# What's New in the Message-Passing Toolkit

Karl Feind, Message-passing Toolkit Engineering Team, SGI

**ABSTRACT:** SGI message-passing software has been enhanced in the past year to support larger Origin 2000 systems and clusters, to provide more of the MPI-2 API, and to enhance performance. Several Message-passing Toolkit (MPT) product releases went out in the past year to deliver the new functionality and performance. Other current issues of importance for message-passing users are also discussed.

## Introduction

This paper will describe significant enhancements to SGI's message-passing software from June 1998 to May 1999. *Message-passing* is a style of parallel programming which uses fast library calls to provide communication between cooperating parallel processes which are usually engaged in CPU-intensive work. SGI-supported message-passing application programming interfaces (APIs) include MPI, SHMEM, and PVM. All three APIs are packaged with the Message-Passing Toolkit (MPT) on IRIX and UNICOS systems. On UNICOS/mk systems SHMEM is packaged in the programming environments, but MPI and PVM are delivered with MPT. See the SGI MPT web page at http://www.sgi.com/software/mpt/ for more background and description of SGI message-passing software.

MPT releases in the past year included 1.2.1 in June 1998, 1.3 in February 1999, and several update releases in between. These releases provide a balance between performance and functionality on all supported SGI and Cray computer systems. Our goal is to provide message-passing implementations, which expose the high performance SGI computer systems on which it runs, while adhering to industry standards for MPI and PVM. At the same time we continue to support the proprietary SHMEM message-passing API both as a high performance stand-alone API as well as a high performance extension to the MPI API.

In the sections that follow, some feature highlights are described for the recent MPT releases. This discussion will only touch the surface; for more detailed information, consult the MPT release documentation or the above-listed MPT web page.

## MPT 1.2.1 Highlights

The MPT 1.2.1 release was made available in June 1998. This release contained a number of significant fixes and features on IRIX, UNICOS, and UNICOS/mk systems.

Prior to MPT 1.2.1, UNICOS and UNICOS/mk systems permitted the use of MPI and SHMEM in the same application, but IRIX systems did not. The MPI and SHMEM APIs both provide communication between a set of processes with identifiers, or *ranks*, in a contiguous range from 0 to N-1. Therefore it is natural for MPI and SHMEM to be peer APIs which provide communication between the same set of processes. The MPT 1.2.1 release added interoperability of MPI with SHMEM on IRIX systems.

In mid 1998, the IRIX Miser resource manager was

Optimizing MPI Programs with SHMEM: A Case Study

The Code

An SGI applications analyst in Eagan was optimizing a spectral climate model running on the Origin 2000..

The Optimization

Replace MPI calls in a transpose loop with SHMEM put/get communication calls.

The Result

The overall code speedup by 20%

being developed and stabilized to be usable for scheduling highly parallel programs, and MPT was enhanced at this time to permit the use of Miser to schedule MPI jobs. An update on the stabilization of Miser can be found later in this paper.

## MPT 1.3 Highlights

Released in February 1999, MPT 1.3 contained enhancements in the areas of large cluster support, performance optimizations, usability improvements, interoperability, and support for more of the MPI-2 standard API.

### Large Cluster Support

The MPI enhancement to support Origin 2000 clusters of up to 6144 processors on 48 hosts were accomplished through restructuring of the internal buffering algorithms and adding support for multiple HIPPI adapters in the same MPI job. If an MPI job can use more than one HIPPI adapter per host and multiple HIPPI switches, two benefits are seen. First of all, connectivity to more remote hosts can occur because of HIPPI switch port limits. Secondly, higher aggregate system bandwidth can be achieved when multiple switches provide alternate data paths between hosts.

### Performance Improvements

The MPI_Barrier operation on Origin 2000 systems was optimized to use the same fetch-op barrier algorithm as is used by SHMEM. This provided a dramatically faster MPI_Barrier. This effort was the beginning of a continuing effort to improve the performance of MPI collective communication on IRIX systems.

### Usability Enhancements

A number of features in MPT 1.3 for IRIX systems helped users diagnose program errors. Program launch errors had not been well reported in the past, but enhancements to MPI and Array Services 3.2 improved matters significantly. Now most common program launch errors are diagnosed with a meaningful error message.

Another class of Fortran MPI program error can now be detected at compile time. The C Programming language has long provided compile-time checking for the number and type of function arguments through the use of prototypes in C header files. The Fortran 90 standard has more recently provided the same capability through use of interface modules. MPI and SHMEM now both provide interface modules which Fortran programmers can use without any changes to portable FORTRAN 77 or Fortran 90 MPI or SHMEM source code. To activate the compile-time interface checking, use the `f90` compiler and the following command-line option:

```
-auto_use mpi_interface,shmem_interface
```

The MPI-2 Standard defines a similar capability which is activated when the MPI module is imported on a USE statement. Unlike SGI's current implementation, the standard-defined capability requires changes to existing MPI-1-conformant programs.

### Performance Analysis and Debugging

The `-stats` option on the `mpirun` command causes a statistics report to be printed at MPI program termination. These statistics characterize the types and quantity of MPI communication—point-to-point, collective, inter-host, intra-host—for every process in the MPI program.

Additional debugging support was provided through increased interoperability with the Totalview debugger. MPT on UNICOS systems added basic support for the Totalview debugger. MPT on

IRIX systems enhanced its interoperability with Dolphin Totalview to permit the display of message queues.

**MPI-2 API Support**

Although many MPI users are content with the scope of the MPI-1 API, increasing numbers of users are requesting some of the functionality defined by the MPI-2 standard. In MPT 1.3 we provided most of the MPI-2 I/O API for IRIX and UNICOS/mk systems. This was done by taking version 1.0.1 of the ROMIO public domain implementation of MPI-2 I/O and integrating it into our MPI libraries.

MPI thread support, defined as an optional capability in the MPI-2 standard, was added on IRIX systems. This functionality will be useful to some users who wish to use thread parallelism and message-passing process parallelism in the same application.

## MPT 1.3.0.1 Highlights

The MPT 1.3.0.1 update was released in May 1999. A variety of features and bug fixes are available in this release. CRAY SV1 cache and memory system support is enhanced in SHMEM. This was significant because previously supported CRAY YMP and CRAY J90 systems had not activated data cache in message-passing programs. The other significant enhancement in this release was on IRIX systems where the MPI_Alltoall collective communication function was enhanced. More discussion of this will follow in the next section.

## High Performance Message Passing

On CRAY T3E systems, many parallel programmers use the SHMEM API to get the best point-to-point and collective communication performance. The SHMEM API is a thin veneer layered on top of remote and local memory copy primitives. SHMEM's thin veneer and high performance are also available on Origin 2000 systems, but for various reasons proportionally more Origin 2000 users than CRAY T3E users are preferring industry standard APIs like MPI. In part this is because MPI supports clustered systems, whereas SHMEM is supported only on systems with globally available direct memory access (DMA) capability. This section describes a few ways SGI's MPI on IRIX systems has been enhanced to provide MPI users with better performance

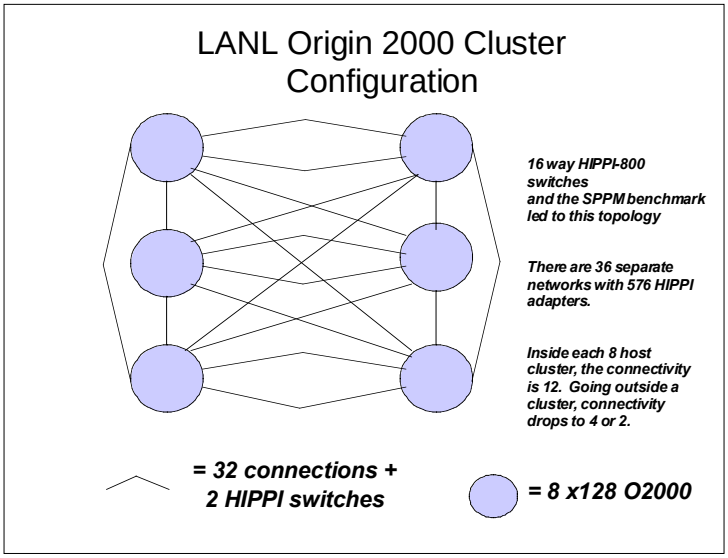


Figure 1

One strategy towards high performance computing is higher levels of parallelism. Parallelism on Origin 2000 clusters was pushed to a new level in MPT 1.3 with the support for larger clusters. In late 1998 the LINPACK

benchmark was run on a 40-host Origin 2000 cluster at Los Alamos National Laboratory (LANL). LINPACK was run on 5040 processors and achieved 1.6 Tflops. This LINPACK run was one step in our process of stabilizing MPI support for large clusters.

Another strategy for improving performance of highly parallel codes is to optimize communication. The MPI collective communication routines are defined in a way, which is conducive to architecture-specific optimizations within an MPI implementation. In MPT releases 1.3 and 1.3.0.1, the `MPI_Barrier` and `MPI_Alltoall` operations on IRIX systems are optimized to take better advantage of underlying hardware capabilities.

**MPI_Barrier Optimization**

Prior to MPT 1.3, `MPI_Barrier` used a portable algorithm layered on MPI send and receive calls. In MPT 1.3 a two-fold strategy was used to improve MPI barrier performance. For programs running on a single host, an MPI-aware version of SHMEM's fetch-op barrier was used. For programs running on multiple hosts, the processes on each host synchronized with each other, and then faster cross-host barrier synchronization was performed using just one representative process from each host. The resulting MPI barrier performance improved by several orders of magnitude for some common cases.

| Origin 2000 MPI_Barrier Time in Microseconds | | | |
|---|---|---|---|
| Hosts x Processes | Prior to Optimization | With Optimization | Improvement |
| 1 x 64 | 3140 | 10 | 300 x |
| 1 x 128 | 24000 | 26 | 1000 x |
| 2 x 4 | 670 | 174 | 4 x |
| 4 x 16 | 26000 | 994 | 26 x |

**MPI_Alltoall Optimization**

A somewhat similar approach was taken in MPT 1.3.0.1 to improve the performance of `MPI_Alltoall`. For programs running within a single host, the collective operation detected cases where the send or receive buffer was in a remotely accessible data area as defined in the `shmem(3)` man page.. For Fortran programs, this includes common blocks and memory allocated by `SHPALLOC(3)`. For C programs, static data and memory allocated with `shmalloc(3)` is remotely accessible. When remotely accessible send or receive buffers are detected, the `MPI_Alltoall` operation is performed by simple memory copies.

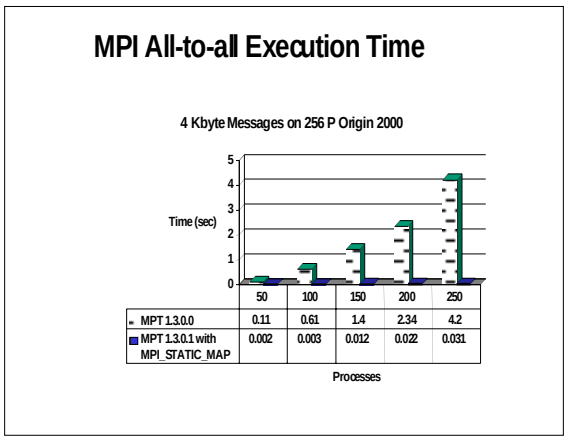


Figure 2

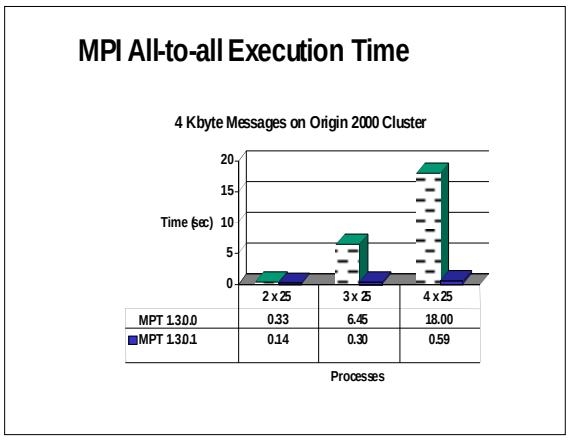


Figure 3

The single-host optimization yielded performance improved by several orders of magnitude, as shown by the table and graph in figure 2.

This optimization is activated by setting the MPI_STATIC_MAP environment variable in the current MPT release. In future releases we plan to activate the optimization by default, to extend the optimization to send and receive buffers allocated by the Fortran 90 ALLOCATE statement, and to use a similar approach to optimize the other MPI collective operations.

A multi-host `MPI_Alltoall` optimization was accomplished by finding a way to send fewer messages of larger size between the hosts. Inter-host communication via HIPPI has a noticeable latency penalty while providing very good peak bandwidth. The optimized algorithm first exchanged the data between hosts. Then several intra-host all-to-all operations shifted the data around into its final location. Figure 3 shows the dramatic performance improvement.

The improved all-to-all performance resulted in highly improved NAS Parallel benchmark performance. The class B FT benchmark was run with and without the optimization. Figures 3 and 4 show the improved performance.

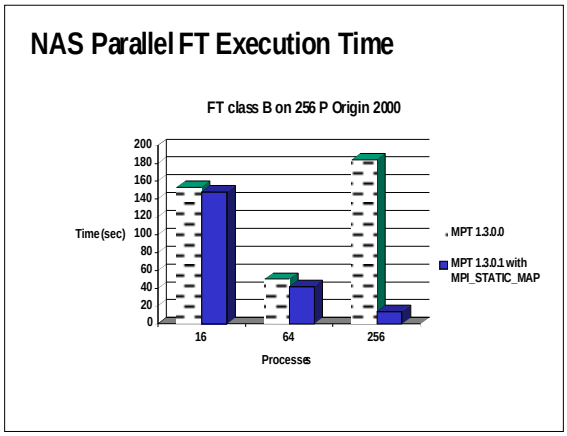


Figure 4

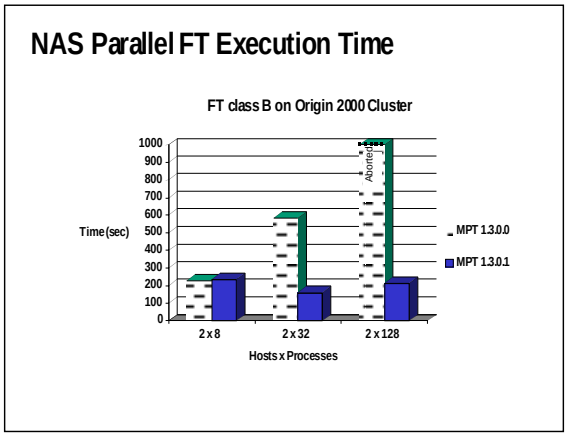


Figure 5

## Scheduling Message Passing Programs on IRIX Systems

In a time-shared environment, highly parallel applications using MPI or SHMEM message passing or OpenMP are best scheduled using a synchronizing scheduler like Miser. When Miser schedules all processes in the application to execute at the same time several important benefits are achieved. Most parallel applications are synchronized single-program, multiple data (SPMD) programs. Thus an early start for a subset of processes in the parallel job only leads to more wait time, often spin wait time, while the early starters wait for the late-starting processes to catch up.

Failure to use synchronized scheduling results in performance non-repeatability and poor system throughput of parallel jobs. Performance non-repeatability means that the real time to completion for the application as a whole will be much different in a time-shared environment when compared with a dedicated system. Lower system throughput arises because the waiting processes consume system memory for a longer period, and the spin waiting consumes CPU time that could otherwise be allocated to processes ready to do user computation.

In IRIX 6.5.4, which was made available to customers in May 1999, Miser has enhanced stability and functionality which helps message passing users on time-shared systems to get the synchronized scheduling they need. SHMEM programs can be scheduled with Miser using this type of command line:

```
setenv NPES 64
miser_submit -q default -o c=64,m=4g,t=10h,static a.out
```

MPI programs can be scheduled with Miser using this type of command line:

```
miser_submit -q default -o c=64,m=4g,t=10h,static \
      mpirun -miser -np 64 a.out
```

See the paper *IRIX Resource Management Plans and Status* in these CUG proceedings for more information about Miser.

## Message-Passing Future Plans

Message passing enhancements planned for the year ahead have several themes--performance enhancements, new hardware support, and continued phase-in of more of the MPI-2 API. The following table summarizes the expected timetable for new message passing features.

| Date | Release | Feature |
|---|---|---|
| October 1999 | MPT 1.4 | HIPPI Resiliency Support for Large Origin Clusters |
| | | MPI Support for CRAY SV1 systems |
| | | MPI-2 C++ Bindings on IRIX systems |
| | | MPI Collectives Optimizations on IRIX systems |
| | | GSN support infrastructure |
| | | Improved Cleanup of Aborted MPI Jobs on IRIX systems |
| 2000 | MPT 1.5 | MPI-2 one-sided Communication on IRIX systems |
| | | MPI-2 I/O Enhancements on IRIX systems |
| | | MPI Support for GSN |
| | | MPI Enhancements to LSF Support |