers. Available at: http://www.top500.org

[18] MVAPICH: MPI over InfiniBand and iWARP. Available at: http://mvapich.cse.ohio-state.edu