Using the Intel MPI Benchmarks (IMB) to Evaluate MPI Implementations on an Infiniband Nehalem Linux Cluster

Ahmed Bukhamsin, Mohamad Sindi, Jallal Al-Jallal
EXPEC Computer Center
Saudi Aramco
Dhahran, Saudi Arabia
Email: {bukham0e, sindimo, jallalja}@aramco.com

Keywords:
IMB, Infiniband, MPI, Nehalem, Saudi Aramco

Abstract

According to Moore's Law, computer speeds are expected to double approximately every 2 years. But with the current challenges that computer manufacturers are facing to double speeds of individual processors, due to various reasons, i.e., processor temperatures; multiprocessor architectures have become more popular nowadays. Eventually, this has led to an increased interest in standards for writing parallel applications. The Message Passing Interface (MPI) has become the de facto standard for writing applications to run on such systems. Our growing needs for the latest high performance computing solutions in Saudi Aramco, the world's largest oil producing company, has given us the opportunity to evaluate three of the most commonly used MPI implementations, MVAPICH2, Open MPI, and Intel MPI on Intel's latest Nehalem processor. In this paper, we describe our test bed environment along with the evaluations that we did using some of the Intel MPI Benchmarks (IMB). We discuss the results in terms of bandwidth speed, execution time, and scalability when running on up to 512 processors on an Infiniband Nehalem Linux cluster with 64 nodes using the three implementations of MPI. We finally state our conclusions, recommendations, and future work.

1. INTRODUCTION

The Message Passing Interface (MPI) has become the de facto standard nowadays for writing applications to run on multi-core systems, such as Linux High Performance Computing (HPC) clusters. We at Saudi Aramco, the world's largest oil producing company [1], were given the opportunity to evaluate three of the most commonly used MPI implementations in the HPC industry, which are MVAPICH2, Open MPI, and Intel MPI. We had the privilege of performing the evaluation on a Linux cluster consisting of 64 nodes and having Intel's latest Nehalem processor along with Infiniband as the main interconnectivity technology. For the evaluation, we decided to run some of Intel MPI benchmarks (IMB) which are designed to assess the performance of the most important MPI functions [2].

The rest of the paper is organized as follows: In section 2, we clarify the drive behind this paper. Section 3 goes over the basics of the MPI standard and presents the three various MPI implementations that were evaluated. In section 4, we describe the Intel MPI Benchmarks package and the benchmarks that were run in our evaluation. In section 5 we present our cluster test bed design, and section 6 describes our experimental evaluations and shows the benchmark results and our interpretations of it. We state our conclusion and future work in the last section.

2. MOTIVATION

Ever since the petroleum industry started using oil reservoir simulations in the oil and gas industry, some of its biggest challenges have been the delivery of high performance computational power for such simulations. Today, our exploration computer center in Saudi Aramco is home for thousands of Linux clustered nodes with some of the latest technologies in the HPC industry, such as Intel's Nehalem processors, along with some of the latest interconnectivity technologies, such as Infiniband and Myrinet. By utilizing these combinations, our exploration computer center is able to process more data than ever before, thereby supporting our company's strategy to stabilize the energy supply to the world for many years to come. The main goal driving our evaluation activity was to search for the best performing MPI implementation in the HPC industry using the latest Nehalem processor and the IMB benchmarks.

The evaluation was performed on-site at Saudi Aramco's EXPEC (Exploration and Petroleum Engineering Center) Computer Center and the benchmarks were performed one MPI implementation at a time using a combination of up to 512 processors of a 64 nodes Linux Nehalem cluster with Infiniband as its main interconnectivity technology.

3. MPI STANDARD/IMPLEMENTATIONS

MPI is considered to be a complex system that is composed of more than 200 functions that provide various functionalities to parallel programs [3-4]. Several implementation of MPI exist and in this paper we have chosen to evaluate three of the most popularly used implementations, which are MVAPICH2 1.4rc2, Open MPI 1.3.3 and Intel MPI 3.2.1.

The MVAPICH2 implementation is confirming to MPI 2.1 standards and is based on two previous implementations of MPI which are MPICH2 and MVICH. This implementation is mainly known for its support for Infiniband interconnect technologies. It works with various operating systems including Linux and it is open source under the BSD license [5].

The Open MPI implementation is based on the MPI-2 standard and is resulted of merging LAM/MPI, LA-MPI and FT-MPI. This implementation works with various interconnects such as Myrinet and Infiniband. It supports most Unix/Linux operating systems as well as Windows and it is open source under the BSD license [6-7].

The Intel MPI implementation is based on the MPICH-2 implementation and it focuses on enhancing application performances on Intel Architecture (IA) based clusters. It supports various interconnects and allows users to switch between them easily through the use of the Direct Access Programming Library (DAPL), which is an interface that is network-independent. It mainly works with the Linux and Windows operating systems and requires a commercial license [8-9].

4. INTEL MPI BENCHMARKS (IMB) [8]

IMB 3.2 was used during this evaluation. It 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 focused on IMB-MPI1 in our evaluation which mainly replaces the formerly known Pallas benchmarks. The IMB-MPI1 benchmarks are classified into three 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. 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 the sendrecv and exchange benchmarks. Collective benchmarks measure the time needed to communicate between a group of processes in different behaviors. For our evaluation we used the reduce, allreduce, scatter, reduce_scatter, gather, alltoall, and bcast benchmarks.

5. CLUSTER DESIGN

A DELL cluster of PowerEdge M610 Blade Servers was used for the evaluation. It consisted of 64 nodes with duel sockets and Intel QuadCore X5570 (Nehalem) 2.93 GHz processors. The operating system used was Red Hat Enterprise Linux Server 5.3 with the 2.6.18-128.el5 kernel. Nodes were equipped with an Infiniband host channel adapter (HCA) supporting 4x-DDR connections between the nodes. Each node also had 12 GB memory.

A master node in the cluster was sharing a Linux EXT3 file system as a network file system (NFS) among the 64 nodes of the cluster. This file system contained the home directory of the test user that was launching the benchmarks as well as the IMB binaries, Intel compilers, Intel MKL libraries, and the binaries of the three MPI implementations that we were evaluating.

As a hardware layout, the cluster consists of three racks, each rack contains two chassis, and each chassis can host up to 12 blade nodes. A central 144-port Qlogic Infiniband switch was is in a rack by itself. Figure 1 shows the hardware cluster design as described.

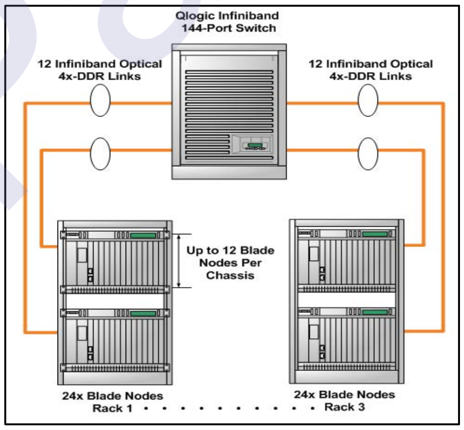

Figure 1. Cluster Design Layout.

6. RESULTS AND PERFORMANCE MEASUREMENTS

In this section, we discuss our measurement criteria and interpret the IMB benchmark results obtained.

In order to evaluate the performance of the three implementations of MPI, the benchmarks were run on the cluster starting with eight processors of one node and up to 512 processes of the entire 64 nodes. The benchmarks were run multiple times and the average results were calculated. The "mpirun" command was used to launch the IMB benchmarks on the various size systems using the various MPI implementations one at a time. The size of the communication message was set to one megabyte (MB).

Table 1 shows the bandwidth and time taken by a 1 MB message to be transferred in ping-pong and ping-ping runs. Open MPI and MVAPICH2 outperformed Intel MPI in the bandwidth and execution time, which is eventually reflected in the results of the other benchmarks performed.

Table 1. Single Transfer Benchmarks Results.

MPI Implementation	Ping-Pong (MB/s)	Ping-Pong (seconds)	Ping-Ping (MB/s)	Ping-Ping (seconds)
Intel MPI	2,775.96	686.08	1,457.56	686.08
MVAPICH2	6,377.59	156.8	3,709.56	269.57
Open MPI	8,267.08	120.96	4,339.35	230.45

In Figure 2 we have plotted the bandwidth in MBs per second of the results obtained from running the sendrecv benchmark. The figure shows that the bandwidth of MVAPICH2 in parallel transfers is slightly better than Open MPI, while Intel MPI shows a lower and almost fixed bandwidth regardless of the number of processors being used.

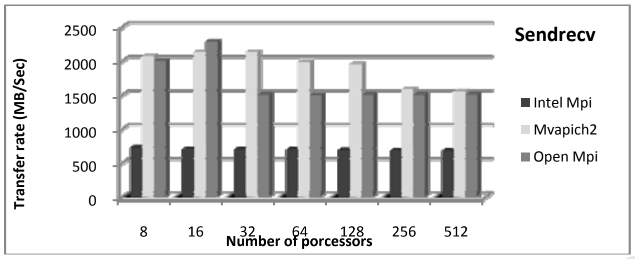

Figure 2. Results of the Sendrecv Benchmark.

The exchange benchmark shows silmilar results to sendrecv except that Open MPI is slightly better than MVAPICH2 in most cases as illustrated in Figure 3.

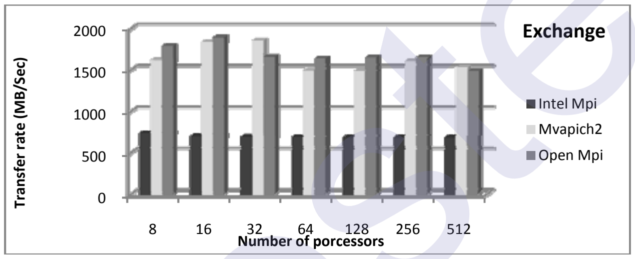

Figure 3. Results of the Exchange Benchmark.

As for the collective benchmarks, the performance of the MPI implementations varied. As illustrated in Figure 4, the reduce runs showed that Open MPI performed the worst while noticing that the time taken by each run increased gradually as we increased of the number of processors. On the other hand, MVAPICH2 and Intel MPI were showing better and more consistent results as the number of processors involved grew larger.

Figure 5 shows that the scatter benchmark performances of the three MPI implementations when running on 8 to 32 processors are close to each other and Open MPI is the best of all, followed by Intel MPI then MVAPICH2. However, the results for Open MPI change after exceeding 32 processors where it becomes the worst out of all and it scales poorly.

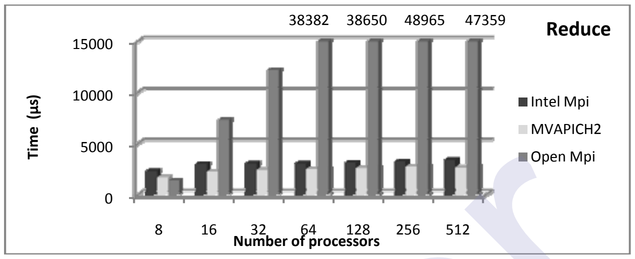

Figure 4. Results of the Reduce Benchmark.

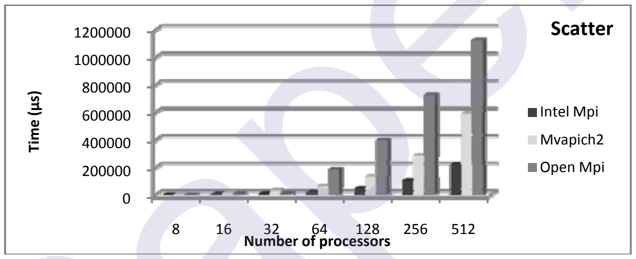

Figure 5. Results of the Scatter Benchmark.

For the gather benchmarks, it is noticeable in Figure 6 that the execution time of Intel MPI and Open MPI is almost the same until we reach 256 processors where Intel MPI starts showing better scalability. MVAPICH2 showed the worst performance compared to the others.

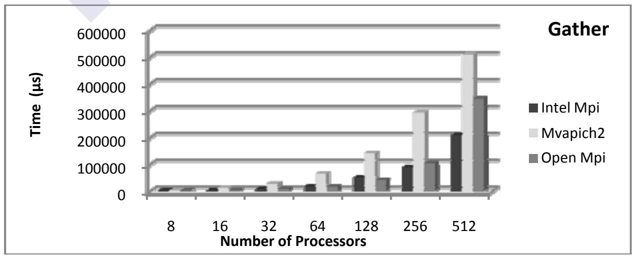

Figure 6. Results of the Gather Benchmark.

As for the alltoall benchmarks, Figure 7 shows that all the MPI implementations had the same behavior as the number of processors increased and they maintain almost identical execution times until Open MPI started becoming the worst performer with the 512 processors.

Among all three MPI implementations, the execution time of the bcast benchmark for Open MPI was showing the best results and scalability with a slight increase in time as the the number of processes increased. MVAPICH2 came in second place until we exceeded 256 processors where its execution time became the worst among all three. Intel MPI is not showing a regular behavior, its execution time keept increasing up to 32 processors until it hit its highest of 5,143 micro-seconds then it went down to 3,156 micro-seconds with the 64 processors run, and after that it started increasing again as illustrated in Figure 8.

As for the results of the allreduce benchmark in Figure 9, the execution time for Intel MPI and MVAPICH2 is consistence with the execution time of the bcast and reduce benchmarks, meaning that the total time of bcast plus reduce equals to the allreduce execution time for every run. This is because the allreduce benchmark can be looked at as reduce followed by bcast. But for Open MPI, the allreduce execution time was interestingly better than the execution time of the reduce benchmark where the execution time of allreduce was 16% less than the execution time of reduce. That could be due to the use of the Modular Component Architecture (MCA) and the advanced algorithm selection functions used in the Open MPI implementation which chooses the best available algorithm for communication based on the communication pattern of the application, network topology, and data size [10]. This implies that the algorithms used to implement the more sophisticated MPI functions in Open MPI are not necessarily based on the algorithms used for the primitive MPI functions.

The same explanation for the allreduce benchmark is also applicable to the results of the reduce_scatter benchmark. The results for Open MPI were the best among all three as seen in Figure 10, while Open MPI was the worst during the reduce benchmark as indicated previously in Figure 4.

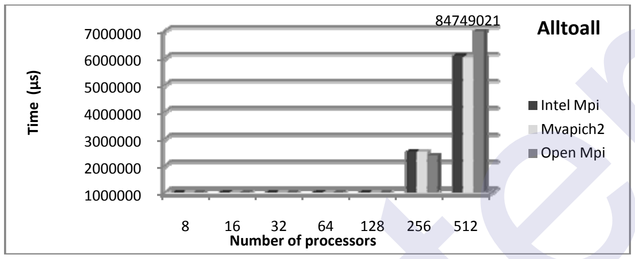

Figure 7. Results of the Alltoall Benchmark.

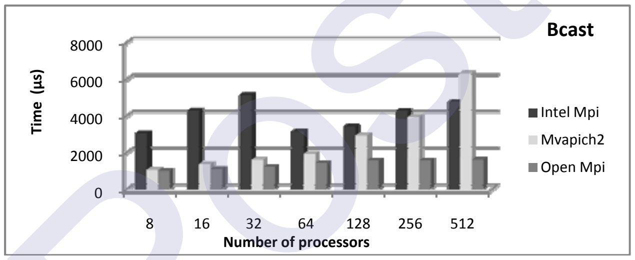

Figure 8. Results of the Bcast Benchmark.

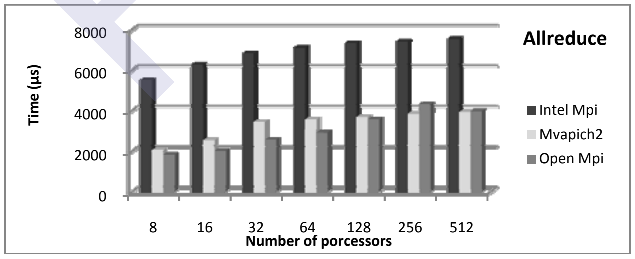

Figure 9. Results of the Allreduce Benchmark.

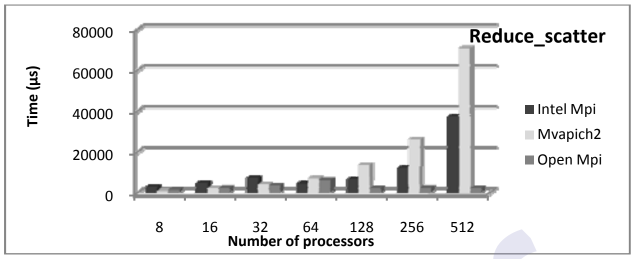

Figure 10. Results of the Reduce_Scatter Benchmark.

CONCLUSION

In this paper we evaluated three of the most popularly used MPI implementations using IMB. Our results show that for a cluster of size 64 nodes, there is no absolute winner. Each MPI implementation performed well with some of the benchmarks, and not so well with others. Therefore the decision of which MPI to use depends on the nature of the application being run. For applications that tend to use allreduce communications frequently, Open MPI or MVAPICH2 would be the choice. For applications that perform frequent gather, scatter, reduce, and alltoall communications, Intel MPI or MVAPICH2 would be the choice. For applications that demand frequent bcast and reduce scatter communications, Open MPI gave the best results. Scalability wise, Intel MPI scaled well in most cases followed by MVAPICH2. Open MPI didn't scale well and it gave the worst results for large runs of the alltoall, reduce, and scatter benchmarks. Our future work includes more scalability evaluations of the MPI implementations on a larger 512 nodes cluster.

REFERENCES

[1] Saudi Aramco Oil Company
http://www.saudiaramco.com

[2] "*Intel MPI Benchmark User Guide and Methodology Description,*" Version 3.2, Intel Corporation.

[3] Ian Foster, "*Designing and Building Parallel Programs,*" Addison Wesley.

[4] William Gropp, Ewing Lusk, Anthony Skjellum: "*Using MPI: Portable Parallel Programming with the Message-Passing Interface,*" MIT Press.

[5] D. K. Panda., "*MVAPICH 1.0 User and Tuning Guide,*" Network-Based Computing Laboratory, Ohio State University.

[6] R.L. Graham, G.M. Shipman, B.W. Barrett, R.H. Castain, G. Bosilca, A. Lumsdaine. "*Open MPI: A High-Performance, Heterogeneous MPI,*" IEEE Conference on Cluster Computing 2006.

[7] Edgar Gabriel, Graham E. Fagg, George Bosilca, Thara Angskun, Jack J. Dongarra, Jeffrey M. Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, Ralph H. Castain, David J. Daniel, Richard L. Graham, Timothy S. Woodall 2004. *"Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation."*European PVM/MPI Users' Group Meeting 2004.

[8] "*Intel MPI Library for Linux OS Reference Manual,*" Revision 3.1, Intel Corporation.

[9] Lei Cha, I. Ranjit Noronha, Dhabaleswar K. Panda, "*MPI over uDAPL: Can High Performance and Portability Exist Across Architectures,*" Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid.

[10] Jeffrey M. Squyres, Andrew Lumsdaine 2004. *"The Component Architecture of Open MPI: Enabling Third-Party Collective Algorithms.* 18th ACM International Conference on Supercomputing.