SGI Message-Passing Status and Plans

Karl Feind, SGI

ABSTRACT: SGI message-passing software has been enhanced in the past year to support additional interconnect fabrics, improve NUMA-awareness, increase MPI-2 content, and provide other improvements. This paper describes the recent enhancements to MPI and SHMEM software and also outlines our roadmap of planned future enhancements.

Introduction

This paper will describe enhancements to SGI’s message-passing software. Message-passing is a style of parallel programming that uses fast library calls to provide communication between cooperating parallel processes typically 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 SGI Message-Passing Toolkit (MPT) on IRIX systems. See the SGI MPT web page at http://www.sgi.com/software/mpt/ for more background and description of SGI message-passing software.

Recent SGI MPT releases were 1.4.0.2 in August 2000, 1.4.0.3 in November 2000, and 1.5 in March 2001. MPT update release 1.5.1 is planned for June 2001 release. In this paper we will highlight some features of the MPT software thematically. For more detailed information about features, consult the MPT release documentation on the above-listed MPT web page or in the relnotes included with each MPT release.

MPT Feature Themes

The goals we have for SGI message-passing software include delivering communication performance, supporting all SGI high performance computing (HPC) hardware platforms, and providing increasing conformance to the MPI-2 standard. Performance features deliver favorable point-to-point communication microbenchmark performance, but more importantly they deliver performance to actual user programs. For example, fast message queuing and tuning of message headers help with high message contention, and runtime-tunable buffer management and single-copy data transfer help out with bandwidth-limited communication. Hardware platform support involves new generations of MIPS-based servers, early work with Intel-based servers, and a number of new and enhanced interconnects. The final feature theme, standards, acknowledges the gradually increasing customer interest in various MPI-2 capabilities and also involves a transition away from our support of PVM.

Performance

Message-passing software at SGI provides a number of performance-oriented features.

Low latency

For message-passing communication over NUMAlink on Origin 3000 servers, send-receive exchange of an 8-byte datum is performed in 5.5 microseconds. A put or get operation incurs a latency of less than 1 microsecond.

High bandwidth

The peak NUMAlink bandwidth for send-receive on Origin 3000 systems is 600 MB/sec when MPI uses the Block Transfer Engine (BTE). The capability to use BTE is planned for upcoming MPT release 1.5.1. When BTE is not being used, bcopy-based bandwidth achieves 280 MB/sec. Messaging bandwidth is also important in situations where all processes are communicating
concurrently, as during data exchanges with logical nearest neighbors in a domain-decomposed application. MPT currently delivers up to 170 MB/sec per transfer on Origin 3000 with NUMAlink in all-communicate scenarios involving two memory nodes and the eight associated processors.

All-communicate Scenario

One-sided Communication
One-sided communication or put/get communication is provided in MPT via the SHMEM and the MPI-2 APIs. This style of communication can provide the lowest possible latency and the highest possible bandwidth for some communication patterns. SGI continues to support the SHMEM API because of its ease of use. The MPI-2 syntax for put/get communication is slightly more complicated to program, but is seeing increased use. The SHMEM API is more flexible and intuitive in some ways. It allows for the use of flag words to signal data delivery with user-written spin-waits. However, both APIs are easy to program when the common (barrier, communicate, barrier, compute) pattern is used in an application. Both APIs for put/get and the associated barrier synchronization routines (shmem_barrier_all and MPI_Win_fence) perform comparably on Origin 3000 systems with the NUMAlink interconnect.

Fast Barriers
Barrier synchronization times on Origin 3000 systems have accelerated proportionately with the reduced memory latency. Using the fetchop tree barrier, which is selected by setting the MPI_BAR_DISSEMM environment variable, barrier synchronization times for 494 processes is 26 microseconds on Origin 2000 systems and 14 microseconds on Origin 3000 systems.

Physical CPU Selection
The MPT 1.5.1 will provide a new capability, which should prove helpful for those who wish to optimize and closely study the performance of parallel applications on dedicated Origin 2000 and Origin 3000 systems. The new MPI_DSM_CPULIST and SMA_DSM_CPULIST environment variables provide an easy way to specify the physical CPUs from /hw/cpunum that the ranks of the message-passing program are to be mapped to. For example, setting MPI_DSM_CPULIST to "4-7,12-15" specifies that ranks 0 through 7 are mapped respectively to CPUs 4, 5, 6, 7, 12, 13, 14 and 15.

Single-Copy Send/Receive
In MPT 1.4.0.1 and later it is possible for MPI to deliver messages using a single data copy. In the typical and default send/receive pathway, MPI will copy the sender's data...
into a temporary buffer from which the receiver then copies the data. This type of
algorithm can lead to contention for MPI message buffers and bandwidth bottlenecks
when large messages are being sent, often by many processes concurrently. The single-
copy send/receive pathway can avoid this contention and improve performance in some
cases.

To select the single-copy messaging pathway, the user needs to set the
MPI_BUFFER_MAX environment variable to some single-copy threshold value. We
recommend setting MPI_BUFFER_MAX to 2000, to activate single-copy only for the
larger messages. Because of the way that MPI maps memory, the single-copy
send/receive pathway will not be taken unless the buffer passed by the sender is
remotely accessible. Remotely accessible memory includes common blocks and
symmetric heap. It excludes stack and private heap variables. We have seen
performance gains realized in user codes that do large broadcasts and message
exchanges similar to the MPI_Alltoallv collective communication. Speed-ups can be by
more than a factor of two because of the combined effects of reduced bandwidth
requirement and reduced message buffer contention.

The improvements from single-copy can be dramatic, but there are some caveats. Some
MPI programs are written with the assumption that MPI_Send buffers its data. This is not
conformant with the standard, but nevertheless some programs have this assumption.
The single-copy pathway requires that MPI_Send not buffer its data. Performance in
many of these data exchanges will not optimize well unless the user is calling the non-
blocking MPI_Isend variant on the sending side.

Reducing Runtime Variability
On single-kernel Origin 3000 systems, MPI and SHMEM programs are cpuset-friendly.
That is to say that when message-passing programs are launched within a cpuset, they
will be scheduled optimally within the cpuset. A number of third party vendors’ batch
workload schedulers take advantage of this. The result is a dramatic reduction in the
interference between two different parallel jobs scheduled to run concurrently. MPT 1.5.1
has further enhanced the interoperability of MPI with cpusets that contain partial memory
nodes or nodes with downed CPUs.

Platforms and Interconnects
The MPI, SHMEM, and PVM APIs are available on differing sets of platform and interconnect
configurations. The following chart shows SGI’s status and plans for support of these APIs. Note that
these are plans, and do not constitute commitments by SGI.

<table>
<thead>
<tr>
<th>Platform</th>
<th>MPI</th>
<th>SHMEM</th>
<th>PVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIPS: single kernel NUMAlink</td>
<td>Supported</td>
<td>Supported</td>
<td>PVM will be retired after MPT 1.6</td>
</tr>
<tr>
<td>MIPS: partitioned NUMAlink</td>
<td>Supported in MPT 1.5.1</td>
<td>Planned late 2001</td>
<td></td>
</tr>
<tr>
<td>MIPS: Myrinet</td>
<td>Supported in MPT 1.5.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MIPS: HIPPI</td>
<td>Supported</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MIPS: Sockets</td>
<td>Supported</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### MPI Standards

SGI's strategy towards the implementation of MPI-2 functionality is to focus on customer needs and prioritize the work on a feature-by-feature basis. Over time, MPI-2 content in MPT will increase. As of MPT 1.5, significant MPI-2 functionality exists in MPT in the following areas:

- One-sided communication
- MPI I/O
- Thread Safety
- Fortran 90 bindings (USE MPI)
- C++ bindings

MPI-2 functionality planned in upcoming releases includes:

- MPI I/O integration with MPI_Wait
- MPI-2 datatypes that replace deprecated MPI-1 Fortran datatypes
- Expanded one-sided communication: lock, unlock, and accumulate
- MPI Process Spawn

### Reading More about MPT Software

A number of references exist for SGI message-passing software information.

**MPT Relnotes**

The "relnotes mpt" command provides information about new features and bugfixes, as well as administration information to help with software installation.

**Man Pages**

The "man mpi" command will display a reference page containing general topics about using MPI. This page lists all runtime environment variables used for runtime tuning of MPI.

The "man shmem" command describes the SHMEM API and also lists environment variables used for runtime tuning of SHMEM.

**Web Pages**

The MPI Programmers Reference Manual is viewable with the "insight" command and also on the web at [http://techpubs.sgi.com](http://techpubs.sgi.com).