ANL/MCS/CP-105480

Programming with MPI on Clusters

Ewing Lusk
Mathematics and Computer Science Division
Argonne, IL 60439
lusk@mcs.anl.gov

Abstract

We discuss the current state of development for the key aspects of MPI programming on clusters. These aspects are the evolution of the MPI Standard itself, developments in cluster hardware and system software that directly affect MPI implementations, and supporting software that facilitates th use of MPI on scalable clusters. In each case we give a brief background and summarize the current status.

1. Introduction

Clusters are becoming a mainstay of high-performance computing. Some applications can take advantage of clusters by exploiting their capability for running large numbers of individual serial jobs, particularly when the cluster is running a batch scheduler. The most demanding applications, however, are implemented with true parallel algorithms and run as single parallel jobs. The development of the programming and execution environment for such applications is ongoing and quite active.

One of the central elements of such environments is the MPI (Message Passing Interface) parallel library specification, which enables programmers to write parallel, portable, scalable programs that can take advantage of the latest in cluster technology and run on a wide variety of high-performance computing platforms, of which large and small clusters are examples.

Here we review and summarize the state of three important elements of MPI programming on clusters: the evolution of the MPI standard itself, the state of current MPI implementations for clusters and promising new developments in the area of MPI implementation, and the software environment outside MPI itself that interacts with MPI applications.

2. The MPI Standard

MPI was defined over the period 1993-1994 by the MPI Forum, a group of parallel compute vendors, computer scientists, and application developers. The goal of the Forum as to develop a community standard that would attract both implementors and users. Key aspects of the MPI specification process was that it was open to anyone and both its internal deliberations and its final results were freely available.

The MPI Standard was released in 1994 [7]. Its innovative features included a wide variety of communication modes, a complete set of collective operations, communicators for encapsulation of communication in libraries, MPI datatypes for describing noncontiguous messages and for dealing with heterogeneous environments, and process topologies. Implementations both vendor and public, appeared shortly, and implementations of this release of the MPI specification are now available on all parallel machines and MPI is widely used.

In 1995-96 the MPI Forum reconvened to add features that had deliberately not been addressed in the first round, including dynamic process management, parallel I/O, remote memory access, and a number of convenience features. this part of the standard is often referred to as MPI-2. Currently there are a number of complete MPI-2 implementations by computer vendors. As yet no complete MPI-2 implementations on clusters exist, but a number of partial implementations are in use.

The Forum is currently addressing small errors and inconsistencies in the MPI-2 standard, and expects to release version 2.1 in the fall. References on MPI itself are [5, 6, 9, 3], and the MPI Forum Web site `www.mpi-forum.org`.

3. MPI Implementation Issues

Currently there are multiple widely distributed MPI implementations for clusters. MPICH [4] is portable to other environments as well, including clusters based on Windows 2000, and is particularly designed to be customized for multiple low-level communications systems. LAM is a daemon-based implementation well-suited for clusters, which has partial support for MPI-2's remote memory access and dynamic process management functions. MPICH and LAM share code for parallel I/O and the C++ bindings. MPI-Pro is a commercial MPI implementation from MPI SoftTech, with both Linux and Win2000 versions. Myricom distributes and supports a customized version of MPICH, called MPICH-GM, with its Myrinet switch hardware.

MPI implementation remains a fertile research area. The MPI Standard gives substantial opportunities for MPI implementors to develop custom algorithms for communication in modern hardware environments, which are evolving rapidly. The increasing use of SMP's in clusters calls for multimethod and/or multithreaded implementations. New low-level communication specifications such as VIA and hardware standards, such as Infiniband, are rapidly pushing cluster implementations beyond their original design point of TCP over Ethernet. New developments like OpenMP encourage implementations to interact smoothly with other mechanisms for expressing parallelism. And "the Grid" encourages MPI implementations to provide interfaces that facilitate communication among different MPI implementations over wade-area networks. One approach is the definition of an interface for multiple implementations to adopt (IMPI); another is a single implementation that coordinates other, vendor implementations (MPICH-G2).

4. Related Software in the Parallel Programming Environment

Any MPI implementation interacts with the systems environment on the cluster. An interesting area of research is in the architecture of such system software in general, including such components as the process manager, job scheduler, resource monitor, accounting system, etc. While these components have been traditionally integrated into monolithic systems, currently there is substantial interest in designing components and component interfaces that will allow a degree of standardization in parallel system software without constraining implementations. Making these system components both scalable and fault-tolerant is a particular challenge.

One area of particular interest is the interface between parallel process managers and MPI implementations. One desires fast startup so that short, interactive MPI programs (like the Scalable Unix Tools[8]) make sense, yet the process manager must provide information to the MPI implementation that supports dynamic process creation and connection. One approach is to create a standard interface, independent of both MPI and of process manager, that will let multiple process managers interact with multiple MPI implementations. Some work along these lines, along with a prototype scalable process manager, is outlined in [1].

Parallel debugging on clusters is still a major challenge. Two approaches that have been taken are the development of a debugger interface that implementations can support and that debuggers can use [2]. This is the approach taken by TotalView and MPICH.

MPI was intended to accelerate the encapsulation of much parallel programming in libraries, and this as occurred, particularly in the area of scientific programming. Important MPI-based libraries include ScaLAPACK (for linear algebra), PETSc (for solving partial differential equations) and Paramesh (for managing data for adaptive mesh refinement).

An unusual recent development is an MPI binding for the Ruby scripting language [10]. In Ruby, everything is an object, and the MPI-Ruby module allows form communication of objects among Ruby programs. Ruby simplifies MPI enormously (no datatypes, for example). This provides an extremely flexible parallel programming environment whose capabilities have just begun to be explored.

5. Conclusion

The evolution of MPI, MPI implementations, and MPI-related software is an active research an development area. The rapidly evolving cluster hardware and system software environment is a good fit for MPI, and stands to both challenge and benefit from MPI.

References

[1] R. Butler, W. Gropp, and E. Lusk. Components and interfaces of a process management system for parallel programs. *Parallel Computing*, 2001. (to appear).

[2] J. Cownie and W. Gropp. A standard interface for debugger access to message queue information in MPI. In J. Dongarra, E. Luque, and T. Margalef, editors, *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, volume 1697 of *Lecture Notes in Computer Science*, pages 51–58. Springer Verlag, 1999. 6th European PVM/MPI Users' Group Meeting, Barcelona, Spain, September 1999.

[3] W. Gropp, S. Huss-Lederman, A. Lumsdaine, E. Lusk, B. Nitzberg, W. Saphir, and M. Snir. *MPI—The Complete Reference: Volume 2, The MPI-2 Extensions.* MIT Press, Cambridge, MA, 1998.

[4] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the MPI Message-Passing Interface standard. *Parallel Computing*, 22(6):789–828, 1996.

[5] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message Passing Interface*, 2nd edition. MIT Press, Cambridge, MA, 1999.

[6] W. Gropp, E. Lusk, and R Thakur. *Using MPI-2: Advanced Features of the Message-Passing Interface*. MIT Press, Cambridge, MA, 1999.

[7] Message Passing Interface Forum. MPI: A Message-Passing Interface standard. *International Journal of Supercomputer Applications*, 8(3/4):165–414, 1994.

[8] E. Ong, E. Lusk, and W. Gropp. Scalable Unix commands for parallel processors: A high-performance implementation. In J. Dongarra and Y. Cotronis, editors, *Proceedings of Euro PVM/MPI*. Springer Verlag, 2001. To appear.

[9] M. Snir, S. W. Otto, S. Huss-Lederman, D. W. Walker, and J. Dongarra. *MPI—The Complete Reference: Volume 1, The MPI Core,* 2nd edition. MIT Press, Cambridge, MA, 1998.

[10] D. Thomas and A. Hunt. *Programming Ruby: The Pragmatic Programmer's Guide*. Addison-Wesley, 2001.

This work was supported by the U.S.
Dept. of Energy under Contract W-31-109-ENG-38.