Compaq MPI

User Guide

May 2001

This guide describes how to install and use the *Compaq Message Passing Interface* (MPI) software; how to perform profiling and tracing; and how to solve any problems that arise when running *Compaq MPI*.

Revision/Update Information:	This version supersedes the manual issued in October 2000 for *Compaq MPI* Version 1.95
Operating System and Versions:	*Compaq Tru64 UNIX* Version 4.0F or higher
Software Version:	*Compaq MPI* Version 1.96

Compaq Computer Corporation
Houston, Texas

May 2001

Argonne National Laboratory Group
W. Gropp: (630) 252-4318; FAX: (630) 252-7852; e-mail: gropp@mcs.anl.gov
E. Lusk: (630) 252-5986; FAX: (630) 252-7852; e-mail: lusk@mcs.anl.gov
Mathematics and Computer Science Division, Argonne National Laboratory, Argonne IL 60439

Mississippi State Group
N. Doss and A. Skjellum:(601) 325-8435; FAX: (601) 325-8997; e-mail: tony@erc.msstate.edu
Mississippi State University, Computer Science Department & NSF Engineering Research Center for Computational Field Simulation, P.O. Box 6176, Mississippi State MS 39762

Contents

4 Profiling and Tracing

5 Troubleshooting

Preface

Purpose of this Guide

This guide contains the information you need to install and use *Compaq® MPI* Version 1.96, and to perform profiling, tracing, and troubleshooting.

Intended Audience

This guide is for users who are familiar with MPI.

Structure of this Guide

This guide is structured as follows:

- Chapter 1 provides a brief introduction to *Compaq MPI*, the requirements for installing *Compaq MPI*, and the installation instructions.
- Chapter 2 explains how to compile and link a program to run with *Compaq MPI*.
- Chapter 3 explains how to run a *Compaq MPI* program, and details the *Compaq MPI* environment variables and process file format.
- Chapter 4 explains how to produce and view timing information for the *Compaq MPI* calls (profiling); and how to print a trace line for every *Compaq MPI* call that is made (tracing).
- Chapter 5 provides information on problem solving, and on problems that may occur when *Compaq MPI* is terminating. It also provides addresses to contact if you have a problem, or wish to make suggestions regarding the software or the manual.

For More Information

In addition to this guide, the *Compaq MPI* documentation set includes the *Compaq MPI* Run Options reference (`man`) page, `dmpirun`.

Documentation Conventions

This guide uses the following documentation conventions:

Convention	Description
%	A percent sign represents the C shell system prompt.
$	A dollar sign represents the system prompt for the Bourne and Korn shells.
#	A number sign represents the superuser prompt.
host-file-name	Italic (slanted) type indicates arguments, variable values, Compaq product names, and complete titles of documents.
`-ump_bufs`	Monospace type indicates options, library names, and procedures.
`mpirun`	Boldface monospace type indicates commands.
MPID_TINY	Uppercase type indicates environment variables.
Underlined type	Underlined type emphasizes important information.
[|] {|}	In syntax definitions, brackets indicate items that are optional and braces indicate items that are required. Vertical bars separating items inside brackets or braces indicate that you choose one item from among those listed.
...	In syntax definitions, a horizontal ellipsis indicates that the preceding item can be repeated one or more times.
MPI_Testany	MPI_ indicates functions.
MB1, MB2, MB3	On a three-button mouse, MB1 indicates the left mouse button, MB2 indicates the middle mouse button, and MB3 indicates the right mouse button.

1

Getting Started with Compaq MPI

This chapter describes *Compaq MPI*, and how to install it. The chapter consists of the following sections:

- Introducing Compaq MPI
- Installing Compaq MPI

1.1 Introducing Compaq MPI

Compaq MPI is a proprietary implementation of MPI software for Alpha™ systems running the *Compaq Tru64™ UNIX* (formerly DIGITAL® UNIX) operating system, and is supported on both standalone SMP systems and *Memory Channel* clusters. *Compaq MPI* is derived from MPICH Version 1.1.1 from Argonne National Laboratories. *Compaq MPI* is optimized to give you low latency, high bandwidth message passing in SMP, and SMP cluster, environments.

The goal of MPI software is to provide a practical, portable, efficient, and flexible standard for message passing programs.

1.2 Installing Compaq MPI

This section describes the following tasks:

- Checking Hardware Requirements (see Section 1.2.1 on page 1–2)
- Checking Software Requirements (see Section 1.2.2 on page 1–2)
- Installing the Kit (see Section 1.2.3 on page 1–2)
- Initializing Machines for Memory Channel (see Section 1.2.4 on page 1–3)
- Setting Shared Memory and Memory Channel Parameters (see Section 1.2.5 on page 1–4)

1.2 Installing Compaq MPI

1.2.1 Checking Hardware Requirements

You need the following hardware to run *Compaq MPI*:

- An *AlphaServer*® system
- *Memory Channel*, if you plan to run MPI applications using more than one *AlphaServer*

  Note: *Compaq MPI* does not support any interconnect except *Memory Channel*.

1.2.2 Checking Software Requirements

You need the following software to run *Compaq MPI*:

- *Tru64 UNIX* Version 4.0F or later
- Licenses and binaries to use the user-level *Memory Channel* application programming interface (API) for the version of the *TruCluster*® product installed on your machine
- X11 Tool Kit Version 4.0 or later to run the *Compaq MPI* version of Upshot

1.2.3 Installing the Kit

To install *Compaq MPI*, follow these steps:

1. Read any release notes that come with the kit.
2. Log in as the root user to the machine running the *Tru64 UNIX* operating system.
3. To install the kit, enter the following commands:

   a. Change to the directory containing the kit, as follows:
   ```
   # cd <directory_containing_the_kit>
   ```

   b. Load the MPI software, as follows:
   ```
   # setld -l MPIBINXX
   ```

   The *Compaq MPI* installation comprises one subset, MPIBIN1*XX*, which consists of the libraries, `include` files, reference (`man`) pages, scripts, and examples.
4. Install the kit on all the systems in the cluster if you are using *Memory Channel*.
5. To verify that *Compaq MPI* is installed correctly, enter the following command:
   ```
   # setld -v MPIBIN1XX
   ```
6. If you intend to use Upshot, modify the first line of the Upshot file to refer to the correct path name for the wish interpreter on your system.

1.2.4 Initializing Machines for Memory Channel

If you are using *Memory Channel*, you must check that each machine in the cluster is initialized for user-level access.

To check, and initialize if necessary, follow these steps:

1. To search for the process `imc_mapper` on each machine, enter this command:
   ```
   # ps ax | grep imc_mapper | grep -v grep
   ```
   The output may be similar to the following:
   ```
   657 ttyp2 U 0:00.01 /usr/sbin/imc_mapper
   ```
2. If the process described in step 1 does not exist on a machine that you intend to use as part of the cluster, enter the following command on the machine:
   ```
   # /usr/sbin/imc_init
   ```
   Once you enter this command, it is in effect until the machine is shut down. (It does not cause a problem if you run this command again while a previously-entered `imc_init` command is still in effect. See the `imc_init` reference (`man`) pages for details.)
3. Ensure that the `/etc/sysconfigtab` file contains the following entry:
   ```
   rm:
       rm-no-inheritance = 1
   ```
   This prevents *Memory Channel* resources from being inherited by a forked process.

   Note:

   If using *Tru64 UNIX* Version 5.1 or higher, this step is not necessary.

   ---
4. If you are using multiple *Memory Channel* rails, the default setting is to use them as a failover pair. While this provides maximum availability, you can increase performance by configuring your cluster so that it does not use the rails as a failover pair. This allows applications to access the aggregate address space of all logical rails, and use aggregate bandwidth.

   To configure non-failover use of the multiple *Memory Channel* rails, change the `/etc/sysconfigtab` file by adding or modifying the following entry:
   ```
   rm:
       rm_rail_style = 0
   ```
   The default value for `rm_rail_style` is 1; that is, configure for failover-pair operations.

1.2 Installing Compaq MPI

1.2.5 Setting Shared Memory and Memory Channel Parameters

Compaq MPI uses shared memory for communication within a host, and *Memory Channel* for communication between hosts. This is achieved by setting up "communication channels", each of which is either a shared memory segment or a group of *Memory Channel* pages.

The size of each communication channel is set by the user, using the `-ump_bufs` option (see Chapter 3, "Compaq MPI Command Line Options", page 3–2). The default value is 128K.

If running very large applications, it may be appropriate to increase the data and stack size using the `limit` command; for example,

```
# limit datasize unlimited
```

1.2.5.1 Changing Shared Memory Parameters

1. When using a Non-Uniform Memory Access machine such as the *AlphaServer* GS320, allocate shared memory using a first-touch algorithm to ensure that most of the shared memory used by MPI is local memory rather than remote memory. The default algorithm uses a striped scheme. Edit the `/etc/sysconfigtab` file and modify the `shm_allocate_striped` entry in the `ipc` stanza to disable striped allocation, as follows:

   ```
   ipc:
   shm_allocate_striped=0
   ```

2. To increase the maximum shared memory size, edit the `/etc/sysconfigtab` file and include or modify the following entry:

   ```
   ipc:
   shm_max=<size in bytes>
   ```

3. There are kernel limits on the number of shared memory segments a process can attach to and the total number of system-wide shared memory segments available.

   To modify the value of the `shm_seg` variable, edit the `/etc/sysconfigtab` file and include or modify the following entry:

   ```
   ipc:
   shm_seg=<number of segments attached per process>
   ```

4. The total number of system-wide shared memory segments allowed is controlled by the `shm_mni` variable.

   To change the system parameter, edit the `/etc/sysconfigtab` file and include or modify the following entry:

   ```
   ipc:
   shm_mni=<total number of system-wide segments>
   ```

5. If you have changed any of the parameters as described in 2, 3, or 4 above, reboot the system.

1.2.5.2 Changing Memory Channel Parameters

Note:

This section does not apply to *Tru64 UNIX* Version 5.0 or higher.

Communication across *Memory Channel* uses a similar scheme to shared memory. The maximum *Memory Channel* allocation allowed is set to 10MB by default. There is a limit of 128MB on *Memory Channel* allocation using *Memory Channel* Version 1.5, and a limit of 512MB on *Memory Channel* Version 2.0.

Before running an MPI program, you should increase the default values. Increasing the default values does not use up memory resources but allows MPI to access more *Memory Channel* space.

To increase from the default 10MB allocation on each host — for example, to 100MB — perform one of the following steps:

1. Log in as root.

   Enter the following command:

   ```
   # imc_init -a 100 -r 100
   ```

or

2. Edit the `/etc/rc.config` file and modify the IMC_MAXALLOC and IMC_MAXRECV entries as follows:

   ```
   IMC_MAXALLOC="100"
   IMC_MAXRECV="100"
   ```

   Compaq recommends that you increase these values to the maximum allowed for your configuration.

   Reboot the system.

1.2 Installing Compaq MPI

1.2.5.3 Changing Virtual Memory Resources

Each shared memory or *Memory Channel* communication channel uses up a virtual memory object. The number of virtual memory objects allowed is a kernel-defined parameter. This parameter must be at least twice as large as the number of communication channels between tasks.

To change this value, follow these steps:

1. Edit the `/etc/sysconfigtab` file and include or modify this entry:

```
vm:
    vm_mapentries=<number of VM objects allowed>
```

Note:

This step does not apply to *Tru64 UNIX* Version 5.0 or higher.

2. Reboot the system.

2

Compiling and Linking MPI Programs

This chapter describes how to compile and link MPI programs. The chapter consists of the following sections:

- Compiling and Linking C Programs
- Compiling and Linking Fortran Programs

2.1 Compiling and Linking C Programs

To compile MPI programs, you must use the appropriate `include` files and libraries in your `make` process.

To compile C programs, follow these steps:

1. Enter the following code into the program:
   `include <mpi.h>`

   This `include` file is installed in the standard location under `/usr/include` and does not need additional compiler directives.

2. Use the following library option:

   Include `-lmpi` in your link command, for example,
   `cc -o myprog myprog.c -lmpi -lrt -pthread`

2.2 Compiling and Linking Fortran Programs

To compile MPI programs, you must use the appropriate `include` files and libraries in your `make` process.

To compile Fortran programs, follow these steps:

1. Enter the following code into the program:
   `include 'mpif.h'`

   This `include` file is installed in the standard location under `/usr/include` and does not need additional compiler directives.

2.2 Compiling and Linking Fortran Programs

2. Use the following library option:

   Include `-lfmpi` and `-lmpi` in your link command, for example,
   `f77 -o myprog myprog.f -lfmpi -lmpi -lrt -pthread`

3

Running Compaq MPI Programs

This chapter describes how to run a *Compaq MPI* program. The chapter consists of the following sections:

- Executing Compaq MPI Programs
- Compaq MPI Command Line Options
- Compaq MPI Environment Variables
- Process File Format
- TotalView Support

3.1 Executing Compaq MPI Programs

To execute a *Compaq MPI* program, enter the command `dmpirun` and the name of the program you wish to execute; for example:

```
$ dmpirun myprog
```

You can execute a *Compaq MPI* program with a small set of options. Refer to `/usr/examples/mpi` for an example program.

For example, to run `myprog` using two processes on one host, enter the following:

```
$ dmpirun -np 2 myprog
```

To run the same program across two hosts — for example, `black` and `white` — follow these steps:

1. Create a file — for example, `myhosts` — with the host names separated by white space (including newline), as follows:
   ```
   black
   white
   ```
2. Add the host file option, `-hf`, to the command line:
   ```
   dmpirun -np 2 -hf myhosts myprog
   ```

3.2 Compaq MPI Command Line Options

You can specify options *via* command line options and arguments, or *via* environment variables. Command line options override the setting of environment variables. Table 3–1 in this section describes the available options and arguments. Table 3–2 in Section 3.3 describes the *Compaq MPI* environment variables.

Note that the `dmpirun` command and its helper programs use the UNIX® commands `wdir`, `hostname`, and `rsh`. It is assumed the commands are located in your UNIX path.

Table 3–1 Compaq MPI Command Line Options

Option	Argument	Description
`-wdir`	*directory*	Specifies the working directory for the application.
`-np`	*number-of-processes* (or tasks)	Default of one per host.
`-hf`	*host-file-name*	Specifies the names of the hosts on which to run the *Compaq MPI* program. The host file *host-file-name* should contain a list of host names separated by whitespace (including a newline). The host on which `dmpirun` is started must be included in the host file. If a `Permission denied` message appears when you execute `dmpirun`, ensure that you have permission to perform an `rsh` command on each host in the host file. Check that the `~/.rhosts` file is set up correctly. The last line of the host file must be terminated with a newline. Host names may now take the following form: `hostname:directory`. If `:directory` is specified, this value is used in preference to the DMPI_DIRECTORY value or the `-wdir` value. This allows processes on each host to run with a different working directory.
`-dbx`		Allows you to run the application processes inside the `dbx` debugger.
`-ladebug`		Allows you to run the application processes inside the `ladebug` debugger.
`-xdbx`		Allows you to run the application processes inside the `xdbx` debugger.
`-gdb`		Allows you to run the application processes inside the `gdb` debugger.

3.2 Compaq MPI Command Line Options

Table 3–1 Compaq MPI Command Line Options

Option	Argument	Description
`-xterm`		Allows you to run each of the application processes inside its own `xterm`.
`-cmddbx`	*dbx-command*	The supplied program will be used in place of the default `dbx` command when the `-dbx` option is used. The full pathname must be supplied.
`-cmdladebug`	*ladebug-command*	The supplied program will be used in place of the default `ladebug` command where the `-ladebug` option is used. The full pathname must be supplied.
`-cmdxdbx`	*xdbx-command*	The supplied program will be used in place of the default `xdbx` command when the `-xdbx` option is used. The full pathname must be supplied.
`-cmdgdb`	*gdb-command*	The supplied program will be used in place of the default `gdb` command when the `-gdb` option is used. The full pathname must be supplied.
`-cmdxterm`	*xterm-command*	The supplied program will be used in place of the default `xterm` command when the `-xterm` option is used. The full pathname must be supplied.
`-cmdrsh`	*rsh-command*	The `dmpirun` program invokes `/usr/bin/rsh` to create remote processes. Use the `-cmdrsh` option to select a different command to use. The full pathname must be supplied.
`-cmddmpirun`	*dmpi-command*	`dmpirun` invokes a copy of itself on each host. The default is `/usr/bin/dmpirun`. Use the `-cmddmpirun` option to select a different location for the command.
`-display`	*x-display*	The X display on which to use `xterm` or a debugger.
`-ump_key`	*ump-key*	The `ump_key` option is used to generate unique identifiers for *Memory Channel* regions used by applications. If you want to run more than one application, each application must have a unique key. When you execute `dmpirun`, a pseudo-random key is chosen by default unless the environment variable UMP_KEY is set. If you specify keys using the `ump_key` option, they must be at least 4000 keys apart.
`-ump_bufs`	*ump_buffer_size*	Determines the size of a channel buffer. Default of 128K or the environment variable UMP_BUFS.

3.2 Compaq MPI Command Line Options

Table 3–1 Compaq MPI Command Line Options

Option	Argument	Description

`-ump_multirail`	*string*	This option controls the assignment of UMP "channels" to the underlying *Memory Channel* physical rail structure. The default rail assignment method evenly distributes UMP "channels" over the available hardware rails. The argument string takes the form: `[report|avoid:N]` where `report` enables the printing of a channel distribution/usage report, and `avoid:N` will cause no channel to be assigned to rail N.
		
		Examples:
		`> setenv UMP_MULTIRAIL "report"`
		`> dmpirun ...`
		will use the default value `pressure` and generate a final usage report.
		
		`> setenv UMP_MULTIRAIL "avoid:1,report"`
		`> dmpirun ...`
		will not assign rail 1 and will also generate a final report.

`-ump_thread_mode`	*mode*	This option controls the use of additional data-moving threads within UMP. By default, an extra thread is not used. This can be modified by setting the option to one of the following: • `merging` selects an optimized multithread implementation. This may be used to some advantage when there are more CPUs available than MPI processes. • `single` (*) is the default value.
`-ump_yield`	*method*	This option is effective when using multiple threads, to control the method by which executing threads yield to a competing thread, as follows: • `kernel` informs the kernel level scheduler when a thread is willing to yield. • `user` informs the thread level scheduler when a thread is willing to yield. • `swap` will alternate between each of the above. • `none` will effectively disable a thread from ever yielding.
`-ump_timeout`	*timeout*	The value of the integer argument is used to determine the timeout value (in seconds) when draining UMP channels during an `ump_close()`. The default value is 300.

3.2 Compaq MPI Command Line Options

Table 3–1 Compaq MPI Command Line Options

Option	Argument	Description
`-ump_fragsize`	*fragsize*	By default, the internal buffer used to transfer messages between processes is divided into four fragments of size UMP_BUFS/4. Setting UMP_FRAGSIZE explicitly sets the size of fragments used, and implicitly sets the number of fragments: UMP_BUFS/UMP_FRAGSIZE.
`-ump_error_mode`	*mode*	This option specifies the level of error checking to be performed for *Memory Channel* transactions. Note that the higher the level of error checking set, the greater the performance degradation. The error-checking levels are as follows: • `none` performs no error checking. • `one` checks for errors once — at process exit — by comparing the error count at process exit with the error count at process startup. • `inter` checks for errors intermittently. This is the default setting. • `always` checks for errors on every transaction. • `recover` will resend data in the event of an error. By default, all *Memory Channel* errors are fatal, including those caused by nodes booting into or leaving a cluster. Errors are considered non-fatal, and cause a warning to be printed, when the string value of `ump_error_mode` includes `+warning`. For example: `-ump_error_mode always+warning` The default mode is `inter+fatal`.
`-mpid_shrt`	*mpid_shrt limit*	*Compaq MPI* messages are normally sent internally as a header packet followed by a data packet. However, for messages smaller than the `mpid_shrt` size, the data is sent as part of the header packet in two UMP[1] operations; this improves performance. The default size is the value specified by the MPID_SHRT environment variable, or 256 bytes if the environment variable is not specified. The short message limit must be a multiple of four, and must not be larger than the UMP buffer size. Some applications perform a large number of short non-blocking sends; the `short` protocol may cause such sends to block. If this happens, set the value of `mpid_shrt` to zero.

3.2 Compaq MPI Command Line Options

Table 3–1 Compaq MPI Command Line Options

Option	Argument	Description
`-mpid_tiny`	*mpid_tiny limit*	*Compaq MPI* messages are normally sent internally as a header packet followed by a data packet. However, for messages smaller than the `mpid_tiny` size, the data is sent as part of the header packet using a single UMP operation; this improves performance. The default size is the value specified by the MPID_TINY environment variable, or 64 bytes if the environment variable is not specified. The value specified must be a multiple of 4, less than or equal to 64, and greater than zero. In some circumstances the `tiny` protocol can lead to the UMP channel becoming clogged up, which causes non-blocking sends to block. If this happens, set the value of `mpid_tiny` to zero.
`-block`		Along with `–cyclic` and `–random`, `-block` is used to select the process layout when the `-hf` option is used. `-block` is the default. The processes are spread across the hosts with a block distribution. For example, with two hosts and four processes, Ranks 0 and 1 will be on the first host, with Ranks 2 and 3 on the second. The processes are spread evenly across the host, but no account is taken of the number of processors on each host.
`-cyclic`		Along with `–block` and `–random`, `-cyclic` is used to select the process layout when the `-hf` option is used. In a `–cyclic` distribution, the processes are located in round-robin fashion. For example, with two hosts and four processes, Ranks 0 and 2 will be on the first host, with Ranks 1 and 3 on the second. The processes are spread evenly across the host, but no account is taken of the number of processors on each host.
`-random`		Along with `–block` and `–cyclic`, `-random` is used to select the process layout when the `-hf` option is used. In a `–random` distribution, processes are placed at random on hosts. The processes are spread evenly across the host, but no account is taken of the number of processors on each host.
`-pf`	*file*	In place of a command line application specification, a file containing the required application processes can be provided. Section 3.4 describes the format of this file. The `-pf` option should not be used with the `-np` or `-hf` options, or a command line executable.

[1]UMP is the communication layer used by *Compaq MPI*. This release includes support for multiple rails.

3.3 Compaq MPI Environment Variables

All of the command line options for `dmpirun` (see Section 3.2, page 3–2) have environment variable counterparts. If you wish to use a non-default value that will not change frequently, set the environment variable to avoid using the command line option. For example, if you would prefer to use `ssh` instead of `rsh`, set the DMPI_CMDRSH environment variable, instead of specifying the `-cmdrsh` option every time.

Table 3–2 on page 3–7 lists all of the environment variables used by *Compaq MPI*. The first column is the variable name; the second is the default value if the variable is unset; and the third column is the corresponding command line option, which takes priority over the environment variable.

Table 3–2 Compaq MPI Environment Variables

Name	Default	Override Flag
DMPI_DIRECTORY	current working directory	`-wdir`
DMPI_HOSTFILE	none	`-hf` or `-machinefile`
DMPI_PROCFILE	none	`-pf`
DMPI_NP	`1`	`-np`
DMPI_UMPKEY	random	`-ump_key`
DMPI_UMPBUFS	`131072` (128K)	`-ump_bufs`
DMPI_MPIDSHRT	`256`	`-mpid_shrt`
DMPI_MPIDTINY	`64`	`-mpid_tiny`
DMPI_DEBUG	.	`-dbx`, `-ladebug`, `-xdbx`, `-xterm`, `-gdb`
DMPI_LAYOUT	`block`	`-block`, `-cyclic`, `-random`
DMPI_LAYOUT_SEED	`dmpirun` *process-id*	none
DMPI_SERVERPORT	`0`	none
DMPI_CMDXTERM	`/usr/bin/X11/xterm`	`-cmdxterm`
DMPI_CMDRSH	`/usr/bin/rsh`	`-cmdrsh`
DMPI_CMDDMPIRUN	`/usr/bin/dmpirun`	`-cmddmpirun`

3.3 Compaq MPI Environment Variables

Table 3–2 Compaq MPI Environment Variables

Name	Default	Override Flag
DMPI_DIRECTORY	current working directory	`-wdir`
DMPI_CMDDBX	`/usr/bin/dbx`	`-cmddbx`
DMPI_CMDXDBX	`/usr/local/bin/X11/xdbx`	`-cmdxdbx`
DMPI_CMDLADEBUG	`/usr/bin/ladebug`	`-cmdladebug`
DMPI_CMDGDB	`/usr/local/bin/gdb`	`-cmdgdb`
UMP_MULTIRAIL	`pressure`	`-ump_multirail`
UMP_THREAD_MODE	`single`	`-ump_thread_mode`
UMP_YIELD	`swap`	`-ump_yield`
UMP_TIMEOUT	`300`	`-ump_timeout`
UMP_FRAGSIZE	`32768` (32K)	`-ump_fragsize`
UMP_ERROR_MODE	`inter`	`-ump_error_mode`

Notes:

Variable	Comment
DMPI_DEBUG	The special value `.` means no debugger. Valid values for this option are thus `.`, `dbx`, `xdbx`, `ladebug`, `xterm`, and `gdb`. `xterm` is not strictly a debugger, but shares the same mechanism.
DMPI_LAYOUT	May take the values `block`, `cyclic`, or `random`.
DMPI_LAYOUT_SEED	Is used to seed the random number generator used by the random layout option. By default, the `dmpirun` *`process-id`* is used. To allow repeatability, a fixed seed may be supplied. There is no command line override for this option.
DMPI_SERVERPORT	`dmpirun` creates a socket which is then used by the host managers to communicate information about the application processes created. This environment variable allows the port number to be set. By default, the value 0 is used, This has special meaning: it requests a currently unused port from the operating system. This avoids collisions with any other well-written software which uses sockets. It has no command line override.

3.4 Process File Format

The new `-pf` option allows you to specify more complex application layouts. In particular, it allows you to use multiple executables within one application. This section describes the file that must be used with the `-pf` option.

The file consists of one or more lines of the following format:

```
number hostname[:path] executable [arguments]
```

[...] indicates optional features.

Each component has the following meaning:.

Name	Explanation
`number`	Takes one of two forms. Either a single number indicating the number of processes required, or a triple of the form lowest;highest;step. In the first form, the next available ranks will be used after all ranks used by the second form have been allocated. In the second, the ranks are specified explicitly as lowest, lowest+step, lowest+(2*step),...<= highest.
`hostname`	The host on which to create these processes.
`path`	Run these processes with `path` set as the working directory, in preference to the `-wdir` or DMPI_DIRECTORY options.
`executable`	The executable to be used for these processes.
`arguments`	The arguments to be used for these processes.

All explicit ranks will be assigned first, then any gaps will be filled with the remaining processes in the order in which they are declared in the process file. Overlapping ranks or missing ranks will result in an error being reported.

The following examples illustrate common cases:

Example 1: An uneven block distribution

```
3 black a.out -x 1
4 white a.out -x 1
2 brown a.out -x 1
3 green a.out -x 1
```

The result will be Ranks 0–2 on `black`, Ranks3–6 on `white`, Ranks 7–8 on `brown`, and Ranks 9–11 on `green`.

3.5 TotalView Support

Example 2: A non-SPMD (Single Process Multiple Data) application

```
1 black master
7 black slave
```

Rank 0 will be an instance of `master`, Ranks 1–7 will be instances of `slave`.

Example 3: A cyclic distribution

```
0;15;4 black a.out -y 0
1;15;4 white a.out -y 0
2;15;4 brown a.out -y 0
3;15;4 green a.out -y 0
```

Ranks 0,4,8,14 will be on `black`; Ranks 1,5,8,13 on `white`; Ranks 2,6,10,14 on `brown`; and Ranks 3,7,11,15 on `green`.

Example 4: Setting some ranks explicitly

```
0;0;1 black source
7;7;1 black sink
6 black worker
```

Rank 0 will be an instance of `source`, Rank 7 will be an instance of `sink`. The remainder, Ranks 1–6, will be instances of `worker`.

3.5 TotalView Support

This version of *Compaq MPI* is compatible with the Dolphin Interconnect Solutions' TotalView debugger, Version 3.7.7 and later. It is used with TotalView in the following way:

```
$ totalview dmpirun -a dmpirun_arguments
```

where *`dmpirun_arguments`* are the command line arguments you wish to give to `dmpirun`.

For example,

```
$ totalview dmpirun -a -np 2 a.out 0 1
```

The use of TotalView in combination with any other debug option, including `xterm`, is not permitted. See the TotalView documentation for usage details.

4
Profiling and Tracing

This chapter describes profiling and how to produce a log of calls to MPI routines and then view the log. It also describes tracing and how to trace MPI calls. The chapter consists of the following sections:

- Introducing Profiling
- Introducing Tracing

4.1 Introducing Profiling

Compaq MPI includes modified versions of the profiling library and Upshot viewer that come with the MPICH kit. You can use the profiling library to produce a log of calls to *Compaq MPI* routines. This records the start and end time of each call and other message information. You can then use Upshot to view, search, and scroll the log.

4.1.1 Differences between Compaq MPI and MPICH Profiling Libraries

The *Compaq MPI* profiling library differs from the MPICH profiling library in the following ways:

- Library code is modified to eliminate many of the unaligned access warnings on Alpha systems
- Time resolution is improved from approximately 1 millisecond to approximately 1 microsecond
- The *Compaq MPI* profiling library works with the MPI Fortran API
- Message sent/received events are generated for MPI_Waitall, MPI_Waitany, MPI_Waitsome, MPI_Testall, MPI_Testany and MPI_Testsome

4.1 Introducing Profiling

4.1.2 Differences between Compaq MPI and MPICH Upshot

The *Compaq MPI* version of Upshot differs from the MPICH version in the following ways:

- Faster loading of log files
- Additional zoom controls
- Additional search capability
- Additional popup window with event, state, and message details
- Additional links from non-blocking events to the wait, test, or cancel that terminates the request

4.1.3 Logging Calls to Compaq MPI Routines

To use the profiling libraries to log calls, follow these steps:

1. Compile your code in the normal way.
2. Link to the profiling libraries instead of the normal MPI library.

   For example, for C programs, enter:

   ```
   $ cc -o myprog.log myprog.c -llmpi -lpmpi -lmpi -lrt -pthread
   ```

   For example, for Fortran programs, enter:

   ```
   $ f77 -o myprog.log myprog.f -lfmpi -llmpi -lpmpi -lmpi -lrt
   -pthread
   ```

3. Run the program and it produces a log file called *executablename*`_profile.log`, where *executablename* is the name of the program.

4.1.4 Viewing the Log using Upshot

To use the *Compaq MPI* version of Upshot, follow these steps:

1. To start Upshot, enter the following command:

   ```
   $ upshot
   ```

   A window appears with an entry field, and Select Logfile, Setup, Options and Quit buttons.

2. Enter the full name of the log file or use the Select Logfile button to select the log file name from the file browser window.

   For example,
   `/usr/examples/mpi/ping_mpi_profile.log`

3. Click on the Setup button.

   A small window appears while the file is loading. When the file is loaded, the timeline window is displayed.

   The features of this window and the actions you can perform are described here:

   - There are a number of buttons at the top of the timeline window:

     a. Horizontal Zoom In/Out: Zooms in or out to display more or less detailed information. Instead of using the Horizontal Zoom In button, you can select the region you want to see by using MB3 on a timeline or on either of the two scales at the bottom of the window. When you click MB3 on a timeline, a yellow vertical line appears at that point. This is used as the zoom point for the zoom buttons. You will notice that a red line appears as you move the mouse. This indicates the area you wish to zoom in on.

     b. Vertical Zoom In/Out: Zooms in or out to display more or less detailed information.

     c. Detailed Zoom: allows you to define the time for the process you wish to view and the factor for displaying it.

     d. Reset: restores the window to the state it was in when the log was loaded.

     e. Print: prints the contents of the window to file.

     f. Find: opens a window where you can search for events. You can search for specific event types in specific timelines, or you can search for the next or previous event of any type as a handy scrolling mechanism.

     g. Close: closes the timeline window.

   - The timeline window shows one line per process (numbered vertically), with events and states on each line.

   - The routines used by the program are placed beneath the buttons at the top of the window and above the processes. Clicking on a routine causes a window to appear that displays a histogram of state durations for the routine.

- The black arrow indicates a message sent from the process at the tail of the arrow to the process at the head of the arrow. You can press MB1 when the cursor is over the arrow to display details about that message, for example, message size, source, destination ranks.
- The yellow arrows indicate the duration of non-blocking requests. Non-blocking MPI communications use ‘requests’ that are typically initiated by a non-blocking send or receive, such as MPI_Isend or MPI_Irecv, and are usually terminated by a wait or test call.

  Messages sent or received this way are logged as though they are sent or received by the terminating wait or test call. Yellow arrows on a timeline indicate the start and end of a request and go from the send or receive call that initiated the request to the test, wait, or cancel call that ends the request.
- You can scroll the timeline area using the scroll bars or by moving the mouse cursor over the timeline while holding down MB2.

4.2 Introducing Tracing

You can use the tracing library to send, to the standard output, information describing each of the MPI calls executed in the program.

The information consists of:

- A trace line containing the rank in MPI_COMM_WORLD of the calling process
- A trace line indicating the call has completed

Most send and receive routines indicate the value of count, tag, and partner (destination for send, source for receives).

4.2.1 Tracing MPI Calls

To use the tracing library to print trace lines for MPI calls, follow these steps:

1. Compile your code in the normal way.
2. Link to the tracing library instead of the normal MPI library. For example, for C programs, enter:

   ```
   $ cc -04 -o myprog.log myprog.c -ltmpi -lpmpi -lmpi -lrt -pthread
   ```

   For example, for Fortran programs, enter:

   ```
   $ f77 -o myprog.log myprog.f -lfmpi -ltmpi -lpmpi -lmpi -lrt
   -pthread
   ```
3. Run the program and the trace lines are produced.

5
Troubleshooting

This chapter describes a problem that might occur when you are running *Compaq MPI* and how to solve it; and potential problems that may occur when *Compaq MPI* is terminating. It also provides contact information in case you wish to send comments or suggestions. The chapter consists of the following sections:

- Problem Solving
- Potential Problems when Terminating
- Contact Information

5.1 Problem Solving

Follow these steps if this message is displayed when you are running `dmpirun`:

```
% dmpirun -np 2 cpi
Permission denied:
```

1. Ensure that the `.rhosts` file in your home directory is set up correctly.
2. To change the protection to user read/write only, enter the following command:
   `$ chmod og-rwx .rhosts`

   For security reasons, only the owner of the `.rhosts` file should have read/write access.
3. Add one line to the `.rhosts` file for each host that you want to use, as follows:
   host username

   For example, if your username is `doe` and you wish to use machines `a.our.org` and `b.our.org`, your `.rhosts` file should contain the following entries:

   ```
   a.our.org doe
   b.our.org doe
   ```

5.2 Potential Problems when Terminating

1. If your program does not terminate cleanly, IPC resources may not be freed. *Compaq MPI* will attempt to clean up IPC resources on all hosts if the program is aborted using Ctrl/C. However, if the *Compaq MPI* processes on a host exit normally, but without calling MPI_ABORT or MPI_FINALIZE, then *Compaq MPI* will not be aware of a problem and will not attempt to clean up.
2. When using some of the debuggers, `dmpirun` may write a temporary file which may not be deleted on termination.

   The command `mpiclean` can be used to tidy up after abnormal terminations. This tidies up IPC resources and temporary files on the local host only.

   Note:

   The command `mpiclean` frees all IPC resources currently in use by the user, including the resources used by other programs started by this user.

   The `mpiclean` command should not be run by the `root` user.

5.3 Contact Information

Please send your comments and suggestions regarding the software or the manual to `Compaq.MPI@compaq.com`

Index