QLOGIC

Simplify

InfiniPath User Guide

Version 2.0

Document Revision History	
Rev. 1.0, 8/20/2005	
Rev. 1.1, 11/15/05	
Rev. 1.2,02/15/06	
Rev. 1.3 Beta 1, 4/15/06	
Rev. 1.3, 6/15/06	
Rev. 2.0 Beta, 9/25/06, QLogic Rev IB6054601 A	
Rev. 2.0 Beta 2, 10/15/06, QLogic Rev IB6054601 B	
Rev. 2.0, 11/30/06, QLogic Rev IB6054601 C	
Changes	Document Sections Affected
Updated Preface and Overview by combining into single section, now called Introduction. Same as introduction in Install Guide.	section 1
Added SLES 9 as new supported distribution	section 1.7
Revised info about MTRR mapping in BIOS. Some BIOSes don't have it, or call it something else.	section C.1, section C.2.1, section C.2.2, section C.2.3
Corrected usage of ipath_core, replacing with ib_ipath	section 2
Added more options to mpirun man page description	section 3.5.10
Added new section on Environment for Multiple Versions of InfiniPath or MPI	section 3.5.8.1
Added info on support for multiple MPIs	section 3.6
Added info about using MPI over uDAPL. Need to load modules rdma_cm and rdma_ucm.	section 3.7
Added section: Error messages generated by mpirun. This explains more about the types of errors found in the sub-sections. Also added error messages related to failed connections between nodes	appendix C.8.12

Added mpirun error message about stray processes to error message section	appendix C.8.12.2
Added driver and link error messages reported by MPI programs	appendix C.8.12.3
Added section about errors occuring when different runtime/compile time MPI versions are used	appendix C.8.7
2.0 mpirun incompatible with 1.3 libraries	appendix C.8.1
Added glossary entry for MTRR	Glossary
Added new index entries for MPI error messages format, corrected index formatting	Index

First Published: August 2005
Printed in U.S.A.

Table of Contents

Figures

Tables

Section 1

Introduction

This chapter describes the objectives, intended audience, and organization of the *InfiniPath User Guide.*

The *InfiniPath User Guide* is intended to give the end users of an InifiniPath cluster what they need to know to use it. In this case, end users are understood to include both the cluster administrator and the MPI application programmers, who have different but overlapping interests in the details of the technology.

For specific instructions about installing the InfiniPath QLE7140 PCI Express™ adapter, the QMI7140 adapter, or the QHT7140 /QHT7040 HTX™ adapters, and the initial installation of the InifiniPath Software, see the *InfiniPath Install Guide*.

1.1 Who Should Read this Guide

This guide is intended both for readers responsible for administration of an InfiniPath cluster network and for readers wanting to use that cluster.

This guide assumes that all readers are familiar with cluster computing, that the cluster administrator reader is familiar with Linux administration and that the application programmer reader is familiar with MPI.

1.2 How this Guide is Organized

The *InfiniPath User Guide* is organized into these sections:

- Section 1"Introduction". This section.
- Section 2 “InfiniPath Cluster Administration” describes the lower levels of the supplied InfiniPath software. This would be of interest mainly to an InfiniPath cluster administrator.
- Section 3 “Using InfiniPath MPI” helps the MPI programmer make best use of the InfiniPath MPI implementation.
- Appendix A “Benchmark Programs”
- Appendix B “Integration with a Batch Queuing System”
- Appendix C “Troubleshooting”. The Troubleshooting section provides information for troubleshooting installation, cluster administration, and MPI.
- Appendix D “Recommended Reading”

- Appendix E Glossary of technical terms
- Index

In addition, the *InfiniPath Install Guide* contains information on InfiniPath hardware and software installation.

1.3 Overview

The material in this documentation pertains to an InfiniPath cluster. This is defined as a collection of nodes, each attached to an InfiniBand™-based fabric through the InfiniPath Interconnect. The nodes are Linux-based computers, each having up to eight processors.

The InfiniPath interconnect is InfiniBand 4X, with a raw data rate of 10 Gb/s (data rate of 8Gb/s).

InfiniPath utilizes standard, off-the-shelf InfiniBand 4X switches and cabling. InfiniPath OpenFabrics software is interoperable with other vendors' InfiniBand HCAs running compatible OpenFabrics releases. There are two options for Subnet Management in your cluster:

- Use the Subnet Manager on one or more managed switches supplied with your Infiniband switches.
- Use the OpenSM component of OpenFabrics.

1.4 Switches

The InfiniPath interconnect is designed to work with all InfiniBand-compliant switches. Use of OpenSM as a subnet manager is now supported. OpenSM is part of the OpenFabrics component of this release.

1.5 Interoperability

InfiniPath participates in the standard InfiniBand Subnet Management protocols for configuration and monitoring. InfiniPath OpenFabrics (including IPoIB) is interoperable with other vendors' InfiniBand HCAs running compatible OpenFabrics releases. The InfiniPath MPI and Ethernet emulation stacks (`ipath_ether`) are not interoperable with other InfiniBand Host Channel Adapters (HCA) and Target Channel Adapters (TCA). Instead, InfiniPath uses an InfiniBand-compliant vendor-specific protocol that is highly optimized for MPI and TCP between InfiniPath-equipped hosts.

NOTE: OpenFabrics was known as OpenIB until March 2006. All relevant references to OpenIB in this documentation have been updated to reflect this change. See the OpenFabrics website at http://www.openfabrics.org for more information on the OpenFabrics Alliance.

1.6 What's New in this Release

QLogic Corp. acquired PathScale in April 2006. In this 2.0 release, product names, internal program and output message names now refer to QLogic rather than PathScale.

The new QLogic and former PathScale adapter model numbers are shown in the table below.

Table 1-1. PathScale-QLogic Adapter Model Numbers

Former PathScale Model Number	New QLogic Model Number	Description
HT-400	IBA6110	Single Port 10GBS InfiniBand to HTX ASIC ROHS
PE-800	IBA6120	Single Port 10GBS InfiniBand to x8 PCI Express ASIC ROHS
HT-460	QHT7040	Single Port 10GBS InfiniBand to HTX Adapter
HT-465	QHT7140	Single Port 10GBS InfiniBand to HTX Adapter
PE-880	QLE7140	Single Port 10GBS InfiniBand to x8 PCI Express Adapter
PE-850	QMI7140	Single Port 10GBS InfiniBand IBM Blade Center Adapter

This version of InfiniPath provides support for all QLogic's HCAs, including:

- InfiniPath QLE7140, which is supported on systems with PCIe x8 or x16 slots
- InfiniPath QMI7140, which runs on Power PC systems, particularly on the IBM® BladeCenter H processor blades
- InfiniPath QHT7040 and QHT7140, which leverage HTX™. The InfiniPath QHT7040 and QHT7140 are exclusively for motherboards that support HTXcards. The QHT7140 has a smaller form factor than the QHT7040, but is otherwise the same. Unless otherwise stated, QHT7140 will refer to both the QHT7040 and QHT7140 in this documentation.

Expanded MPI scalability enhancements for PCI Express have been added. The QHT7040 and QHT7140 can support 2 processes per context for a total of 16. The QLE7140 and QMI7140 also support 2 processes per context, for a total of 8.

Support for multiple versions of MPI has been added. You can use a different version of MPI and achieve the high-bandwidth and low-latency performance that is standard with InfiniPath MPI.

Also included is expanded operating system support, and support for the latest OpenFabrics software stack.

Multiple InfiniPath cards per node are supported. A single software installation works for all the cards.

Additional up-to-date information can be found on the QLogic web site:

http://www.qlogic.com

1.7 Supported Distributions and Kernels

The InfiniPath interconnect runs on AMD Opteron, Intel EM64T, and IBM Power Blade Center H) systems running Linux. The currently supported distributions and associated Linux kernel versions for InfiniPath and OpenFabrics are listed in the following table. The kernels are the ones that shipped with the distributions, unless otherwise noted.

Table 1-2. InfiniPath/OpenFabrics Supported Distributions and Kernels

Distribution	InfiniPath/OpenFabrics supported kernels
Fedora Core 3 (FC3)	2.6.12 (x86_64)
Fedora Core 4 (FC4)	2.6.16, 2.6.17 (x86_64)
Red Hat Enterprise Linux 4 (RHEL4)	2.6.9-22, 2.6.9-34, 2.6.9-42(U2/U3/U4) (x86_64)
CentOS 4.2-4.4 (Rocks 4.2-4.4)	2.6.9 (x86_64)
SUSE Linux 9.3 (SUSE 9.3)	2.6.11 (x86_64)
SUSE LInux Enterprise Server (SLES 9)	2.6.5 (x86_64)
SUSE LInux Enterprise Server (SLES 10)	2.6.16 (x86_64 and ppc64)

NOTE: IBM Power systems run only with the SLES 10 distribution.

The SUSE10 release series is no longer supported as of this InfiniPath 2.0 release. Fedora Core 4 kernels prior to 2.6.16 are also no longer supported.

1.8 Software Components

The software provided with the InfiniPath Interconnect product consists of:

- InfiniPath driver (including OpenFabrics)
- InfiniPath ethernet emulation
- InfiniPath libraries
- InfiniPath utilities, configuration, and support tools
- InfiniPath MPI
- InfiniPath MPI benchmarks
- OpenFabrics protocols, including Subnet Management Agent
- OpenFabrics libraries and utilities

OpenFabrics kernel module support is now built and installed as part of the InfiniPath RPM install. The InfiniPath release 2.0 runs on the same code base as OpenFabrics Enterprise Distribution (OFED) version 1.1. It also includes the OpenFabrics 1.1-based library and utility RPMs. InfiniBand protocols are interoperable between InfiniPath 2.0 and OFED 1.1.

This release provides support for the following protocols:

- IPoIB (TCP/IP networking)
- SDP (Sockets Direct Protocol)
- OpenSM
- UD (Unreliable Datagram)
- RC (Reliable Connection)
- UC (Unreliable Connection)
- SRQ (Shared Receive Queue)
- uDAPL (user Direct Access Provider Library)

This release includes a technology preview of:

- SRP (SCSI RDMA Protocol)

Future releases will provide support for:

- iSER (iSCSI Extensions for RDMA)

No support is provided for RD.

NOTE: 32 bit OpenFabrics programs using the verb interfaces are not supported in this InfiniPath release, but will be supported in a future release.

1.9 Conventions Used in this Document

This Guide uses these typographical conventions:

Table 1-3. Typographical Conventions

Convention	Meaning
`command`	Fixed-space font is used for literal items such as commands, functions, programs, files and pathnames, and program output;
`variable`	Italic fixed-space font is used for variable names in programs and command lines.
concept	Italic font is used for emphasis, concepts.
`user input`	Bold fixed-space font is used for literal items in commands or constructs that you type in.
`$`	Indicates a command line prompt.
`#`	Indicates a command line prompt as root when using bash or sh.
[ ]	Brackets enclose optional elements of a command or program construct.
...	Ellipses indicate that a preceding element can be repeated.
>	Right caret identifies the cascading path of menu commands used in a procedure.
2.0	The current version number of the software is included in the RPM names and within this documentation.
NOTE:	Indicates important information.

1.10 Documentation and Technical Support

The InfiniPath product documentation includes:

- The *InfiniPath Install Guide*
- The *InfiniPath User Guide*
- Release Notes
- Quick Start Guide
- Readme file

The Troubleshooting Appendix for installation, InfiniPath and OpenFabrics administration, and MPI issues is located in the *InfiniPath User Guide*.

Visit the QLogic support Web site for documentation and the latest software updates.

http://www.qlogic.com

Notes

Section 2

InfiniPath Cluster Administration

This chapter describes what the cluster administrator needs to know about the InfiniPath software and system administration.

2.1 Introduction

The InfiniPath driver `ib_ipath`, layered Ethernet driver `ipath_ether`, OpenSM, and other modules and the protocol and MPI support libraries are the components of the InfiniPath software providing the foundation that supports the MPI implementation.

Figure 2-1, below, shows these relationships.

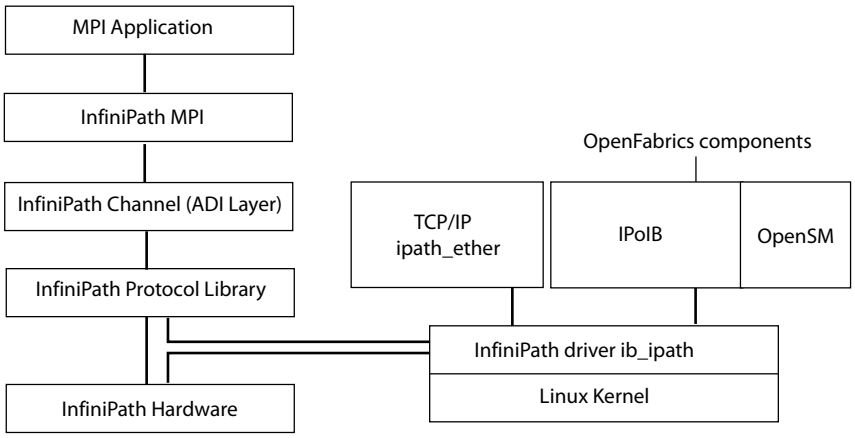

Figure 2-1. InfiniPath Software Structure

2.2 Installed Layout

The InfiniPath software is supplied as a set of RPM files, described in detail in the *InfiniPath Install Guide*. This section describes the directory structure that the installation leaves on each node's file system.

The InfiniPath shared libraries are installed in:

```
/usr/lib for 32-bit applications
/usr/lib64 for 64-bit applications
```

MPI include files are in:

```
/usr/include
```

MPI programming examples and source for several MPI benchmarks are in:

```
/usr/share/mpich/examples
```

InfiniPath utility programs, as well as MPI utilities and benchmarks are installed in:

```
/usr/bin
```

The InfiniPath kernel modules are installed in the standard module locations in:

```
/lib/modules (version dependent)
```

They are compiled and installed when the `infinipath-kernel` RPM is installed. They must be rebuilt and re-installed when the kernel is upgraded. This can be done by running the script:

```
/usr/src/infinipath/drivers/make-install.sh
```

Documentation can be found in:

```
/usr/share/man
/usr/share/doc/infinipath
/usr/share/doc/mpich-infinipath
```

2.3 Memory Footprint

The following is a preliminary guideline for estimating the memory footprint of the InfiniPath adapter on Linux x86_64systems. Memory consumption is linear based on system configuration. OpenFabrics support is under development and has not been fully characterized. This table summarizes the guidelines.

Table 2-1. Memory Footprint of the InfiniPath Adapter on Linux x86_64 Systems

Adapter component	Required/ optional	Memory Footprint	Comment
InfiniPath Driver	Required	9 MB	Includes accelerated IP support. Includes tables space to support up to 1000 node systems. Clusters larger than 1000 nodes can also be configured.
MPI	Optional	71 MB per process with default parameters: 60 MB + 512*2172 (sendbufs) + 4096*2176 (recvbufs) + 1024*1K (misc allocations) + 32 MB per node when multiple processes communicate via shared memory + 264 Bytes per MPI node on the subnet	Several of these parameters (sendbufs, recvbufs and size of the shared memory region) are tunable if reduced memory footprint is desired.
OpenFabrics	Optional	1~6 MB + ~500 bytes per QP + TBD bytes per MR + ~500 bytes per EE Context + OpenFabrics stack from openfabrics.org (size not included in these guidelines)	This not been fully characterized as of this writing.

Here is an example for a 1024 processor system:

- 1024 cores over 256 nodes (each node has 2 sockets with dual-core processors)
- 1 adapter per node
- Each core runs an MPI process, with the 4 processes per node communicating via shared memory.
- Each core uses OpenFabrics to connect with storage and file system targets using 50 QPs and 50 EECs per core.

This breaks down to a memory footprint of 331MB per node, as follows:

Table 2-2. Memory Footprint, 331 MB per Node

Component	Footprint (in MB)	Breakdown
Driver	9	Per node
MPI	316	4*71 MB (MPI per process) + 32 MB (shared memory per node)
OpenFabrics	6	6 MB + 200 KB per node

2.4 Configuration and Startup

2.4.1 BIOS Settings

A properly configured BIOS is required. The BIOS settings, which are stored in non-volatile memory, contain certain parameters characterizing the system,. These parameters may include date and time, configuration settings, and information about the installed hardware.

There are currently two issues concerning BIOS settings that you need to be aware of:

- ACPI needs to be enabled
- MTRR mapping needs to be set to “Discrete”

MTRR (Memory Type Range Registers) is used by the InfiniPath driver to enable write combining to the InfiniPath on-chip transmit buffers. This improves write bandwidth to the InfiniPath chip by writing multiple words in a single bus transaction (typically 64). This applies only to x86_64 systems.

However, some BIOSes don’t have the MTRR mapping option. It may be referred to in a different way, dependent upon chipset, vendor, BIOS, or other factors. For example, it is sometimes referred to as "32 bit memory hole", which should be enabled.

If there is no setting for MTRR mapping or 32 bit memory hole, please contact your system or motherboard vendor and inquire as to how write combining may be enabled.

ACPI and MTRR mapping issues are discussed in greater detail in the Troubleshooting section of the *InfiniPath User Guide*.

NOTE: BIOS settings on IBM Blade Center H (Power) systems do not need adjustment.

You can check and adjust these BIOS settings using the BIOS Setup Utility. For specific instructions on how to do this, follow the hardware documentation that came with your system.

2.4.2 InfiniPath Driver Startup

The `ib_ipath` module provides low level InfiniPath hardware support. It does hardware initialization, handles infinipath-specific memory management, and provides services to other InfiniPath and OpenFabrics modules. It provides the management functions for InfiniPath MPI programs, the `ipath_ether` ethernet emulation, and general OpenFabrics protocols such as IPoIB, and SDP. It also contains a Subnet Management Agent.

The InfiniPath driver software is generally started at system startup under control of these scripts:

```
/etc/init.d/infinipath
/etc/sysconfig/infinipath
```

These scripts are configured by the installation. Debug messages are printed with the function name preceding the message.

The cluster administrator does not normally need to be concerned with the configuration parameters. Assuming that all the InfiniPath and OpenFabrics software has been installed, the default settings upon startup will be:

- InfiniPath `ib_ipath` is enabled
- InfiniPath `ipath_ether` is not running until configured
- OpenFabrics IPoIB is not running until configured
- OpenSM is enabled on startup. Disable it on all nodes except where it will be used as subnet manager.

2.4.3 InfiniPath Driver Software Configuration

The `ib_ipath` driver has several configuration variables which provide for setting reserved buffers for the software, defining events to create trace records, and setting debug level. See the `ib_ipath` man page for details.

2.4.4 InfiniPath Driver Filesystem

The InfiniPath driver supplies a filesystem for exporting certain binary statistics to user applications. By default, this filesystem is mounted in the `/ipathfs` directory when the infinipath script is invoked with the "start" option (e.g. at system startup)

and unmounted when the infinipath script is invoked with the "stop" option (e.g. at system shutdown).

The layout of the filesystem is as follows:

```
atomic_stats
00/
01/
...
```

The `atomic_stats` file contains general driver statistics. There is one numbered directory per InfiniPath device on the system. Each numbered directory contains the following files of per-device statistics:

```
atomic_counters
node_info
port_info
```

The `atomic_counters` file contains counters for the device: examples would be interrupts received, bytes and packets in and out, and so on. The `node_info` file contains information such as the device's GUID. The `port_info` file contains information for each port on the device. An example would be the port LID.

2.4.5 Subnet Management Agent

Each node in an InfiniPath cluster runs a Subnet Management Agent (SMA), which carries out two-way communication with the Subnet Manager (SM) running on one or more managed switches. The Subnet Manager is responsible for network initialization (topology discovery), configuration, and maintenance. The Subnet Manager also assigns and manages InfiniBand multicast groups, such as the group used for broadcast purposes by the `ipath_ether` driver. The primary functions of the SMA are to keep the SM informed whether a node is alive and to get the node's assigned identifier (LID) from the SM.

2.4.6 Layered Ethernet Driver

The layered Ethernet component `ipath_ether` provides almost complete Ethernet software functionality over the InfiniPath fabric. At startup this is bound to some Ethernet device `ethx`. All Ethernet functions are available through this device in a transparent way, except that Ethernet multicasting is not supported. Broadcasting is supported. You can use all the usual command line and GUI-based configuration tools on this Ethernet. Configuration of `ipath_ether` is optional.

These instructions are for enabling TCP-IP networking over the InfiniPath link. To enable IPoIB networking, see section 2.4.7.1.

You must create a network device configuration file for the layered Ethernet device on the InfiniPath adapter. This configuration file will resemble the configuration files for the other Ethernet devices on the nodes. Typically on servers there are two Ethernet devices present, numbered as 0 (`eth0`) and 1 (`eth1`). This examples assumes we create a third device, `eth2`.

NOTE: When multiple InfiniPath chips are present, the configuration for `eth3`, `eth4`, and so on follow the same format as for adding eth2 in the examples below.

Two slightly different procedures are given below for the ipath configuration; one for Fedora and one for SUSE, SLES9, or SLES 10.

Many of the entries that are used in the configuration directions below are explained in the file `sysconfig.txt`. To familiarize yourself with these, please see:

`/usr/share/doc/initscripts-*/sysconfig.txt`

2.4.6.1
`ipath_ether` Configuration on Fedora and RHEL4

These configuration steps will cause the `ipath_ether` network interfaces to be automatically configured when you next reboot the system. These instructions are for the Fedora Core 3, Fedora Core 4 and Red Hat Enterprise Linux 4 distributions.

Typically on servers there are two Ethernet devices present, numbered as 0 (`eth0`) and 1 (`eth1`). This example assumes we create a third device, `eth2`.

NOTE: When multiple InfiniPath chips are present, the configuration for `eth3`, `eth4`, and so on follow the same format as for adding `eth2` in the examples below.

1. Check for the number of Ethernet drivers you currently have by either one of the two following commands :

   ```
   $ ifconfig -a
   $ ls /sys/class/net
   ```

   As mentioned above we assume that two Ethernet devices (numbered 0 and 1) are already present.

2. Edit the file `/etc/modprobe.conf` (as root) by adding the following line:

   ```
   alias eth2 ipath_ether
   ```

3. Create or edit the following file (as root).

   `/etc/sysconfig/network-scripts/ifcfg-eth2`

If you are using DHCP (dynamic host configuration protocol), add the following lines to `ifcfg-eth2`:

```
# QLogic Interconnect Ethernet
DEVICE=eth2
ONBOOT=yes
BOOTPROTO=dhcp
```

If you are using static IP addresses, use the following lines instead, substituting your own IP address for the sample one given here.The normal matching netmask is shown.

```
# QLogic Interconnect Ethernet
DEVICE=eth2
BOOTPROTO=static
ONBOOT=YES
IPADDR=192.168.5.101 #Substitute your IP address here
NETMASK="255.255.255.0"#Normal matching netmask
TYPE=Ethernet
```

This will cause the ipath_ether Ethernet driver to be loaded and configured during system startup. To check your configuration, and make the ipath_ether Ethernet driver available immediately, use the command (as root):

```
# /sbin/ifup eth2
```

4. Check whether the Ethernet driver has been loaded with:

   ```
   $ lsmod | grep ipath_ether
   ```

5. Verify that the driver is up with:

   ```
   $ ifconfig -a
   ```

2.4.6.2

ipath_ether Configuration on SUSE 9.3, SLES 9, and SLES 10

These configuration steps will cause the `ipath_ether` network interfaces to be automatically configured when you next reboot the system. These instructions are for the SUSE 9.3, SLES 9 and SLES 10 distributions.

Typically on servers there are two Ethernet devices present, numbered as 0 (`eth0`) and 1 (`eth1`). This example assumes we create a third device, `eth2`.

NOTE: When multiple InfiniPath chips are present, the configuration for `eth3`, `eth4`, and so on follow the same format as for adding `eth2` in the examples below. Similarly , in step 2, add one to the unit number, so replace `.../00/guid` with `/01/guid` for the second InfiniPath interface, and so on.

Step 3 is applicable only to SLES 10; it is required because SLES 10 uses a newer version of the `udev` subsystem.

NOTE: The MAC address (media access control address) is a unique identifier attached to most forms of networking equipment. Step 2 below determines the MAC address to use, and will be referred to as $MAC in the subsequent steps. $MAC must be replaced in each case with the string printed in step 2.

The following steps must all be executed as the root user.

1. Be sure that the ipath_ether module is loaded:

   ```
   # lsmod | grep -q ipath_ether || modprobe ipath_ether
   ```

2. Determine the MAC address that will be used:

   ```
   # sed 's/^\(..:..:..\):..:../\1/' \
   /sys/bus/pci/drivers/ib_ipath/00/guid
   ```

   NOTE: Care should be taken when cutting and pasting commands such as the above from PDF documents, as quotes are special characters and may not be translated correctly.

   The output should appear similar to this (6 hex digit pairs, separated by colons):

   ```
   00:11:75:04:e0:11
   ```

   The GUID can also be returned by running:

   ```
   # ipath_control -i
   $Id: QLogic Release2.0 $ $Date: 2006-10-15-04:16 $
   00: Version: Driver 2.0, InfiniPath_QHT7140, InfiniPath1 3.2,
   PCI 2, SW Compat 2
   00: Status: 0xe1 Initted Present IB_link_up IB_configured
   00: LID=0x30 MLID=0x0 GUID=00:11:75:00:00:04:e0:11 Serial:
   1236070407
   ```

   Note that removing the middle two `00:00` octets from the GUID in the above output will form the MAC address

   If either step 1 or step 2 fails in some fashion, the problem must be found and corrected before continuing. Verify that the RPMs are installed correctly, and that `infinipath` has correctly been started. If problems continue, run `ipathbug-helper` and report the results to your reseller or InfiniPath support organization.

3. Skip to Step 4 if you are using SUSE 9.3 or SLES 9. This step is only done on SLES 10 systems. Edit the file:

   ```
   /etc/udev/rules.d/30-net_persistent_names.rules
   ```

   If this file does not exist, skip to Step 4.

Check each of the lines starting with `SUBSYSTEM=`, to find the highest numbered interface. (For standard motherboards, the highest numbered interface will typically be 1.)

Add a new line at the end of the file, incrementing the interface number by one. In this example, it becomes `eth2`. The new line will look like this:

```
SUBSYSTEM=="net", ACTION=="add", SYSFS{address}=="$MAC",
IMPORT="/sbin/ rename_netiface %k eth2"
```

This will appear as a single line in the file. $MAC is replaced by the string from step 2 above.

4. Create the network module file:

   `/etc/sysconfig/hardware/hwcfg-eth-id-$MAC`
   Add the following lines to the file:

   ```
   MODULE=ipath_ether
   STARTMODE=auto
   ```

   This will cause the `ipath_ether` Ethernet driver to be loaded and configured during system startup.

5. Create the network configuration file:

   `/etc/sysconfig/network/ifcfg-eth2`

   If you are using DHCP (dynamically assigned IP addresses), add these lines to the file:

   ```
   STARTMODE=onboot
   BOOTPROTO=dhcp
   NAME='InfiniPath Network Card'
   _nm_name=eth-id-$MAC
   ```

   Proceed to Step 6.

   If you are you are using static IP addresses (not DHCP), add these lines to the file:

   ```
   STARTMODE=onboot
   BOOTPROTO=static
   NAME='InfiniPath Network Card'
   NETWORK=192.168.5.0
   NETMASK=255.255.255.0
   BROADCAST=192.168.5.255
   IPADDR=192.168.5.211
   _nm_name=eth-id-$MAC
   ```

   Make sure that you substitute your own IP address for the sample IPADDR shown here. The BROADCAST, NETMASK, and NETWORK lines need to match for your network.

6. To verify that the configuration files are correct, you will normally now be able to run the commands:

```
# ifup eth2
# ifconfig eth2
```

Note that it may be necessary to reboot the system before the configuration changes will work.

2.4.7 OpenFabrics Configuration and Startup

In the prior InfiniPath 1.3 release the InfiniPath (`ipath_core`) and OpenFabrics (`ib_ipath`) modules were separate. In this release there is now one module, `ib_ipath`, which provides both low level InfiniPath support and management functions for OpenFabrics protocols. The startup script for `ib_ipath` is installed automatically as part of the software installation, and normally does not need to be changed.

However, the IPoIB network interface and OpenSM components of OpenFabrics can be configured to be on or off. IPoIB is **off** by default; OpenSM is **on** by default.

IPoIB and OpenSM configuration is explained in greater detail in the following sections.

NOTE: The following instructions work for FC4, SUSE9.3, SLES 9, and SLES 10.

2.4.7.1 Configuring the IPoIB Network Interface

Instructions are given here to manually configure your OpenFabrics IPoIB network interface. This example assumes that you are using `sh` or `bash` as your shell, and that all required InfiniPath and OpenFabrics RPMs are installed, and your startup scripts have been run, either manually or at system boot.

For this example, we assume that your IPoIB network is 10.1.17.0 (one of the networks reserved for private use, and thus not routable on the internet), with a /8 host portion, and therefore requires that the netmask be specified.

This example assumes that no hosts files exist, and that the host being configured has the IP address 10.1.17.3, and that DHCP is not being used.

NOTE: We supply instructions only for this static IP address case. Configuration methods for using DHCP will be supplied in a later release.

Type the following commands (as root):

```
# ifconfig ib0 10.1.17.3 netmask 0xffffff00
```

To verify the configuration, type:

```
# ifconfig ib0
```

The output from this command should be similar to this:

```
ib0   Link encap:InfiniBand  HWaddr
00:00:00:02:FE:80:00:00:00:00:00:00:00:00:00:00:00:00:00:00
inet addr:10.1.17.3  Bcast:10.1.17.255 Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST  MTU:2044  Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:128
RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

Next, type:

```
# ping -c 2 -b 10.1.17.255
```

The output of the `ping` command should be similar to that below, with a line for each host already configured and connected:

```
WARNING: pinging broadcast address
PING 10.1.17.255 (10.1.17.255) 517(84) bytes of data.
174 bytes from 10.1.17.3: icmp_seq=0 ttl=174 time=0.022 ms
64 bytes from 10.1.17.1: icmp_seq=0 ttl=64 time=0.070 ms (DUP!)
64 bytes from 10.1.17.7: icmp_seq=0 ttl=64 time=0.073 ms (DUP!)
```

The IPoIB network interface is now configured.

NOTE: The configuration must be repeated each time the system is rebooted.

2.4.8 OpenSM

OpenSM is an optional component of the OpenFabrics project that provides a subnet manager for InfiniBand networks. This package can be installed on all machines, but only needs to be enabled on the machine in your cluster that is going to act as a subnet manager. You do not need to use OpenSM if any of your InfiniBand switches provide a subnet manager.

After installing the `opensm` package, OpenSM is configured to be **on** on the next machine reboot. It only needs to be enabled on the node which acts as the subnet manager, so use the `chkconfig` command (as root) to disable it on the other nodes:

```
# chkconfig opensmd off
```

The command to enable it on reboot is:

```
# chkconfig opensmd on
```

You can start `opensmd` without rebooting your machine as follows:

```
# /etc/init.d/opensmd start
```

and you can stop it again like this:

```
# /etc/init.d/opensmd stop
```

If you wish to pass any arguments to the OpenSM program, modify the file:

```
/etc/init.d/opensmd
```

and add the arguments to the "OPTIONS" variable. Here is an example:

```
# Use the UPDN algorithm instead of the Min Hop algorithm.
OPTIONS="-u"
```

2.5 SRP

SRP stands for SCSI RDMA Protocol. It was originally intended to allow the SCSI protocol to run over InfiniBand for SAN usage. SRP interfaces directly to the Linux file system through the SRP Upper Layer Protocol. SRP storage can be treated as just another device.

In this release SRP is provided as a technology preview. Add `ib_srp` to the module list in `/etc/sysconfig/infinipath` to have it automatically loaded.

NOTE: SRP does not yet work with IBM Power Systems.This will be fixed in a future release.

2.6 Further Information on Configuring and Loading Drivers

See the `modprobe(8)`, `modprobe.conf(5)`, `lsmod(8)`, man pages for more information. Also see the file `/usr/share/doc/initscripts-*/sysconfig.txt` for more general information on configuration files. Section 2.7, below, may also be useful.

2.7 Starting and Stopping the InfiniPath Software

The InfiniPath driver software runs as a system service, normally started at system startup. Normally you will not need to restart the software, but you may wish to do so after installing a new InfiniPath release, or after changing driver options, or if doing manual testing.

The following commands can be used to check or configure state. These methods will not reboot the system.

To check the configuration state, use the command:

```
$ chkconfig --list infinipath
```

To enable the driver, use the command (as root):

```
# chkconfig infinipath on 2345
```

To disable the driver on the next system boot, use the command (as root):

```
# chkconfig infinipath off
```

NOTE: This does not stop and unload the driver, if it is already loaded.

You can start, stop, or restart (as root) the InfiniPath support with:

```
# /etc/init.d/infinipath [start | stop | restart]
```

This method will not reboot the system. The following set of commands shows how this script can be used. Please take note of the following:

- You should omit the commands to start/stop `opensmd` if you are not running it on that node.
- You should omit the `ifdown` and `ifup` step if you are not using `ipath_ether` on that node.

The sequence of commands to restart `infinipath` are given below. Note that this next example assumes that `ipath_ether` is configured as `eth2`.

```
# /etc/init.d/opensmd stop
# ifdown eth2
# /etc/init.d/infinipath stop
...
# /etc/init.d/infinipath start
# ifup eth2
# /etc/init.d/opensmd start
```

The `...` represents whatever activity you are engaged in after InfiniPath is stopped.

An equivalent way to specify this is to use same sequence as above, except use the **`restart`** command instead of **`start`** and **`stop`**:

```
# /etc/init.d/opensmd stop
# ifdown eth2
# /etc/init.d/infinipath restart
# ifup eth2
# /etc/init.d/opensmd start
```

NOTE: Restarting InfiniPath will terminate any InfiniPath MPI processes, as well as any OpenFabrics processes that are running at the time. Processes using networking over `ipath_ether` will return errors.

You can check to see if `opensmd` is running by using the following command; if there is no output, `opensmd` is not configured to run:

```
# /sbin/chkconfig --list opensmd | grep -w on
```

You can check to see if `ipath_ether` is running by using the following command. If it prints no output, it is not running.

```
$ /sbin/lsmod | grep ipath_ether
```

If there is output, you should look at the output from this command to determine if it is configured:

```
$ /sbin/ifconfig -a
```

Finally, if you need to find which InfiniPath and OpenFabrics modules are running, try the following command:

```
$ lsmod | egrep 'ipath_|ib_|rdma_|findex'
```

2.8 Software Status

InfiniBand status can be checked by running the program `ipath_control`. Here is sample usage and output:

```
$ ipath_control -i
      $Id: QLogic Release2.0 $ $Date: 2006-09-15-04:16 $
      00: Version: Driver 2.0, InfiniPath_QHT7140, InfiniPath1 3.2,
      PCI 2, SW Compat 2
      00: Status: 0xe1 Initted Present IB_link_up IB_configured
      00: LID=0x30 MLID=0x0 GUID=00:11:75:00:00:07:11:97 Serial:
      1236070407
```

Another useful program is `ibstatus`. Sample usage and output is as follows:

```
$ ibstatus

Infiniband device 'ipath0' port 1 status:
        default gid:     fe80:0000:0000:0000:0011:7500:0005:602f
        base lid:        0x35
        sm lid:          0x2
        state:           4: ACTIVE
        phys state:      5: LinkUp
        rate:            10 Gb/sec (4X)
```

For more information on these programs, See appendix C.9.9 and appendix C.9.5.

2.9 Configuring ssh and sshd Using shosts.equiv

Running MPI programs on an InfiniPath cluster depends, by default, on secure shell `ssh` to launch node programs on the nodes. Jobs must be able to start up without the need for interactive password entry on every node. Here we see how the cluster administrator can lift this burden from the user through the use of the `shosts.equiv` mechanism. This method is recommended, provided that your cluster is behind a firewall and accessible only to trusted users.

Later, in section 3.5.1, we show how an individual user can accomplish this end through the use of `ssh-agent`.

This next example assumes the following:

- Both the cluster nodes and the front end system are running the openssh package as distributed in current Linux systems.
- All cluster users have accounts with the same account name on the front end and on each node, either by using NIS or some other means of distributing the password file.
- The front end is called ip-fe.
- Root or superuser access is required on ip-fe and on each node in order to configure ssh.
- ssh, including the host's key, has already been configured on the system ip-fe. See the sshd and ssh-keygen man pages for more information.

The example proceeds as follows:

1. On the system ip-fe, the front end node, change /etc/ssh/ssh_config to allow host-based authentication. Specifically, this file must contain the following four lines, set to 'yes'. If they are already present but commented out with an initial #, remove the #.

   ```
   RhostsAuthentication yes
   RhostsRSAAuthentication yes
   HostbasedAuthentication yes
   EnableSSHKeysign yes
   ```

2. On each of the InfiniPath node systems, create or edit the file /etc/ssh/shosts.equiv, adding the name of the front end system. You'll need to add the line:

   ```
   ip-fe
   ```

   Change the file to mode 600 when finished editing.

3. On each of the InfiniPath node systems, create or edit the file /etc/ssh/ssh_known_hosts. You'll need to copy the contents of the file /etc/ssh/ssh_host_dsa_key.pub from ip-fe to this file (as a single line), and then edit that line to insert ip-fe ssh-dss at the beginning of the line. This is very similar to the standard known_hosts file for ssh. An example line might look like this (displayed as multiple lines, but a single line in the file):

   ```
   ip-fe ssh-dss
   AAzAB3NzaC1kc3MAAACBAPoyES6+Akk+z3RfCkEHCkmYuYzqL2+1nwo4LeTVWp
   CD1QsvrYRmpsfwpzYLXiSJdZSA8hfePWmMfrkvAAk4ueN8L3ZT4QfCTwqvHVvS
   ctpibf8n
   aUmzloovBndOX9TIHyP/Ljfzzep4wL17+5hr1AHXldzrmgeEKp6ect1wxAAAAF
   QDR56dAKFA4WgAiRmUJailtLFp8swAAAIBB1yrhF5P0jO+vpSnZrvrHa0Ok+Y9
   apeJp3sessee30NlqKbJqWj5DOoRejr2VfTxZROf8LKuOY8tD6I59I0vlcQ812
   E5iw1GCZfNefBmWbegWVKFwGlNbqBnZK7kDRLSOKQtuhYbGPcrVlSjuVpsfWEj
   u64FTqKEetA8l8QEgAAAIBNtPDDwdmXRvDyc0gvAm6lPOIsRLmgmdgKXTGOZUZ
   0zwxSL7GP1nEyFk9wAxCrXv3xPKxQaezQKs+KL95FouJvJ4qrSxxHdd1NYNR0D
   ```

```
avEBVQgCaspgWvWQ8cL
0aUQmTbggLrtD9zETVU5PCgRlQL6I3Y5sCCHuO7/UvTH9nneCg==
```

Change the file to mode 600 when finished editing.

4. On each node, the system file `/etc/ssh/sshd_config` must be edited, so that the following four lines uncommented (no `#` at the start of the line) and are set to `yes`. Each of these lines is normally present, but commented out and set to `no` by default.

```
RhostsAuthentication yes
RhostsRSAAuthentication yes
HostbasedAuthentication yes
PAMAuthenticationViaKbdInt yes
```

5. After creating or editing these three files in steps 2, 3 and 4, `sshd` must be restarted on each system. If you are already logged in via `ssh` (or any other user is logged in via `ssh`), their sessions or programs will be terminated, so do this only on idle nodes. Tell `sshd` to use the new configuration files by typing (as root):

```
# killall -HUP sshd
```

NOTE: This will terminate all ssh sessions into that system. Run from the console, or have a way to log into the console in case of any problem.

At this point, any user should be able to login to the `ip-fe` front end system, and then use `ssh` to login to any InfiniPath node without being prompted for a password or pass phrase.

2.9.1 Process Limitation with `ssh`

MPI jobs that use more than 8 processes per node may encounter an SSH throttling mechanism that limits the amount of concurrent per-node connections to 10. If you need to use more processes, you or your system administrator should increase the value of 'MaxStartups' in your sshd configurations. See appendix C.8.8 for an example of an error message associated with this limitation.

2.10 Performance and Management Tips

The following section gives some suggestions for improving performance and simplifying management of the cluster.

2.10.1 Remove Unneeded Services

An important step that the cluster administrator can take to enhance application performance is to minimize the set of system services running on the compute nodes. Since these are presumed to be specialized computing appliances, they

do not need many of the service daemons normally running on a general Linux computer.

Following are several groups constituting a minimal necessary set of services. These are all services controlled by `chkconfig`. To see the list of services that are enabled, use the command:

```
$ /sbin/chkconfig --list | grep -w on
```

Basic network services:

```
network
ntpd
syslog
xinetd
sshd
```

For system housekeeping:

```
anacron
atd
crond
```

If you are using NFS or yp passwords:

```
rpcidmapd
ypbind
portmap
nfs
nfslock
autofs
```

To watch for disk problems:

```
smartd
readahead
```

The service comprising the InfiniPath driver and SMA:

```
infinipath
```

Other services may be required by your batch queuing system or user community.

2.10.2
Disable Powersaving Features

If you are running benchmarks or large numbers of short jobs, it is beneficial to disable the powersaving features of the Opteron. The reason is that these features may be slow to respond to changes in system load.

For rhel4, fc3 and fc4, run this command as root:

```
# /sbin/chkconfig --level 12345 cpuspeed off
```

For SUSE 9.3 and 10.0 run this command as root:

```
# /sbin/chkconfig --level 12345 powersaved off
```

After running either of these commands, the system will need to be rebooted for these changes to take effect.

2.10.3 Balanced Processor Power

Higher processor speed is good. However, adding more processors is good only if processor speed is balanced. Adding processors with different speeds can result in load imbalance.

2.10.4 SDP Module Parameters for Best Performance

To get the best performance from SDP, especially for bandwidth tests, edit one of these files:

```
/etc/modprobe.conf (on Fedora and RHEL)
/etc/modprobe.conf.local (on SUSE and SLES)
```

Add the line:

```
options ib_sdp sdp_debug_level=4
sdp_zcopy_thrsh_src_default=10000000
```

This should be a single line in the file. This sets both the debug level and the zero copy threshold.

2.10.5 CPU Affinity

InfiniPath will attempt to run each node program with CPU affinity set to a separate logical processor, up to the number of available logical processors. If CPU affinity is already set (with `sched_setaffinity()`, or with the `taskset` utility), then InfiniPath will not change the setting.

The `taskset` utility can be used with `mpirun` to specify the mapping of MPI processes to logical processors. This is useful, for example, to make best use of available memory bandwidth or cache locality when running on dual-core SMP cluster nodes.

In the following example we use the NAS Parallel Benchmark's MG (multi-grid) benchmark and the `-c` option to `taskset`.

```
$ mpirun -np 4 -ppn 2 -m $hosts taskset -c 0,2 bin/mg.B.4
$ mpirun -np 4 -ppn 2 -m $hosts taskset -c 1,3 bin/mg.B.4
```

The first command forces the programs to run on CPUs (or cores) 0 and 2. The second forces the programs to run on CPUs 1 and 3. Please see the `man` page for `taskset` for more information on usage.

2.10.6
Hyper-Threading

If using Intel processors that support Hyper-Threading, it is recommended that HyperThreading is turned off in the BIOS. This will provide more consistent performance. You can check and adjust this setting using the BIOS Setup Utility. For specific instructions on how to do this, follow the hardware documentation that came with your system.

2.10.7
Homogeneous Nodes

To minimize management problems, the compute nodes of the cluster should have very similar hardware configurations and identical software installations. A mismatch between the InfiniPath software versions may also cause problems. Old and new libraries should not be run within the same job. It may also be useful to distinguish between the InfiniPath-specific drivers and those that are associated with kernel.org, OpenFabrics, or are distribution-built. The most useful tools are:

```
ipathbug-helper
ipath_control
rpm
mpirun
ident
strings
ipath_checkout
```

NOTE: Run these tools to gather information before reporting problems and requesting support.

`ipathbug_helper`

The InfiniPath software includes a shell script `ipathbug-helper`, which can gather status and history information for use in analyzing InfiniPath problems. This tool is also useful for verifying homogeneity. It is best to run `ipathbug-helper` with root privilege, since some of the queries require it. There is also a `--verbose` option which greatly increases the amount of gathered information. Simply run it on several nodes and examine the output for differences.

`ipath_control`

Run the shell script `ipath_control` as follows:

```
% ipath_control -i
$Id: QLogic Release2.0 $ $Date: 2006-09-15-04:16 $
00: Version: Driver 2.0, InfiniPath_QHT7140, InfiniPath1 3.2, PCI
2, SW Compat 2
00: Status: 0xe1 Initted Present IB_link_up IB_configured
```

```
00: LID=0x30 MLID=0x0 GUID=00:11:75:00:00:07:11:97 Serial:
1236070407
```

Note that `ipath_control` will report whether the installed adapter is the QHT7040, QHT7140, or the QLE7140. It will also report whether the driver is InfiniPath-specific or not with the output associated with `$Id`.

rpm

To check the contents of an RPM, commands of these types may be useful:

```
$ rpm -qa infinipath\* mpi-\*
$ rpm -q --info infinipath # (etc)
```

The option `-q` will query and `-qa` will query all.

mpirun

mpirun can give information on whether the program is being run against a QLogic or non-QLogic driver. Sample commands and results are given below.

QLogic-built:

```
$ mpirun -np 2 -m /tmp/id1 -d0x101 mpi_latency 1 0
asus-01:0.ipath_setaffinity: Set CPU affinity to 1, port 0:2:0 (1
active chips)
asus-01:0.ipath_userinit: Driver is QLogic-built
```

Non-QLogic built:

```
$ mpirun -np 2 -m /tmp/id1 -d0x101 mpi_latency 1 0
asus-01:0.ipath_setaffinity: Set CPU affinity to 1, port 0:2:0 (1
active chips)
asus-01:0.ipath_userinit: Driver is not QLogic-built
```

ident

`ident` strings are available in `ib_ipath.ko`. Running `ident` (as root) will yield information similar to the following. For QLogic RPMs, it will look like:

```
# ident /lib/modules/$(uname -r)/updates/*ipath.ko
/lib/modules/2.6.16.21-0.8-smp/updates/ib_ipath.ko:
     $Id: QLogic Release2.0 $
     $Date: 2006-09-15-04:16 $
     $Id: QLogic Release2.0 $
     $Date: 2006-09-15-04:16 $
```

For non-QLogic RPMs, it will look like:

```
# ident /lib/modules/$(uname -r)/updates/*ipath_ether.ko
/lib/modules/2.6.16.21-0.8-smp/updates/infinipath.ko:
```

```
  $Id: kernel.org InfiniPath Release 2.0 $
  $Date: 2006-09-15-04:16 $
/lib/modules/2.6.16.21-0.8-smp/updates/ipath.ko:
  $Id: kernel.org InfiniPath Release2.0 $
  $Date: 2006-09-15-04:20 $
```

NOTE: `ident` is in the optional `rcs` RPM, and is not always installed.

`strings`

The command `strings` can also be used. Here is a sample:

`$` **`strings /usr/lib/libinfinipath.so.4.0 | grep Date:`**

will produce output like this:

```
Date: 2006-09-15 04:07 Release2.0 InfiniPath $
```

NOTE: `strings` is part of `binutils` (a development RPM), and may not be available on all machines.

`ipath_checkout`

`ipath_checkout` is a `bash` script used to verify that the installation is correct, and that all the nodes are functioning. It is run on a front end node and requires a hosts file:

`$` **`ipath_checkout [options] hostsfile`**

More complete information on `ipath_checkout` is given below in section 2.11 and in section C.9.8.

2.11 Customer Acceptance Utility

`ipath_checkout` is a `bash` script used to verify that the installation is correct and that all the nodes of the network are functioning and mutually connected by the InfiniPath fabric. It is to be run on a front end node, and requires specification of a hosts file:

`$` **`ipath_checkout [options]`** ***`hostsfile`***

where `hostsfile` designates a file listing the hostnames of the nodes of the cluster, one hostname per line. The format of `hostsfile` is as follows:

```
hostname1
hostname2
...
```

`ipath_checkout` performs the following seven tests on the cluster:

1. `ping` all nodes to verify all are reachable from the frontend.
2. `ssh` to each node to verify correct configuration of `ssh`.

3. Gather and analyze system configuration from nodes.
4. Gather and analyze RPMs installed on nodes.
5. Verify InfiniPath hardware and software status and configuration.
6. Verify ability to `mpirun` jobs on nodes.
7. Run bandwidth and latency test on every pair of nodes and analyze results.

The possible options to `ipath_checkout` are:

`-h, --help`
Displays help messages giving defined usage.

`-v, --verbose`
`-vv, --vverbose`
`-vvv, --vvverbose`
These specify three successively higher levels of detail in reporting results of tests. So, there are four levels of detail in all, including the case of where none these options are given.

`-c, --continue`
When not specified, the test terminates when any test fails. When specified, the tests continue after a failure, with failing nodes excluded from subsequent tests.

`--workdir=DIR`
Use DIR to hold intermediate files created while running tests. DIR must not already exist.

`-k, --keep`
Keep intermediate files that were created while performing tests and compiling reports. Results will be saved in a directory created by `mktemp` and named `infinipath_XXXXXX` or in the directory name given to `--workdir`.

`--skip=LIST`
Skip the tests in `LIST` (e.g. --skip=2,4,5,7 will skip tests 2, 4, 5, and 7)

`-d, --debug`
Turn on `-x` and `-v` flags in bash.

In most cases of failure, the script suggests recommended actions. Please see the `ipath_checkout` man page for further information and updates.

Section 3

Using InfiniPath MPI

This chapter provides information on using InfiniPath MPI. Examples are provided for compiling and running MPI programs.

3.1 InfiniPath MPI

QLogic's implementation of the MPI standard is derived from the MPICH reference implementation Version 1.2.6. The InfiniPath MPI libraries have been highly tuned for the InfiniPath Interconnect, and will not run over other interconnects.

InfiniPath MPI is an implementation of the original MPI 1.2 standard. The MPI-2 standard provides several enhancements of the original standard. Of the MPI-2 features, InfiniPath MPI includes only the MPI-IO features implemented in ROMIO version 1.2.6 and the generalized MPI_Alltoallw communication exchange.

In this Version 2.0 release, the InfiniPath MPI implementation supports hybrid MPI/OpenMP, and other multi-threaded programs, as long as only one thread uses MPI. For more information, see section 3.10.

3.2 Other MPI Implementations

As of this release, other MPI implementations can now be run over InfiniPath. The currently supported implementations are HP-MPI, OpenMPI and Scali. For more information see section 3.6.

3.3 Getting Started with MPI

In this section you will learn how to compile and run some simple example programs that are included in the InfiniPath software product. Compiling and running these examples lets you verify that InfiniPath MPI and its components have been properly installed on your cluster. See appendix C.8 if you have problems compiling or running these examples.

These examples assume that:

- Your cluster administrator has properly installed InfiniPath MPI and the PathScale compilers.
- Your cluster's policy allows you to use the `mpirun` script directly, without having to submit the job to a batch queuing system.
- You or your administrator has properly set up your `ssh` keys and associated files on your cluster. See section 3.5.1 and section 2.9 for details on `ssh` administration.

To begin, copy the examples to your working directory:

```
$ cp /usr/share/mpich/examples/basic/* .
```

Next, create an MPI hosts file in the same working directory. It contains the host names of the nodes in your cluster on which you want to run the examples, with one host name per line. Name this file `mpihosts`. The contents can be in the following format:

```
hostname1
hostname2
...
```

There is more information on the `mpihosts` file in section 3.5.6.

3.3.1 An Example C Program

InfiniPath MPI uses some shell scripts to find the appropriate include files and libraries for each supported language. Use the script `mpicc` to compile an MPI program in C and the script `mpirun` to execute it.

The supplied example program `cpi.c` computes an approximation to pi. First, compile it to an executable named `cpi`.

```
$ mpicc -o cpi cpi.c
```

`mpicc`, by default, runs the PathScale `pathcc` or `gcc` compiler, and is used for both compiling and linking, exactly as you'd use the `pathcc` command.

NOTE: On ppc64 systems, `gcc` is the default compiler. For information on using other compilers, see section 3.5.3.

Then, run it with several different specifications for the number of processes:

```
$ mpirun -np 2 -m mpihosts ./cpi
Process 0 on hostname1
  Process 1 on hostname2
  pi is approximately 3.1416009869231241,
Error is 0.0000083333333309
wall clock time = 0.000149
```

Here `./cpi` designates the executable of the example program in the working directory. The `-np` parameter to `mpirun` defines the number of processes to be used in the parallel computation. Now try it with four processes:

```
$ mpirun -np 4 -m mpihosts ./cpi
Process 3 on hostname1
Process 0 on hostname2
Process 2 on hostname2
Process 1 on hostname1
pi is approximately 3.1416009869231249,
Error is 0.0000083333333318
wall clock time = 0.000603
```

If you run the program several times with the same value of the `-np` parameter, you may get the output lines in different orders. This is because they are issued by independent asynchronous processes, so their order is non-deterministic.

The number of processes can be greater than the number of nodes. In this four-process example, the `mpihosts` file listed only two hosts, hostname1 and hostname2. Generally, `mpirun` will try to distribute the specified number of processes evenly among the nodes listed in the `mpihosts` file, but if the number of processes exceeds the number of nodes listed in the `mpihosts` file, then some nodes will be assigned more than one instance of the program.

Up to a limit, the number of processes can even exceed the total number of processors on the specified set of nodes, although it is usually detrimental to performance to have more than one node program per processor. This limit is eight processes per node with the QHT7140, and four processes per node with the QLE7140. See section 3.5.9 for further discussion.

Details on alternate means of specifying the `mpihosts` file are given in section 3.5.6. Further information on the `mpirun` options are in section 3.5.5, section 3.5.9 and section 3.5.10.

3.3.2 Examples Using Other Languages

This section gives more examples, one for Fortran77, one for Fortran90, and one for C++. Fortran95 usage will be similar to that for Fortran90.

`fpi.f` is a Fortran77 program that computes pi in a way similar to `cpi.c`. Compile and link it with:

```
$ mpif77 -o fpi3 fpi3.f
```

and run it with:

```
$ mpirun -np 2 -m mpihosts ./fpi3
```

`pi3f90.f90` in the same directory is a Fortran90 program that does essentially the same computation. Compile and link it with:

```
$ mpif90 -o pi3f90 pi3f90.f90
```

and run it with:

```
$ mpirun -np 2 -m mpihosts ./pi3f90
```

The C++ program `hello++.cc` is a parallel processing version of the traditional "Hello, World" program. Notice that this version makes use of the external C bindings of the MPI functions if the C++ bindings are not present.

Compile it:

```
$ mpicxx -o hello hello++.cc
```

and run it:

```
$ mpirun -np 10 -m mpihosts ./hello
Hello World! I am 9 of 10
Hello World! I am 2 of 10
Hello World! I am 4 of 10
Hello World! I am 1 of 10
Hello World! I am 7 of 10
Hello World! I am 6 of 10
Hello World! I am 3 of 10
Hello World! I am 0 of 10
Hello World! I am 5 of 10
Hello World! I am 8 of 10
```

Each of the scripts invokes the PathScale compiler for the respective language and the linker. See section 3.5.3 for an example of how to use the `gcc` compiler. The use of `mpirun` is the same for programs in all languages.

3.4 Configuring MPI Programs for InfiniPath MPI

When configuring an MPI program (generating header files and/or Makefiles), for InfiniPath MPI, you will usually need to specify `mpicc`, `mpif90`, etc. as the compiler, rather than `pathcc`, `pathf90`, etc.

Typically this is done with commands similar to these (this assumes you are using `sh` or `bash` as your shell):

```
$ export CC=mpicc
$ export CXX=mpicxx
$ export F77=mpif77
$ export F90=mpif90
$ export F95=mpif95
```

The shell variables will vary with the program being configured, but these examples show frequently used variable names. Users of `csh` would instead use commands similar to:

```
$ setenv CC mpicc
```

You may need to instead pass arguments to `configure` directly, in a fashion similar to this:

```
$ ./configure -cc=mpicc -fc=mpif77 -c++=mpicxx
-c++linker=mpicxx
```

Sometimes you may need to edit a Makefile to achieve this result, adding lines similar to:

```
CC=mpicc
F77=mpif77
F90=mpif90
F95=mpif95
CXX=mpicxx
```

In some cases, the configuration process may specify the linker. It is recommended that the linker be specified as `mpicc`, `mpif90`, etc. in these cases. That will automatically include the correct flags and libraries, rather than trying to configure to pass the flags and libraries explicitly. For example:

```
LD=mpicc
LD=mpif90
```

These scripts pass appropriate options to the various compiler passes to include header files, required libraries, etc. While the same effect can be achieved by passing the arguments explicitly as flags, the required arguments may vary from release to release, so it's good practice to use the provided scripts.

3.5 InfiniPath MPI Details

This section gives more details on the use of InfiniPath MPI. We assume the reader has some familiarity with standard MPI. See the references in appendix D.1. This implementation does include the `man` pages from the MPICH implementation for the numerous MPI functions.

3.5.1 Configuring for `ssh` Using `ssh-agent`

The command `mpirun` can be run on the front end or on any other node. In InfiniPath MPI, this uses the secure shell command `ssh` to start instances of the given MPI program on the remote compute nodes. To use `ssh`, the user must have generated RSA or DSA keys, public and private. The public keys must be distributed to all the compute nodes so that connections to the remote machines can be established without supplying a password. Each user can accomplish this through use of the `ssh-agent`. `ssh-agent` is a daemon that caches decrypted private keys. You use `ssh-add` to add your private keys to `ssh-agent`'s cache. When `ssh` establishes a new connection, it communicates with `ssh-agent` in order to acquire these keys, rather than prompting you for a passphrase.

The process is shown in the following steps:

1. Create a key pair. Use the default file name, and be sure to enter a passphrase.

   ```
   $ ssh-keygen -t rsa
   ```

2. Enter a passphrase for your key pair when prompted. Note that the key agent does not survive X11 logout or system reboot:

   ```
   $ ssh-add
   ```

3. This tells `ssh` that your key pair should let you in:

   ```
   $ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
   ```

   edit `~/.ssh/config` so that it reads like this:

   ```
   Host*
   ForwardAgent    yes
   ForwardX11    yes
   CheckHostIP    no
   StrictHostKeyChecking    no
   ```

   This forwards the key agent requests back to your desktop. When you log into a front end node, you can ssh to compute nodes without passwords.

4. Start ssh-agent by adding the following line to your ~/.bash_profile (or equivalent in another shell):

   ```
   eval `ssh-agent`
   ```

   Use back-quotes rather than normal single-quotes. Programs started in your login shell will then be able to locate `ssh-agent` and query it for keys.

5. Finally, test by logging into the front end node, and from the front end node to a compute node as follows:

   ```
   $ ssh frontend_node_name
   $ ssh compute_node_name
   ```

For more information, see the man pages for `ssh(1)`, `ssh-keygen(1)`, `ssh-add(1)`, and `ssh-agent(1)`.

Alternatively, the cluster administrator can accomplish this for all users through the `shosts.equiv` mechanism, as described in section 2.9 .

3.5.2 Compiling and Linking

These scripts invoke the compiler and linker for programs in each of the respective languages, and take care of referring to the correct include files and libraries in each case.

```
mpicc
mpicxx
mpif77
mpif90
mpif95
```

On x86_64, by default these call the PathScale compiler and linker. To use other compilers, see section 3.5.3.

NOTE: The 2.x PathScale compilers aren't currently supported on systems that use the GNU 4.x compiler and environment. This includes FC4, FC5 and SLES10. For suggestions on how to work around this issue, see section 3.5.4. The 3.0 compiler release will support the GNU 4.x compiler environment.

These scripts all provide the following command line options:

`-help`
Provides help.

`-show`
Lists each of the compiling and linking commands that would be called without actually calling them.

`-echo`
Gets verbose output of all the commands in the script.

`-compile_info`
Shows how to compile a program.

`-link_info`
Shows how to link a program.

Further, each of these scripts allows a command line option for specifying the use of a different compiler/linker as an alternative to the PathScale Compiler Suite. These are described in the next section.

Most other command line options are passed on to the invoked compiler and linker. The PathScale compiler and the usual alternatives all admit numerous command line options. See the PathScale compiler documentation and the `man` pages for `pathcc` and `pathf90` for complete information on its options. See the corresponding documentation for any other compiler/linker you may call for its options.

3.5.3
To Use Another Compiler

In addition to the PathScale Compiler Suite, InfiniPath MPI supports a number of other compilers. These include PGI 5.2 and 6.0, Intel 9.0, the GNU gcc 3.3.x, 3.4.x, and 4.0.x compiler suites and gfortran. The IBM XL family of compilers is also supported on ppc64 (Power) systems.

NOTE: The 2.x PathScale compilers aren't currently supported on systems that have the GNU 4.x compilers and compiler environment (header files and libraries). This includes Fedora Core 4, Fedora Core 5, SUSE 10, and SLES 10. To run on those distributions, you can compile your application on a system that does support the PathScale compiler. Then you can run the executable on one of the systems that uses the GNU 4.x compiler and environment. For more information on setting up for cross-compilation, see section 3.5.4. The GNU 4.x compiler environment will be supported by the PathScale Compiler Suite 3.0 release.

NOTE: In addition, `gfortran` is not currently supported on Fedora Core 3, as it has dependencies on the GNU 4.x suite.

The following example shows how to use `gcc` for compiling and linking MPI programs in C:

```
$ mpicc -cc=gcc .......
```

To use `gcc` for compiling and linking C++ programs use:

```
$ mpicxx -CC=g++ .......
```

To use gcc for compiling and linking Fortran77 programs use:

```
$ mpif77 -fc=g77 .......
```

In each case, `.....` stands for the remaining options to the `mpicxx` script, the options to the compiler in question, and the names of the files it is to operate upon.

Using the same pattern you will see that this next example is similar, except that it uses the PGI (pgcc) compiler for compiling and linking in C:

```
$ mpicc -cc=pgcc .....
```

To use PGI for Fortran90/Fortran95 programs, use:

```
$ mpif90 -f90=pgf90 .....
$ mpif95 -f95=pgf95 .....
```

This example uses the Intel C compiler (icc):

```
$ mpicc -cc=icc .....
```

To use the Intel compiler for Fortran90/Fortran95 programs, use:

```
$ mpif90 -f90=ifort .....
$ mpif95 -f95=ifort .....
```

Usage for other compilers will be similar to the examples above, substituting the options following `-cc`, `-CC`, `-f77`, `-f90`, or `-f95`. Consult the documentation for specific compilers for more details.

Also, use `mpif77`, `mpif90`, or `mpif95` for linking, otherwise you may have problems with `.true.` having the wrong value. If you are not using the provided scripts for linking, you should link a sample program using the `-show` option as a test, to see what libraries to add to your link line. Some examples follow.

For Fortran90 programs:

```
$ mpif90 -f90=pgf90 -show pi3f90.f90 -o pi3f90
pgf90 -I/usr/include/mpich/pgi5/x86_64 -c -I/usr/include
pi3f90.f90 -c
pgf90 pi3f90.o -o pi3f90 -lmpichf90 -lmpich -lmpichabiglue_pgi5
```

Fortran95 programs will be similar to the above.

For C programs:

```
$ mpicc -cc=pgcc -show cpi.c
pgcc -c cpi.c
pgcc cpi.o -lmpich -lpgftnrtl -lmpichabiglue_pgi5
```

3.5.3.1
Compiler and Linker Variables

If you use environment variables (e.g., $MPICH_CC) to select which compiler `mpicc`, et al. should use, the scripts will also set the matching linker variable (e.g. $MPICH_CLINKER), if not already set. If both the environment variable and command line options are used (e.g, `-cc=gcc`), the command line variable is used.

If both the compiler and linker variables are set, and they do not match for the compiler you are using, it is likely that the MPI program will fail to link, or if it links, it may not execute correctly. For a sample error message, please see section C.8.3 in the Troubleshooting chapter.

3.5.4
Cross-compilation Issues

The 2.x PathScale compilers aren't currently supported on systems that use the GNU 4.x compilers and compiler environment (header files and libraries). This includes Fedora Core 4, Fedora Core 5 and SLES 10. The GNU 4.x environment will be supported in the PathScale Complier Suite 3.0 release.

The current workaround for this is to compile on a supported and compatible distribution, then run the executable on one of the systems that uses the GNU 4.x compilers and environment.

- To run on FC4 or FC5, install FC3 or RHEL4/CentOS on your build machine. Compile your application on this machine.
- To run on SLES 10, install SUSE 9.3 on your build machine. Compile your application on this machine.
- Alternatively, `gcc` can be used as the default compiler. Set `mpicc -cc=gcc` as described in section 3.5.3.

Next, on the machines in your cluster on which the job will run, install compatibility libraries. These libraries include C++ and Fortran compatibility shared libraries and libgcc.

For an FC4 or FC5 system, you would need:

- pathscale-compilers-libs (for FC3)
- compat-gcc-32
- compat-gcc-32-g77
- compat-libstdc++-33

On a SLES 10 system, you would need:

- compat-libstdc++ (for FC3)
- compat-libstdc++5 (for SLES 10)

Depending upon the application, you may need to use the `-W1,-Bstatic` option to use the static versions of some libraries.

3.5.5
Running MPI Programs

The script `mpirun` lets you start your parallel MPI program on a set of nodes in a cluster. It starts, monitors, and terminates the node programs. `mpirun` uses ssh (secure shell) to log in to individual cluster machines and prints any messages that the node program prints on `stdout` or `stderr` on the terminal from which mpirun is invoked. It is therefore usually desirable to either configure all cluster nodes to use `shosts.equiv` (see section 2.9), or for users to use `ssh-agent` (see section 3.5.1) in order to allow MPI programs to be run without requiring that a password be entered for each node in the job.

The general syntax is:

```
$ mpirun [mpirun_options...] program-name [program options]
```

`program-name` will generally be the pathname to the executable MPI program. If the MPI program resides in the current directory and the current directory is not in your search path, then *`program-name`* must begin with ‘./’, such as:

./program-name

Unless you want to run only one instance of the program, you need to use the `-np` option, as in:

`$` **`mpirun -np`** ***`n [other options] program-name`***

This spawns `n` instances of *`program-name`*. We usually call these instances *node programs*.

Each node program is started as a process on one node. While it is certainly possible for a node program to fork child processes, the children must not themselves call MPI functions.

`mpirun` monitors the parallel MPI job, terminating when all the node programs in that job exit normally, or if any of them terminates abnormally.

Killing the `mpirun` program kills all the processes in the job. Use `Ctrl-C` to do this.

3.5.6 The `mpihosts` File

As noted in section 3.3 you have created an `mpihosts` file (also called a *machines file, node file*, or *hosts file*) in your current working directory. This file names the nodes on which the node programs may run. The `mpihosts` file contains lines of the form:

```
hostname[:p]
```

The optional part `:p` specifies the number of node programs that can be spawned on that node. When not specified, the default value is 1. The two supported formats for the `mpihosts` file are:

```
hostname1
hostname2
...
```

or

```
hostname1:process_count
hostname2:process_count
...
```

In the first format, if the -np count is greater than the number of lines in the machine file, the hostnames will be repeated (in order) as many times as necessary for the requested number of node programs.

In the second format `process_count` can be different for each host, and is normally the number of available processors on the node. Up to `process_count` node programs will be started on that host before using the next entry in the `mpihosts` file. If the full `mpihosts` file is processed, and there are still more processes requested, processing starts again at the start of the file.

You have several alternative ways of specifying the mpihosts file.

1. First, as noted in section 3.3.1, you can use the command line option -m:

   $ **mpirun -np *n* -m mpihosts *[other options] program-name***

   In this case, if the named file cannot be opened, the MPI job fails.
2. If the -m option is omitted, mpirun checks the environment variable MPIHOSTS for the name of the MPI hosts file. If this variable is defined and the file it names cannot be opened, then the MPI job fails.
3. In the absence of both the -m option and the MPIHOSTS environment variable, mpirun uses the file ./mpihosts, if it exists.
4. If none of these three methods of specifying the hosts file are used, mpirun looks for the file ~/.mpihosts.

If you are working in the context of a batch queuing system, it may provide you with a job submission script that generates an appropriate mpihosts file.

3.5.7
Console I/O in MPI Programs

mpirun sends any output printed to stdout or stderr by any node program to the terminal. This output is line-buffered, so the lines output from the various node programs will be non-deterministically interleaved on the terminal. Using the -l option to mpirun will label each line with the rank of the node program that produced it.

Node programs do not normally use interactive input on stdin, and by default, stdin is bound to /dev/null. However, for applications that require standard input redirection, InfiniPath MPI supports two mechanisms to redirect stdin:

1. If mpirun is run from the same node as MPI rank 0, all input piped to the mpirun command will be redirected to rank 0.
2. If mpirun is not run from the same node as MPI rank 0 or if the input must be redirected to all or specific MPI processes, the -stdin option can be used to redirect a file as standard input to all nodes or to a particular node as specified by the -stdin-target option.

3.5.8
Environment for Node Programs

The environment variables existing on the front end node on which you run mpirun are *not* propagated to the other nodes. You can set the paths, such as LD_LIBRARY_PATH, and other environment variables for the node programs through the use of the -rcfile option of mpirun:

$ **mpirun -np *n* -m mpihosts -rcfile *mpirunrc program***

In the absence of this option, mpirun checks to see if a file called $HOME/.mpirunrc exists in the user's home directory. In either case, the file is sourced by the shell on each node at time of startup of the node program.

The .mpirunrc should not contain any interactive commands. It may contain commands that output on stdout or stderr.

When you do not specify an mpirunrc file, either through the option or the default ~/.mpirunrc, the environment on each node is whatever it would be for the user's login via ssh, unless you are using MPD. (See section 3.8.)

There is a global options file that can be used for mpirun arguments. The default location of this file is:

/opt/infinipath/etc/mpirun.defaults

You can use an alternate file by setting the environment variable $PSC_MPIRUN_DEFAULTS_PATH. See the mpirun man page for more information.

3.5.8.1
Environment for Multiple Versions of InfiniPath or MPI

The variable INFINIPATH_ROOT sets a root prefix for all Infinipath-related paths. It is used by mpirun to try to find the mpirun-ipath-ssh executable, and it is also used to set up LD_LIBRARY_PATH for new programs. This allows multiple versions of the InfiniPath software releases to be installed on some or all nodes, as well as having InfiniPath MPI and other version(s) of MPI installed at the same time. It may be set in the environment, in mpirun.defaults, or in an rcfile (such as .mpirunrc, .bashrc or .cshrc) that will be invoked on remote nodes.

If you have used the --prefix argument with the rpm command to change the root prefix for the InfiniPath installation, then set INFINIPATH_ROOT to the same value.

If INFINIPATH_ROOT is not set, the normal PATH is used unless mpirun is invoked with a full pathname.

NOTE: mpirun-ssh was renamed mpirun-ipath-ssh so as to avoid name collisions with other MPI implementations.

3.5.9
Multiprocessor Nodes

Another command line option, -ppn, instructs mpirun to assign a fixed number *p* of node programs to each node, as it distributes the n instances among the nodes:

$ **mpirun -np *n* -m mpihosts -ppn *p* *program-name***

This option overrides the :p specifications, if any, in the lines of the MPI hosts file.

As a general rule, `mpirun` tries to distribute the *n* node programs among the nodes without exceeding on any node the maximum number of instances specified by the `:p` option. The value of the `:p` option is specified by either the `-ppn` command line option or in the `mpihosts` file.

NOTE: When the `-np` value is larger than the number of nodes in the mpi hostsfile times the `-ppn` value, `mpirun` will cycle back through the hostsfile, assigning additional node programs per host.

Normally, the number of node programs should be no larger than the number of processors on the node, at least not for compute-bound problems. In the current implementation of the InfiniPath interconnect, no node can run more than eight node programs.

For improved performance, InfiniPath MPI uses shared memory to pass messages between node programs running on the same host.

3.5.10 `mpirun` Options

Here is a list summarizing the most commonly used options to `mpirun`. See the man page for a more complete listing.

`-np` *`np`*
Number of processes to spawn.

`-ppn` *`processes-per-node`*
Create up to specified number of processes per node.

`-machinefile` *`filename`*`, -m` *`filename`*
Machines (mpihosts) file, the list of hosts to be used for this job.
Default: `$MPIHOSTS`, then `./mpihosts`, then `~/.mpihosts`

`-M`
Print a formatted list of MPI-level stats of interest for the MPI programmer

`-verbose`
Print diagnostic messages from `mpirun` itself. Can be useful in troubleshooting
Default: Off

`-version, -v`
Print MPI version. Default: Off

`-help, -h`
Print `mpirun` help message. Default: Off

`-rcfile` *`node-shell-script`*
Startup script for setting environment on nodes.
Default: `$HOME/.mpirunrc`

`-in-xterm`
Run each process in an xterm window. Default: Off

`-display` *`X-server`*
X Display for xterm. Default: None

`-debug`
Run each process under debugger in an xterm window. Uses `gdb` by default.
Default: Off

Set `-q 0` when using `-debug`.

`-debug-no-pause`
Like debug, except doesn't pause at beginning. Uses `gdb` by default.
Default: Off

`-debugger` *`gdb`*|*`pathdb`*|*`strace`*
Which debugger to use.Default: `gdb`

`-psc-debug-level` *`mask`*
Controls the verbosity of MPI and InfiniPath debug messages for node programs. A synonym is `-d` *`mask.`*
Default: 1

`-nonmpi`
Run a non-MPI program. Required if the node program makes no MPI calls. Default: Off

`-quiescence-timeout, seconds`
Wait time in seconds for quiescence (absence of MPI communication) on the nodes. Useful for detecting deadlocks. 0 disables quiescence detection.
Default: 900

`-disable-mpi-progress-check`
This option disables MPI communication progress check, without disabling the `ping` reply check. Default: `Off`.

`-l`
Label each line of output on `stdout` and `stderr`with the rank of the MPI process which produces the output.

`-labelstyle string`
Specify the label that is prefixed to error messages and statistics. Process rank is the default prefix.

`-stdin filename`
Filename that should be fed as stdin to the node program. Default: `/dev/null`

`-stdin-target 0..np-1 | -1`
Process rank that should receive the file specified with the `-stdin` option. -1 means all ranks. Default: -1

`-wdir path-to-working_dir`
Sets the working directory for the node program.
Default: `-wdir current-working-dir`

`-print-stats`
Causes each node program to print various MPI statistics to `stderr` on job termination. Can be useful for troubleshooting. Default: `off`. For details, see appendix C.8.13.

`-statsfile file-prefix`
Specifies alternate file to receive the output from the `-print-stats` option.
Default: `stderr`

3.6 Using Other MPI Implementations

Support for multiple MPI implementations has been added. You can use a different version of MPI and achieve the high-bandwidth and low-latency performance that it is standard with InfiniPath MPI.

The currently supported implementations are HP-MPI, OpenMPI and Scali.

These MPI implementations will run on multiple interconnects, and have their own mechanisms for selecting which one you will run on. Please see the documentation provided with the version of MPI that you wish to use.

If you have downloaded and installed another MPI implementation, you will need to set your `PATH` up to pick up the version of MPI you wish to use.

You will also need to set `LD_LIBRARY_PATH`, both in your local environment and in an rcfile (such as `.mpirunrc`, `.bashrc` or `.cshrc`) that will be invoked on remote nodes. See section 3.5.8 and section 3.5.3.1 for information on setting up your environment and section C.8.6 for information on setting your run-time library path. See also section C.8.7 for information on run time errors that may occur if there are MPI version mismatches.

3.7 MPI Over uDAPL

Some MPI implementations can be run over uDAPL. uDAPL is the user mode version of the Direct Access Provider Library (DAPL). Examples of such MPI implementations are Intel MPI and one option on OpenMPI.

If you are running such an MPI implementation, the `rdma_cm` and `rdma_ucm` modules will need to be loaded. To test these modules, use these commands (as root):

```
# modprobe rdma_cm
# modprobe rdma_ucm
```

To ensure that the modules are loaded whenever the driver is loaded, add `rdma_cm` and `rdma_ucm` to the `OPENFABRICS_MODULES` assignment in `/etc/sysconfig/infinipath`.

3.8 MPD

MPD is an alternative to `mpirun` for launching MPI jobs. It is described briefly in the following sections.

3.8.1 MPD Description

The Multi-Purpose Daemon (MPD) was developed by Argonne National Laboratory (ANL), as part of the MPICH-2 system. While the ANL MPD had certain advantages over the use of their `mpirun` (faster launching, better cleanup after crashes, better tolerance of node failures), the InfiniPath `mpirun` offers the same advantages.

The disadvantage of MPD is reduced security, since it does not use `ssh` to launch node programs. It is also a little more complex to use than `mpirun` because it requires starting a ring of MPD daemons on the nodes. Therefore, most users should use the normal `mpirun` mechanism for starting jobs as described in the previous chapter. However, for users who wish to use MPD, it is included in the InfiniPath software.

3.8.2 Using MPD

To start an MPD environment, use the `mpdboot` program. You must provide `mpdboot` with a file listing the machines on which to run the `mpd` daemon. The format of this file is the same as for the mpihosts file in the `mpirun` command.

Here is an example of how to run mpdboot:

```
$ mpdboot -f hostsfile
```

After `mpdboot` has started the MPD daemons, it will print a status message and drop you into a new shell.

To leave the MPD environment, exit from this shell. This will terminate the daemons.

To run an MPI program from within the MPD environment, use the `mpirun` command. You do not need to provide a `mpihosts` file or a count of CPUs; by default, `mpirun` will use all nodes and CPUs available within the MPD environment.

To check the status of the MPD daemons, use the `mpdping` command

NOTE: To use MPD, the software package `mpi-frontend-2.0*.rpm` must be installed on all nodes. See the *InfiniPath Install Guide* for more details on software installation..

3.9 File I/O in MPI

File I/O in MPI is discussed briefly in the following two sections.

3.9.1 Linux File I/O in MPI Programs

MPI node programs are Linux programs, which can do file I/O to local or remote files in the usual ways through APIs of the language in use. Remote files are

accessed via some network file system, typically NFS. Parallel programs usually need to have some data in files to be shared by all of the processes of an MPI job. Node programs may also use non-shared, node-specific files, such as for scratch storage for intermediate results or for a node's share of a distributed database.

There are different styles of handling file I/O of shared data in parallel programming. You may have one process, typically on the front end node or on a file server, which is the only process to touch the shared files, and which passes data to and from the other processes via MPI messages. On the other hand, the shared data files could be accessed directly by each node program. In this case, the shared files would be available through some network file support, such as NFS. Also, in this case, the application programmer would be responsible for ensuring file consistency, either through proper use of file locking mechanisms offered by the OS and the programming language, such as `fcntl` in C, or by the use of MPI synchronization operations.

3.9.2 MPI-IO with ROMIO

MPI-IO is the part of the MPI2 standard, supporting collective and parallel file IO. One of the advantages in using MPI-IO is that it can take care of managing file locks in case of file data shared among nodes.

InfiniPath MPI includes ROMIO version 1.2.6, a high-performance, portable implementation of MPI-IO from Argonne National Laboratory. ROMIO includes everything defined in the MPI-2 I/O chapter of the MPI-2 standard except support for file interoperability and user-defined error handlers for files. Of the MPI-2 features, InfiniPath MPI includes only the MPI-IO features implemented in ROMIO version 1.2.6 and the generalized MPI_Alltoallw communication exchange. See the ROMIO documentation in http://www.mcs.anl.gov/romio for details.

3.10 InfiniPath MPI and Hybrid MPI/OpenMP Applications

InfiniPath MPI supports hybrid MPI/OpenMP applications provided that MPI routines are only called by the master OpenMP thread. This is called the *funneled thread model*. Instead of MPI_Init/MPI_INIT (for C/C++ and Fortran respectively), the program can call MPI_Init_thread/MPI_INIT_THREAD to determine the level of thread support and the value MPI_THREAD_FUNNELED will be returned.

To use this feature the application should be compiled with both OpenMP and MPI code enabled. To do this, use the `-mp` flag on the `mpicc` compile line.

As mentioned above, MPI routines must only be called by the master OpenMP thread. The hybrid executable is executed as usual using `mpirun`, but typically only one MPI process is run per node and the OpenMP library will create additional threads to utilize all CPUs on that node. If there are sufficient CPUs on a node, it

may be desirable to run multiple MPI processes and multiple OpenMP threads per node.

The number of OpenMP threads is typically controlled by the OMP_NUM_THREADS environment variable in the `.mpirunrc` file. This may be used to adjust the split between MPI processes and OpenMP threads. Usually the number of MPI processes (per node) times the number of OpenMP threads will be set to match the number of CPUs per node. An example case would be a node with 4 CPUs, running 1 MPI process and 4 OpenMP threads. In this case, OMP_NUM_THREADS is set to 4. OMP_NUM_THREADS is on a per-node basis.

See the section 3.5.8 for information on setting environment variables.

The MPI_THREAD_SERIALIZED and MPI_THREAD_MULTIPLE models are not yet supported.

NOTE: If there are more threads than CPUs, then both MPI and OpenMP performance can be significantly degraded due to over-subscription of the CPUs.

3.11 Debugging MPI Programs

Debugging parallel programs is substantially more difficult than debugging serial programs. Thoroughly debugging the serial parts of your code before parallelizing is good programming practice.

3.11.1 MPI Errors

Almost all MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error code; as the function return value in C functions or as the last argument in a Fortran subroutine call. Before the value is returned, the current MPI error handler is called. By default, this error handler aborts the MPI job. Therefore you can get information about MPI exceptions in your code by providing your own handler for `MPI_ERRORS_RETURN`. See the `man` page for `MPI_Errhandler_set` for details.

NOTE: MPI does not guarantee that an MPI program can continue past an error.

See the standard MPI documentation referenced in appendix D for details on the MPI error codes.

3.11.2 Using Debuggers

The InfiniPath software supports the use of multiple debuggers, including `pathdb`, `gdb`, and the system call tracing utility `strace`. These debuggers let you set breakpoints in a running program, and examine and set its variables.

Symbolic debugging is easier than machine language debugging. To enable symbolic debugging you must have compiled with the `-g` option to `mpicc` so that the compiler will have included symbol tables in the compiled object code.

To run your MPI program with a debugger use the `-debug` or `-debug-no-pause` and `-debugger` options to `mpirun`. See the `man` pages to `pathdb`, `gdb`, and `strace` for details. When you run under a debugger, you get an xterm window on the front end machine for each node process. Thus, you can control the different node processes as desired.

To use `strace` with your MPI program, the syntax would be:

```
$ mpirun -np n -m mpihosts strace program-name
```

The following features of InfiniPath MPI especially facilitate debugging:

- Stack backtraces are provided for programs that crash.
- `-debug` and `-debug-no-pause` options are provided for `mpirun` that can make each node program start with debugging enabled. The `-debug` option allows you to set breakpoints, and start running programs individually. The `-debug-no-pause` option allows postmortem inspection. Note that you should set `-q 0` when using `-debug`.
- Communication between `mpirun` and node programs can be printed by specifying the `mpirun -verbose` option.
- MPI implementation debug messages can be printed by specifying the `mpirun -psc-debug-level` option. Note that this can substantially impact the performance of the node program.
- Support is provided for progress timeout specifications, deadlock detection, and generating information about where a program is stuck.
- Several misconfigurations (such as mixed use of 32-bit/64-bit executables) are detected by the runtime.
- A formatted list containing information useful for high-level MPI application profiling is provided by using the `-print-stats` option with `mpirun`. Statistics include minimum, maximum and median values for message transmission protocols as well as a more detailed information for expected and unexpected message reception. See appendix C.8.13 for more information and a sample output listing.

3.12 InfiniPath MPI Limitations

The current version of InfiniPath MPI has the following limitations:

By default, at most eight node programs per node with the QHT7140 are allowed, and at most four node programs per node with the QLE7140. The error message when this limit is exceeded is:

```
No ports available on /dev/ipath
```

NOTE: If port sharing is enabled, this limit is raised to 16 and 8 respectively. To enable port sharing, set PSM_SHAREDPORTS=1 in your environment

There are no C++ bindings to MPI -- use the extern C MPI function calls.

In MPI-IO file I/O calls in the Fortran binding, offset or displacement arguments are limited to 32 bits. Thus, for example, the second argument of `MPI_File_seek` must lie between -2^{31} and $2^{31}-1$, and the argument to `MPI_File_read_at` must lie between 0 and $2^{32}-1$.

Appendix A
Benchmark Programs

Several MPI performance measurement programs are installed from the `mpi-benchmark` RPM. This Appendix describes these useful benchmarks and how to run them. These programs are based on code from the group of Dr. Dhabaleswar K. Panda at the Network-Based Computing Laboratory at the Ohio State University. For more information, see:

http://nowlab.cis.ohio-state.edu/

These programs allow you to measure the MPI latency and bandwidth between two or more nodes in your cluster. Both the executables, and the source for those executables, are shipped. The executables are shipped in the `mpi-benchmark` RPM, and installed under `/usr/bin`. The source is shipped in the `mpi-devel` RPM and installed under `/usr/share/mpich/examples/performance`.

The examples given below are intended only to show the syntax for invoking these programs and the meaning of the output. They are NOT representations of actual InfiniPath performance characteristics.

A.1 Benchmark 1: Measuring MPI Latency Between Two Nodes

In the MPI community, *latency for a message of given size* is defined to be the time difference between a node program's calling `MPI_Send` and the time that the corresponding `MPI_Recv` in the receiving node program returns. By *latency*, alone without a qualifying message size, we mean the latency for a message of size zero. This latency represents the minimum overhead for sending messages, due both to software overhead and to delays in the electronics of the fabric. To simplify the timing measurement, latencies are usually measured with a *ping-pong* method, timing a round-trip and dividing by two.

The program `osu_latency`, from Ohio State University, measures the latency for a range of messages sizes from 0 to 4 megabytes. It uses a ping-pong method, in which the rank 0 process initiates a series of sends and the rank 1 process echoes them back, using the blocking MPI send and receive calls for all operations. Half the time interval observed by the rank 0 process for each such exchange is a measure of the latency for messages of that size, as defined above. The program uses a loop, executing many such exchanges for each message size, in order to get an average. It defers the timing until the message has been sent and received a number of times, in order to be sure that all the caches in the pipeline have been filled.

This benchmark always involves just two node programs. You can run it with the command:

```
$ mpirun -np 2 -ppn 1 -m mpihosts osu_latency
```

The `-ppn 1` option is needed to be certain that the two communicating processes are on different nodes. Otherwise, in the case of multiprocessor nodes, `mpirun` might assign the two processes to the same node, and so the result would not be indicative of the latency of the InfiniPath fabric, but rather of the shared memory transport mechanism. Here is what the output of the program looks like:

```
# OSU MPI Latency Test (Version 2.0)
# Size          Latency (us)
0               1.26
1               1.26
2               1.26
4               1.26
8               1.26
16              1.45
32              1.47
64              1.52
128             1.63
256             1.88
512             2.34
1024            3.25
2048            5.13
4096            7.34
8192            11.58
16384           20.25
32768           37.56
65536           78.69
131072          149.84
262144          287.49
524288          565.84
1048576         1119.18
2097152         2220.18
4194304         4424.59
```

The first column gives the message size in bytes, the second gives the average (one-way) latency in microseconds. Again, this example is given to show the syntax of the command and the format of the output, and is not meant to represent actual values that might be obtained on any particular InfiniPath installation.

A.2 Benchmark 2: Measuring MPI Bandwidth Between Two Nodes

The `osu_bw` benchmark is meant to measure the maximum rate at which you can pump data between two nodes. It also uses a ping-pong mechanism, similar to the `osu_latency` code, except in this case, the originator of the messages pumps a number of them (64 in the installed version) in succession using the non-blocking

MPI_Isend function, while the receiving node consumes them as quickly as it can using the non-blocking MPI_Irecv, and then returns a zero-length acknowledgement when all of the set has been received.

You can run this program with:

```
$ mpirun -np 2 -ppn 1 -m mpihosts osu_bw
```

Typical output might look like:

```
# OSU MPI Bandwidth Test (Version 2.0)
# Size          Bandwidth (MB/s)
1               2.250465
2               4.475789
4               8.979276
8               17.952547
16              27.615041
32              52.676363
64              104.704225
128             198.347505
256             335.396929
512             521.273433
1024            829.369420
2048            884.249845
4096            926.723948
8192            934.093084
16384           941.191459
32768           938.179872
65536           945.163478
131072          950.206048
262144          951.938802
524288          952.912385
1048576         953.716825
2097152         953.922714
4194304        954.119999
```

Note that the increase in measured bandwidth with messages size results from the fact that latency's contribution to the measured time interval becomes relatively smaller.

A.3 Benchmark 3: Messaging Rate Microbenchmarks

`mpi_multibw` is the microbenchmark used to highlight QLogic's messaging rate results. This benchmark is a modified form of the OSU NOWlab's `osu_bw` benchmark (as shown in the example above). It has been enhanced with the following additional functionality:

- Messaging rate reported as well as bandwidth
- N/2 dynamically calculated at end of run
- Allows user to run multiple processes per node and see aggregate bandwidth and messaging rates

The benchmark has been updated with code to dynamically determine which processes are on which host. This is an example showing the type of output you will see when you run `mpi_multibw`:

```
$ mpirun -np 8 ./mpi_multibw
```

This will run on four processes per node. Typical output might look like:

```
# PathScale Modified OSU MPI Bandwidth Test
(OSU Version 2.2, PathScale $Revision: 1.1 $)
# Running on 4 procs per node
# Size        Aggregate Bandwidth    (MB/s) Messages/s
1             8.150462               8150461.697283
2             16.693747              8346873.631841
4             33.086567              8271641.814960
8             66.733488              8341686.016159
16            108.097082             6756067.602089
32            213.733132             6679160.388156
64            389.544112             6086626.744516
128           569.671531             4450558.832794
256           725.826904             2835261.345093
512           839.450014             1639550.807757
1024          913.428063             892019.592596
2048          954.482747             466056.028717
4096          954.474461             233025.991467
8192          954.452712             116510.340772
16384         954.496729             58257.857017
32768         954.547225             29130.469523
65536         949.074433             14481.726572
131072        951.786548             7261.555084
262144        952.193849             3632.331272
524288        952.391830             1816.543255
1048576       952.490368             908.365600
2097152       952.539382             454.206172
4194304       952.566591             227.109573

Searching for N/2 bandwidth. Maximum Bandwidth of 954.547225
MB/s...
Found N/2 bandwidth of 476.993060 MB/s at size 94 bytes
```

This microbenchmark is available and can be downloaded from the QLogic website:

http://www.qlogic.com

A.4 Benchmark 4: Measuring MPI Latency in Host Rings

The program `mpi_latency` can be used to measure latency in a ring of hosts. Its syntax is a bit different from Benchmark 1 in that it takes command line arguments that let you specify the message size and the number of messages over which to average the results. So, for example, if you have a hosts file listing four or more nodes, the command:

```
$ mpirun -np 4 -ppn 1 -m mpihosts mpi_latency 100 0
```

might produce output like this:

```
0       1.760125
```

This indicates that it took an average of 1.76 microseconds per hop to send a zero-length message from the first host, to the second, to the third, to the fourth, and then get replies back in the other direction.

Appendix B

Integration with a Batch Queuing System

Most cluster systems use some kind of batch queuing system as an orderly way to provide users with access to the resources they need to meet their job's performance requirements. One of the tasks of the cluster administrator is to provide means for users to submit MPI jobs through such batch queuing systems. This can take the form of a script, which your users can invoke much as they would invoke `mpirun` to submit their MPI jobs. A sample script is presented in this section.

B.1 A Batch Queuing Script

We give an example of the some of the functions that such a script might perform, in the context of the Simple Linux Utility Resource Manager (SLURM) developed at Lawrence Livermore National Laboratory. These functions assume the use of the `bash` shell. We will call this script `batch_mpirun`. It is provided here:

```
#! /bin/sh
# Very simple example batch script for InfiniPath MPI, using slurm
#   (http://www.llnl.gov/linux/slurm/)
# Invoked as:
#  batch_mpirun #cpus mpi_program_name mpi_program_args ...
#
np=$1 mpi_prog="$2" # assume arguments to script are correct
shift 2 # program args are now $@
eval `srun --allocate --ntasks=$np --no-shell`
mpihosts_file=`mktemp -p /tmp mpihosts_file.XXXXXX`
srun --jobid=${SLURM_JOBID} hostname -s | sort | uniq -c \
   | awk '{printf "%s:%s\n", $2, $1}' > $mpihosts_file
mpirun -np $np -m $mpihosts_file "$mpi_prog" $@
exit_code=$?
scancel ${SLURM_JOBID}
rm -f $mpihosts_file
exit $exit_code
```

In the following sections, setup and the various functions of the script are discussed in further detail.

B.1.1 Allocating Resources

When the `mpirun` command starts, it requires specification of the number of node programs it must spawn (via the `-np` option) and specification of an *`mpihosts`* file listing the nodes on which the node programs may be run. (See section 3.5.8 for more information.) Normally, since performance is usually important, a user might

require that his node program be the only application running on each node CPU. In a typical batch environment, the MPI user would still specify the number of node programs, but would depend on the batch system to allocate specific nodes when the required number of CPUs becomes available. Thus, `batch_mpirun` would take at least an argument specifying the number of node programs and an argument specifying the MPI program to be instantiated. For example,

```
$ batch_mpirun -np n my_mpi_program
```

After parsing the command line arguments, the next step of `batch_mpirun` would be to request an allocation of *n* processors from the batch system. In SLURM, this would use the command

```
eval `srun --allocate --ntasks=$np --no-shell`
```

Make sure to use back-quotes rather than normal single-quotes. `$np` is the shell variable that your script has set from the parsing of its command line options. The `--no-shell` option to `srun` prevents SLURM from starting a subshell. The `srun` command is run with `eval` in order to set the `SLURM_JOBID` shell variable from the output of the `srun` command.

With these specified arguments, the SLURM function `srun` blocks until there are `$np` processors available to commit to the caller. When the requested resources are available, this command opens a new shell and allocates the requested number of processors to it.

B.1.2 Generating the `mpihosts` File

Once the batch system has allocated the required resources, your script must generate a `mpihosts` file, which contains a list of nodes that will be used. To do this, it must find out which nodes the batch system has allocated, and how many processes we can start on each node. This is the part of the script `batch_mpirun` that performs these tasks:

```
mpihosts_file=`mktemp -p /tmp mpihosts_file.XXXXXX`
srun --jobid=${SLURM_JOBID} hostname -s | sort | uniq -c \
   | awk '{printf "%s:%s\n", $2, $1}' > $mpihosts_file
```

The first command creates a temporary hosts file with a random name, and assigns the name to the variable `mpihosts file` it has generated.

The next instance of the SLURM `srun` command runs `hostname -s` once per process slot that SLURM has allocated to us. If SLURM has allocated two slots on one node, we thus get the output of `hostname -s` twice for that node.

The `sort | uniq -c` component tells us the number of times each unique line was printed. The `awk` command converts the result into the `mpihosts` file format used

by `mpirun`. Each line consists of a node name, a colon, and the number of processes to start on that node.

NOTE: This is one of two formats that the file may use. See section 3.5.6 for more information.

B.1.3 Simple Process Management

At this point, your script has enough information to be able to run an MPI program. All that remains is to start the program when the batch system tells us that we can do so, and notify the batch system when the job completes. This is done in the final part of `batch_mpirun`:

```
mpirun -np $np -m $mpihosts_file "$mpi_prog" $@
exit_code=$?
scancel ${SLURM_JOBID}
rm -f $mpihosts_file
exit $exit_code
```

B.1.4 Clean Termination of MPI Processes

The InfiniPath software will normally ensure clean termination of all MPI programs when a job ends, but in some rare circumstances an MPI process will remain alive, and potentially interfere with future MPI jobs. To avoid this problem, the usual solution is to run a script before and after each batch job which kills all unwanted processes. QLogic does not provide such a script, but it is useful to know how to find out which processes on a node are using the InfiniPath interconnect. The easiest way to do this is through use of the `fuser` command, which is normally installed in `/sbin`. Run as root:

```
# /sbin/fuser -v /dev/ipath

/dev/ipath:  22648m 22651m
```

In this example, processes 22648 and 22651 are using the InfiniPath interconnect. It is also possible to use this command (as root):

```
# lsof /dev/ipath
```

This gets a list of processes using InfiniPath. Additionally, to get all processes, including `stats` programs, `ipath_sma`, diags, and others, run the program in this way:

```
# /sbin/fuser -v /dev/ipath*
```

`losf` can also take the same form:

```
# lsof /dev/ipath*
```

The following command will terminate all processes using the InfiniPath interconnect:

```
# /sbin/fuser -k /dev/ipath
```

For more information, see the man pages for `fuser(1)` and `lsof(8)`.

NOTE: Run these commands as root to insure that all processes are reported.

B.2 Lock Enough Memory on Nodes When Using SLURM

This is identical to information provided in appendix C.8.11. It is repeated here for your convenience.

InfiniPath MPI requires the ability to lock (pin) memory during data transfers on each compute node. This is normally done via `/etc/initscript`, which is created or modified during the installation of the `infinipath` RPM (setting a limit of 64MB, with the command "`ulimit -l 65536`").

Some batch systems, such as SLURM, propagate the user's environment from the node where you start the job to all the other nodes. For these batch systems, you may need to make the same change on the node from which you start your batch jobs.

If this file is not present or the node has not been rebooted after the `infinipath` RPM has been installed, a failure message similar to this will be generated:

```
$ mpirun -m ~/tmp/sm -np 2 -mpi_latency 1000 1000000
node-00:1.ipath_update_tid_err: failed: Cannot allocate memory
mpi_latency:
/fs2/scratch/infinipath-build-1.3/mpi-1.3/mpich/psm/src
mq_ips.c:691:
mq_ipath_sendcts: Assertion 'rc == 0' failed. MPIRUN: Node program
unexpectedly quit. Exiting.
```

You can check the `ulimit -l` on all the nodes by running `ipath_checkout`. A warning will be given if `ulimit -l` is less that 4096.

There are two possible solutions to this. If `infinipath` is not installed on the node where you start the job, set this value in the following way. You must be root to set it:

```
# ulimit -l 65536
```

Or, if you have installed `infinipath` on the node, reboot it to insure that `/etc/initscript` is run.

Appendix C
Troubleshooting

This Appendix describes some of the existing provisions for diagnosing and fixing problems. The sections are organized in the following order:

- C.1 “Troubleshooting InfiniPath adapter installation”
- C.2 “BIOS settings”
- C.3 “Software installation issues”
- C.4 “Kernel and initialization issues”
- C.5 “OpenFabrics issues”
- C.6 “System administration troubleshooting”
- C.7 “Performance issues”
- C.8 “InfiniPath MPI troubleshooting”
- C.9 “Useful programs and files for debugging”

C.1 Troubleshooting InfiniPath Adapter Installation

This section lists conditions you may encounter while installing the InfiniPath QLE7140 or QHT7140 adapter, and offers suggestions for working around them.

C.1.1 Mechanical and Electrical Considerations

The LEDs function as link and data indicators once the InfiniPath hardware and software has been installed, the driver has been loaded, and the fabric is being actively managed by a Subnet Manager. The following table shows the possible

states of the LEDs. The green LED will normally illuminate first. The normal state is Green On, Amber On.

Table C-1. LED Link and Data Indicators

LED	Color	Status	
Power	Green	ON Signal detected. Ready to talk to an SM to bring link fully up.	OFF Switch not powered up. Software not installed or started. Loss of signal. Check cabling.
Link	Amber	ON Link configured. Properly connected and ready to receive data and link packets.	OFF SM may be missing. Link may not be configured. Check the connection.

If a node repeatedly and spontaneously reboots when attempting to load the InfiniPath driver, it may be a symptom that its InfiniPath interconnect board is not well seated in the HTX or PCIe slot.

C.1.2
Some HTX Motherboards May Need 2 or More CPUs in Use

Some HTX motherboards may require that 2 or more of the CPUs be in use for the HTX InfiniPath card to be recognized. This is most evident in four-socket motherboards.

C.2
BIOS Settings

This section covers issues related to improper BIOS settings.The two most important settings are:

- ACPI needs to be enabled
- MTRR mapping needs to be set to “Discrete”

If ACPI has been disabled, it may result in initialization problems, as described in appendix C.4.4.

An improper setting for MTRR mapping can result in reduced performance. See appendix C.2.2, appendix C.2.3, and appendix C.2.3 for details.

NOTE: BIOS settings on IBM Blade Center H (Power) systems do not need adjustment.

C.2.1
MTRR Mapping and Write Combining

MTRR (Memory Type Range Registers) is used by the InfiniPath driver to enable write combining to the InfiniPath on-chip transmit buffers. This improves write bandwidth to the InfiniPath chip by writing multiple words in a single bus transaction (typically 64). This applies only to x86_64 systems. To see if is working correctly and to check your bandwidth use this command:

```
$ ipath_pkt_test -B
```

When configured correctly, PCIe InfiniPath will normally report in the range of 1150-1500 MB/s, while HTX InfiniPath cards will normally report in the range of 2300-2650 MB/s.

However, some BIOSes don't have the MTRR mapping option. It may be referred to in a different way, dependent upon chipset, vendor, BIOS, or other factors. For example, it is sometimes referred to as "32 bit memory hole", which should be enabled.

If there is no setting for MTRR mapping or 32 bit memory hole, please contact your system or motherboard vendor and inquire as to how write combining may be enabled.

C.2.2
Incorrect MTRR Mapping

In some cases, the InfiniPath driver may be unable to configure the CPU Write Combining attributes for the QLogic InfiniPath IBA6110. This would normally be seen for a new system, or after the system's BIOS has been upgraded or reconfigured.

If this error occurs, the InfiniPath interconnect will operate, but in a degraded performance mode. Typically the latency will increase to several microseconds, and the bandwidth may decrease to as little as 200 MBytes/sec.

A message similar to this will be printed on the console, and normally to the system log (typically in /var/log/messages):

```
infinipath: mtrr_add(feb00000,0x100000,WC,0) failed (-22)
infinipath: probe of 0000:04:01.0 failed with error -22
```

If you see this error message, you should edit the BIOS setting for MTRR Mapping. The setting should look like this:

```
MTRR Mapping          [Discrete]
```

You can check and adjust the BIOS settings using the BIOS Setup Utility. Check the hardware documentation that came with your system for more information on how to do this. Section C.2.3, below, documents a related issue.

C.2.3
Incorrect MTRR Mapping Causes Unexpected Low Bandwidth

This same MTRR Mapping setting as described in the previous section can also cause unexpected low bandwidth if it is set incorrectly.

The setting should look like this:

```
MTRR Mapping            [Discrete]
```

The MTRR Mapping needs to be set to Discrete if there is 4GB or more memory in the system; it affects where the PCI, PCIe, and HyperTransport i/o addresses (BARs) are mapped. If there is 4GB or more memory in the system, and this is not set to Discrete, you will get very low bandwidth (under 250 MB/sec) on anything that would normally run near full bandwidth. The exact symptoms can vary with BIOS, amount of memory, etc., but typically there will be no errors or warnings.

To check your bandwidth try:

```
$ ipath_pkt_test -B
```

When configured correctly, PICIe InfiniPath will normally report in the range of 1150-1500 MB/s, while HTX InfiniPath cards will normally report in the range of 2300-2650 MB/s. `ipath_checkout` can also be used to check bandwidth.

You can check and adjust the BIOS settings using the BIOS Setup Utility. Check the hardware documentation that came with your system for more information on how to do this.

C.2.4
Change Setting for Mapping Memory

In some cases, on systems with 4GB or more memory on Opteron systems with InfiniPath HTX cards (QHT7040 or QHT7140), and the Red Hat Enterprise Linux 4 release with 2.6.9 Linux kernels, MPI jobs may fail to initialize or may terminate early. This can be worked around by changing the setting for mapping memory around the PCI configuration space ("SoftWare Memory Hole") to "Disabled" in the Chipset, Northbridge screen in the BIOS. This will result in a small loss in usable memory.

C.2.5
Issue with SuperMicro H8DCE-HTe and QHT7040

The InfiniPath card may not be recognized on startup when using the SuperMicro H8DCE-HT-e and the QHT7040 adapter. To fix this problem, the OS selector option in the BIOS should be set for Linux. The option will look like this:

```
OS Installation [Linux]
```

C.3 Software Installation Issues

This section covers issues related to software installation.

C.3.1 OpenFabrics Dependencies

You need to install `sysfsutils` for your distribution before installing the OpenFabrics RPMs, as there are dependencies. If `sysfsutils` has not been installed, you might see error messages like this:

```
error: Failed dependencies:
libsysfs.so.1()(64bit) is needed by
libipathverbs-2.0-1_100.77_fc3_psc.x86_64
 libsysfs.so.1()(64bit) is needed by
libibverbs-utils-2.0-1_100.77_fc3_psc.x86_64
/usr/include/sysfs/libsysfs.h is needed by
libibverbs-devel-2.0-1_100.77_fc3_psc.x86_64
```

Check your distribution's documentation for information about `sysfsutils`.

C.3.2 Install Warning with RHEL4U2

You may see a warning similar to this when installing InfiniPath and OpenFabrics modules on RHEL4U2.

```
infinipath-2.0-7277.1538_fc3_psc
Building and installing InfiniPath and OpenIB modules for
2.6.9-22.ELsmp kernel
Building modules, stage 2.
Warning: could not find versions for .tmp_versions/ib_mthca.mod
```

This warning may be safely ignored.

C.3.3 `mpirun` Installation Requires 32-bit Support

On a 64-bit system, 32-bit `glibc` must be installed before installing the `mpi-frontend-*` RPM. `mpirun`, which is part of the `mpi-frontend-*` RPM, requires 32-bit support.

If 32-bit `glibc` is not installed on a 64-bit system, you will now see an error like this when installing `mpi-frontend`:

```
# rpm -Uv ~/tmp/mpi-frontend-2.0-2250.735_fc3_psc.i386.rpm
error: Failed dependencies:
/lib/libc.so.6 is needed by mpi-frontend-2.0 2250.735_fc3_psc.i386
```

In older distributions, such as RHEL4, the 32-bit `glibc` will be contained in the `libgcc` RPM. The RPM will be named similarly to:

libgcc-3.4.3-9.EL4.i386.rpm

In newer distributions, glibc is an RPM name. The 32-bit glibc will be named similarly to:

glibc-2.3.4-2.i686.rpm

or

glibc-2.3.4-2.i386.rpm

Check your distribution for the exact RPM name.

C.3.4 Installing Newer Drivers from Other Distributions

The driver source now resides in infinipath-kernel. This means that newer drivers can be installed as they become available. Those who wish to install newer drivers, for example, from OFED (Open Fabrics Enterprise Distribution), should be able to do so. However, some extra steps need to be taken in order to install properly.

1. Install all InfiniPath RPMs, including infinipath-kernel. The RPM infinipath-kernel installs into:

   /lib/modules/$(uname -r)/updates

   This should not affect any other installed InfiniPath or OpenFabrics drivers.

2. Reload the InfiniPath and OpenFabrics modules to verify that the installation works by using this command (as root):

   # **/etc/init.d/infinipath restart**

3. Run ipath_checkout or other OpenFabrics test program to verify that the InfiniPath card(s) work properly.

4. Unload the InfiniPath and OpenFabrics modules with the command:

   # **/etc/init.d/infinipath stop**

5. Remove the InfiniPath kernel components with the command:

   $ **rpm -e infinipath-kernel --nodeps**

   The option --nodeps is required because the other InfiniPath RPMs depend on infinipath-kernel.

6. Verify that no InfiniPath or OpenFabrics modules are present in the /lib/modules/$(uname -r)/updates directory.

7. If not yet installed, install the InfiniPath and OpenFabrics modules from your alternate set of RPMs.

8. Reload all modules by using this command (as root):

   # **/etc/init.d/infinipath start**

An alternate mechanism can be used, if provided as part of your alternate installation.

9. Run an OpenFabrics test program, such as `ibstatus`, to verify that your InfiniPath card(s) work correctly.

C.3.5 Installing for Your Distribution

You may be using a kernel which is compatible with one of the supported distributions, but which may not be picked up during `infinipath-kernel` installation. It may also happen when using `make-install.sh` to manually recompile the drivers.

In this case, you can set your distribution with the `$IPATH_DISTRO` override. Run this command before installation, or before running `make-install.sh`. We use the RHEL4 Update 4 distribution as an example in this command for `bash` or `sh` users:

```
$ export IPATH_DISTRO=rhel4_U4
```

The distribution arguments that are currently understood are listed below. They are found in the file `build-guards.sh`.

These are used for RHEL, CentOS(Rocks), and Scientific Linux.

```
rhel4_U2
rhel4_U3
rhel4_U4
```

These are used for SLES, SUSE, and Fedora:

```
sles9
sles10
suse9.3
fc3
fc4
```

`make-install.sh` and `build-guards.sh` are both found in this directory:

```
/usr/src/infinipath/drivers
```

C.4 Kernel and Initialization Issues

Issues that may prevent the system from coming up properly are described.

C.4.1
Kernel Needs CONFIG_PCI_MSI=y

If the InfiniPath driver is being compiled on a machine without CONFIG_PCI_MSI=y configured, you will get a compilation error similar to this:

```
ib_ipath/ipath_driver.c:46:2: #error "InfiniPath driver can only
be used with kernels with CONFIG_PCI_MSI=y"
make[3]: *** [ib_ipath/ipath_driver.o]
Error 1
```

Some kernels, such as some versions of FC4 (2.6.16), have `CONFIG_PCI_MSI=n` as the default. This default may also be introduced with updates to other Linux distributions or local configuration changes. This needs to be changed to CONFIG_PCI_MSI=y in order for the InfiniPath driver to function.

The suggested remedy is to install one of the supported Linux kernels (see section 1.7), or download a patched kernel from the QLogic website.

Pre-built kernels and patches for these distributions are available for download on the website. Please go to:

http://www.qlogic.com

Follow the links to the download page.

NOTE: As of this writing, kernels later than 2.6.16-1.2108_FC4smp on FC4 no longer have this problem.

C.4.2
pci_msi_quirk

A change was made in the kernel.org 2.6.12 kernel that can cause an InfiniPath driver runtime error with the QLE7140. This change is found in most linux distributions with 2.6.12 - 2.6.16 kernels, including Fedora Core 3, Fedora Core 4, and SUSE Linux 10.0. Affected systems are thoses that contain the AMD8131 PCI bridge. Such systems may experience a problem with MSI (Message Signaled Interrupt) that impairs the operation of the InfiniPath QLE7140 adapter. The InfiniPath driver will not be able to configure the InfiniBand link to the Active state. If messages similar to those below are displayed on the console during boot, or are in `/var/log/messages`, then you probably have the problem:

```
PCI: MSI quirk detected. pci_msi_quirk set.
path_core 0000:03:00.0: pci_enable_msi failed: -22, interrupts may
not work
```

Pre-built kernels and patches for these distributions are available for download on the website. Please go to:

http://www.qlogic.com

Follow the links to the downloads page.

NOTE: This problem has been fixed in the 2.6.17 kernel.org kernel.

C.4.3 Driver Load Fails Due to Unsupported Kernel

If you try to load the InfiniPath driver on a kernel that InfiniPath software does not support, the load fails. Error messages similar to this appear:

```
modprobe: error inserting
'/lib/modules/2.6.3-1.1659-smp/kernel/drivers/infiniband/hw/ipath
/ib_ipath.ko': -1 Invalid module format
```

To correct this, install one of the appropriate supported Linux kernel versions as listed in section 2.3.3, then reload the driver.

C.4.4 InfiniPath Interrupts Not Working

The InfiniPath driver will not be able to configure the InfiniPath link to a usable state unless interrupts are working. Check for this with the commands:

```
$ grep ib_ipath /proc/interrupts
```

Normal output will like similar to this:

```
CPU0       CPU1
  0:   22577705   22968429    IO-APIC-edge  timer
  4:        415        774    IO-APIC-edge  serial
  8:          0          0    IO-APIC-edge  rtc
  9:          0          0   IO-APIC-level  acpi
 14:      15750      64559    IO-APIC-edge  ide0
 15:          0         23    IO-APIC-edge  ide1
169:     533817        921   IO-APIC-level  eth0
177:          0      22767   IO-APIC-level  eth1
185:     364263          0   IO-APIC-level  ib_ipath
193:          0          0   IO-APIC-level  libata
201:          0          0   IO-APIC-level  ohci_hcd:usb1, ohci_hcd:usb2
NMI:      45641      45570
LOC:   45540410   45540372
ERR:          0
MIS:          0
```

If there is no output at all, driver initialization has failed. For further information on driver problems, see appendix C.4.1, appendix C.4.3, or appendix C.4.6.

However, if the output appears similar to one of these lines, then interrupts are not being delivered to the driver:

```
66:          0          0         PCI-MSI         ib_ipath
185:         0          0         IO-APIC-level   ib_ipath
```

NOTE: The output you see may vary depending on board type, distribution, or update level.

A zero count in all CPU columns means that no interrupts have been delivered to the processor.

Possible causes are:

- Booting the linux kernel with ACPI (Advanced Configuration and Power Interface) disabled on the boot command line, or in the BIOS configuration
- Other `infinipath` initialization failures

To check if the kernel was booted with the `"noacpi"` or `"pci=noacpi"` options, use this command:

```
$ grep -i acpi /proc/cmdline
```

If output is displayed, fix your kernel boot command line so that ACPI is enabled. This can be set in various ways, depending on your distribution. If no output is displayed, check to be sure that ACPI is enabled in your BIOS settings.

To track down other initialization failures, see appendix C.4.6.

The program `ipath_checkout` can also help flag these kinds of problems. See appendix C.9.8 for more information.

C.4.5
OpenFabrics Load Errors If `ib_ipath` Driver Load Fails

When the `ib_ipath` driver fails to load for any reason, all of the OpenFabrics drivers/modules loaded by /etc/init.d/infinipath fail with "Unknown symbol" errors:

```
ib_mad: Unknown symbol ib_unregister_client
ib_mad: Unknown symbol ib_query_ah
.
ib_sa: Unknown symbol ib_unregister_client
ib_sa: Unknown symbol ib_unpack
.
ib_ipath: Unknown symbol ib_modify_qp_is_ok
ib_ipath: Unknown symbol ib_unregister_device
.
ipath_ether: Unknown symbol ipath_layer_get_mac
ipath_ether: Unknown symbol ipath_layer_get_lid
.
```

NOTE: Not all the error messages are shown here.

C.4.6
InfiniPath `ib_ipath` Initialization Failure

There may be cases where `ib_ipath` was not properly initialized. Symptoms of this may show up in error messages from an MPI job or another program. Here is a sample command and error message:

```
$ mpirun -np 2 -m ~/tmp/mbu13 osu_latency
<nodename>:The link is down
MPIRUN: Node program unexpectedly quit. Exiting.
```

First, check to be sure that the InfiniPath driver is loaded:

```
$ lsmod | grep ib_ipath
```

If no output is displayed, the driver did not load for some reason. Try the commands (as root):

```
# modprobe -v ib_ipath
# lsmod | grep ib_ipath
# dmesg | grep ipath | tail -25
```

This will indicate whether the driver has loaded. Printing out messages using `dmesg` may help to locate any problems with `ib_ipath`.

If the driver loaded, but MPI or other programs are not working, check to see if problems were detected during the driver and InfiniPath hardware initialization with the command:

```
$ dmesg | grep -i ipath
```

This may generate more than one screen of output. Also, check the link status with the commands:

```
$ cat /sys/bus/pci/driver/ib_ipath/0?/status_str
```

These commands are normally executed by the `ipathbug-helper` script, but running them separately may help locate the problem.

Refer also to appendix C.9.16 and appendix C.9.8.

C.4.7
MPI Job Failures Due to Initialization Problems

If one or more nodes do not have the interconnect in a usable state, messages similar to the following will occur when the MPI program is started:

```
userinit: userinit ioctl failed: Network is down [1]: device init
failed
userinit: userinit ioctl failed: Fatal Error in keypriv.c(520):
device init failed
```

This could indicate that a cable is not connected, the switch is down, SM is not running, or a hardware error has occurred.

C.5 OpenFabrics Issues

This section covers items related to OpenFabrics, including OpenSM.

C.5.1 Stop OpenSM Before Stopping/Restarting InfiniPath

OpenSM must be stopped before stopping or restarting InfiniPath. If not, error messages such as the following will occur:

```
# /etc/init.d/infinipath stop

Unloading infiniband modules: sdp cm umad uverbs ipoib sa ipath
mad coreFATAL:Module ib_umad is in use.

Unloading infinipath modules FATAL: Module ib_ipath is in use.
[FAILED]
```

C.5.2 Load and Configure IPoIB Before Loading SDP

SDP will generate "Connection Refused" errors if it is loaded before IPoIB has been loaded and configured. Loading and configuring IPoIB first should solve the problem.

C.5.3 Set $IBPATH for OpenFabrics Scripts

The environment variable $IBPATH should be set to `/usr/bin`. If this has not been set, or if you have it set to a location other than the installed location, you may see error messages similar to this when running some OpenFabrics scripts:

```
/usr/bin/ibhosts: line 30: /usr/local/bin/ibnetdiscover: No such
file or directory
```

For the OpenFabrics commands supplied with this InfiniPath release, you should set the variable (if it has not been set already), to `/usr/bin` as follows:

```
$ export IBPATH=/usr/bin
```

C.6 System Administration Troubleshooting

The following section gives details on locating problems related to system administration.

C.6.1 Broken Intermediate Link

Sometimes message traffic passes through the fabric while other traffic appears to be blocked. In this case, MPI jobs fail to run.

In large cluster configurations, switches may be attached to other switches in order to supply the necessary inter-node connectivity. Problems with these inter-switch (or intermediate) links are sometime more difficult to diagnose than failure of the final link between a switch and a node. The failure of an intermediate link may allow some traffic to pass through the fabric while other traffic is blocked or degraded.

If you encounter such behavior in a multi-layer fabric, check that all switch cable connections are correct. Statistics for managed switches are available on a per-port basis, and may help with debugging. See your switch vendor for more information.

C.7 Performance Issues

Performance issues that are currently being addressed are covered in this section.

C.7.1 MVAPICH Performance Issues

MVAPICH over OpenFabrics over InfiniPath performance tuning has not yet been done. Improved performance will be delivered in future releases.

C.8 InfiniPath MPI Troubleshooting

Problems specific to compiling and running MPI programs are detailed below.

C.8.1 Mixed Releases of MPI RPMs

Make sure that all of the MPI RPMs are from the same release. When using `mpirun`, an error message will occur if different components of the MPI RPMs are from different releases. This is a sample message in the case where `mpirun` from release 1.3 is being used with a 2.0 library:

```
$ mpirun -np 2 -m ~/tmp/x2 osu_latency
MPI_runscript-xqa-14.0: ssh -x> Cannot detect InfiniPath
interconnect.
MPI_runscript-xqa-14.0: ssh -x> Seek help on loading InfiniPath
interconnect driver.
MPI_runscript-xqa-15.1: ssh -x> Cannot detect InfiniPath
interconnect.
MPI_runscript-xqa-15.1: ssh -x> Seek help on loading InfiniPath
interconnect driver.
MPIRUN: Node program(s) exitted during connection setup

$ mpirun -v
MPIRUN:Infinipath Release2.0 : Built on Wed Nov 19 17:28:58 PDT
2006 by mee
```

The following is the error that occurs when `mpirun` from the 2.0 release is being used with the 1.3 libraries:

```
$ mpirun-ipath-ssh -np 2 -ppn 1 -m ~/tmp/idev osu_latency

MPIRUN: mpirun from the 2.0 software distribution requires all
node processes to be running 2.0 software. At least node
<nodename> uses non-2.0 MPI libraries
```

C.8.2 Cross-compilation Issues

The 2.x PathScale compilers aren't currently supported on systems that use the GNU 4.x compilers and compiler environment (header files and libraries). This includes Fedora Core 4, Fedora Core 5 and SLES 10. The GNU 4.x environment will be supported in the PathScale Complier Suite 3.0 release.

The current workaround for this is to compile on a supported and compatible distribution, then run the executable on one of the systems that uses the GNU 4.x compilers and environment.

- To run on FC4 or FC5, install FC3 or RHEL4/CentOS on your build machine. Compile your application on this machine.
- To run on SLES 10, install SUSE 9.3 on your build machine. Compile your application on this machine.
- Alternatively, `gcc` can be used as the default compiler. Set `mpicc -cc=gcc` as described in section 3.5.3.

Next, on the machines in your cluster on which the job will run, install compatibility libraries. These libraries include C++ and Fortran compatibility shared libraries and libgcc.

For an FC4 or FC5 system, you would need:

- pathscale-compilers-libs (for FC3)
- compat-gcc-32
- compat-gcc-32-g77
- compat-libstdc++-33

On a SLES 10 system, you would need:

- compat-libstdc++ (for FC3)
- compat-libstdc++5 (for SLES 10)

Depending upon the application, you may need to use the `-W1,-Bstatic` option to use the static versions of some libraries.

C.8.3 Compiler/Linker Mismatch

This is a typical error message if the compiler and linker are not matching in C and C++ programs:

```
$ export MPICH_CC=gcc
$ mpicc mpiworld.c
/usr/bin/ld: cannot find -lmpichabiglue_gcc3
collect2: ld returned 1 exit status
```

C.8.4 Compiler Can't Find Include, Module or Library Files

RPMs can be installed in any location by using the `--prefix` option. This can introduce errors when compiling, if the compiler cannot find the include files (and module files for Fortran90 and Fortran95) from `mpi-devel*` , and the libraries from `mpi-libs*` in the new locations. Compiler errors similar to this can occur:

```
$ mpicc myprogram.c
/usr/bin/ld: cannot find -lmpich
collect2: ld returned 1 exit status
```

NOTE: As noted in Section 3.5.2 of the *InfiniPath Install Guide*, all development files now reside in specific *-Devel subdirectories.

On development nodes, programs must be compiled with the appropriate options so that the include files and the libraries can be found in the new locations. In addition, when running programs on compute nodes, you need to insure that the run-time library path is the same as the path that was used to compile the program.

The examples below show what compiler options to use for include files and libraries on the development nodes, and how to specify this new library path on the compute nodes for the runtime linker. The affected RPMs are:

```
mpi-devel* (on the development nodes)
mpi-libs* (on the development or compute nodes)
```

For these examples in Section C.8.5 below, we assume that these new locations are:

```
/path/to/devel (for mpi-devel-*)
/path/to/libs (for mpi-libs-*)
```

C.8.5 Compiling on Development Nodes

If the `mpi-devel-*` rpm is installed with the `--prefix /path/to/devel` option then `mpicc`, etc. will need to be passed `-I/path/to/devel/include` in order for the compiler to find the MPI include files, as in this example:

```
$ mpicc myprogram.c -I/path/to/devel/include
```

If you are using Fortran90 or Fortran95, a similar option is needed for the compiler to find the module files:

```
$ mpif90 myprogramf90.f90 -I/path/to/devel/include
```

If the `mpi-lib-*` rpm is installed on these development nodes with the `--prefix /path/to/libs` option, then the compiler will need to be given the `-L/path/to/libs` option so it can find the libraries. Here is the example for mpicc:

```
$ mpicc myprogram.c  -L/path/to/libs/lib (for 32 bit)
$ mpicc myprogram.c  -L/path/to/libs/lib64 (for 64bit)
```

To find both the include files and the libraries with these non-standard locations, we would now see an example like this:

```
$ mpicc myprogram.c -I/path/to/devel/include -L/path/to/libs/lib
```

C.8.6 Specifying the Run-time Library Path

There are several ways to specify the run-time library path so that when the programs are run the appropriate libraries are found in the new location. There are three different ways to do this:

- Use the `-Wl,-rpath,` option when compiling on the development node.
- Update the `/etc/ld.so.conf` file on the the compute nodes to include the path.
- Export the path in the `.mpirunrc` file.

These methods are explained in more detail below.

1. An additional linker option, `-Wl,-rpath,` supplies the run-time library path when compiling on the development node. The compiler options now look like this:

   ```
   $ mpicc myprogram.c -I/path/to/devel/include
   -L/path/to/libs/lib -Wl,-rpath,/path/to/libs/lib
   ```

   The above compiler command insures that the program will run using this path on any machine.

   For the second option, we change the file `/etc/ld.so.conf` on the compute nodes rather than using the `-Wl,-rpath,` option when compiling on the development node. We assume that the `mpi-lib-*` rpm is installed on the compute nodes with the same `--prefix /path/to/libs` option as on the development nodes. Then, on the computer nodes we then add the following lines to the file `/etc/ld.so.conf`.

   ```
   /path/to/libs/lib
   /path/to/libs/lib64
   ```

   Then, to make sure that the changes are picked up, run (as root):

   ```
   # /etc/ldconfig
   ```

   The libraries can now be found by the runtime linker on the compute nodes. This method has the advantage that it will work for all InfiniPath programs, without having to remember to change the compile/link lines.

2. Instead of either of the two above mechanisms, you can also put this line in the `~/.mpirunrc` file:

   ```
   export LD_LIBRARY_PATH=/path/to/libs/{lib,lib64}
   ```

   See Section 3.5.8 in the chapter “Using InfiniPath MPI” for more information on using the `-rcfile` option to `mpirun`.

   Choices between these options are left up to the cluster administrator and the MPI developer. See the documentation for your compiler for more information on the compiler options.

C.8.7 Run Time Errors With Different MPI Implementations

It is now possible to run different implementations of MPI, such as HP-MPI, over InfiniPath. Many of these implementations share command (such as `mpirun`) and library names, so it is important to distinguish which MPI version is in use. This is done primarily through careful programming practices.

Examples are given below.

In the following command, the HP-MPI version of `mpirun` is invoked by the full pathname. However, the program `mpi_nxnlatbw` was compiled with the QLogic version of `mpicc`. The mismatch will produce errors similar this:

```
$ /opt/hpmpi/bin/mpirun -hostlist "bbb-01,bbb-02,bbb-03,bbb-04"
-np 4 /usr/bin/mpi_nxnlatbw
bbb-02: Not running from mpirun?.
MPI Application rank 1 exited before MPI_Init() with status 1
bbb-03: Not running from mpirun?.
MPI Application rank 2 exited before MPI_Init() with status 1
bbb-01: Not running from mpirun?.
bbb-04: Not running from mpirun?.
MPI Application rank 3 exited before MPI_Init() with status 1
MPI Application rank 0 exited before MPI_Init() with status 1
```

In the case below, `mpi_nxnlatbw.c` is compiled with the HP-MPI version of `mpicc`, and given the name of `hpmpi-mpi_nxnlatbw`, so that it is easy to see which version was used. However, it is run with the QLogic `mpirun`, which will produce errors similar to this:

```
$ /opt/hpmpi/bin/mpicc \
/usr/share/mpich/examples/performance/mpi_nxnlatbw.c -o
hpmpi-mpi_nxnlatbw
$ mpirun -m ~/host-bbb  -np 4 ./hpmpi-mpi_nxnlatbw
./hpmpi-mpi_nxnlatbw: error while loading shared libraries:
libmpio.so.1: cannot open shared object file: No such file or
directory
./hpmpi-mpi_nxnlatbw: error while loading shared libraries:
libmpio.so.1: cannot open shared object file: No such file or
directory
./hpmpi-mpi_nxnlatbw: error while loading shared libraries:
libmpio.so.1: cannot open shared object file: No such file or
directory
./hpmpi-mpi_nxnlatbw: error while loading shared libraries:
libmpio.so.1: cannot open shared object file: No such file or
directory
MPIRUN: Node program(s) exitted during connection setup
```

The following two commands will both work properly:

QLogic `mpirun` and executable used together:

```
$ mpirun -m ~/host-bbb  -np 4 /usr/bin/mpi_nxnlatbw
```

HP-MPI `mpirun` and executable used together:

```
$ /opt/hpmpi/bin/mpirun -hostlist \
"bbb-01,bbb-02,bbb-03,bbb-04" -np 4 ./hpmpi-mpi_nxnlatbw
```

Hints:

Use the `rpm` command to find out which RPM is installed in the standard installed layout. For example:

```
# rpm -qf /usr/bin/mpirun
mpi-frontend-2.0-964.731_fc3_psc.i386.rpm
```

Check all rcfiles and `/opt/infinipath/etc/mpirun.defaults` to make sure that the paths for binaries and libraries (`$PATH` and `$LD_LIBRARY _PATH`) are consistent.

When compiling, use descriptive names for the object files.

See section C.8.4, section C.8.5, and section C.8.6 for additional information.

C.8.8 Process Limitation with `ssh`

MPI jobs that use more than 8 processes per node may encounter an `ssh` throttling mechanism that limits the amount of concurrent per-node connections to 10. If you have this problem, you will see a message similar to this when using mpirun:

```
$ mpirun -m tmp  -np 11 ~/mpi/mpiworld/mpiworld
ssh_exchange_identification: Connection closed by remote host
MPIRUN: Node program(s) exitted during connection setup
```

If you encounter a message like this, you or your system administrator should increase the value of 'MaxStartups' in your sshd configurations.

C.8.9 Using MPI.mod Files

MPI.mod (or mpi.mod) are the Fortran90/Fortran95 mpi modules files. These contain the Fortran90/Fortran95 interface to the platform-specific MPI library. The module file is invoked by 'USE MPI' or 'use mpi' in your application. If the application has an argument list that doesn't match what mpi.mod expects, errors such as this can occur:

```
$ mpif90 -O3 -OPT:fast_math -c communicate.F
           call mpi_recv(nrecv,1,mpi_integer,rpart(nswap),0,
                         ^
pathf95-389 pathf90: ERROR BORDERS, File = communicate.F, Line =
407, Column = 18
No specific match can be found for the generic subprogram call
"MPI_RECV".
```

If it is necessary to use a non-standard argument list, it is advisable to create your own MPI module file, and compile the application with it, rather than the standard MPI module file that is shipped in the mpi-devel-* RPM.

The default search path for the module file is:

```
/usr/include
```

To include your own `MPI.mod` rather than the standard version, use `-I/your/search/directory` which will cause `/your/search/directory` to be checked before `/usr/include`:

```
$ mpif90 -I/your/search/directory  myprogram.f90
```

Usage for Fortran95 will be similar to the example for Fortran90.

C.8.10 Extending MPI Modules

MPI implementations provide certain procedures which accept an argument having any data type, any precision, and any rank, but it isn't practical for an MPI module to enumerate every possible combination of type, kind, and rank. Therefore the strict type checking required by Fortran 90 may generate errors.

For example, if the MPI module tells the compiler that "mpi_bcast" can operate on an integer but does not also say that it can operate on a character string, you may see a message similar to the following one:

```
pathf95: ERROR INPUT, File = input.F, Line = 32, Column = 14
No specific match can be found for the generic subprogram call
"MPI_BCAST".
```

If you know that an argument can in fact accept a data type which the MPI module doesn't explicitly allow, you can extend the interface for yourself. For example, here's a program which illustrates how to extend the interface for "mpi_bcast" so that it accepts a character type as its first argument, without losing the ability to accept an integer type as well:

```
module additional_bcast
   use mpi
   implicit none
   interface mpi_bcast
      module procedure additional_mpi_bcast_for_character
   end interface mpi_bcast
contains
   subroutine additional_mpi_bcast_for_character(buffer, count,
   datatype, & root, comm, ierror)
   character*(*) buffer
   integer count, datatype, root, comm, ierror
   ! Call the Fortran 77 style implicit interface to "mpi_bcast"
   external mpi_bcast
   call mpi_bcast(buffer, count, datatype, root, comm, ierror)
   end subroutine additional_mpi_bcast_for_character
end module additional_bcast
program myprogram
   use mpi
   use additional_bcast
   implicit none
   character*4 c
   integer master, ierr, i
   ! Explicit integer version obtained from module "mpi"
   call mpi_bcast(i, 1, MPI_INTEGER, master, MPI_COMM_WORLD, ierr)
   ! Explicit character version obtained from module "additional_bcast"
```

```
   call mpi_bcast(c, 4, MPI_CHARACTER, master, MPI_COMM_WORLD, ierr)
end program myprogram
```

This is equally applicable if the module "mpi" provides only a lower-rank interface and you want to add a higher-rank interface. An example would be where the module explicitly provides for 1-D and 2-D integer arrays but you need to pass a 3-D integer array.

However, some care must be taken. One should only do this if:

- The module "mpi" provides an explicit Fortran 90 style interface for "mpi_bcast." If the module "mpi" does not, the program will use an implicit Fortran 77 style interface, which does not perform any type checking. Adding an interface will cause type-checking error messages where there previously were none.
- The underlying function really does accept any data type. It is appropriate for the first argument of "mpi_bcast" because the function operates on the underlying bits, without attempting to interpret them as integer or character data.

C.8.11
Lock Enough Memory on Nodes When Using a Batch Queuing System

InfiniPath MPI requires the ability to lock (pin) memory during data transfers on each compute node. This is normally done via `/etc/initscript`, which is created or modified during the installation of the `infinipath` RPM (setting a limit of 64MB, with the command "`ulimit -l 65536`").

Some batch systems, such as SLURM, propagate the user's environment from the node where you start the job to all the other nodes. For these batch systems, you may need to make the same change on the node from which you start your batch jobs.

If this file is not present or the node has not been rebooted after the `infinipath` RPM has been installed, a failure message similar to this will be generated:

```
$ mpirun -m ~/tmp/sm -np 2 -mpi_latency 1000 1000000
 node-00:1.ipath_update_tid_err: failed: Cannot allocate memory
 mpi_latency:
 /fs2/scratch/infinipath-build-2.0/mpi-2.0/mpich/psm/src
mq_ips.c:691:
mq_ipath_sendcts: Assertion 'rc == 0' failed. MPIRUN: Node program
unexpectedly quit. Exiting.
```

You can check the `ulimit -l` on all the nodes by running `ipath_checkout`. A warning will be given if `ulimit -l` is less that 4096.

There are two possible solutions to this. If InfiniPath is not installed on the node where you start the job, set this value in the following way (as root).

```
# ulimit -l 65536
```

Or, if you have installed InfiniPath on the node, reboot it to insure that `/etc/initscript` is run.

C.8.12 Error Messages Generated by `mpirun`

In the sections below, types of mpirun error messages are described. They fall into these categories:

- Messages from the InfiniPath Library
- MPI messages
- Messages relating to the InfiniPath driver and InfiniBand links

Messages generated by `mpirun` follow a general format:

```
program_name: message
function_name: message
```

Messages may also have different prefixes, such and `ipath_` or `psm_`, which will indicate in which part of the software the errors are occurring.

C.8.12.1 Messages from the InfiniPath Library

These messages may appear in the `mpirun` output.

The first set are error messages, which indicate internal problems and should be reported to Support.

```
Trying to cancel invalid timer (EOC)
sender rank rank is out of range (notification)
sender rank rank  is out of range (ack)
Reached TIMER_TYPE_EOC while processing timers
Found unknown timer type type
unknown frame type type
recv done: available_tids now n, but max is m (freed p)
cancel recv available_tids now n, but max is m (freed %p)
[n] Src lid error: sender: x, exp send: y
Frame receive from unknown sender. exp. sender = x, came from y
Failed to allocate memory for eager buffer addresses: str
```

The following error messages probably indicate a hardware or connectivity problem:

```
Failed to get IB Unit LID for any unit
Failed to get our IB LID
Failed to get number of Infinipath units
```

In these cases you can try to reboot, then call Support.

The following indicate a mismatch between the InfiniPath interconnect hardware in use and the version for which the software was compiled:

```
Number of buffer avail registers is wrong; have n, expected m
build mismatch, tidmap has n bits, ts_map m
```

These indicate a mismatch between the InfiniPath software and hardware versions. Consult Support after verifying that current drivers and libraries are installed.

The following are all informative messages about driver initialization problems. They are not necessarily fatal themselves, but sometimes indicate problems that interfere with the application. In the actual printed output all of them are prefixed with the name of the function that produced them.

```
Failed to get LID for unit u: str
Failed to get number of units: str
GETPORT ioctl failed: str
can't allocate memory for ipath_ctrl_typ: type
can't stat infinipath device to determine type: type
file descriptor is not for a real device, failing
get info ioctl failed: str
ipath_get_num_units called before init
ipath_get_unit_lid called before init
mmap64 of egr bufs from h failed: str
mmap64 of pio buffers at %llx failed: str
mmap64 of pioavail registers (%llx) failed: str
mmap64 of rcvhdr q failed: str
mmap64 of user registers at %llx failed: str
userinit allocation of rcvtail memory failed: str
userinit ioctl failed: str
Failed to set close on exec for device: str
```

NOTE: These messages should never occur. Please inform Support if they do.

The following message indicates that a node program may not be processing incoming packets, perhaps due to a very high system load:

```
eager array full after overflow, flushing (head h, tail t)
```

The following indicates an invalid InfiniPath link protocol version:

```
InfiniPath version ERROR: Expected version v, found w (memkey h)
```

The following error messages should rarely occur and indicate internal software problems:

```
ExpSend opcode h tid=j, rhf_error k: str
Asked to set timeout w/delay l, gives time in past (t2 < t1)
Error in sending packet: str
Fatal error in sending packet, exiting: str
Fatal error in sending packet: str
```

Here the `str` can give additional clues to the reason for the failure.

The following probably indicates a node failure or malfunctioning link in the fabric:

```
Couldn't connect to NODENAME, rank RANK#. Time elapsed HH:MM:SS.
Still trying
```

NODENAME is the node (host) name, RANK# is the MPI rank, and HH:MM:SS are the hours, minutes, and seconds since we started trying to connect.

If you get messages similar to the following, it may mean that you are trying to receive to an invalid (unallocated) memory address, perhaps due to a logic error in the program, usually related to malloc/free:

```
ipath_update_tid_err: Failed TID update for rendevous, allocation
problem
kernel: infinipath: get_user_pages (0x41 pages starting at
0x2aaaaeb50000
kernel: infinipath: Failed to lock addr 0002aaaaeb50000, 65 pages:
errno 12
```

TID is short for Token ID, and is part of the InfiniPath hardware. This error indicates a failure of the program, not the hardware or driver.

C.8.12.2
MPI Messages

Some MPI error messages are issued from the parts of the code inherited from the MPICH implementation. See the MPICH documentation for descriptions of these. This section presents the error messages specific to the InfiniPath MPI implementation.

These messages appear in the `mpirun` output. Most are followed by an abort, and possibly a backtrace. Each is preceded by the name of the function in which the exception occurred.

```
Error sending packet: description
Error receiving packet: description
```
A fatal protocol error occurred while trying to send an InfiniPath packet.

```
On Node n, process p seems to have forked.
The new process id is q. Forking is illegal under
InfiniPath. Exiting.
```
An MPI process has forked and its child process has attempted to make MPI calls. This is not allowed.

```
processlabel Fatal Error in filename line_no: error_string
```
This is always followed by an abort. The `processlabel` usually takes the form of host name followed by process rank.

At time of writing, the possible `error_string`s are:

```
Illegal label format character.
Recv Error.
Memory allocation failed.
Error creating shared memory object.
Error setting size of shared memory object.
Error mmapping shared memory.
Error opening shared memory object.
Error attaching to shared memory.
invalid remaining buffers !!
Node table has inconsistent length!
Timeout waiting for nodetab!
```

The following indicates an unknown host:

```
$ mpirun -np 2 -m ~/tmp/q  mpi_latency 100 100
MPIRUN: Cannot obtain IP address of <nodename>: Unknown host
<nodename> 15:35_~.1019
```

There is no route to a valid host:

```
$ mpirun -np 2 -m ~/tmp/q  mpi_latency 100 100
ssh: connect to host <nodename> port 22: No route to host
MPIRUN: Some node programs ended prematurely without connecting to
mpirun.
MPIRUN: No connection received from 1 node process on node
<nodename>
```

There is no route to any host:

```
$ mpirun -np 2 -m ~/tmp/q  mpi_latency 100 100
ssh: connect to host <nodename> port 22: No route to host
ssh: connect to host <nodename> port 22: No route to host
MPIRUN: All node programs ended prematurely without connecting to
mpirun.
```

Node jobs have started, but one host couldn't connect back to `mpirun`:

```
$ mpirun -np 2 -m ~/tmp/q  mpi_latency 100 100
9139.psc_skt_connect: Error connecting to socket: No route to host
<nodename> Cannot connect to mpirun within 60 seconds.
MPIRUN: Some node programs ended prematurely without connecting to
mpirun.
MPIRUN: No connection received from 1 node process on node
<nodename>
```

Node jobs have started, both hosts couldn't connect back to mpirun:

```
$ mpirun -np 2 -m ~/tmp/q  mpi_latency 100 100
9158.psc_skt_connect: Error connecting to socket: No route to host
<nodename> Cannot connect to mpirun within 60 seconds.
6083.psc_skt_connect: Error connecting to socket: No route to host
<nodename> Cannot connect to mpirun within 60 seconds.
MPIRUN: All node programs ended prematurely without connecting to
mpirun.
$ mpirun -np 2 -m ~/tmp/q  mpi_latency 1000000 1000000
MPIRUN: <nodename> node program unexpectedly quit: Exiting.
```

One program on one node died:

```
$ mpirun -np 2 -m ~/tmp/q mpi_latency 100000 1000000
MPIRUN: <nodename> node program unexpectedly quit: Exiting.
```

The `quiescence detected` message is printed when an MPI job does not seem to be making progress. The default timeout is 900 seconds. After this length of time all the node processes will be terminated. This timeout can be extended or disabled with the `-quiescence-timeout` option in `mpirun`.

```
$ mpirun -np 2 -m ~/tmp/q  -q 60 mpi_latency 1000000 1000000
MPIRUN: MPI progress Quiescence Detected after 9000 seconds.
MPIRUN: 2 out of 2 ranks showed no MPI send or receive progress.
MPIRUN: Per-rank details are the following:
MPIRUN: Rank    0 (<nodename> ) caused MPI progress Quiescence.
MPIRUN: Rank    1 (<nodename> ) caused MPI progress Quiescence.
MPIRUN: both MPI progress and Ping Quiescence Detected after 120
seconds.
```

Occasionally a stray process will continue to exist out of its context. `mpirun` checks for stray processes; they are killed after detection.The following is an example of the type of message you will see in this case:

```
$ mpirun -np 2 -ppn 1 -m ~/tmp/mfast mpi_latency 500000 2000
iqa-38: Received 1 out-of-context eager message(s) from stray
process PID=29745
running on host 192.168.9.218
iqa-35: PSM pid 10513 on host IP 192.168.9.221 has detected that I
am a stray process, exiting.
2000    5.222116
iqa-38:1.ips_ptl_report_strays: Process PID=29745 on host
IP=192.168.9.218 sent
1 stray message(s) and was told so 1 time(s) (first stray message
at 0.7s (13%),last at 0.7s (13%) into application run)
```

The following should never occur. Please inform Support if it does:

```
Internal Error: NULL function/argument found:func_ptr(arg_ptr)
```

C.8.12.3
Driver and Link Error Messages Reported by MPI Programs

Two types of error messages are described below.

1. When the InfiniBand link fails during a job, a message will be reported once per occurrence. The message will be similar to this:

   ```
   ipath_check_unit_status: IB Link is down
   ```

   This can happen when a cable is disconnected, a switch is rebooted, or if there are other problems with the link. The job will continue retrying until the quiescence interval expires. See the `mpirun -q` option for information on quiescence.

2. If a hardware problem occurs, an error similar to this will be reported:

   ```
   infinipath:  [error strings ] Hardware error
   ```

   This will cause the MPI program to terminate. The error string may provide additional information as to the problem. To further determine the source of the problem, examine `syslog` on the node reporting the problem.

C.8.13
MPI Stats

Using the `-print-stats` option to `mpirun` will result in a listing to `stderr` of various MPI statistics. Here is example output for the `-print-stats` option when used with an 8-rank runof the HPCC benchmark.

```
MPIRUN:  MPI Statistics Summary    (min,max,median @ rank)
MPIRUN:  Messages sent
MPIRUN:   Eager count              (min=652.54K @ 0, max=653.39K @ 7, med= 653.15K)
MPIRUN:   Eager aggregate bytes    (min=  2.08G @ 0, max=  2.08G @ 2, med=   2.08G)
MPIRUN:   Rendezvous count         (None)
MPIRUN:   Rendezvous agg. bytes    (None)
MPIRUN:
MPIRUN:  Messages received
MPIRUN:   Expected count           (min=590.48K @ 2, max=624.90K @ 6, med= 619.01K)
MPIRUN:   Expected aggregate bytes (min=  2.03G @ 2, max=  2.04G @ 1, med=   2.04G)
MPIRUN:   Unexpected count         (min= 27.89K @ 6, max= 62.69K @ 2, med=  39.20K)
MPIRUN:   Unexpected agg. bytes    (min= 44.57M @ 1, max= 57.95M @ 2, med=  48.04M)
MPIRUN:   Unexpected count %       (min=     4% @ 6, max=     9% @ 2, med=      6%)
```

Message statistics are available for transmitted and received messages. In allcases, the MPI rank number responsible for a minimum or maximum value is reported with the relevant value. For application runs of at least 3 ranks, a median is also available.

Since transmitted messages employ either an *Eager* or a *Rendezvous* protocol, results are available relative to both message count and aggregated bytes. Message count represents the amount of messages transmitted by each protocol on a per-rank basis. Aggregated amounts of message bytes indicate the total amount of data that was moved on each rank by a particular protocol.

On the receive side, messages are split into either *expected* or *unexpected* messages. Unexpected messages cause the MPI implementation to buffer the transmitted data until the receiver is able to produce a matching MPI receive buffer. Expected messages refer to the inverse case, which should be the common case in most MPI applications. An additional metric, *Unexpected count %*, representing the proportion of unexpected messages in relation to the total number of messages received is also shown because of the notable effect unexpected messages have on performance.

For more precise information, users are encouraged to make use of MPI profilers such as mpiP. For more information on mpiP, see :

http://www.llnl.gov/CASC/mpip/

For reference on the HPCC benchmark, see:

http://icl.cs.utk.edu/hpcc/

C.9 Useful Programs and Files for Debugging

The most useful programs and files for debugging are listed in the sections below. Many of these programs and files have been discussed elsewhere in the documentation: this information is summarized and repeated here for your convenience.

C.9.1 Check Cluster Homogeneity with `ipath_checkout`

Many problems can be attributed to the lack of homogeneity in the cluster environment. Use the following items as a checklist for verifying homogeneity. A difference in any one of these items in your cluster may cause problems:

- Kernels
- Distributions
- Versions of the InfiniPath boards
- Runtime and build environments
- .o files from different compilers
- Libraries
- Processor speeds

With the exception of finding any differences between the runtime and build environments, `ipath_checkout` will pick up information on all the above items. Other programs useful for verifying homogeneity are listed in Table C-2.

C.9.2
Restarting InfiniPath

If, on any node, the driver status appears abnormal, you can try restarting (as root):

```
# /etc/init.d/infinipath restart
```

These two commands perform the same functions:

```
# /etc/init.d/infinipath stop
# /etc/init.d/infinipath start
```

It may also be useful to inspect the file `/var/log/messages`, to check for any abnormal activity.

C.9.3
Summary of Useful Programs and Files

Useful programs and files are summarized in the table below. Descriptions for some of the programs and files follow. Check `man` pages for more information on the programs.

Table C-2. Useful Programs and Files

Program or file name	Function	Use to verify homogeneity?
`boardversion`	File. Check the version of the installed InfiniPath software.	Yes
`chkconfig`	Check configuration state, enable/disable services, including drivers.	No
`ibstatus`	Checks status of InfiniBand devices when OpenFabrics is enabled.	No
`ibv_devinfo`	Lists info about InfiniBand devices in use. Use when OpenFabrics is enabled.	No
`ident`	Identifies `RCS` keyword strings in files. Can check for dates, release versions, and other identifying information.	Yes
`ipath_checkout`	A `bash` shell script that performs sanity testing on cluster using InfiniPath hardware and software. If the program is run without errors, the node is properly configured.	Yes
`ipath_control`	A shell script that can be used to manipulate various parameters for the InfiniPath driver. This script gathers the same information contained in `boardversion`, `status_str`, and `version`.	Yes

Table C-2. Useful Programs and Files (Continued)

Program or file name	Function	Use to verify homogeneity?
ipathbug-helper	A shell script that gathers status and history information for use in analyzing InfiniPath problems.	Yes
ipath_pkt_test	Tests the InfiniBand link and bandwidth between two InfiniPath HCAs, or, using an InfiniBand loopback connector, within a single InfiniPath HCA.	No
ipathstats	Displays both driver statistics, and hardware counters, including both performance and "error" (including status) counters	No
lsmod	Shows status of modules in the Linux kernel. Can use to check whether drivers are loaded.	No
modprobe	Adds or removes modules from the Linux kernel. Used to configure ipath_ether module on SUSE.	No
mpirun	A front end program that starts an MPI job on an InfiniPath cluster. Can be used to check the origin of the drivers.	Yes
ps	Displays information on current active processes. Use to check whether all necessary processes have been started.	No
rpm	Package manager used to install, query, verify, update, or erase software packages. Can use to check contents of a package.	Yes
strings	Prints the strings of printable characters in a file. Useful for determining contents of non-text files such as date and version.	Yes
status_str	File. Verifies that the InfiniPath software is loaded and functioning.	No
version	File.Provides version information of installed software /drivers.	Yes
/var/log/messages	File. Various programs write messages to this logfile. Use to track activity on your system.	No

C.9.4 boardversion

It may be useful to keep track of the current version of the installed software. You can check the version of the installed InfiniPath software by looking in:

```
/sys/bus/pci/drivers/ib_ipath/00/boardversion
```

Example contents are:

```
Driver 2.0,InfiniPath_QHT7140,InfiniPath13.2,PCI 2,SW Compat 2
```

This information is useful when for reporting problems when requesting support.

NOTE: This file returns information on which form factor adapter is installed. The HTX full height short form factor is referred to as the QHT7040, the HTX low profile form factor is referred to as the QHT7140, and the PCIe half height short form factor is the QLE7140. This information will make it easier for Support to help with any problems.

C.9.5 ibstatus

This program displays basic information on the status of InfiniBand devices that are currently in use when the OpenFabrics modules are loaded.

C.9.6 ibv_devinfo

This program displays information about InfiniBand devices, including various kinds of identification and status data. Use this program when OpenFabrics is enabled.

C.9.7 ident

`ident` strings are available in `ib_ipath.ko`. Running `ident` (as root) will yield information similar to the following. For QLogic RPMs, it will look like:

```
# ident /lib/modules/$(uname -r)/updates/*ipath.ko
/lib/modules/2.6.16.21-0.8-smp/updates/ib_ipath.ko:
     $Id: QLogic Release2.0 $
     $Date: 2006-09-15-04:16 $
     $Id: QLogic Release2.0 $
     $Date: 2006-09-15-04:16 $
```

For non-QLogic RPMs, it will look like:

```
# ident /lib/modules/$(uname -r)/updates/*ipath_ether.ko
/lib/modules/2.6.16.21-0.8-smp/updates/infinipath.ko:
  $Id: kernel.org InfiniPath Release 2.0 $
  $Date: 2006-09-15-04:16 $
/lib/modules/2.6.16.21-0.8-smp/updates/ipath.ko:
  $Id: kernel.org InfiniPath Release2.0 $
  $Date: 2006-09-15-04:20 $
```

NOTE: $ident is in the optional rcs RPM, and is not always installed.

C.9.8 ipath_checkout

ipath_checkout is a bash script used to verify that the installation is correct and that all the nodes of the network are functioning and mutually connected by the InfiniPath fabric. It is to be run on a front end node, and requires specification of a hosts file:

```
$ ipath_checkout [options] hostsfile
```

where hostsfile designates a file listing the hostnames of the nodes of the cluster, one hostname per line. The format of hostsfile is as follows:

```
hostname1
hostname2
...
```

ipath_checkout performs the following seven tests on the cluster:

1. ping all nodes to verify all are reachable from the frontend.
2. ssh to each node to verify correct configuration of ssh.
3. Gather and analyze system configuration from nodes.
4. Gather and analyze RPMs installed on nodes.
5. Verify InfiniPath hardware and software status and configuration.
6. Verify ability to mpirun jobs on nodes.
7. Run bandwidth and latency test on every pair of nodes and analyze results.

The possible options to ipath_checkout are:

-h, --help
Displays help messages giving defined usage.

-v, --verbose
-vv, --vverbose
-vvv, --vvverbose
These specify three successively higher levels of detail in reporting results of tests. So, there are four levels of detail in all, including the case of where none these options are given.

-c, --continue
When not specified, the test terminates when any test fails. When specified, the tests continue after a failure, with failing nodes excluded from subsequent tests.

`--workdir=DIR`

Use DIR to hold intermediate files created while running tests. DIR must not already exist.

`-k, --keep`

Keep intermediate files that were created while performing tests and compiling reports. Results will be saved in a directory created by `mktemp` and named `infinipath_XXXXXX` or in the directory name given to `--workdir`.

`--skip=LIST`

Skip the tests in `LIST` (e.g. --skip=2,4,5,7 will skip tests 2, 4, 5, and 7)

`-d, --debug`

Turn on `-x` and `-v` flags in bash.

In most cases of failure, the script suggests recommended actions. Please see the `ipath_checkout` man page for further information and updates.

C.9.9 ipath_control

This is a shell script that can be used to manipulate various parameters for the InfiniPath driver. Many of them are intended to be used only when diagnosing problems, and may require special system configurations. Use of the options may require restarting the driver or utility programs in order to recover from incorrect parameters.

Most of the functionality is accessed via the `/sys` filesystem. This shell script gathers the same information contained in these files:

```
/sys/bus/pci/drivers/ib_ipath/00/boardversion
/sys/bus/pci/drivers/ib_ipath/00/status_str
/sys/bus/pci/drivers/ib_ipath/version
```

These files are also documented in table C-2 and table C-3.

Other than the `-i` option, this script will need to be run with root permissions. The `-i` option is listed here, as it is the most commonly used. See the man pages for `ipath_control` for more details.

Here is sample usage and output:

```
$ ipath_control -i
$Id: QLogic Release2.0 $ $Date: 2006-09-15-04:16 $
00: Version: Driver 2.0, InfiniPath_QHT7140, InfiniPath1 3.2, PCI
2, SW Compat 2
```

```
00: Status: 0xe1 Initted Present IB_link_up IB_configured
00: LID=0x30 MLID=0x0 GUID=00:11:75:00:00:07:11:97 Serial:
1236070407
```

C.9.10 ipathbug-helper

The tool `ipathbug-helper` is useful for verifying homogeneity. Prior to seeking assistance from QLogic technical support, you should run this script on the head node of your cluster and the compute nodes which are suspected to have problems. Inspection of the output will often help you to see the problem. Simply run it on several nodes and examine the output for differences.

It is best to run `ipathbug-helper` with root privilege, since some of the queries it makes requires it. There is also a `--verbose` which greatly increases the amount of gathered information.

If you are unable to see the problem, send its `stdout` output to your reseller, along with information on the version of the InfiniPath software you are using.

C.9.11 ipath_pkt_test

This is a simple program that can be used to test the InfiniBand link and bandwidth between two InfiniPath HCAs, or, using an InfiniBand loopback connector, within a single InfiniPath HCA. It is runs in either ping-pong mode (send a packet, wait for a reply, repeat), or in stream mode (send packets as quickly as possible, receive responses as they come back).

On completion, the sending side prints statistics on the packet bandwidth, showing both the payload bandwidth, and the total bandwidth (including InfiniBand and InfiniPath headers). See the man page for more information.

C.9.12 ipathstats

The `ipathstats` program can be useful for diagnosing InfiniPath problems, particularly those that are performance related. It displays both driver statistics, and hardware counters, including both performance and "error" (including status) counters.

Running "`ipathstats -c 10`", for example, will show the number of packets and 32 bit words of data being transferred on a node in each10 second interval. This may show differences in traffic patterns on different nodes, or at different stages of execution. For more information see the man page.

C.9.13 lsmod

If you need to find which InfiniPath and OpenFabrics modules are running, try the following command:

```
# lsmod | egrep 'ipath_|ib_|rdma_|findex'
```

C.9.14 mpirun

`mpirun` can give information on whether the program is being run against a QLogic or non-QLogic driver. Sample commands and results are given below.

QLogic-built:

```
$ mpirun -np 2 -m /tmp/id1 -d0x101 mpi_latency 1 0
asus-01:0.ipath_setaffinity: Set CPU affinity to 1, port 0:2:0 (1
active chips)
asus-01:0.ipath_userinit: Driver is QLogic-built
```

Non-QLogic built:

```
$ mpirun -np 2 -m /tmp/id1 -d0x101 mpi_latency 1 0
asus-01:0.ipath_setaffinity: Set CPU affinity to 1, port 0:2:0 (1
active chips)
asus-01:0.ipath_userinit: Driver is not QLogic-built
```

C.9.15 rpm

To check the contents of an RPM, use these commands:

```
$ rpm -qa infinipath\* mpi-\*
$ rpm -q --info infinipath # (etc)
```

The option `-q` will query and `--qa` will query all.

C.9.16 status_str

Check the file `status_str` to verify that the InfiniPath software is loaded and functioning. To locate this file go to:

```
/sys/bus/pci/drivers/ib_ipath/
```

and look for a subdirectory with the InfiniPath unit numbers 00, 01, and so on. `status_str` will be found in this directory. The following table shows the possible contents of the file, with brief explanations of the entries.

Table C-3. status_str File

File contents	Description
`Initted`	The driver has loaded and successfully initialized the IBA6110.
`Present`	The IBA6110 has been detected (but not initialized unless Initted is also here).
`IB_link_up`	The IB link has been configured and is in the active state; packets can be sent and received.
`IB_configured`	The IB link has been configured. It may or may not be up and usable.
`NOIBcable`	Unable to detect link present. Can be caused by no cable plugged into the QHT7140 or QLE7140, or connected there but not to a switch, or the switch it is connected to is down.
`Fatal_Hardware_Error`	Only appears if there is trouble.

In this same directory are other files containing information related to status. They are summarized in Table C-4, below.

Table C-4. Other Files Related to Status

File name	Contents
`lid`	InfiniBand Local ID (LID). The address on the IB fabric, similar conceptually to an IP address for TCP/IP. The "Local" refers to it being unique only within a single IB fabric.
`mlid`	The Multicast Local ID (MLID), for IB multicast. Used for doing InfiniPath ether broadcasts, since IB has no concept of broadcast.
`guid`	The Globally Unique ID (GUID) for the InfiniPath chip. Equivalent to an Ethernet MAC address.
`nguid`	The number of GUIDs that are used. If nguids == 2, and two chips are discovered, the first one will be assigned the requested GUID (from `eeprom`, or `ipath_sma`), and the second chip gets that GUID+1.
`serial`	The serial number of the QHT7140 or QLE7140 board.
`unit`	Unique number for each card or chip in a system.
`status`	The numeric version of the `status_str` file, described in the preceding table.

C.9.17 strings

The command `strings` can also be used. Its format is as follows:

```
$ strings /usr/lib/libinfinipath.so.4.0 | grep Date:
```

will produce output like this:

```
$Date: 2006-09-15 04:07 Release2.0 InfiniPath $
```

NOTE: `strings` is part of `binutils` (a development RPM), and may not be available on all machines.

C.9.18 version

You can check the version of the installed InfiniPath software by looking in:

```
/sys/bus/pci/drivers/ib_ipath/version
```

Example contents for QLogic-built drivers:

```
$Id: QLogic Release2.0 $ $Date: 2006-09-15-04:16 $
```

For non-Qlogic-built drivers (in this case kernel.org), it will look like this:

```
$Id: kernel.org InfiniPath Release2.0 $ $Date: 2006-09-15-04:18 $
```

Appendix D
Recommended Reading

Reference material for further reading is provided here.

D.1 References for MPI

The MPI Standard specification documents.
http://www.mpi-forum.org/docs

The MPICH implementation of MPI and its documentation.
http://www-unix.mcs.anl.gov/mpi/mpich/

The ROMIO distribution and its documentation.
http://www.mcs.anl.gov/romio

D.2 Books for Learning MPI Programming

Gropp, William, Ewing Lusk, and Anthony Skjellum, *Using MPI*, Second Edition, 1999, MIT Press, ISBN 0-262-57134-X.

Gropp, William, Ewing Lusk, and Anthony Skjellum, *Using MPI-2*, Second Edition, 1999, MIT Press, ISBN 0-262-57133-1.

Pacheco, *Parallel Programming with MPI*, 1997, Morgan Kaufman Publishers, ISBN 1-55860

D.3 Reference and Source for SLURM

The open-source resource manager designed for Linux clusters.
http://www.llnl.gov/linux/slurm/

D.4 InfiniBand

The InfiniBand specification, found at the InfiniBand Trade Association site.
http://www.infinibandta.org/

D.5 OpenFabrics

Open InfiniBand Alliance.
http://www.openfabrics.org

D.6 Clusters

Gropp, William, Ewing Lusk, and Thomas Sterling, *Beowulf Cluster Computing with Linux*, Second Edition, 2003, MIT Press, ISBN 0-262-69292-9.

D.7 Rocks

Extensive documentation on installing Rocks and custom Rolls.
http://www.rocksclusters.org/

Appendix E
Glossary

A glossary is provided below for technical terms used in the documentation.

bandwidth	The rate at which data can be transmitted. This represents the capacity of the network connection. Theoretical peak bandwidth is fixed, but the *effective bandwidth*, the ideal rate is modified by overhead in hardware and the computer operating system. Usually measured in bits/megabits or bytes/megabytes per second. Bandwidth is related to latency.
BIOS	For *Basic Input/Output System*. It typically contains code for initial hardware setup and bootstrapping.
build node	A machine on which source code, examples or benchmarks can be compiled.
compute node	A machine used to run a job.
DAPL	For *Direct Access Provider Library*. The reference implementation for RDMA transports. Consists of both kernel mode (kDAPL) and user mode (uDAPL) versions.
development node	Same as *build node*.
DHCP	For *Dynamic Host Configuration Protocol*. A communications protocol for allocating IP addresses. Also provides other basic networking information, such as router addresses and name servers.
EATX	For *Extended Advanced Technology Extended* motherboard.
fabric	The InfiniBand interconnect infrastructure, consisting of a set of HCAs (and possibly TCAs) connected by switches, such that each end node can directly reach all other nodes.
front end node	The machine or machines used to launch jobs.
funneled thread model	Only the main (master) thread may execute MPI calls. In InfiniPath MPI, hybrid MPI/OpenMP applications are supported, provided the MPI routines are called only by the master OpenMP thread.

GID	For *Global Identifier*. Used for routing between different InfiniBand subnets.
GUID	For *Globally Unique Identifier* for the InfiniPath chip. Equivalent to Ethernet MAC address.
head node	Same as *front end node*.
HCA	For *Host Channel Adapter.* HCAs are I/O engines located within processing nodes, connecting them to the InfiniBand fabric.
hosts file	Same as *mpihosts file.* Not the same as the /etc/hosts file.
HTX	A specification that defines a connector and form factor for HyperTransport-enabled daughtercards and EATX motherboards.
InfiniBand	Also referred to as *IB*. An input/output architecture used in high-end servers. It is also a specification for the serial transmission of data between processors and I/O devices. InfiniBand typically uses switched, point-to-point channels. These channels are usually created by attaching host channel adapters (HCAs) and target channel adapters (TCAs) through InfiniBand switches.
IPoIB	For *Internet Protocol over InfiniBand*, as per the OpenFabrics standards effort. This protocol layer allows the traditional Internet protocol(IP) to run over an InfiniBand fabric.
iSER	For *iSCSI Extensions for RDMA*. An upper layer protocol.
kDAPL	For *kernel Direct Access Provider Library*.kDAPL is the kernel mode version of the DAPL protocol.
latency	The delay inherent in processing network data. In terms of MPI, it is the time required to send a message from one node to another, independent of message size. Latency can be further split into sender and receiver processing overheads, as well as wire and switch overhead.
launch node	Same as *front end node*.
layered driver	A driver that does not directly manage any target devices. The layered driver calls another driver's routines, which in turn manages the target devices.

LID	For *Local Identifier.* Assigned by the Subnet Manager (SM) to each visible node within a single InfiniBand fabric. It is similar conceptually to an IP address for TCP/IP.
Lustre	Open source project to develop scalable cluster file systems.
MAC Address	For *Media Access Control Address*. It is a unique identifier attached to most forms of networking equipment.
machines file	Same as *mpihostsfile*.
MADs	For *Management Datagrams*. Subnet Managers (SMs) and Subnet Management Agents (SMAs) communicate via MADs.
managed switch	A switch that can be configured to run an embedded Subnet Manager (SM).
MGID	For *Multicast Group ID*. An identifier for a multicast group. This can be assigned by the SM at multicast group creation time, although frequently it is chosen by the application or protocol instead.
MLID	For *Multicast Local ID* for InfiniBand multicast. This is the identifier a member of a multicast group uses for addressing messages to other members of the group.
MPD	For *Multi-Purpose Daemon*. An alternative to `mpirun` to launch MPI jobs, providing support for MPICH. Developed at Argonne National laboratory.
MPI	For *Message-Passing Interface*. MPI is a message-passing library or collection of routines used in distributed-memory parallel programming. It is used in data exchange and task synchronization between processes. The goal of MPI is to provide portability and efficient implementation across different platforms and architectures.
MPICH	A freely available, portable implementation of MPI.
mpihosts file	A file containing a list of the hostnames of the nodes in a cluster on which node programs may be run. Also referred to as *node file*, *hosts file*, or *machine(s) file*.

MTRR	For *Memory Type Range Registers*. MTRR For "Memory Type Range Registers". Used by the InfiniPath driver to enable write combining to the InfiniPath on-chip transmit buffers. This improves write bandwidth to the InfiniPath chip, by writing multiple words in a single bus transaction (typically 64). Applies only to x86_64 systems.
MTU	For *Maximum Transfer Unit*. The largest packet size that can be transmitted over a given network.
multicast group	A mechanism that a group of nodes use to communicate amongst either other. It is an efficient mechanism for broadcasting messages to many nodes, as messages sent to the group are received by all members of the group without the sender having to explicitly send it to each individual member (or even having to know who the members are.) Nodes can join or leave the group at any time.
node file	Same as *hostsfile*.
node program	Each individual process that is part of the parallel MPI job. The machine on which it is executed is called a *node*.
OpenIB	The previous name of OpenFabrics.
OpenFabrics	The open source InfiniBand protocol stack.
OpenMP	Specification that provides an open source model for parallel programming that is portable across shared memory architectures from different vendors.
OpenSM	Open source SM (Subnet Manager**)** that provides basic functionality for subnet discovery and activation.
PCIe	For *PCI Express*. Based on PCI concepts and standards, PCIe uses a faster serial connection mechanism.
RDMA	For *Remote Direct Memory Access*. A communications protocol that enables data transmission from the memory of one computer to the memory of another without involving the CPU. The most common form of RDMA is over InfiniBand.
RPM	For *Red Hat Package Manager.* A tool for packaging, installing, and managing software for Linux distributions.

SDP	For S*ockets Direct Protocol*. An InfiniBand-specific upper layer protocol. It defines a standard wire protocol to support stream sockets networking over InfiniBand.
SRP	For *SCSI RDMA Protocol*. The implementation of this protocol is under development for utilizing block storage devices over an InfiniBand fabric.
SM	For *Subnet Manager*. A subnet contains a master Subnet Manager which is responsible for network initialization (topology discovery), configuration, and maintenance. The Subnet Manager discovers and configures all the reachable nodes in the InfiniBand fabric. It discovers them at switch startup, and continues monitoring changes in the physical network connectivity and topology. It is responsible for assigning local identifiers, called *LID*s, to the visible nodes. It also handles multicast group setup. When the network contains multiple managed switches, they negotiate among themselves which will be the controlling Subnet Manager. It communicates with the *SMA*s that exist on all nodes in a cluster.
SMA	For *Subnet Management Agent*. SMAs exist on all nodes, and are responsible for interacting with the subnet manager to configure an individual node and report node parameters and statistics.
subnet	A single InfiniBand network.
switch	Used to connect HCAs and TCAs. Packets are forwarded from one port to another within the switch, based on the LID of the packet. The fabric is the connected group of switches.
TCA	For *Target Channel Adapter.* A TCA is a channel adapter for I/O nodes, such as shared storage devices.
TCP	For *Transmission Control Protocol*. One of the core protocols of the Internet protocol suite. A transport mechanism that ensures that data arrives complete and in order.
TID	For *Token ID*. A method of identifying a memory region. Part of the InfiniPath hardware.
uDAPL	For *user Direct Access Provider Library*. uDAPL is the user space implementation of the DAPL protocol.
unmanaged switch	A switch that does not have an active Subnet Manager (SM).

Index

L

M

O

P

R

S

T