

Myrinet Protocol Module Validation Guide for Sun HPC ClusterTools 4.0

Myricom, Inc.
Draft: 5 April 2003

Table of Contents

<u>I. Introduction</u>	3
<u>II. Install and Validate GM</u>	3
<u>1. Configuring and compiling GM.</u>	4
<u>2. Installing the GM Driver.</u>	5
<u>3. Run the GM mapper.</u>	6
<u>4. Enabling IP over Myrinet (Ethernet Emulation).</u>	8
<u>5. Testing/Validation.</u>	8
<u>a. Check the LEDs on each switch port and interface.</u>	8
<u>b. Run gm_board_info on one host.</u>	9
<u>c. Run gm_debug to test the PCI bandwidth.</u>	10
<u>d. Run gm_allsize to test links in the network.</u>	10
<u>e. Run gm_stress to test the network.</u>	12
<u>III. Install and Configure Myrinet PM for Sun HPC ClusterTools 4.0.</u>	15
<u>1. Install Sun HPC ClusterTools 4.0</u>	15
<u>2. Install the Myrinet PM libraries</u>	15
<u>3. Set up GM hostnames.</u>	16
<u>4. Edit the GM configuration file</u>	16
<u>5. Edit the hpc configuration file</u>	17
<u>6. Start the daemon (requires root privileges)</u>	18
<u>7. Initialize the "all" partition (if necessary).</u>	18
<u>8. Rerun the daemon (if necessary)</u>	19
<u>9. Run-Time Tuning Options.</u>	19
<u>IV. Test Myrinet PM Using the Intel Testing Suite.</u>	20
<u>V. Test Myrinet PM Using the MPICH Testing Suite.</u>	22
<u>1. Download MPICH-GM</u>	22
<u>2. Testing the basic function cpi</u>	22
<u>3. Testing the point-to-point programs</u>	22
<u>4. Testing the performance test program -- mpptest</u>	23
<u>VI. Test Myrinet PM Using the PALLAS Benchmark Suite.</u>	24

I. Introduction

The Sun HPC ClusterTools is a high performance cluster software package, providing an implementation of MPI. The Myrinet PM is a loadable protocol module for the ClusterTools software to carry MPI traffic directly over GM (the message passing system of Myrinet) over Myrinet networks with low latency and high data rates. Myrinet PM is implemented and supported by Myricom. Features of the Myrinet PM include:

- *Uses RTE (run time environment) for initialization. This provides good scalability for large configurations. Also supports a minimum configuration of two hosts with a point-to-point Myrinet connection without a Myrinet switch.*
- *Each process allocates a GM port locally and dynamically.*
- *Uses efficient pipelined memory copy and GM directed send (PUT) for large messages.*
- *Uses different protocols – eager, rendezvous or multiphase rendezvous protocol, for messages of different sizes. The thresholds are tunable at run-time. The default eager/rendezvous threshold is (16K-96) bytes for 32-bit applications and (16K-112) bytes for 64-bit applications respectively. The default rendezvous/multiphase-rendezvous threshold is (64K-8) bytes.*

Validating Myrinet PM for Sun ClusterTools 4.0 involves 4 steps:

- *Install and validate GM*
- *Install and configure Myrinet PM*
- *Test Myrinet PM using the Intel testing suite*
- *Test Myrinet PM using the MPICH testing suite*
- *Test Myrinet PM using the PALLAS Benchmark Suite*

II. Install and Validate GM

The current GM software release for Solaris on UltraSPARC hosts is gm-1.6.4_Solaris. Compared to the previous GM 1.6 software, this gm-1.6.4_Solaris release provides significant performance improvements in both latency and bandwidth.

Starting from gm-1.6_Solaris, the following important features have been included:

- *GM 1.6 and later supports 32-bit and 64-bit user applications concurrently.*
- *GM 1.6 and later provides a shared library in addition to the static library. Applications that are linked with the GM shared library can use a new GM installation without re-compiling or re-linking once the paths are properly set.*
- *GM 1.6 and later improves throughput on the SPARC architecture by safely using streaming rather than consistent DMA.*

Solaris users are encouraged to check

<http://www.myri.com/scs/solaris.html>

for the newest release of GM.

The GM software for Solaris is distributed in both source and binary forms. The installation procedures for the source and binary forms of GM are similar except that installing GM from a source release requires an extra step for compilation.

Starting from source compilation, GM installation is performed in 5 easy steps:

1. Configuring and compiling GM.
2. Installing the GM driver.
3. Running the GM mapper.
4. Enabling IP over Myrinet (ethernet emulation).
5. Testing/Validation.

1. Configuring and compiling GM.

- **Configuring and Compilation from Source Release**

Building GM from source form requires the Sun C++ compiler and Gnu make.

Download GM

```
ftp://comrade@ftp.myri.com/pub/GM/gm-1.6.4_Solaris.tar.gz
```

```
gunzip -c gm-1.6.4_Solaris.tar.gz | tar xvf -
cd gm-1.6.4_Solaris
autoconf
autoheader
./configure
```

Add the "--disable-64b" option if you are building a 32-bit driver on a machine running a 64-bit kernel, or "--enable-64b" if you are building a 64-bit driver on a machine running a 32-bit kernel. Without these options, a driver matching the running kernel on the compilation machine will be built.

```
make depend
make
cd binary
```

- **Unpacking GM from Binary Release**

For PCI64B/C (LANai 7/9), download the current GM binary releases for machines running Solaris 64-bit kernel

```
ftp://comrade@ftp.myri.com/pub/GM/gm-1.6.4_Solaris64-sun4u-SunOS-5.7-8port-heart-beat-on.tar.gz
```

```
gunzip -c gm-1.6.4_Solaris64-sun4u-SunOS-5.7-8port-heart-beat-on.tar.gz|tar xvf -
cd gm-1.6.4_Solaris-sun4u-SunOS-5.7
```

2. Installing the GM Driver.

Select an installation directory path *<GM_install_path>*. It is usually best for *<GM_install_path>* to be the path to an NFS-mounted directory, accessible using *<GM_install_path>* from all machines that are to share the installation. *<GM_install_path>* must be an absolute path; it must start with /. However, *<GM_install_path>* may contain symbolic links.

Note: The <GM_install_path> installation directory must be created prior to invoking the GM_INSTALL script.

```
./GM_INSTALL <GM_install_path>
```

If you omit *<GM_install_path>*, the driver will be installed in the default directory, **/opt/gm**.

Next, you must run

```
su root
<GM_install_path>/sbin/gm_install_drivers
/etc/init.d/gm start
```

on each machine to install the drivers on that machine.

The **gm_install_drivers** script performs the following operations:

- Shuts down existing IP over Myrinet
- Unloads existing GM module (if it exists)
- Creates the devices (/dev/gm* and /dev/gmp*)
- Loads the GM module

If you wish the driver to auto-load at boot, you must create appropriate links in the **/etc/rcN.d** directories to the **/etc/init.d/gm** and **/etc/init.d/myri** scripts. Alternatively, you may start and stop the driver manually using

```
su root
```

```
/etc/init.d/gm start
/etc/init.d/gm stop
```

OR

```
su root
/etc/init.d/gm restart
```

Once the **gm_install_drivers** script has been run on each host, the yellow/amber "LANai" LED should illuminate on the faceplate of the Myrinet PCI/Host interface. The yellow "LANai" LED is controlled by the LANai processor, and will pulse like a heartbeat while the GM MCP/firmware is running, and will pulse faster when there is more packet-sending activity (including sending acknowledge packets in reply to packets received.) If the yellow LED is not pulsing, the GM MCP is not loaded or is not running.

Note:

1. GM is not in the critical performance path so it does not need to be built with specialized compilers and flags.
2. GM should be installed in an NFS-mounted area.
3. **gm_install_drivers** needs to be run on all nodes in the cluster!
4. By default, we assume that you have PCI64, PCI64A, PCI64B, or PCI64C interfaces. (PCI32 interfaces are not supported in gm-1.6.4)
5. If a host is rebooted, you must reload the GM driver (and rerun the GM mapper).

3. Run the GM mapper.

```
cd <GM_install_path>/sbin/
su root
./mapper ../etc/gm/map_once.args
```

Important points to note:

- The GM mapper is ONLY run on one node in the cluster. You should choose one node in the cluster to be the *mapper node*, and any subsequent invocations of the mapper should be done on this node only.
- The GM mapper must be run before any communication over Myrinet can occur.
- If a host is rebooted, you must reload the GM driver and rerun the GM mapper.
- If any topological change occurs in the cluster, the GM mapper must be rerun.
- Never run the GM mapper on multiple nodes at the same time as serious routing confusion will result.

The aforementioned mapping procedure uses the most common form of mapping: "Map Once" Mapping. Depending upon your needs, there are three ways to run the GM mapper:

- Map Once Mapping
- Static or “File” Mapping
- High Availability (HA) Mapping

“Map Once” Mapping is by far the most common way of running the GM mapper. In this method, the mapper is run on one host in the network (any of the hosts). It is rerun if a host (re)boots or a hostname is changed or after a change of Myrinet topology (swapping of ports on a switch). The command for this method of running the GM mapper is:

```
cd <GM_install_path>/sbin/
su root
./mapper ../etc/gm/map_once.args
```

“Static” Mapping is another way in which the GM mapper may be used. In this method, an active mapper is run once when ALL of the hosts are up and running the GM driver.

- This initial active mapper will generate a map file and a host file.
- These files are then shared by NFS, or copied to all of the hosts in the network.
- An entry in the boot scripts will allow each host to read the map file and the host file and update the routing table on its local Myrinet interface(s).

The command for this method of running the GM mapper is:

```
cd <GM_install_path>/sbin/
su root
./mapper ../etc/gm/static.args
```

If GM is not installed in an NFS-mounted area, copy the three files created by this command (**static.map**, **static.routes**, and **static.hosts**) to each <GM_install_path>/sbin/ directory on each host.

For auto-mapping at boot time, add the following command to the boot scripts of the host (scripts in /etc/init.d or /etc/rcN.d)

```
cd <GM_install_path>/sbin/
su root
./mapper ../etc/gm/static.args
```

“High Availability” Mapping is the third way in which the GM mapper may be used. This method is for users who have a need for High Availability (HA) in an aggressive computing environment. The command for this method of running the GM mapper is:

```
cd <install_path>/binary/sbin/
su root
./mapper ../etc/gm/active.args &
```

“High Availability” Mapping will continuously run the GM mapper in the background to detect and add any new hosts or remove any non-responding hosts, to detect any change of topology (change of slots in the switch, change of innerswitch topology), and periodically update the routing tables of the Myrinet cards (by default, every 30 seconds).

You should note that this HA mapping method is slightly intrusive. Since the GM mapper uses unreliable messages that may be dropped in case of heavy contention, this method can lead to hosts involved in a long computation being marked as “non-responding” and removed from the routing tables because they are unreachable.

For the majority of users, the "map_once" GM mapping method is sufficient. For the users with more production-level constraints, the "static mapping" is the recommended method. For fault-tolerant GM applications, the third method provides the best alternative.

4. Enabling IP over Myrinet (Ethernet Emulation).

Enable IP over GM using the following command:

```
/sbin/ifconfig myri0 <ip_address> up
```

where you may replace *myri0* with the appropriate name (*myri1*, *myri2*, etc.) if you have more than one Myrinet interface per host, it is not required by Myrinet PM to have IP enabled on the Myrinet interface other than *myri0*.

5. Testing/Validation.

Once the GM software is properly installed on all hosts in your cluster, you are ready to "validate" your Myrinet installation by performing the following sequence of tests.

- Check the LEDs on each switch port and interface
- Run `gm_board_info` on one host
- Run `gm_debug` to test the PCI bandwidth
- Run `gm_allsize` to test the links in the network

These steps are detailed below and are also described in the “Troubleshooting” section of the FAQ (<http://www.myri.com/scs/faq/faq-troubleshooting.html#debug9>). Once you have followed these steps, you will have a solid Myrinet installation.

a. Check the LEDs on each switch port and interface

After the hardware installation and GM software installation have been completed, there will be a green LED and flashing yellow/amber LED illuminated on each interface, and a green LED illuminated on each connected switch port.

If a green LED is not illuminated on each connected interface and switch port, then

refer to http://www.myri.com/scs/doc/troubleshooting_guide.pdf Section III (page 6).

If a yellow/amber LED is not illuminated on each connected interface, then refer to http://www.myri.com/scs/doc/troubleshooting_guide.pdf Section V (pages 8—10).

b. Run `gm_board_info` on one host

If all the LEDs are illuminated, you can now test that the GM mapper has correctly detected all of the hosts in your Myrinet network by using following command on one host:

```
cd <GM_install_path>/bin/  
./gm_board_info
```

Assuming that the `gm` module has been loaded and the GM mapper has been run, `gm_board_info` displays information about each Myrinet interface (board) installed on this host, as well as a list (routing table) of the Myrinet nodes with which it is able to communicate.

- If you see ***** No routes found ***** at the bottom of the display for each board, this is an indication that the GM mapper has not been run on the Myrinet network since this host was powered on or rebooted. You must run the GM mapper (as previously described) before you can proceed.
- If the list of nodes (routing table) is not complete (that is, it does not contain all of the nodes in the cluster) you will need to determine why some nodes are not listed. Refer to the following guidelines.
 1. Was GM properly loaded on all hosts in the Myrinet network?
 - If so, you should see the `gm` module loaded (**modinfo**) and the devices (`/dev/gm*` and `/dev/gmp*`) created. Proceed to (2.) below.
 - If not, run `gm_install_drivers` on the missing hosts, and then rerun the GM mapper. If `gm_install_drivers` fails, refer to the FAQ entry (<http://www.myri.com/scs/faq/faq-install.html#install6b>).
 2. If the `gm` module is loaded and the devices are created but you still do not see a specific host in the routing table, check the basic connectivity on the missing host by running the following **loopback-mode gm_allsize test** on this particular host:

```
gm_simpleroute --loopback [--board=n]  
gm_allsize --geometric -exit-on-error [--board=n]
```

If `gm_allsize` fails, then the host is disconnected; check the cable from the interface to the switch to make sure that it is well-connected and that a green "link" LED is illuminated on the interface and on the port on the switch to which the cable is connected. You should also check the kernel log messages (with "dmesg") on this host for any GM errors. If there are abnormal messages in the kernel log, then check the FAQ or contact help@myri.com. If there are

not any abnormal messages in the kernel log, the problem may be hardware-related. You should try a different cable and then try a different port on a switch line card to see if this interface can see the Myrinet network.

Note: After running tests with **gm_simpleroute** the mapper must be run again so that this host will have the correct routes to other hosts.

Note that **gm_simpleroute** is an alternative to the GM mapper that you may prefer for early testing. (Loopback and point-to-point (back-to-back) network topologies require **gm_simpleroute** be run instead of the GM mapper.) **gm_simpleroute** may be used to set up a one-node "network", thus enabling sanity tests like **gm_board_info** and loopback-mode **gm_allsize**.

c. Run **gm_debug** to test the PCI bandwidth

We recommend the following test to verify your GM performance.

```
cd <GM_install_path>/bin/  
gm_debug -L
```

This **gm_debug** test displays the results of the hardware benchmark test of the PCI bus with the DMA engine of the Myrinet interface. The output of this command indicates the maximum sustained bandwidth that can be obtained from the PCI bus, and thus provides an upper bound on GM performance. A detailed description of this benchmark can be found in the FAQ (<http://www.myri.com/scs/faq/faq-gm.html#gmq15>).

The output of this command also tells you if the Myrinet interface was correctly detected as 64-bit / 66 MHz, for example. If the interface was not correctly detected by the BIOS, you should suspect a riser card problem or a PCI slot problem.

d. Run **gm_allsize** to test links in the network

If **gm_board_info** shows all of the hosts in the cluster, you are ready to proceed to running **gm_allsize**. As previously mentioned, the test program **gm_allsize** can be used as a basic connectivity test (loopback-mode); however, **gm_allsize** can also be used to measure latency and bandwidth. One-way latency and bandwidth tests between two hosts in the Myrinet network can be performed with **gm_allsize** by running in slave mode on one node and as master on the GM node to be tested, specifying the target host as the slave host. This is done by adding the **gm_allsize** run-time option **--slave** on the command line of the slave node and the option **--remote-host=<host1>** on the command line of the master node, where **<host1>** is the name of the host running in slave mode.

(The name of each Myrinet host can be found by running **gm_board_info** on one of the hosts.)

Adding the **--verify** flag to any **gm_allsize** command will augment the test with verification of the contents of all messages, at the cost of significantly degraded

performance. For a list of all options to **gm_allsize**, type `gm_allsize -help` or refer to the FAQ. For sample output of **gm_allsize**, refer to the FAQ (<http://www.myri.com/scs/faq/faq-troubleshooting.html#debug9>).

- **Latency**

To test the **latency** between two hosts (*host1* and *host2*), type the following on *host1*:

```
gm_allsize --slave --size=15
```

and on *host2* type:

```
gm_allsize --remote-host=host1 --geometric --size=15
```

The output from this command will consist of two columns of data: the first column lists the message size (in bytes), and the second column lists the latency (in microseconds).

- **Unidirectional Bandwidth**

To test the **unidirectional bandwidth** between two hosts (*host1* and *host2*), type the following on *host1*:

```
gm_allsize --slave --size=15
```

and on *host2* type:

```
gm_allsize --unidirectional --bandwidth \  
--remote-host=host1 --size=15 --geometric
```

Unidirectional bandwidth is a PingPing test, measuring the startup and throughput of a single message sent between two processes, where messages are obstructed by oncoming messages.

The output from this command will consist of two columns of data: the first column lists the message size (in bytes) and the second column lists the bandwidth (in MB/s).

- **Ping Pong Bandwidth**

To test the **ping pong bandwidth** between two hosts (*host1* and *host2*), type the following on *host1*:

```
gm_allsize --slave --size=15
```

and on *host2* type:

```
gm_allsize --bidirectional --bandwidth --remote-host=host1 \  

```

```
--size=15 --geometric
```

This test performs a ping pong test: send one packet back and forth with only a single node sending at a time, which turns out to be half of the unidirectional bandwidth for a two-node test.

The output from this command will consist of two columns of data: the first column lists the message size (in bytes) and the second column lists the bandwidth (in MB/s).

- **Summed Bidirectional Bandwidth**

To test the **summed bidirectional bandwidth** between two hosts (*host1* and *host2*), type the following on **host1**:

```
gm_allsize --slave --size=15
```

and on *host2* type:

```
gm_allsize --both-ways -bandwidth --remote-host=host1 \  
--size=15 -geometric
```

This test has GM streaming packets in both directions (both nodes are always sending) and it causes GM to report the sum of the send and receive bandwidths.

The output from this command will consist of two columns of data: the first column lists the message size (in bytes) and the second column lists the bandwidth (in MB/s).

Performance graphs for GM on Solaris are available at

<http://www.myri.com/scs/solaris/>.

Refer to the section entitled "GM Performance" in the <GM_source_path>/README in the source release for complete details on expected GM performance.

e. Run **gm_stress** to test the network

gm_stress.c is an all-to-all test program for GM. It is available in GM 1.5.1 and higher in the <GM_install_path>/bin/ subdirectory. The most up-to-date version, which may be newer than in any GM release, is located on the Myrinet FAQ.

(<http://www.myri.com/scs/faq/faq-troubleshooting.html#debug-stress>).

Run **gm_stress** on every host in a cluster to validate GM. If you decide to use run-time options, they must be the same on all hosts. You can use **rsh** to launch **gm_stress** on remote nodes. When **gm_stress** terminates on the local node, it has terminated on all the other nodes too, even if **rsh** doesn't clean up nicely on them. If the test failed on any node, then **gm_stress** on the local node will report an error. You can also send datagrams with this test, but they don't count towards terminating the

program. Thus, if you send 100% datagrams **gm_stress** will run forever. By default **gm_stress** runs all-to-all on the entire cluster. Use the **-f** run-time option to test on a subset.

The options to **gm_stress** can be obtained by invoking the following:

```
xxx@xxx:<GM_install_path>/bin$ ./gm_stress -h
options:
-u, --unit=NUM           [0],      myrinet board number.
-p, --port=NUM           [2],      GM port number.
-n, --num-sends=NUM      [1024],  sends per host.
-m, --max-size=NUM       [19],      max send "size" (in log2 units).
-f, --file=FILENAME      [none],    file of hostnames, one per line
-c, --checksum           [off],     compute checksums.
-r, --min-receives=NUM   [4],      minimum number of receive buffers
per size.
-b, --barrier-seconds=NUM [300],   timeout for barrier.
-x, --max-resends=NUM     [30],     max resends per host destination
for send timeouts.
-d, --datagram=NUM        [0],      percent datagrams.
```

The format of the file specified by FILENAME is a list of hostnames (one per line) as reported by the routes in **gm_board_info**.

Example Usage on a subset of nodes in a cluster

You can use an **rsh** script like the following to launch it on the cluster.

```
xxx% cat /home/gm/tests/rsh_stress
#!/bin/bash

for i in `cat $1`
do
rsh $i $2 $3 $4 $5 $6 $7 $8 $9 ${10} ${11} ${12} ${13} ${14} ${15}
${16} ${17} &
done

xxx% cat /home/gm/tests/hosts
racksaver1
racksaver2
racksaver3
racksaver4
racksaver5
racksaver6
racksaver7

racksaver1% /home/gm/tests/rsh_stress hosts \
            <GM_install_path>/bin/gm_stress -f /home/gm/tests/hosts
racksaver1%
7 hosts.
7 hosts.
7 hosts.
7 hosts.
7 hosts.
7 hosts.
```

7 hosts.

```
sent / received 1024 messages per host
in lengths up to 524240 bytes.
at a rate of 93 / 92 MB/s
sent / received 1024 messages per host
in lengths up to 524240 bytes.
at a rate of 93 / 90 MB/s
sent / received 1024 messages per host
in lengths up to 524240 bytes.
at a rate of 93 / 93 MB/s
sent / received 1024 messages per host
in lengths up to 524240 bytes.
at a rate of 88 / 89 MB/s
sent / received 1024 messages per host
in lengths up to 524240 bytes.
at a rate of 88 / 91 MB/s
sent / received 1024 messages per host
in lengths up to 524240 bytes.
at a rate of 88 / 86 MB/s
sent / received 1024 messages per host
in lengths up to 524240 bytes.
at a rate of 93 / 94 MB/s
```

How does gm_stress work?

Every host sends a given number of random sized packets out of a big chunk of **gm_dma_malloc** memory to each host randomly and terminates when there is an error or when it has received the given number of packets from each host, (including itself). It uses **gm_zone.c**, which is part of **libgm**, for **malloc** and **free** from the send heap. If any host terminates without error then it means the test has succeeded globally. Otherwise the host that detected the error will terminate first, and then the others after sends fail. The test prints out a host count before the test starts. If it doesn't print anything out, then the test is hanging on the initial barrier, which indicates a user error: for example, the command was not started on all the hosts or GM is not installed properly on a host.

The **gm_stress** test sends message to itself and will work in loopback but not point-to-point topology, where there is no route-to-self.

The datagrams option adds some percentage of uncounted datagrams to the stream. 100 percent is legal but will never terminate.

What happens when gm_stress fails?

The **gm_stress** program is meant to *stress* all of the connections in the Myrinet network. There are 3 ways in which **gm_stress** can fail.

1. **connectivity problem.** Eventually a send will fail, usually with the "destination unreachable" error. On the host where the send failed, **gm_stress** will terminate. Subsequent sends from the other hosts to this first host will fail. On these hosts, **gm_stress** will terminate too, with "remote port closed" errors. Eventually, **gm_stress** will terminate globally in this

manner. It will not happen immediately. If you are impatient, you can kill the lingering `gm_stress` tests. If you do this, however, you may also need to wait for the GM long timeout (around 30 seconds) before those ports will be usable again, because GM may still be retransmitting some packets. The error messages will be a combination of send/recv errors.

2. **User error.** For instance, inconsistent host files as inputs, not running `gm_stress` on all hosts specified in the host file, or using mismatched size parameters. These may result in assertion failures on one host, followed by the kind of distributed shutdown described above. Or the test may seem to hang. You may have to kill the `gm_stress` tests by hand and wait for the 30-second GM long timeout for ports to close.
3. **Congestion.** If `gm_stress` fails with the "exceeded --max-resends" error on any node, this is just a problem with the test program, and doesn't indicate anything wrong with the cluster. Due to congestion at the receiver, send timeouts can occur. You will need to tune the parameters a bit. Try incrementally increasing `--min-receives` and decreasing `--max-size`, starting from their defaults (4 and 19). Or try increasing `--max-resends` to something huge, like 300.

You have now successfully installed GM.

III. Install and Configure Myrinet PM for Sun HPC ClusterTools 4.0

After GM is successfully installed, you can install Myrinet PM for Sun ClusterTools 4.0 in the following 8 steps:

1. Install Sun HPC Clustertools 4.0
2. Install the Myrinet PM libraries
3. Setup GM hostnames
4. Create GM configure file
5. Modify ClusterTools configure file `hpc.conf`
6. Start the daemon
7. Initialize the "all" partition (if necessary)
8. Rerun the daemon (if necessary)

1. Install Sun HPC ClusterTools 4.0

Select an installation directory path `<CT_install_path>` and install Sun HPC ClusterTools 4.0 (default at `/opt/SUNWhpc`).

2. Install the Myrinet PM libraries

Download the current Myrinet PM release

```
ftp://comrade@ftp.myri.com/pub/GM/clustertools/CT-4.0-Myrinet-PM-0.5-no-
mt.tar.gz
```

and utility function *ip2hostname*

```
ftp:// comrade@ftp.myri.com/pub/GM/clustertools/util/ip2hostname
```

Solaris users are encouraged to consult

<http://www.myri.com/scs/solaris.html>

for the newest release of Myrinet PM.

```
gunzip -c CT-4.0-Myrinet-PM-0.5-no-mt.tar.gz |tar xvf -
su root
cp myr*.so* <CT_install_path>/lib
cp sparcv9/myr*.so* <CT_install_path>/lib/sparcv9
```

3. Set up GM hostnames

For each node of the cluster, run *ip2hostname* to obtain the hostname corresponding to the IP address of myri0 on that node

```
ip2hostname x.x.x.x (myri0's IP address)
```

and set the node's GM name to this hostname.

```
su root
cd <GM_install_path>
./bin/gm_setname -h hostname
```

Rerun the GM mapper to globally propagate the change to all nodes. Note that the GM mapper should be rerun whenever a new node is added to the network, the network topology has changed, or a node's GM name is changed.

```
su root
./sbin/mapper ./etc/gm/map_once.args
```

4. Edit the GM configuration file

The configuration file usually locates at <CT_install_path>/conf/gm.conf. An example file for a cluster consisting of two nodes is given as follows:

```
2
# node_name board_num port_num port_ids
u81-t 1 12 4 5 6 7 8 9 10 11 12 13 14 15
u82-t 1 12 4 5 6 7 8 9 11 10 12 13 14 15
```

In this file, the first line specifies the total number of nodes in the cluster. Starting

from the second line, the first column specifies the node's GM name; the second column specifies the number of Myrinet interfaces on that node; the third column specifies the number of GM ports available for CT; starting from the fourth column are the GM port ids available for CT.

Set the environment parameter `MPI_MYR_CONF` to this file:

```
setenv MPI_MYR_CONF <CT_install_path>/conf/gm.conf
```

When no `MPI_MYR_CONF` is set, file `</opt/SUNWhpc/conf/gm.conf>` is used as default.

5. Edit the hpc configuration file

`<CT_install_path>/conf/hpc.conf` file is a configuration file used by ClusterTools to select runtime environment variables. It sets the rank for different protocol modules (PM) and the rank for each kind of network interfaces when multiple network interfaces are available for the same protocol module, such as TCP PM. A smaller rank integer corresponds to a higher priority. The following text is a sample `<CT_install_path>/conf/hpc.conf` file with Myrinet PM set to a higher priority (rank 30) than TCP PM but lower priority than SHM PM. The Myrinet interface can also be used as one of the TCP PM interfaces since it can emulate Ethernet. Thus, "myri" is set with a higher priority (rank 161) than the "qfe", "hme" and "le" interfaces.

```
.....

# List the available Protocol Modules
# PMODULE LIBRARY
Begin PMODULES
shm      ()
rsm      ()
myr      ()
tcp      ()
End PMODULES

# SHM settings
# NAME  RANK
Begin PM=shm
shm     5
End PM

# RSM settings
# NAME  RANK  AVAIL
Begin PM=rsm
wrsm    20    1
End PM

# MYR settings
# NAME  RANK
Begin PM=myr
myr     30
End PM
```

```

.....
# TCP settings
# NAME RANK MTU STRIPE LATENCY BANDWIDTH
Begin PM=tcp
midn 0 16384 0 20 150
idn 10 16384 0 20 150
.....
myri 161 4096 0 20 150
mqfe 163 4096 0 20 150
qfe 167 4096 0 20 150
mhme 170 4096 0 20 150
hme 180 4096 0 20 150
meri 183 4096 0 20 150
eri 187 4096 0 20 150
mle 190 4096 0 20 150
le 200 4096 0 20 150
msmc 210 4096 0 20 150
smc 220 4096 0 20 150
lo 230 4096 0 20 150
End PM

```

6. Start the daemon (requires root privileges)

On the master host type:

```

su root
/etc/init.d/sunhpc.cre_master start
/etc/init.d/sunhpc.cre_node start

```

On each of the non-master hosts type:

```

su root
/etc/init.d/sunhpc.cre_node start

```

7. Initialize the "all" partition (if necessary)

Initialize the "all" partition (requires root privileges) when "mpinfo -N" does not have all nodes in "all" partition.

```

su root
<CT_install_path>/etc/part_initialize

```

Example: output of `mpinfo -N` that shows not all nodes are in "all" partition.

```

NAME UP PARTITION OS OSREL NCPU FMEM FSWP LOAD1 LOAD5 LOAD15
u81 y all SunOS 5.8 4 3901 3910 0.02 0.04 0.02
u82 y - SunOS 5.8 4 3873 3873 3.05 2.48 1.42

```

8. Rerun the daemon (if necessary)

If file `<CT_install_path>/conf/hpc.conf` is modified (eg. changes in the rank, interface being added/removed, PM being added/removed, etc.), or the status of any listed interface being changed (eg. interface up/down, etc.), the daemon should be rerun.

Stop the daemon (requires root privileges).

On the non-master nodes,

```
su root
/etc/init.d/sunhpc.cre_node stop
```

On the master node,

```
su root
/etc/init.d/sunhpc.cre_node stop
/etc/init.d/sunhpc.cre_master stop
```

Restart the daemon.

Repeat step 6.

Prior to running an MPI program, permissions must be set properly on the hosts.

If **rsh** is used, a customer is required to have his **.rhosts** file set appropriately for all of the hosts in the Myrinet network. If this has not been done, the customer will see *'Permission Denied'* messages at run-time. If **ssh** is used, it must be configured to not prompt the user for a password. Otherwise, the customer will also see a *'Connection Refused'* or *'Permission Denied'* message.

You have now successfully installed Myrinet PM for ClusterTools 4.0.

9. Run-Time Tuning Options

A number of run-time tuning options can be supplied to Myrinet PM using environment parameters.

- **MPI_MYR_CONF**
Configuration file for Myrinet boards and ports available for CT on every host, default `/opt/SUNWhpc/conf/gm.conf`.
- **MPI_MYR_RENDVSIZE**
The eager/rendezvous thresholds, default (16K-96) bytes for 32-bit applications and (16K-112) for 64-bit applications respectively.
- **MPI_MYR_MULTIPHASE_RENDVSIZE**
The rendezvous/multiphase-rendezvous thresholds, default (64K-8) bytes.

IV. Test Myrinet PM Using the Intel Testing Suite

The source release of Sun HPC ClusterTools 4.0 includes a SCSL MPI conformance test suite. This is a modified version of the Intel MPI v1.1 Validation Suite. The original version of the test suite and its documentation can be obtained from <http://www.netlib.org/mpi/mpi-1.1-test/>.

The user may run the tests through TET environment by following the instruction in <SCSL_ROOT>/README.mpi.test -- "MPI Test Suite User Guide".

The user may choose to run the tests directly without setting TET environment using following script. This script is also available at comrade@ftp.myri.com/pub/GM/clustertools/SCSL_MPITEST.srp.

```
#!/bin/sh

if test x"$SCSL_ROOT" = x
then
    echo "Could not find \$SCSL_ROOT. Set \$SCSL_ROOT before running
this script."
    exit 1
fi

# Report testing environment
echo "% mpinfo -N"
mpinfo -N
echo "% mprun -np 2 -Ns uname -a"
mprun -np 2 -Ns uname -a
echo "% mprun -np 2 -Ns isalist"
mprun -np 2 -Ns isalist

cd $SCSL_ROOT/mpi/MPITEST/Test/c

dirs="blocking collective datatype env_manager grp_ctxt_comm misc \
      nonblocking persist_request probe_cancel sendrecv topo"

#===== build executable =====#

echo "Build intel MPI test suite\n"
for dir in $dirs
do
    cd $SCSL_ROOT/mpi/MPITEST/Test/c/$dir/functional
    files=`ls |grep MPI_ ; ls |grep async; ls |grep rings`

    for file in $files
    do
        cd $SCSL_ROOT/mpi/MPITEST/Test/c/$dir/functional/$file
        make execs MPITEST_TESTNAME=$file
    done
done

done

#===== run tests with 2, 4 and 8 processes =====#
```

```

echo "Run all but topo tests with 2, 4, 8 processes"
echo "Run topo tests with 4, 8 and 24 processes"
for dir in $dirs
do
    cd $SCSL_ROOT/mpi/MPITEST/Test/c/$dir/functional
    files=`ls | grep MPI_; ls | grep async; ls |grep rings`

    for file in $files
    do

# run MPI tests with 2 processes
        if [ "$dir" != "topo" ]; then
            echo "\nmprun -np 2 -Ns $dir/$file/node.execs_bx"
            mprun -np 2 -Ns $file/node.execs_bx
            fi

# run MPI tests with 4 processes
            echo "\nmprun -np 4 -W $dir/$file/node.execs_bx"
            mprun -np 4 -W $file/node.execs_bx

# run MPI tests with 8 processes
            echo "\nmprun -np 8 -W $dir/$file/node.execs_bx"
            mprun -np 8 -W $file/node.execs_bx

# run MPI tests with 24 processes
            if [ "$dir" = "topo" ]; then
                echo "\nmprun -np 24 -W $dir/$file/node.execs_bx"
                mprun -np 24 -W $file/node.execs_bx
                fi

        done
    done
done

```

A total of 11 groups of tests will be performed:

- a) MPI blocking tests
- b) MPI collective tests
- c) MPI datatype tests
- d) MPI env_manager tests
- e) MPI grp_ctxt_comm. tests
- f) MPI misc tests
- g) MPI nonblocking tests
- h) MPI persist_request tests
- i) MPI probe_cancel tests
- j) MPI sendrecv tests
- k) MPI topo tests

When running these tests, we assume a 2-node cluster, each node with one Myrinet interface, loaded with the 16-port GM driver. MPI jobs are run with 2, 4 and 8 processes. MPI “*topo*” tests are run with 4, 8 and 24 processes.

V. Test Myrinet PM Using the MPICH Testing Suite

1. Download MPICH-GM

Download the current MPICH-GM source from <http://www.myri.com/scs/index.html>.

```
gunzip -c mpich-1.2.5..9.tar.gz | tar xvf -
```

We will refer to the source directory of MPICH-GM as <MPICH-GM>.

```
cd <MPICH-GM>
./configure
```

Sample test programs are located under the directory <MPICH-GM>/examples. We are going to test Myrinet PM using the basic **mpi** test program, the point-to-point test programs and the performance test program -- **mpptest**.

2. Testing the basic function mpi

Build mpi test program:

```
cd <MPICH-GM>/examples/basic
tmcc mpi.c -lmpi -lm -o mpi
```

Run the test:

```
mprun -np 2 -Ns mpi
```

3. Testing the point-to-point programs

Build point-to-point test programs:

```
cd <MPICH-GM>/examples/test/pt2pt
```

Modify the Makefile for the following entries:

```
CC      = tmcc
CLINKER = tmcc
CFLAGS  =
LIBS    = -lmpi
```

Then, type

```
make
```

Use the following script to run the point-to-point MPI tests, using 2, 4, 8 and 24 processes respectively.

```
#!/bin/sh
```

```

files_2="bSENDtest    cancelibm      dtyperecv \
flood2             htmsg         irectvtest \
isndrcv            issend2       issendinit \
issendtest         longmsgs     overtake \
pack               probe         probel \
reqfree            selfvsworld  sendrecv \
ssendtest          testtest1    typetest"

files_all="cancel     cancel2       cancel3 \
cancelmessages     commit       dataalign \
dtypelife          exittest     fifth \
flood              fourth       getelm \
hindexed           hvec         hvectest \
hvectest2          irsend       irsendinit \
isendtest          nblock      nbtest \
nullproc           nullproc2    order \
persist            persist2     persistent \
relrank            reqcreate    sendmany \
sendorder          sendrecv2    sendrecv3 \
sendrecv4          sixth        sndrcv \
sndrcvrep          sndrcvrpl2  ssendtest2 \
testall            testsome     third \
trunc              truncmult    typebase \
typecreate         typeub       typeub2 \
typeub3            waitall     waitall2 \
waitany"

# Run 2-process tests
for file in $files_2 $files_all
do
    echo "\nmprun -np 2 -Ns $file"
    mprun -np 2 -Ns $file
done

# Run tests with 4, 8 and 24 processes
for proc in 4 8 24
do
    for file in $files_all
    do
        echo "\nmprun -np $proc -W $file"
        mprun -np $proc -W $file
    done
done
done

```

4. Testing the performance test program -- mpptest

Build the performance test program:

```
cd <MPICH-GM>/examples/perftest
```

Modify the Makefile for the following entries:

```
CC      = tmcc
LIBS    = -lmpi
```

Then type:

```
make mpptest
```

Use the following script to run the performance test between two nodes for message length from 0 to 4194304 bytes. The results will report latency (in microseconds) and bandwidth (in MB/s).

```
#!/bin/sh

mprun -np 2 -Ns mpptest -mbytes -cachesize 525000 -async -auto

mprun -np 2 -Ns mpptest -mbytes -cachesize 525000 \
      -size 1024 16384 128 -async

mprun -np 2 -Ns mpptest -mbytes -cachesize 525000 \
      -size 16384 65536 1024

mprun -np 2 -Ns mpptest -mbytes -cachesize 525000 \
      -size 65536 1048576 16384 -async -reps 20

mprun -np 2 -Ns mpptest -mbytes -cachesize 2100000 \
      -size 1048576 4194304 524288 -async -reps 20
```

VI. Test Myrinet PM Using the PALLAS Benchmark Suite

Download the Pallas Benchmark Suite from <http://www.pallas.com/e/products/pmb>

```
gunzip -c PMB2.2.tar.gz | tar xvf -
cd SRC
```

Set environment parameters appropriately as suggested in the file **make_solaris**

```
setenv MPI_HOME /opt/SUNWhpc
setenv MPI_INCLUDE $(MPI_HOME)/include
setenv LIBS -lmpi -lm -lnsl -lsocket
setenv LIB_PATH -L$(MPI_HOME)/lib
setenv CC cc
setenv CLINKER cc
make
```

Run the PMB-MPI1 program between 2 nodes:

```
mprun -np 2 -Ns PMB-MPI1
```

The results will report latency (in microseconds) and bandwidth (in MB/s).

If you encounter any problems during these Myrinet PM validation procedures, please collect all the relevant information and contact help@myri.com.