Journal of Physics: Conference Series **331** (2011) 062012 doi:10.1088/1742-6596/331/6/062012

Message Passing Framework for Globally Interconnected Clusters

M Hafeez [2], S Asghar[1], U A Malik[1], A Rehman[1] and N Riaz[2]

[1]ASC, Advanced Scientific Computing, National Centre for Physics, Quaid-i-Azam University Campus, Islamabad, Pakistan
[2]Department of Computer Science, SZABIST, Shaheed Zulfikar Ali Bhutto Institute of Science and Technology, Plot 67, Street 9, H-8/4, Islamabad, Pakistan

Mehnaz.Hafeez@cern.ch, Sajjad.Asghar@ncp.edu.pk, Usman.Malik@ncp.edu.pk, Adeel.Rehman@ncp.edu.pk, N.R.Ansari@szabist-isb.edu.pk

Abstract. In prevailing technology trends it is apparent that the network requirements and technologies will advance in future. Therefore the need of High Performance Computing (HPC) based implementation for interconnecting clusters is comprehensible for scalability of clusters. Grid computing provides global infrastructure of interconnecting clusters consisting of dispersed computing resources over Internet. On the other hand the leading model for HPC programming is Message Passing Interface (MPI). As compared to Grid computing, MPI is better suited for solving most of the complex computational problems. MPI itself is restricted to a single cluster. It does not support message passing over the internet to use the computing resources of different clusters in an optimal way. We propose a model that provides message passing capabilities between parallel applications over the internet. The proposed model is based on Architecture for Java Universal Message Passing (A-JUMP) framework and Enterprise Service Bus (ESB) named as High Performance Computing Bus. The HPC Bus is built using ActiveMQ. HPC Bus is responsible for communication and message passing in an asynchronous manner. Asynchronous mode of communication offers an assurance for message delivery as well as a fault tolerance mechanism for message passing. The idea presented in this paper effectively utilizes wide-area intercluster networks. It also provides scheduling, dynamic resource discovery and allocation, and sub-clustering of resources for different jobs. Performance analysis and comparison study of the proposed framework with P2P-MPI are also presented in this paper.

1. Introduction

Grids provide an effective and efficient way to perform complex computations, especially in science and research. Grids are dynamic in nature and provide access to the computing resources such as; systems, data, applications and services over the Internet. Along with the increase in computing power Grids also support the voluminous data exchange. Many popular grid systems are running in the world successfully. These grid systems are generally specific to certain scientific application. Grids are not able to replace clusters because clusters are generic in nature and they provide computational resources to a range of different applications which may execute in parallel. Today, MPI is a de facto standard to provide parallel code execution on clusters. MPI itself is not specific to the applications and projects. It can be used to solve a variety of computational problems using parallel programming

paradigm on homogenous clusters. However MPI is static in its nature. It uses a static set of hardware resources which are defined in an initial configuration file. Therefore static and homogenous model of MPI is not compatible with dynamic and heterogeneous environment of Grid where availability of different type of resources keeps changing. Moreover MPI applications are Operating System dependent. Thus, Java becomes an excellent basis for achieving portability of applications.

MPI programming on grids may require the use of several communication paradigms. Current grid applications are tightly coupled with the communication layer implementations. Furthermore running distributed message-passing applications on heterogeneous environment over internet is difficult, as these applications have to deal with multiple communication interfaces, low-level protocols, data encodings, data compressions and quality of service in order to achieve acceptable performance. Also a parallel MPI code expects low latency and high bandwidth communication while communication between MPI processes on different clusters shows high latencies and small bandwidth consumption. Therefore to design MPI applications for Grids is a complex task.

This paper presents A-JUMP to support interconnected clusters communication. The proposed framework supports the dynamic behaviour of interconnected clusters with MPI functionality. It is organized as follows: section 2 gives synopsis of A-JUMP; section 3 presents a detailed component level description of A-JUMP for interconnected clusters and section 4 covers the literature overview of the related work. In section 5 functionality tests and performance results are presented, while the concluding remarks and future directions are covered in section 6.

2. A-JUMP Framework

Architecture for Java Universal Message Passing (A-JUMP) provides explicit parallelism for distributed memory multi-core systems. A-JUMP is a pure Java implementation [12], [13]. The backbone of A-JUMP is HPC bus which provides the interoperability between different hardware resources, communication protocols and mediums. The HPC bus is built upon well-established industry standards; Java Messaging Service (JMS) and Java programming language. In HPC bus, ActiveMQ is responsible for communication and message passing in an asynchronous manner. It also provides the facility of message persistence to support fault tolerance in case messages are lost. The HPC bus ensures that any changes in communication protocol and network topology will remain transparent to the end users.

3. A-JUMP for interconnected clusters

In this paper we propose the implementation of A-JUMP for interconnected clusters to run parallel code with the heterogeneous network layouts and multiple underlying communication protocols. Figure 1 illustrates the layered architecture of A-JUMP for interconnected clusters.

A-JUMP for interconnected clusters provides facility to write parallel MPI code using different programming languages. It offers message exchange between different languages. The current implementation is in Java. A framework on internet must be dynamic in nature to take on changes in computing resources. A-JUMP dynamically adopts the hardware as well as software changes in computing resources. Current MPI implementations share executable and data among the machines. Thus it is needed for every newly added resource to the cluster to get configured for a specific job and add the required executable and data on it. A-JUMP distributes executable and data together at the job execution time. Therefore executable and the data are not required to be a part of cluster. A-JUMP provides sub-clustering for resource allocation of a job. This prevents any single job to occupy all the distributed resources over interconnected clusters. A-JUMP supports different network topologies over LAN, WAN, P2P networks and Grids. A-JUMP supports different communication protocols for different networks using HPC bus. These protocols may include HTTP, TCP and UDP etc.

3.1. Components of A-JUMP for interconnected clusters

Details and relationship between the components of the A-JUMP for interconnected clusters is shown in figure 2.

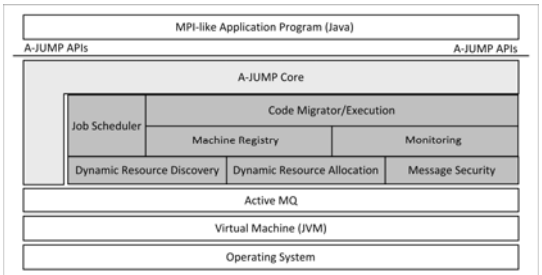

Figure 1. Layered architecture of A-JUMP for interconnected clusters.

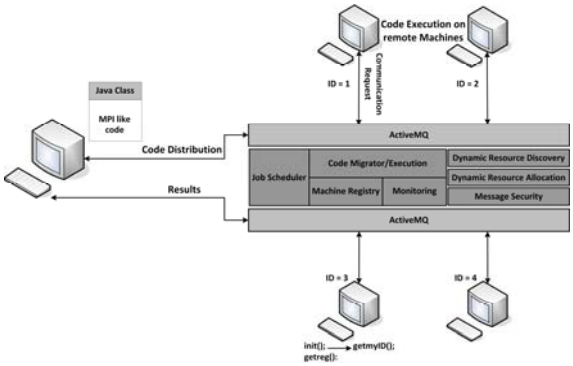

Figure 2. Components of A-JUMP for interconnected clusters.

The Dynamic Resource Discovery (DRD) keeps track of the resources that are the part of A-JUMP clusters. It gets the information from Registry to index the available machines that are registered with the Machine Registry. The DRD fetches the information about the state of the machines from Monitoring. The states are defined as free and busy. The Dynamic Resource Allocation (DRA) finds the availability of free resources on A-JUMP clusters. Client sends the request for the required numbers of free processes that are needed to run the job. DRA searches the free available resources from the index maintained by DRD. It creates the list of the machines which will be the part of the sub-cluster required by the user. The Message Security provides the facility to encrypt the outgoing messages. Current implementation is based on simple symmetric encryption technique.

The Job Scheduler distributes and redirects the incoming jobs according to the sub clustering information defined by DRA. The Monitoring component records number of busy and free CPUs while the Machine Registry maintains a list of available machines in clusters dynamically. The HPC Bus is the communication backbone and is based on ActiveMQ. The selective and collective communications between the tasks as well as distribution of code are done in an asynchronous mode through HPC bus. A built-in communication API is a part of this component that enables the tasks to do point to point, selective and collective communication.

4. Related Work

This section presents the critical evaluation and comparison of A-JUMP for interconnected clusters with the reviewed literature. Unlike A-JUMP which is a pure Java implementation, mpiJava[1] is based on the native C++ binding to MPI. Moreover it is a Java wrapper to access MPI functionality written in C++ from Java code. MPJ Express [2] supports two types of communication libraries; java.nio and native interfaces to access Myrinet protocol whereas the communication layer of A-JUMP is independent of application code. F-MPJ [3] provides support for shared and distributed memory models for fast communication. The communication layer is based on Java Fast Sockets (JFS). F-MPJ framework is tightly coupled with its communication libraries. On the other hand, A-JUMP framework is loosely coupled with its communication layer. P2P-MPI [4] is written to support grids for message passing. P2P-MPI and A-JUMP almost provide similar functionality for interconnected clusters. P2P-MPI uses a customized peer-to-peer infrastructure, which is based on concept of SuperNode. While A-JUMP is designed for interconnected clusters provides message passing using multiple protocols.

JMPI [5] has a communication layer based on RMI & Java sockets while A-JUMP communication layer is based on asynchronous ActiveMQ messaging. The network layer of MPJ/Ibis [6] relies on native code for performance improvement. Like the proposed framework, it is highly scalable. MPI implementation of MPICH, MPICH-G2 and PACX-MPI support heterogeneous computing environment with different design considerations of MPI implementations such as MPICH offers

flexible communication method, MPICH-G2 supports Grid environment and PACX-MPI provides extensibility for future computer architecture.

MagPIe [7] and G-JavaMPI [8] are based on MPICH and MPICH-G2 respectively. Therefore, both implementations inherit the shortcomings of MPICH and MPICH-G2 libraries. Unlike MagPIe and G-JavaMPI, A-JUMP for interconnected clusters has support for multiple vendor provided implementations as HPC bus of A-JUMP is based on JMS standards. PACX-MPI [9] and MPI/Madeleine III [10] both have limitations of scalability whereas it is one of the important features of the proposed model. GridMPI [11] is based on MPI-1.2 specification while A-JUMP follows MPJ standards. For interconnected cluster message passing, GridMPI uses IMPI whereas A-JUMP message passing communication is based on ActiveMQ using Java. Therefore, A-JUMP provides heterogeneity at protocol level as well as at operating system level. Thus A-JUMP for interconnected clusters becomes distinct among other related MPI like frameworks. The other features include; ease of use for writing parallel code, support for hybrid architectures, scalability, security and asynchronous communication.

5. Performance measurement

Performance measurement analysis is presented using the two broader categories of HPC standard benchmark tests. Code Speed-up performance analysis is done using; embarrassingly parallel (EP) benchmark and JGF crypt benchmark. Communication performance measurement includes ping pong latency test and network throughput measurement.

5.1. Test environment

Results are collected on an environment that simulates globally interconnected clusters. The hardware resources involved in conducting the tests include; 9 single core machines with 3.2GHz single core CPUs, 1GB RAM and a Gigabit LAN interface; divided into two clusters. The clusters are connected with 1 Gbps bandwidth. The machines are running Windows Server 2003 (with SP2) and Windows XP (with SP3). The TCP window size is default on all the machines in clusters. There is no optimization done at the level of OS as well as at the level of hardware.

5.2. Code speed-up performance analysis

The comparison of EP benchmark performance results of P2P-MPI and A-JUMP for interconnected clusters is given in Figure 3. It shows that A-JUMP performs better than P2P-MPI. Figure 4 evidently demonstrates the performance gain with the increase in the number of CPUs. The test results for JGF Crypt are compiled for two data sizes; Class A contains 3,000,000 whereas Class B comprises of 20,000,000 integer data elements respectively. The results presented in Figure 5 and 6 illustrate the performance comparison with dataset type A & B discretely. For class A dataset on two and four CPUs A-JUMP has performed better than P2P-MPI.

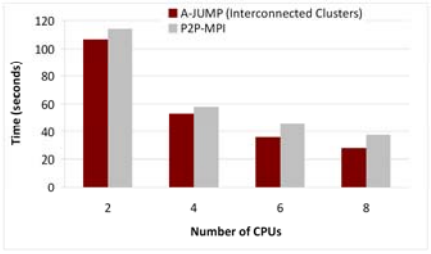

Figure 3. NAS EP benchmark results.

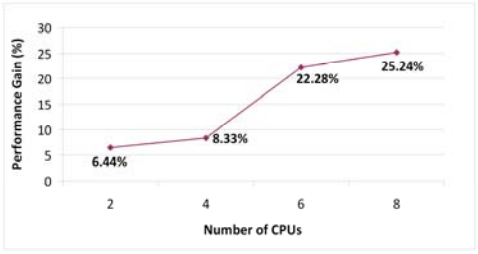

Figure 4. Performance gain of A-JUMP.

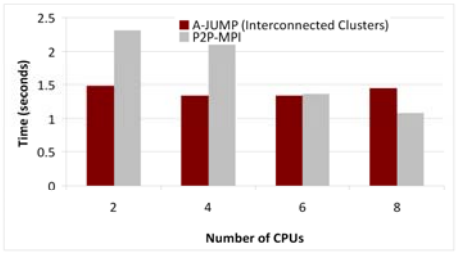

Figure 5. JGF crypt benchmark for class A.

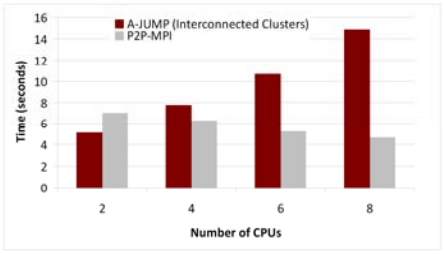

Figure 6. JGF crypt benchmark for class B.

On six and eight CPUs P2P-MPI has shown better performance. Whereas for the larger datasets of class B, P2P-MPI has shown better performance than A-JUMP. In code speed-up performance analysis, it is observed that for EP benchmark tests the proposed framework performs faster than P2P-MPI. Whereas, JGF Crypt results for P2P-MPI are better than the A-JUMP. It is due to the fact that the current implementation of the A-JUMP is not optimized to handle the communication of the large message sizes. This limitation is imposed by the ActiveMQ that can not send large object messages efficiently.

5.3. Communication performance measurement

For communication performance measurement, tests have been performed are; ping pong latency test and network throughput measurement. Ping pong communication test presents latency results on unidirectional sending and receiving of messages. It is repeated 1000 times for various message sizes. Figure 7 shows the comparison of ping pong latency between P2P-MPI and A-JUMP for interconnected clusters. For smaller message sizes, A-JUMP has lesser network latencies. However if message size increases A-JUMP performs comparable to the P2P-MPI. This demonstrates that even though there are some network communication overheads due to ActiveMQ, still A-JUMP for interconnected clusters shows promising results. Network throughput measurement shows how efficiently the framework consumes available network bandwidth. The network throughput results in Figure 8 show that the A-JUMP has better performance for various message sizes. The difference between the performance of A-JUMP and P2P-MPI narrows down as message size is increased to 256K.

Current implementation of A-JUMP has a limitation to send message greater than 64k in size efficiently. We expect that the future implementation will yield better performance for standard benchmark tests using larger message sizes. Current results are for point-to-point and collective communication paradigms. The results are promising and comparable with P2P-MPI with the message sizes up to 256k.

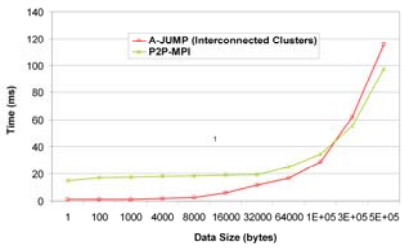

Figure 7. Ping pong latency measurements.

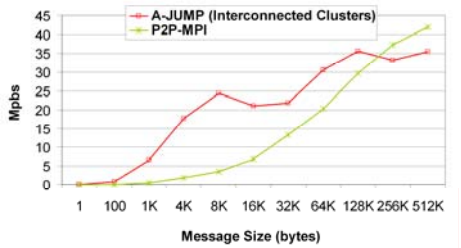

Figure 8. Network throughput measurements.

6. Conclusion and future work

A-JUMP for interconnected clusters supports both homogeneous and heterogeneous clusters for parallel code execution. It segregates the communication layer from the application code and offers

message exchange between different languages. The current implementation is in Java. In future C# and Python will also be incorporated to provide support for cross-language message exchange between different languages. Moreover it performs dynamic resource management, scheduling and sub-clustering of resources for different jobs. Its communication is based on ActiveMQ that offers transparency for protocol conversions for the machines running with different communication protocols. It provides message security through encryption. A set of easy to use APIs are part of A-JUMP for interconnected clusters.

We have also shown code speed-up, ping pong latency, and network throughput measurement results and their comparison with P2P-MPI. Current results are for point-to-point and collective communication paradigms. The results are promising and comparable with P2P-MPI with the message sizes up to 256k. Future work includes; improving the communication library for message sizes greater than 256k, support for different security techniques, multilingual support, improve network latencies, and lessen the communication overheads of A-JUMP for interconnected clusters. In addition, we intend to explore applications, which may use the proposed framework.

7. References

[1] Baker M, Carpenter B, Fox G, Ko S, and Lim S 1999 mpi-Java: an Object-Oriented Java Interface to MPI *Lecture Notes in Computer Science* **1586** 748–62

[2] Shafi A, Carpenter B, and Baker M 2009 Nested Parallelism for Multi-core HPC Systems using Java *Journal of Parallel and Distributed Computing* **69(6)** 532-45

[3] Taboada G L, Tourino J, and Doallo R 2009 F-MPJ: scalable Java message-passing communications on parallel systems *Journal of Supercomputing*

[4] Genaud S and Rattanapoka C 2007 P2P-MPI: A Peer-to-Peer Framework for Robust Execution of Message Passing *Parallel Programs Journal of Grid Computing* **5(1)** 27-42

[5] Bang S and Ahn J 2007 Implementation and Performance Evaluation of Socket and RMI based Java Message Passing *Systems Proc. 5th Intl. Conf. on Software Engineering Research, Management and Applications (Busan, Korea 2007)* pp 153-59

[6] Bornemann M, Nieuwpoort R V v and T. Kielmann MPJ/Ibis: A Flexible and Efficient Message Passing Platform for Java *Proc. 12th European PVM/MPI Users' Group Meeting (Sorrento Italy 2005) (Lecture Notes in Computer Science* vol 3666**)** pp 217-24

[7] Kielmann T , Hofman R F H , Bal H E , Plaat A and Bhoedjang R A F 1999 MagPIe: MPI's collective communication operations for clustered wide area systems *ACM SIGPLAN Notices* **34(*08)*** 131-40

[8] Chen L, Wang C, Lau F C M and Ma R K K A 2003 Grid Middleware for Distributed Java Computing with MPI Binding and Process Migration Supports *Journal of Computer Science and Technology* **18(04)** 505-14

[9] PACX-MPI http://www.hlrs.de/organization/av/amt/research/pacx-mpi [Online]

[10] Aumage O Heterogeneous multi-cluster networking with the madeleine III communication library *16th IEEE International Parallel and Distributed Processing Symposium (IPDPS '02 (IPPS & SPDP) (Washington - Brussels – Tokyo 2002)* pp 85

[11] GridMPI: http://www.gridmpi.org/index.jsp [Online]

[12] Hafeez M, Asghar S, Malik U, A Rehman A and Riaz N Survey of MPI Implementations *International Conference on Digital Information and Communication Technology and its Applications (DICTAP 2011), Dijon, France, CCIS/LNCS* vol 167 pp 1011–1025

[13] Asghar S, Hafeez M, Malik U A, Rehman A and Riaz N A-JUMP Architecture for Java Universal Message Passing *Proc. FIT 2010 (Islamabad, Pakistan 2010)* doi:[10.1145/1943628.1943662]