Late Papers

Implementation of MPI over HTTP

S.Lakshminarayanan
lakme@mecna.iitm.ernet.in
Dept. of Computer Science
Indian Institute of Technology
Madras, India

S.S.Ghosh N.Balakrishnan
ghosh@serc.iisc.ernet.in balki@serc.iisc.ernet.in
Supercomputer Education and Research Center
Indian Institute of Science
Bangalore, India.

Abstract

Message Passing Interface[2] is the de facto standard for multi-computer and cluster message passing. In this paper we explore a new paradigm of high performance distributed computing by implementing a message passing interface over HTTP[1]. This provides a platform independent implementation of MPI and also develops a base for web based computation to achieve global parallel processing. The conventional approach to message passing is through the use of Parallel Virtual Machine(PVM) or MPI software packages that optimize the communication and the synchronization functions for the message passing paradigm needed in high performance distributed computing environments. However the recent trend in Internet communication has been to use the HTTP for inter-linking information and files across the World Wide Web. The HTTP [1], over the years has matured to be a very efficient tool for communication and is fit enough to warrant consideration as a vehicle for inter-processor communication. The performance of our implementation was compared with the standard MPI implementation over a cluster of workstations connected by a Shared Ethernet network and the results are very encouraging.

1 Introduction

Nowadays, massively parallel architectures are built by interconnecting a cluster of workstations and accomplishing massive parallel processing using the workstations as the nodes of processing.There is also a growing number of web-enabled machines and the development of web-windows giving the productivity tools for a true distributed high performance computing environment. Message passing is a paradigm used widely on certain classes of parallel machines, especially those with distributed memory. The interface that provides for message passing should establish a practical, portable, efficient and flexible standard for message passing. MPI is the de facto standard for multi-computer and cluster message passing[2].

The Hypertext Transfer Protocol is an application level request-response protocol for distributed, collaborative, hypermedia information systems [1]. HTTP as a protocol has matured over the years and many communication optimization ideas and synchronization routines have been entwined with it. The primitives of HTTP can be used to implement a client-server model over the Web. The recent spurt of enhancements of HTTP like persistent connections[1] and pipelining make HTTP a wonderful protocol for message passing over the World Wide Web.

Present day implementations of MPI are platform specific and optimizations of MPI are performed over a specific platform [3]. By implementing MPI over HTTP one gets a platform independent implementation of MPI. This allows for the development of portable parallel software on different parallel architectures and also provides a wonderful paradigm for web-based computation.

We have implemented the basic MPI calls and other commonly used MPI calls using HTTP primitives and have tested the performance of our implementation with the standard MPI implementation. We have also tested the performance of our MPI_Send and MPI_Recv calls separately. Results indicate that the saturating bandwidth obtained in both implementations are comparable.

In section 2 we present our design and implementation details. In section 3 a detailed performance analysis of our implementation is given. In section 4 we summarize our results and also give a sketch of our future work.

2 Design and Implementation

The processors that execute a parallel program are treated as HTTP clients which submit their requests to HTTP servers to accomplish message passing. The Message passing interface developed over HTTP is present as a library at every client executing the MPI programs. Any MPI function that needs to send or receive messages, generates a particular request which is sent to the server for servicing. The server in turn sends a response to the client which is subsequently parsed by the client to get the status of the request. The status indicates a success or a failure in the completion of the request. In order to reduce the connection setup overhead, we follow a connection management policy and maintain long live connections to the servers.

All the basic operations of MPI and other commonly used MPI operations are implemented as part of the client's library. MPI operations like MPI_Send and MPI_Recv which actually perform message passing are im-

plemented with the help of HTTP primitives GET,HEAD and POST. The MPI-operations implemented include the six basic MPI operations namely MPI_Init, MPI_Finalize, MPI_Comm_rank, MPI_Comm_size, MPI_Send and MPI_Recv. All MPI programs can be written using these six basic MPI functions. Other commonly used MPI operations MPI_Wait, MPI_Bcast, MPI_Gather and MPI_Reduce are also implemented.

A list of all the processor nodes is created in the MPI_Init call and the MPI_Comm_rank and MPI_Comm_size calls make use of this list to report the rank and the size for an MPI program. The important communicating MPI calls, MPI_Send and MPI_Recv, implemented efficiently using HTTP primitives. In our framework both the communicating clients agree on a HTTP server with which communication is carried out in the least amount of time, in order that the above objective be satisfied.The message is posted to this server as part of the entity body of the request by a POST request to a cgi-bin executable at the server which stores the message at the server end. As part of the MPI_Recv call, the receiver keeps polling the server for the message, and it through a GET request once it has arrived at the server.

All the collective MPI-operations like MPI_Bcast, MPI_Gather and MPI _Reduce involve one to many or many to one transmissions. In-order to implement them efficiently all the nodes are divided in the beginning into various groups and every group is associated with a group leader. The collective operation is performed in two stages; one between the central node and the group leader and the other among all nodes in a group.

3 Performance Results

The above implementation was tested over a 10 Mbps Ethernet link connecting a cluster of machines through a hub. The HTTP daemon is made to run on one of the machines. Message passing is accomplished between two other nodes in the cluster acting as HTTP clients. Various kinds of message passing experiments on this kind of set-up and the results obtained are more than encouraging. From the results of the experiments done below, we can note that the saturating bandwidth obtained using our implementation is comparable to that of conventional MPI implementation. The effective bandwidth obtained for packets of small size increases with the maintenance of long-live connections. The performance of MPI_Recv is poor because of high disk access time at the server end for large packets.

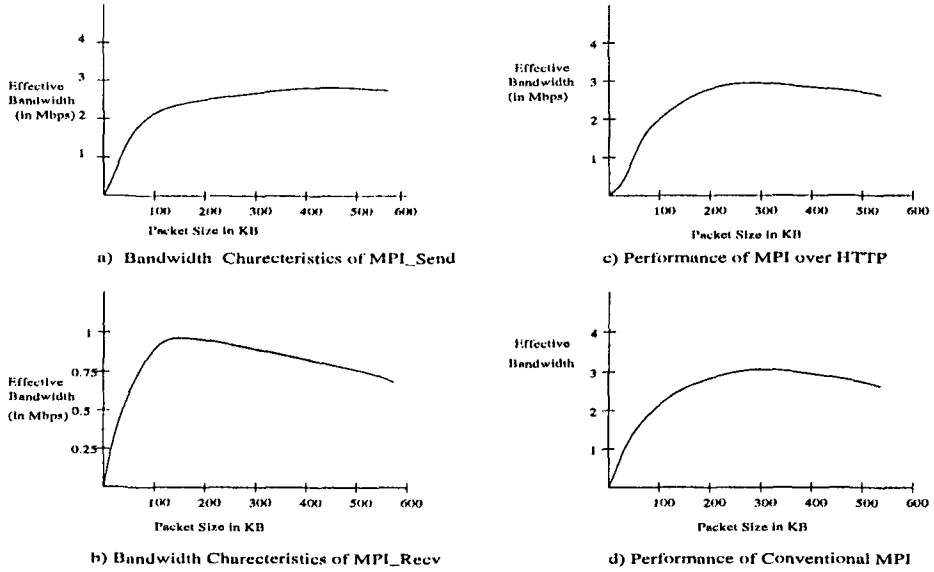

Figure 1: Results of the Experiments

4 Conclusion

In this paper we have opened up a new paradigm for Web based computation. MPI over HTTP also provides a pervasive base for global parallel processing and many problems that require huge computation power can be solved using our approach.We have also obtained a platform independent implementation of MPI and HTTP enhancements like pipelining are bound to improve the performance of our implementation tremendously.

References

[1] R.Fielding,J.Gettys,J.Mogul, H.Frystyk and T.Berners-Lee, Hypertext transfer protocol- HTTP/1.1, RFC 2068, *Internet Request for comments*, Jan 1997.

[2] J.J.Dongarra,S.W.Otto,M.Snir and D.W.Walker, An Introduction to the MPI Standard,*Communications of the ACM* ,Jan 1995.

[3] H.Franke, P.Hochschild,P.Pattnaik and M.Snir, An Efficient Implementation of MPI on IBM-SP1, *Proc. Of the 1994 Int. Conf. on Parallel Processing*, Aug. 1994