Early Applications in the Message-Passing Interface (MPI)*

Anthony Skjellum‡
Department of Computer Science &
NSF Engineering Research Center for Computational Field Simulation
Mississippi State University
PO Box 6176
Mississippi State, MS 39762

Ewing Lusk, William Gropp§
Argonne National Laboratory
Mathematics and Computer Science Division
9700 South Cass Avenue
Argonne, IL 60439

June 15, 1994

Abstract

In this paper, we describe a number of early efforts to make use of the Message Passing Interface (MPI) standard in real applications. An informal survey, described herein, was posted by the authors on common USENet groups as well as to common MPI mailing lists. The responses contained here represent all but one the responses received; it is not exhaustive, but rather represents those researchers who were willing to respond during a limited period of time (about two weeks).

Rather than a definitive statement of MPI development work, this paper restricts itself to describing the initial successes, progress, and impressions of application developers have with MPI, according to the responses received (many other projects, not described here, are also evidently on-going). An understanding of message-passing, and access to the MPI standard are prerequisites for appreciating this paper. We summarize the important aspects of each response, and draw conclusions about the spread of MPI into applications. Clearly, studying the state of MPI application development is on-going: here, we demonstrate that it is well underway, and that it has breadth in the type of applications, and the target architectures.

*Submitted to Special Issue of *International Journal of Supercomputing Applications*. Revision 1.0; June 15, 1994.

†Work supported in part by the NSF Engineering Research Center for Computational Field Simulation, Mississippi State University.

‡Corresponding author; e-mail: *tony@cs.msstate.edu*; phone: 601-325-8435.

§Work supported by U.S. Department of Energy under contract W-31-109-Eng-38.

1 Introduction

In this paper, we describe a number of early efforts to make use of the Message Passing Interface (MPI) standard in real applications [7, 8]. MPI was developed as part of a multi-national effort (supported partially by NSF, ARPA, and ESPRIT) to replace the many, marginally compatible vendor and portability systems currently in use with a coherent system that makes use of the experience of previous message-passing systems (such as P4, PVM, NX, etc), while providing greater functionality, unified design and an emphasis on user access to high performance protocols and hardware. Participants included a core of 40 or so forum members, including developers of all the major then existing message-passing systems, as well as computer vendors, and application developers. MPI encompasses and extrapolates the best ideas from existing practice, and adds features designed to secure high performance for applications on the hardware that exists today, and for hardware expected in the next few years.

The initial round of that MPI standardization effort began in April 1992 and culminated at Supercomputing '93 (November, 1993), where the standard was presented to the public[1]. By the time of its public introduction, several implementations were already underway, notably the Argonne/Mississippi State "model implementation" (also known as MPICH) [5], which was also demonstrated at Supercomputing '93; other portable implementations have followed. During the first half of this year, IBM and Cray already have launched commercial MPI efforts (*e.g.*, [9]); IBM demonstrated its experimental "MPI-F" at Supercomputing '93 as well. Intel, Thinking Machines, Meiko, and others also have implementation efforts or plans. One book has already been written about MPI, to be released in November, 1994, (*Using MPI* by Gropp, Lusk, and Skjellum [10]), and a second (*MPI Annotated Reference Manual* by Otto et al. [6]) is in progress, to be completed by the latter part of 1994. Hopefully, both of these books will contribute to the available references on the new standard, increase awareness of MPI, and, most importantly, help application programmers.

Rather than a definitive statement of MPI development work or a comprehensive explanation of *how* to develop MPI-based applications, this paper restricts itself to describing the initial successes, progress, and impressions of selected application developers have with MPI, according to the responses received (yet several other projects, not described here, are also evidently on-going). An understanding of message-passing, and access to the MPI standard are prerequisites for appreciating this paper. We summarize the responses, highlighting important facets of each contributed application survey, and describe an overall impression of the state of MPI applications.

Application programmers were informed of the opportunity to contribute to this paper, through an informal survey mechanism published on popular mailing lists, and on USENet groups. The responses described here represent all but one of the respondents during the two-week period alloted for the survey (one response was too preliminary to contain any reasonable information about the work undertaken; it will appear in an

[1] However, the standard document was finalized only in May, 1994, with on-going errata, so the existence of several applications at the writing of this paper in June, 1994 is thought to be quite significant.

updated version of this paper.) Responses came in the following categories:

- Applications:
  - Many-body Methods for Light Nuclei,
  - Solution of Unsteady Incompressible Viscous Flows,
  - NAMMU — Groundwater Modeling for Continuous Media,
  - PARMICS-MP — Microscopic Traffic Simulation,
  - Volume Visualization,
  - Reservoir Simulation (not included).
- Application-Enabling Libraries:
  - A Fast Particle Summation Library,
  - CVL — C Vector Library,
  - "ParX" (Parallel X environment).

This paper is organized as follows. First, we outline the survey itself. Then, each of the surveyed applications are described following the format of the survey. Subsequent to this discussion is an interpretation of the responses to the survey, followed by the summary, conclusions, and future directions section.

2 The Survey

In order to collect data from the widest possible source of application programmers, but consistent with a short deadline, we posted requests for application work on major USENet newsgroups (*comp.parallel*, *comp.parallel.pvm*), as well as to the MPI "reflectors" maintained by University of Tennessee (*mpi-comm@cs.utk.edu*) and Argonne National Laboratory (*mpi-users@mcs.anl.gov*). Respondents were asked to respond with applications underway, completed, or even just getting started.

The following information was requested for contributions to the survey:

1. Describe the MPI-based application, who's working on it, and experiences thus far. Specifically,
   - authors,
   - a description of the application area,
   - goals of application project (high speed, cheap cycles, new physics, etc),
   - proper citation(s) to the project, and techniques it is based on, and/or on-line locations of relevant articles.

2. Provide anectodal information about process of converting from other message-passing systems (if applicable):
   - level of effort,
   - motivation for conversion,
   - degree of satisfaction.
3. Determine which features of MPI have proven most helpful in applications development and/or conversion, such as
   - communicators (intra-communicators),
   - inter-communicators,
   - topologies,
   - pack/unpack capabilities,
   - datatypes.
4. Determine which parts of MPI are most understandable and/or most confusing for application developers.
5. Show early indications of performance results (such as on IBM SP/1, and Paragon).
6. Indicate which MPI implementation or implementations have been used in conjunction with the applications,
7. Provide miscellaneous feedback from application programmers and users, if relevant. For instance, desire of such programmers or users for other/different features in MPI.

3 Applications

In this section, we describe five responses to the survey, each of which is a pure application project.

3.1 Many-body Methods for Light Nuclei

Researchers (Steven Pieper and Robert Wiringa[2]) in Argonne's Physics Division, in collaboration with a colleague at the University of Illinois at Urbana-Champaign, are computing the properties of light (up to 40 neutrons and protons) nuclei using realistic two- and three-nucleon interactions. This research involves

[2]Theoretical Physics Group Physics Division, Argonne National Laboratory, Argonne IL 60439. e-mail: *pieper@phy.anl.gov, wiringa@phy.anl.gov.*

developing many-body methods for reliably computing the properties of a nucleus for complicated forces that are strongly dependent on the spins and charge states of the nucleons.

Unlike the Coulomb force used in atomic or condensed-matter calculations, no useful fundamental theory tells us what this force is. The two-body force can be partially constrained by fitting nucleon-nucleon scattering data, but many-body calculations are required to test other properties of this force as well as the three-body interaction. Thus the researchers are refining their knowledge of the forces and at the same time using this knowledge to make predictions about nuclei.

3.1.1 Goals of Application Project

The goal of the application project is to do new physics: that is, to compute binding energies more accurately than ever before. New calculations being done on the IBM SP1 parallel computer at Argonne are using a new nuclear interaction and are obtaining much better (when compared with experiment) results for the binding energy and density profile of oxygen than had previously been obtained. In particular, physicists are now able to find parameters of the minimal energy solution for sixteen nucleons much more precisely than ever before. Before this computation, theory did not clearly predict that the nucleus of an oxygen atom would remain stable; the previous computational results would have allowed it to disintegrate into four helium atoms. Thus in some sense this research has proven that oxygen can exist! Work on calcium (forty nucleons), which could not be contemplated before, is now in progress.

3.1.2 Parallel Formulation/Conversion

This program was already written for portable parallel execution using p4. It took one afternoon to convert it to MPI. Motivation was to learn about MPI and prepare the code for continued portability on high-performance machines as further MPI implementations become available. The developers stated that their degree of satisfaction was complete.

3.1.3 Helpful Features of MPI

The algorithm used was straightforward. No advanced features of MPI were used. So the most helpful aspect of MPI was that it is similar to existing message-passing systems, rendering porting straightforward as well.

3.1.4 Most Understandable/Most Frustrating Parts of MPI

Basic message passing was easy to understand since MPI is much like other systems in this regard. No confusing parts of MPI were encountered.

3.1.5 Early indications of Performance Results

Table 1 contains information about the size and performance of this application on the 128-node SP-1 at Argonne.

Lines of Fortran code and comments	23,000
Lines in time-critical subroutines	1000
Total program single-processor speed (megaflops)	48
Total 128-node speed (gigaflops)	5.9
Speed in the critical subroutines (megaflops)	36-56

Table 1: Summary of the Argonne nuclear structure code.

3.1.6 MPI Implementation Used

This application used the portable model MPI implementation developed at Argonne National Laboratory and Mississippi State University.

3.1.7 Miscellenaous Feedback

One feature that the p4 implementation provided that was unavailable in MPI is a form of broadcast that is received by an ordinary receive instead of a corresponding broadcast operation.

3.2 Solution of Unsteady Incompressible Viscous Flows

The computation of flow regimes over complex configurations involves the numerical solution of a system of coupled non-linear partial differential equations, the Navier-Stokes equations. Researchers in the Computational Fluid Dynamics Laboratory at the Mississippi State University-NSF Engineering Research Center for Computational Field Simulation have developed an implicit, finite-volume code (known as UNCLE) for solving the unsteady three-dimensional incompressible Euler and Navier-Stokes equations using dynamic multiblock grids [14, 15, 18]. The flow solver can be used in a variety of applications ranging from maneuvering underwater vehicles to centrifugal compressors. Ramesh Pankajakshan[3] and W. Roger Briley, also of the MSU ERC, have undertaken the scalable parallelization research and development of this solver.

Long runtimes and large memory requirements restrict the size and complexity of the problems that can be handled using the sequential version of the UNCLE code. Therefore, a scalable portable parallel version is being developed that can take advantage of existing as well as emerging parallel platforms. The goal is to use parallel computing to enable solution of large-scale field simulation problems on both parallel supercomputers and idle and/or dedicated workstation clusters. The message-passing interface required for the parallel implementation has to support collective operations within user-defined groups as well as provide safe communication contexts for overlapping sets of collective operations. Motivations for using MPI include these features, as well as

- portability,

[3]Contact address: MSU ERC, PO Box 6176, Mississippi State, MS 39762. e-mail: *ramesh@erc.msstate.edu*.

- ease of software development,
- positive performance expectations, and
- broad base of participants in MPI effort.

3.2.1 Parallel Formulation and Implementation

The parallel formulation [12] employs spatial decomposition of the overall grid into sub-blocks assigned to separate processes, and exploits coarse-grained parallelism and message passing within sub-iterations of an implicit time-dependent solution algorithm. The solution at points shared by neighboring processes is updated between each sub-iteration by means of a message exchange.

Key areas of the parallel implementation include

- initialization of the flow field,
- duplication of stored data for points near block interfaces,
- exchange of data during sub-iterations, for points having duplicated storage, and
- treatment of line searches and integrals along coordinates emanating from solid boundaries, which arise from the particular algebraic turbulence model used.

In the parallel implementation of this code [12], the domain is partitioned into a number of nearly equally sized sub-domains, each of which is assigned to a node of the multicomputer. The local data dependencies at the boundary of each block are taken into account by a two-cell-deep layer of buffer cells whose values are updated from the appropriate neighboring block. The data duplication and updating at the block boundaries is implemented using the `MPI_Sendrecv` routine. The connectivity of the processes gives rise to a Cartesian virtual topology having empty nodes in it. Each process is tagged using an ordered triplet that represents its coordinate on the virtual Cartesian grid. These coordinates are then used to define communicators for processes that are oriented along any of the three axes of the grid. This involves repeated use of the `MPI_Comm_split` routine. The line searches and integrals of the turbulence model are implemented using the `MPI_Bcast` routine and user-defined `MPI_Allreduce` operations over subsets of processes.

3.2.2 Parallel Formulation/Conversion

Work has progressed to the point of a working (but as yet preliminary) MPI application. Constructing this MPI code from an existing serial FORTRAN code took about three weeks by one of the researchers (Pankajakshan), who was familiar with the existing serial code and who also had a working knowledge of MPI and C programming at the outset. From their experience so far, the researchers find that MPI appears well suited for this application. MPI was especially useful in performing collective operations on subsets of processes, a need which arises in connection with line searches and integrals in the algebraic turbulence model. Communicators and datatypes are features of MPI which have been or will be helpful in developing this application.

Datatypes seem to be the most confusing aspect of MPI encountered so far in this application development.

3.2.3 MPI Implementations Used

The parallel code has used both the IBM's MPI-F and Argonne/Mississippi State implementations of MPI. The code has run on the IBM-SP1 at the High Performance Computing Research Facility, Mathematics and Computer Science Division, Argonne National Laboratory, and on an MSU Sun workstation/ethernet cluster.

3.3 NAMMU — Groundwater Modeling

NAMMU is a package for predicting groundwater flow, heat, and mass transport through continuous porous media. The application is based on an efficient implementation of the finite-element method, which in turns makes heavy use of the Harwell Frontal Direct matrix solver routine 'MA42'. The work described here is a parallelization of the solver through domain decomposition of the mesh. NAMMU was developed by AEA Technology, modification of the MA42 package was done by Harwell, and the message passing (in MPI) was done by the Edinburgh Parallel Computing Centre. The authors of NAMMU are as follows:

- Hon W. Yau (Edinburgh Parallel Computing Centre)[4],
- K. Andrew Cliffe (AEA Technology, Harwell),
- Jennifer A. Scott (Atlas Centre, Harwell),
- David Brear (AEA Technology, Harwell).

Hon Yau is the designer (and one of the implementors) of the parallelized version of NAMMU.

3.3.1 Goals of Application Project

The aim is to produce a portable, parallelized version of NAMMU that will best exploit parallel architectures. Together with two other AEA Technology safety assessment codes, this will form a suite of parallelized groundwater simulation codes.

3.3.2 Parallel Formulation/Conversion

The conversion effort was considerably eased by the availability of a high quality in-house implementation of the MPI specifications. Moreover, a 'toy' MPI implementation of the solver using MA42 had already

[4]Contact address: Edinburgh Parallel Computing Centre (EPCC), James Clerk Maxwell Building, King's Buildings, Mayfield Road, Edinburgh EH9–3JZ, U.K. e-mail: *hwyau@epcc.ed.ac.uk*.

been successfully demonstrated prior to work on the NAMMU package. From the design of the NAMMU implementation to the successful execution of the available test cases, took approximately three months. This included roughly 20 person-days for the conversion of NAMMU to use domain decomposition, and one person-month to write and debug the MPI message-passing constructs needed.

The authors were motivated to convert to MPI because they wanted a portable implementation that demonstrates good scalability within the limitations of the parallel algorithm. The authors reported a high degree of satisfaction with the current MPI specifications, in particular with the richness of features. They noted that it seemed a thoroughly thought-out product.

3.3.3 Helpful Features of MPI

Only a duplicate of the global group of processes (represented in `MPI_COMM_WORLD`') was used in the application, since the intension was to use one or more of the collective communications functions. Derived datatypes were heavily used, as an efficient alternative to multiple messages and packing/unpacking; this feature had much to be commended.

3.3.4 Most Understandable/Most Frustrating Parts of MPI

The most obvious source of problem is with the Specifications Document [7]. As the only currently available introduction to MPI it is seen by the developers as daunting — in particular to those with no prior message passing experience. It is, however, also seen as reasonably complete, though in need of more Fortran examples and a glossary of terms.

3.3.5 Early indications of Performance Results

At present, performance figures are only available for the aforementioned 'toy' implementation of the solver. With this, good scalability was demonstrated for various mesh sizes decomposed into four domains. Thus far, the parallel implementation of NAMMU has reproduced solutions obtained by the serial code on four standard test cases for this application.

3.3.6 MPI Implementation Used

An in-house EPCC implementation of MPI, based on CHIMP, was used. This version of MPI runs on a variety of machines, including Sun and Silicon Graphics workstations, and the Meiko CS-1/i860.

3.3.7 Miscellaneous Feedback

The NAMMU authors recommended the following:

- Turn the MPI specifications document into a proper reference manual, and provide a simpler introductory guide[5].
- Tidy up the use of derived datatyes. At present, using these requires the following steps:
  - Finding the address of each block,
  - Set the number of elements in each block,
  - Set the types of each block,
  - Create a new datatype using the above information,
  - 'Commit'ing the new datatype.

It would be useful to have a utility routine that performs the first four above functions at once, for many situations.

3.4 PARMICS-MP — Microscopic Traffic Simulation

PARAMICS-MP builds on earlier work on the PARAMICS project, which developed a PARAllel MICroscopic traffic Simulator. The PARAMICS system, in its current state, can simulate over 250,000 vehicles at speeds faster than real time, on a road network based on real data for the Scottish Trunk Road Network. As far as the researchers know, this is the largest microscopic traffic simulation ever implemented. PARMICS-MP is being developed by Gordon Cameron, Gordon Duncan[6], David McArthur, of the Edinburgh Parallel Computing Centre.

The primary goal for PARAMICS-MP is to port the existing PARAMICS simulator, originally developed as a data-parallel implementation for the Connection Machine CM-200, to a Message-Passing version running on the recently purchased 256-processor Cray T3D located in Edinburgh. The "MP" version will use the MPI library currently under development at EPCC.

It is expected that the T3D's superior compute power will enable the integration to the simulator of a significantly more detailed vehicle dynamics model than is currently possible. In addition, more advanced simulation features will be added, such as the effects of dynamic routing as proposed by reactive Road Traffic Information systems.

3.4.1 Parallel Formulation/Conversion

The current status of the project is that a prototype version of the MP simulator is running on a distributed system, using EPCC's own message passing system CHIMP (which has recently added an MPI interface). It is intended that as soon as the MPI implementation being developed locally for the T3D becomes available the application will be converted to use it.

---

[5] This is being done in the book by Otto et al. [6].

[6] Contact address: e-mail: *gd@epcc.ed.ac.uk*; phone: +44 (0)31 650 5818.

This move is part of an EPCC organization-wide policy of migration to the MPI standard in all forthcoming applications.

3.4.2 Implementation(s) of MPI Used

The PARAMICS-MP project is using the MPI for the Cray T3D, developed for Cray by EPCC.

3.4.3 Miscellaneous Feedback

Current functionality seems sufficient, but the project still is in its developmental stages.

3.5 Volume Visualization

With the advent of faster and cheaper computer hardware, scientists and researchers have been able to increase the size and complexity of numerical simulations to the point where large, three-dimensional simulations are routinely performed. As the volume of data produced increases, however, researchers encounter significant difficulty when trying to interpret the results of the computations. This is especially true with three-dimensional data sets.

Volume visualization techniques were created to allow researchers to examine the data and spatial relationships contained in volume data sets, but, in general, the algorithms are relatively slow on even moderately sized data sets. This problem is compounded as the size of the data increases, and additional problems with memory size may be encountered in generating a visualization for large data sets on a conventional machine. It is thought that some of these problems can be addressed by distributed memory parallel computers, by allowing the parallel machine to perform the calculation of the scene, and employing conventional machines to display that scene. Parallel computers have added potential over conventional machines for scene rendering in that a large data volume can be distributed over the processors in the machine and operated on concurrently, reducing at once the computational time and and memory requirement per processor (scalability).

John E. West[7], Michael Stephens, and Louis Turcotte, of the Waterways Experiment Station (WES) are investigating the application of MPI to the development of algorithms for performing volume visualization of large data sets on parallel machines. The goal is to create preliminary versions of volume visualization algorithms for parallel machines both to investigate the use of a specific machine, the nCUBE/2 at WES, for this type of application, and to gain insight into the issues surrounding parallel visualization in general. The algorithms under examination represent the two general classes of algorithms available for extracting information out of 3-D data sets: marching cubes, which is prototypical of surface-extraction algorithms, and ray tracing, which is prototypical of volume-rendering algorithms.

[7]This work is being performed at the DOD High Performance Computing Center at the Army Corps of Engineers Waterways Experiment Station. e-mail: *hatter@athena.wes.army.mil*; phone: 601-634-4610.

It is expected that after the preliminary study is completed, some or all of the algorithms will be ported to other parallel machines and evaluated for hardware-specific performance features. Using MPI is expected to reduce the cost of this software port dramatically as well as simplify code development.

3.5.1 Parallel Formulation/Conversion Process

Conversion issues do not apply to this project, since the code is being created from scratch using MPI. According to the researchers, this represents a distinct advantage for them, in that, not only is the initial development time expected to be decreased, but the time for subsequent ports of the code to different platforms will also be reduced.

3.5.2 Helpful Features of MPI

Thus far, the features of MPI that have proven most helpful are the communicators and topologies. Having developed codes for varying platforms in the past using native message passing constructs (NX, Vertex, etc), these researchers attest to the fact that the presence of the communicators and the utility function that are available for their manipulation greatly simplify the development of a communication protocol within an application. The topology functions have also been useful in performing the data distribution. They allow the creation of a process topology that matches the original data topology in only a few simple steps. This further simplifies the data distribution and communication problems typically encountered in other message passing protocols.

It is expected that datatypes, and in particular strided datatypes, will also prove especially useful in the future.

3.5.3 Most Understandable / Least Understandable Aspects of MPI

MPI was found to be easy to use and understand.

3.5.4 Early Indications of Performance Results

The preliminary anecdotal results indicate that, for this application, the performance of the nCUBE using MPI is comparable to a high-end graphics workstation when applied to small data sets. However, for large data sets, the parallel algorithms have some performance advantage because of the reduced cost in memory access time per node (because the data is localized to nodes), and in actual execution time deriving from concurrency of the computation. Work at quantifying these advantages remains to be done.

3.5.5 Implementation Used

Nathan Doss (MSU ERC) and John West ported the Argonne/Mississippi State MPI implementation to the WES nCUBE. This port will require significant, non-trivial modification in the future to add support for the nCUBE's host/node programming paradigm. Currently, nCUBE nodes communicate with each other only in the MPI model; host-node communication still requires the use of the Vertex calls.

4 Application-Enabling Libraries

In this section, we describe application-related responses to the survey. Each of these responses represents a project that is either part of an application, or enables a class of applications.

4.1 A Fast Particle Summation Library

A general-purpose "Fast Particle Summation" library, which is, in turn used in several "applications," is under development by John K. Salmon[8] of Caltech, Michael S. Warren[9] of Los Alamos, and David J. Edelsohn[10] of Syracuse University. The code itself and it's original application in gravitational N-body simulations has been described in [13, 16, 17].

A description of an application solving potential flow problems and another involving generation of Gaussian random fields is in preparation. All the relevant papers, including some with an emphasis on astrophysics, rather than computation, are available by anonymous ftp in *ftp://ftp.ccsf.caltech.edu/nbody*, or on Mosaic (HTML), as *http://www.ccsf.caltech.edu/ johns/papers.html.* Also, papers describing the evolution and interaction of disk galaxies by David Edelsohn (NPAC, Syracuse University) and Bruce Elmegreen (IBM Research) are in preparation.

4.1.1 Parallel Formulation/Conversion

The application is written to the researchers' own set of message-passing primitives, (*e.g.*, asynchronous send, asynchronous receive, exchange, etc). These primitives are, by design, extremely simple. They are implemented using the message-passing programming interface of each system. Because the primitives previously had been ported to the IBM MPL (EUI), converting to MPI was greatly simplified, in view of the similarities between the two message-passing definitions. The most difficult element is adapting to system peculiarities for querying about the status of outstanding operations and statistics about received messages. The researchers ported to MPI because it was simple to accomplish, expanded their target environments, and improved flexibility and maintenance for future systems on which this application will run.

At the moment the authors only use the lowest level of MPI functionality. Using inter-communicators for interactive, real-time control and display of simulation results is a long-term goal for them.

[8]Physics Department, California Institute of Technology, *johns@ccsf.caltech.edu,* phone: +1 818 395 2907.
[9]Theoretical Astrophysics, Los Alamos National Laboratory, *msw@tdo-serv.lanl.gov*, phone: +1 505 665 5023.
[10]Northeast Parallel Architectures Center, Syracuse University, *edelsohn@npac.syr.edu*, phone: +1 914 945 3011.

4.1.2 Criticisms of MPI

Of the features that the authors wanted to use, the multiple completion functions are much less flexible than they desired. They would have would prefer functions that act upon dynamic sets or collections of communication objects which can be updated efficiently and then used by a generalized POSIX "select()" function to allow rapid discovery of completion of one of many outstanding communication operations, because they may have hundreds of such outstanding operations.

Their conclusion is that the current MPI array semantics does not allow for an efficient implementation of the functionality they desire for their applications and limits usefulness to applications with a relatively short and mostly static list of communication handles.[11]

4.1.3 Implementation Used

The IBM MPI-F prototype implementation running on an IBM SP1 with the High Performance Switch achieves a latency of 30 microseconds and a bandwidth of 8.7 MBytes per second. This implementation is an early research prototype developed within IBM Research that conforms to and implements the entire, final MPI document [7]. Comparing the application running with MPI-F and MPL/p (EUI-H) on the IBM SP1, they observed no significant change in performance [9].

To date, the only MPI implementation with which the application has been run is the above mentioned IBM MPI-F prototype on the IBM SP1.

4.1.4 Miscellaneous

The researchers would like the MPI Forum to reconsider their communication-completion function modification proposal for the next round. Also, asynchronous forms of collective communication, that is, collective communication which is overlapped with computation, is perceived to be useful.

4.2 CVL - C Vector Library

CVL (C Vector Library) is a library of low-level vector routines callable from C developed by Jonathan C. Hardwick[12]. It is used by the SCANDAL project at CMU as a basis for implementing NESL, a portable nested data-parallel language, and also by the Proteus project at UNC to implement the parallel prototyping language Proteus. The library includes a wide variety of vector operations such as element-wise function applications, scans, reduces, and permutations. Most CVL routines are defined for both segmented and unsegmented vectors, since segmentation is critical for implementing nested data parallelism. Previously,

[11] In our experience, many applications do not have more than a handful of outstanding communication operations pending, unlike the algorithms considered by these researchers. Alternative functionality proposed by these researchers near the end of the MPI standardization process, was consequently rejected as an alternative. Had it been proposed as a supplement to the easy-to-use arrays of status objects, rather than a replacement, its inclusion would have been much more likely.

[12] Contact address: *jch@cs.cmu.edu*.

CVL has been ported to each new parallel machine architecture (Cray, CM-2, CM-5, MasPar, etc), using each manufacturer's own low-level communication routines [1, 2].

The main goal of the SCANDAL project is the development of a portable, interactive environment for programming a wide range of supercomputers, using a high-level data-parallel language that supports nested parallelism. The researcher and his colleagues are especially interested in the implementation of algorithms with irregular or dynamic structures.

4.2.1 Conversion Effort

The MPI port of CVL has taken a couple of man-months. It benefited from using algorithms for a distributed memory architecture initially developed for the CM-5 port, although these had to be adapted to use coarse-grain rather than fine-grain communication.

The overwhelming motivation for writing an MPI port of CVL was the prospect of "never having to do another CVL port again." That is, the effort required to port to future machines will be reduced to tuning the MPI port to best fit a particular manufacturer's implementation of MPI.

The author has been satisfied with MPI; of all the CVL ports (other than the serial version), he reports that this was the easiest.

4.2.2 Features of MPI Helpful to the Port

In no particular order, the most helpful features of MPI for the CVL implementation were seen by the author of CVL to be as follows:

- Built-in support for scans and reductions, although this is not as complete as it could have been (see below),
- Non-blocking sends and receives to overlap communication and computation,
- Choice of send semantics with respect to blocking and buffering; this makes it easy to get a quick-and-dirty version running, which can then be refined to increase performance.

4.2.3 Most Understandable / Most Frustrating Features of MPI

The "standard" message-passing features are the most understandable. The author singles out figure 4.1 of [7] as a great summary of 19 pages of text. The most confusing section is that on constructed datatypes, although this is perhaps inevitable given the complexity of the topic. CVL does not use either communicators or topologies, so the author did not comment on those sections.

4.2.4 Implementation Used

The author used the portable Argonne/Mississippi State implementation. Being able to develop and debug MPI code on a workstation has been seen to be a great help; supercomputer debugging environments are generally inferior to those on workstations, and interactive debugging can consume large amounts of project cycles. The author intends to test on other MPI implementations before releasing the code.

4.2.5 Miscellaneous

Scans should be exclusive rather than inclusive: it is seen as trivial to convert an exclusive result into an inclusive result, but the reverse must involve extra communication for operations with no inverse (*e.g.,* max, min). The author did not agree with the rationale cited in the standard that providing an identity for user-defined operations is "too complex" (the stated rationale for choosing inclusive scans). The author noted that segmented scans and reductions, as supplied by High Performance Fortran, TMC Paris (CM-2), TMC CMMD (CM-5), etc, would be a definite plus, although they can be "kludged together" using the user-defined scan operators in MPI.

5 ParX

Steve Ball, of Australian National University[13], has developed ParX. ParX is a port of the X Window System to a distributed-memory multicomputer (in the first instance, the Fujitsu AP1000). Both the Xlib library and an X server are to be ported. This will allow an application running on the multicomputer to display graphics on a local framebuffer or a remote display using the same application programmer interface. There is no need to off-load data to a host processor subsequently to generate X calls. There will be an implementation of the Xlib library that allows an application program to connect to an X server (local or remote) as well as an implementation of an X server for a distributed framebuffer. This server will itself be distributed across the multicomputer.

5.0.6 Conversion Effort

The author was just starting a re-write of the Xlib implementation, when X11R6 was released as well as an implementation of MPI for the AP1000. Thus no extra effort has been necessary to convert to MPI, apart from learning about MPI itself.

In the not-to-distant future, the author will be porting ParX's Xlib to the CM5 (CMMD), as CM5 users require MIMD X (as opposed to CMX11, which is SIMD). MPI allows the code to be written portably to the CM5, as well as other distributed systems.

MPI has satisfied all of the author's requirements, apart from the semantics of broadcasting. A non-blocking, non-synchronizing broadcast should overcome this (he will be pursuing this addition).

[13] Contact address: *Steve.Ball@cs.anu.edu.au.*

5.0.7 Helpful Features of MPI

Intra- and inter-communicators are vital for this project, as it is a library that needs to insulate itself from the user's application. Pack/unpack may be useful later for optimizing memory access.

5.0.8 Most Understandable / Most Frustrating

The point-to-point and collective operations are fairly straight-forward (the author skipped over the datatype features). Communicators took a little while to work through, but they also turned out to be straight-forward, according to the author.

5.0.9 Implementation of MPI Used

MPI release 0.1 for the Fujitsu AP1000 (port done by ANU), evidently based on Argonne/Mississippi State University portable implementation.

5.0.10 Miscellaneous

The author noted that, at this stage, the only missing feature to his mind was non-blocking collective operations.

6 Interpretation of the Survey

We begin by summarizing key aspects of the applications surveyed here. Then we interpret the input received further.

6.1 Highlights of Surveys

For all the applications surveyed, portability was a key goal. Other highlights included:

- Many-body Methods for Light Nuclei
  - Conversion in an afternoon,
  - Uses only simple MPI calls,
  - Performance results already,
  - Application already able to obtain news physics results,
  - 100% FORTRAN application.
- UNCLE — Solution of Unsteady Incompressible Viscous Flows
  - Conversion in about three person weeks (preliminary),

- Uses multiple communicators to exploit structure,
- Uses user-defined generalized cartesian topologies,
- Eventually will use datatypes to simplify data transfer, and for possibly higher performance,
- 100% FORTRAN application.

- NAMMU — Groundwater Modeling
  - About 50 person days for conversion:
    1. 20 person days to restructure algorithm for parallelism,
    2. 30 person days to incorporate MPI calls into solver.
  - Simple communicators (duplicates of `MPI_COMM_WORLD`)
  - Collective operations only on "world,"
  - Extensive use of datatypes.
- PARAMICS-MP — Microscopic Traffic Simulation
  - Discrete event simulation,
  - Conversion from SIMD paradigm,
  - Target is Cray T3D with MPI,
  - MPI is now EPCC standard message-passing system.
- Volume Visualization
  - Communicators and topologies used extensively,
  - Strided datatypes will be used as application progresses,
  - Preliminary conversion of Argonne/Mississippi State MPI to nCUBE/2.
- A Fast Particle Summation Library
  - Library is a basis for many applications,
  - Performance was not significantly different from using EUI-H on IBM SP-1 (but would be much higher than the performance of the IBM EUI product),
  - Stresses the current MPI architecture for persistent communication objects.
- CVL
  - A couple of person months for port,
  - Optimization reduced to making sure MPI is fast on a platform,

  - Easiest CVL port undertaken by author to date.
  - Stressed the current MPI "scan" operations, which are inclusive, not exclusive. Indicated need for segmented scans and reductions.

- ParX

  - Represents distributed/parallel computing application,
  - Puts parallel graphics into MPI applications,
  - Utilizes inter-communicators,
  - First environment is Fujitsu AP-1000 running MPI,
  - Could have exploited asynchronous collective operations not in MPI.

Surveyed applications used either C or FORTRAN, with no particular bias. The variety of machines targeted was significant, beyond the requirement of portability. The Cray T3D, IBM SP-1, Fujitsu AP-1000, nCUBE/2, Meiko CS1/i860, and TMC CM-5 were mentioned specifically, as well as clusters of workstations; because of the strong portability features of MPI, other machines and cluster environments will evidently be targetted as well.

Although the performance results described in the foregoing are limited, it is clear that researchers are already getting good results on MPI-based applications. The Light Nuclei results are the most quantified here, but it is evident that others have also made timings, particularly the work on the Fast Summation Library, where the researchers mention comparable performance to the fastest non-MPI message-passing system available on the IBM SP-1. For other researchers, portability and development have preceeded application tuning, so their detailed performance results will come only later.

Several of the researchers utilized only the simple aspects of MPI. Others made extensive use of the collective operations; it is from these researchers that the most feedback has come about possible changes to MPI. Furthermore, the researchers who tried datatypes found them almost universally confusing.

6.2 Comments about how MPI should be improved

- A common request was for a broadcast primitive to be matched up with a simple receive.
- A common request was for simplifying the interface to derived datatypes, and to making these clearer.
- Asynchronous collective communication functions were desired by some (such as asynchronous broadcast, non-blocking reductions).
- One set of authors want a more efficient mechanism for request objects; current semantics imply higher-than-needed complexity for large numbers of outstanding sends and receives.
- Clearer, simpler documentation is obviously needed.

- More examples are needed, at varying levels of difficulty, in both C and Fortran.

7 Summary, Conclusions, Future Work

We surveyed application programmers and researchers concerning early applications in the Message-Passing Interface (MPI), whose standards document was finalized in May, 1994. We received nine responses, eight of which are described in this paper. Five of the responses were purely application projects, a sixth aimed at a class of applications, and the latter two responses were more generic application-enabling libraries.

Portability was universally a theme in all the responses received. Furthermore, there were only positive comments about the process of converting to MPI, with the exception that some of the researchers had specific needs for collective operations not addressed in the initial version of MPI. Furthermore, the weakness of the datatypes feature of MPI was shown repeatedly. The interface was seen to be confusing.

When the researchers commented on the standards document, they indicated that it needs to be made more understandable, have more examples (in Fortran as well as C), and that other simpler explanations are needed. We noted that several such projects (such as an annotated reference manual) are underway.

The variety of machines mentioned by the researchers was extremely interesting. Not only do the re-seachers use the IBM SP-1, CM-5, and nCUBE/2, newer machines like the Cray T3D, and less well known machines, like the Fujitsu AP-1000, were also mentioned. All are currently able to run a portable version of MPI, some will have commercial versions of MPI this year. This indicates significant penetration of the MPI standard after only a few months. Furthermore, the variety of implementations of MPI in use were interesting. The Argonne/Mississippi State implementation, IBM implementation, and Edinburgh Parallel Computing Centre implementations were mentioned. It is clear that at least two commercial MPI versions will emerge in 1994, as indicated in the body of this paper. Thus, application development is not at all premature in MPI.

It is interesting to note that several surveys mentioned using MPI on clusters of workstations, but there were no high-performance networks mentioned as yet. Although we know of a working version of MPI for Fiberchannel, availability of MPI for high-speed networks is probably not significant as yet, as gauged by the applications surveyed. We expect that, in the future, MPI will have a wide audience on high-performance clusters, nonetheless.

This paper will is necessarily the first step of an on-going study of the progress of these (and hopefully additional) key applications under MPI. We intend to update this paper from time to time, following the progress of applications, where possible, and adding new applications that demonstrate the effectiveness of MPI, and show where MPI needs additional or modified functionality in future releases.

8 Acknowledgements

We would like to thank the following researchers for contributing to the survey and to this paper:

- S. Pieper and R. Wiringa, Physics Division, Argonne National Laboratory,
- R. Pankajakshan and W. R. Briley, and J. Zhu, Mississippi State University, NSF Engineering Research Center for Computational Field Simulation.
- J. West, M. Stevens, L. Turcotte, Army Corps of Engineerings, Vicksburg,
- H. W. Yau, Edinburgh Parallel Computing Centre,
- K. A. Cliffe and D. Brear, AEA Technology, Harwell,
- J. A. Scott, Atlas Centre, Harwell,
- J. Salmon of Caltech,
- M. S. Warren of Los Alamos National Laboratory,
- D. J. Edelson of Northeast Parallel Architectures Center, Syracuse University,
- J. C. Hardwick of Carnegie Mellon University, and
- S. Ball, of Australian National University.

We also thank the editor of this special issue, Louis Turcotte, for his help, guidance, and patience.

We thank the members of the MPI Forum as a whole, for their efforts in creating MPI.

References

[1] Guy E. Blelloch, Siddhartha Chatterjee, Jonathan C. Hardwick, Margaret Reid-Miller, Jay Sipelstein, and Marco Zagha. CVL: A C vector library. Technical Report CMU-CS-93-114, School of Computer Science, Carnegie Mellon University, February 1993.

[2] Guy E. Blelloch, Siddhartha Chatterjee, Jonathan C. Hardwick, Jay Sipelstein, and Marco Zagha. Implementation of a portable nested data-parallel language. *Journal of Parallel and Distributed Computing*, 21(1):4–14, April 1994.

[3] Cameron, G, Smith, M, White, M, Wylie, B.J.N, and McArthur, D. The PARAMICS Parallel Microscopic Traffic Simulator. In *UK IT Forum Conference Digest*, 1994.

[4] Cameron, G, Wylie, B.J.N, and McArthur, D. PARAMICS: Moving Vehicles on the Connection Machine. In *Submitted to Supercomputing '94*, 1994.

[5] Nathan Doss, William Gropp, Ewing Lusk, and Anthony Skjellum. A model implementation of MPI. Technical report, Argonne National Laboratory, 1993.

[6] Steve Otto et al. *MPI Annotated Reference Manual.* MIT Press, 1994.

[7] Message Passing Interface Forum. Document for a standard message-passing interface. Technical Report Technical Report No. CS-93-214 (revised), University of Tennessee, April 1994. Available on **netlib**.

[8] Message Passing Interface Forum. MPI: A message-passing interface standard. Computer Science Dept. Technical Report CS-94-230, University of Tennessee, Knoxville, TN, April 1994. (To appear in the International Journal of Supercomputer Applications, Volume 8, Number 3/4, 1994).

[9] Hubertus Franke, Peter Hochschild, Pratap Pattnaik, and Marc Snir. An efficient implementation of MPI. (contact: Hubertus Franke, frankeh@watson.ibm.com), 1993.

[10] William Gropp, Ewing Lusk, and Anthony Skjellum. *Using MPI: Portable Parallel Programming with the Message-Passing Interface.* MIT Press, 1994.

[11] McArthur, D. The PARAMICS Model: Present and Future Directions. In *current working paper, to be submitted to the Transportation Research Journal*, 1994.

[12] Ramesh Pankajakshan and W. Roger Briley. Parallel solution of unsteady incompressible viscous flows. (in preparation).

[13] John K. Salmon, Grégoire S. Winckelmans, and Michael S. Warren. Fast parallel treecodes for gravitational and fluid dynamical N-body problems. *J. Supercomputer Appl.*, 8, 1994. (to appear) available as ftp/nbody/ijsa.ps.Z@sampson.ccsf.caltech.edu.

[14] Lafe K. Taylor, Judy A. Busby, Min Yee Jiang, Abdollah Arabshahi, Kidambi Sreenivas, and David L. Whitfield. Time accurate incompressible Navier-Stokes simulation of the flapping foil experiment, August 2–5 1993. Presented at the Sixth International Conference on Numerical Ship Hydrodynamics, Iowa City, Iowa.

[15] Lafe K. Taylor and David L. Whitfield. Unsteady three-dimensional incompressible Euler and Navier-Stokes solver for stationary and dynamic grids, June 24–26 1991. AIAA 91-1650. Presented at the AIAA 22nd Fluid Dynamics, Plasma Dynamics and Lasers Conference, Honolulu, Hawaii.

[16] Michael S. Warren and John K. Salmon. A parallel hashed oct-tree N-body algorithm. In *Supercomputing '93*, pages 12–21, Los Alamitos, 1993. IEEE Comp. Soc.

[17] Michael S. Warren and John K. Salmon. A portable parallel particle program. *Computer Physics Communications*, 1994. (submitted).

[18] David L. Whitfield and Lafe K. Taylor. Discretized Newton-relaxation solution of high resolution flux-difference split schemes, June 24–26 1991. AIAA-91-1539. Presented at the AIAA 10th Computational Fluid Dynamics Conference, Honolulu, Hawaii.

[19] Wylie, B.J.N, Cameron, G, White, M, Smith, M, and McArthur, D. PARAMICS: Parallel Microscopic Traffic Simulator. In *Proceedings of Second European Connection Machine Users Conference*, 1993.