MPI-2: Extending the Message-Passing Interface

Al Geist[1] and William Gropp[2] and Steve Huss-Lederman[3] and Andrew Lumsdaine[4] and Ewing Lusk[2] and William Saphir[5] and Tony Skjellum[6] and Marc Snir[7]

[1] Oak Ridge National Laboratory
[2] Argonne National Laboratory
[3] University of Wisconsin
[4] University of Notre Dame
[5] NASA Ames
[6] Mississippi State University
[7] IBM Research Yorktown

Abstract. This paper describes current activities of the MPI-2 Forum. The MPI-2 Forum is a group of parallel computer vendors, library writers, and application specialists working together to define a set of extensions to MPI (Message Passing Interface). MPI was defined by the same process and now has many implementations, both vendor-proprietary and publicly available, for a wide variety of parallel computing environments. In this paper we present the salient aspects of the evolving MPI-2 document as it now stands. We discuss proposed extensions and enhancements to MPI in the areas of dynamic process management, one-sided operations, collective operations, new language binding, real-time computing, external interfaces, and miscellaneous topics.

1 Introduction

During 1993 and 1994, a group of parallel computer vendors, library writers, and application scientists met regularly to define a standard interface for message-passing libraries. The result of this effort was MPI (Message-Passing Interface) [7]. Implementations of MPI are now widely available, including portable and freely available implementations [2, 3, 8] and specialized versions from vendors. General information on MPI is available at [1]. For the purposes of this paper, it will be useful to refer to the result of the initial MPI standardization effort as "MPI-1."

MPI-1 defined an interface for a specific *message-passing model* of parallel computation, in which a fixed number of processes with disjoint address spaces communicate through a cooperative mechanism (when two processes communicate, one sends and the other receives). MPI provides many types of point-to-point communication, to incorporate requirements for robustness, expressivity, and performance. Messages are strictly typed and scoped, allowing for communication in a heterogeneous environment. MPI also contains an extensive set of collective operations, process topology functions, and a profiling interface.

The most distinctive feature of the current MPI-2 proposals described in this paper is that they go beyond the strict message-passing model defined above. In MPI-2, processes may create other processes, so that the number of processes in an MPI computation can change dynamically (Section 2). Processes can interact directly with the memory of other processes (Section 3). Extensions, semantic modifications, and subset definitions in support of real-time and embedded systems (Section 4) also represent changes to the computational model.

Other topics being discussed in MPI-2 include extending MPI-1's collective operations to intercommunicators and nonblocking operations (Section 5), bindings for C++ and Fortran 90 (Section 6), and interface definitions for some of MPI's opaque objects so that they can be used more effectively in support of profiling and other libraries (Section 7). Finally, a number of issues, such as interlanguage communication, a portable startup mechanism, and minor repairs to the MPI-1 specification (Section 8), are under consideration in MPI-2.

In the rest of this paper, we present an overview of each of these areas. We assume familiarity with the current MPI Standard. In the Conclusion we describe the current status of these proposals and prospects for their early appearance in implementations.

2 Dynamic Process Management

MPI-1 describes how a group of processes can communicate with one another. It does not specify how those processes are created, nor does it allow processes to enter or leave a parallel application after the application has started. This static process model enables the specification of deterministic semantics and facilitates efficient implementations of MPI.

Nevertheless, a number of important applications cannot use MPI-1 because of the constraints imposed by its static process model. These include manager-worker applications, where the number and type of workers are not known until the manager has started, task farms, applications that can adapt to changing resources, applications with varying resource requirements, and client/server applications. Much of the impetus for relaxing the static process model comes from the PVM community, which is familiar with PVM's relatively rich support for dynamism.

2.1 The Interface

A fundamental concept in MPI-1 is `MPI_COMM_WORLD`, which defines the communication space containing all processes in an MPI application. With MPI-2's ability to add more processes to an application, the definition is modified to be the communication space containing all processes started together. Groups of newly started processes each have their own unique `MPI_COMM_WORLD`, but they also have an intercommunicator that allows them to merge with their parent group, forming a single bigger communicator. MPI-2 also provides an attribute,