QLOGIC®

The Ultimate in Performance

Draft

QLogic HCA and QLogic OFED Software Users Guide

QLogic OFED Version 1.4

IB6054601-00 G.02

Information furnished in this manual is believed to be accurate and reliable. However, QLogic Corporation assumes no responsibility for its use, nor for any infringements of patents or other rights of third parties which may result from its use. QLogic Corporation reserves the right to change product specifications at any time without notice. Applications described in this document for any of these products are for illustrative purposes only. QLogic Corporation makes no representation nor warranty that such applications are suitable for the specified use without further testing or modification. QLogic Corporation assumes no responsibility for any errors that may appear in this document.

No part of this document may be copied nor reproduced by any means, nor translated nor transmitted to any magnetic medium without the express written consent of QLogic Corporation. In accordance with the terms of their valid QLogic agreements, customers are permitted to make electronic and paper copies of this document for their own exclusive use.

The QHT7040, QHT7140, QLE7140, QLE7240, and QLE7280 QLogic Host Channel Adapters are covered by the following patent: 7308535.

Draft

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	
Rev. 2.0, 3/23/07, QLogic Rev IB6054601 D	
Rev. 2.1, 8/24/07, QLogic Rev IB6054601 E	
Rev. 2.2, 5/27/08, QLogic Rev IB6054601 F	
Rev. 2.2, 9/5/08, QLogic Rev IB6054601 G	
Rev. QLogic OFED 1.4, 4/7/09, QLogic Rev IB6054601 G.02	
Changes	**Sections Affected**
Product name changed from *InfiniPath* to *QLogic OFED*. Version number is set to 1.4. Instances of InfiniPath changed where appropriate; some file-names and output messages keep old name.	All
Updated Contact Information	“Contact Information” on page 1-5
Added more information on VNIC Interface	“OpenSM” on page 4-7
Remove sections on configuring `ipath_ether`. *ipath_ether* now removed.	Was “Configuring the ipath_ether Network Interface” on page 3-7.
Updated the supported distributions information.	“Supported Distributions and Kernels” on page 2-3

Delete reference to *ipath_ether*	"InfiniPath and OpenFabrics Driver Overview" on page 4-5, "Manual Shutdown or Restart May Hang if NFS in Use" on page D-7, Table F-7 on page F-18, "Glossary" on page Glossary-1, "Managing the InfiniPath Driver" on page 4-18
Delete error entry for *ipath_ether* in Troubleshooting	Was ifup on ipath_ether on SLES 10 Reports "unknown device" on page C-9
Deleted ***NOTE*** about MPICH_ROOT replacing INFINIPATH_ROOT. No longer needed.	Table 5-7 on page 5-18
Change What's New in this Release to Features. Deleted old Features section	Appendix 2
Updated compiler support information	"Compiler Support" on page 2-4
Updated information on software components.	"Software Components" on page 2-5
Removed paragraph about this release adding support for QLE7240/7280, since this is outdated. Changed *InfiniPath* to *QLogic OFED* where appropriate.	"Features" on page 2-1
Updated Installed Layout section.	"Installed Layout" on page 4-2
Updated InfiniPath and OpenFabrics Driver Overview section.	"InfiniPath and OpenFabrics Driver Overview" on page 4-5
Updated Configuring the IPoIB Network Interface section.	"Configuring the IPoIB Network Interface" on page 4-6
Updated VNIC interface information	"Configuring and Administering the VNIC Interface" on page 4-10
Updated SRP section	"SRP" on page 4-8
Added new content for configuring Intel MPI.	"Setup" on page 6-3
Updated information for starting and stopping the InfiniPath software. Added ***NOTE*** about how to stop driver/modules manually.	"Managing the InfiniPath Driver" on page 4-18
Added ***NOTE*** about using ibhosts to create a nodefile or hostsfile.	"Console I/O in MPI Programs" on page 5-17, "`ipath_checkout`" on page F-7
Moved Performance and Management Tips section to the end of Cluster Administration. Reordered the subsections in this section.	"Performance Settings and Management Tips" on page 4-21

Draft

Removed references to Fedora 6; not supported in this release.	"Use Wrapper Scripts for Compiling and Linking" on page 5-6, "SDP Module Not Loading" on page D-8, "Cross-Compilation Issues" on page D-14
Added more information on processor affinity. Reorganized methods 1 and 2.	"Erratic Performance" on page D-11
sysctl parameters don't help TCP/SDP performance in all systems. Dropped reference to "TCP/SDP" in this section.	"Use `sysctl` to Configure Kernel Parameters" on page 5-21
Replace PathScale compilers with GNU compilers as default compilers.	"Getting Started with MPI" on page 5-3, "As noted in "Create the mpihosts File" on page 5-3, an `mpihosts` file (also called a machines file, nodefile, or hostsfile) has been created in your current working directory. This file names the nodes on which the node programs may run." on page 5-13, "Use Wrapper Scripts for Compiling and Linking" on page 5-6
Change reference to PathScale 3.0 release to 3.1	"Use Wrapper Scripts for Compiling and Linking" on page 5-6
Move "mpirun options summary" to new Appendix A.	Appendix A
Added/changed these options to mpirun options summary. Added `-H`, `-i`, `-job-info`, `-no-syslog`, `-statsmode` Updated `-np`, `-rcfile`, `psc-debug-level`, `-debug options`	"`mpirun` Options Summary" on page A-1
Added setup checklist.	"Step-by-Step Cluster Setup and MPI Usage Checklists" on page 3-1
Renamed Compiling and Linking section.	"Use Wrapper Scripts for Compiling and Linking" on page 5-6
Combined sections on configuring `ssh`.	"Configuring `ssh` and `sshd` Using `shosts.equiv`" on page 4-25, "Configuring for `ssh` Using `ssh-agent`" on page 4-27
Main section on MPI File I/O removed; use only subsections Linux File I/O in MPI Programs, and MPI-IO with ROMIO to new Introduction in the beginning of the MPI section.	"Introduction" on page 5-1 "Linux File I/O in MPI Programs" on page 5-2 "MPI-IO with ROMIO" on page 5-2

Draft

Clarified descriptions of:
`-long-len, -L [default: 64000]`
`-long-len-shmem, -s [default: 16000]`
	"`mpirun` Tunable Options" on page 5-22

Split MPI chapter into two parts: "Using QLogic MPI" and "Using Other MPIs"	"Using QLogic MPI" on page 5-1, "Using Other MPIs" on page 6-1
More introductory material about MPI.	"Using QLogic MPI" on page 5-1,
Added introductory remarks about `mpirun`.	"Using `mpirun`" on page 5-15,
New sections for the other MPIs	"Open MPI" on page 6-3 "HP-MPI" on page 6-7 "Platform (Scali) MPI" on page 6-8 "Intel MPI" on page 6-10 "Improving Performance of Other MPIs Over IB Verbs" on page 6-12
Updated Other Supported MPI Implementations table	Table 6-1 on page 6-1
Deleted ***NOTE*** about using `mpi-selector`; information is now contained in table.	Was below Table 6-1 on page 6-1
Added new issue to Troubleshooting	"`ibsrpdm` Command Hangs When Two HCAs are Installed but Only Unit 1 is Connected to the Switch" on page D-8
Added new issue to Troubleshooting	"Number of Processes Exceeds `ulimit` for Number of Open Files" on page D-20
Added Glossary entry for Verbs.	"Verbs" on page Glossary-6
Added Glossary entry for PAT.	"PAT" on page Glossary-4
InfiniPath Cluster Administration renamed to InfiniPath Cluster Setup and Administration.	"InfiniPath Cluster Setup and Administration" on page 4-1
Location of kernel modules changed to: `/lib/modules/$(uname -r)/updates/kernel/drivers/infiniband/hw/ipath` Also deleted last paragraph after the pathname above. (Information about replacing kernel modules now obsolete). Also added that documentation can be downloaded from the web site; it is not in /usr/share/doc/infinipath.	"Installed Layout" on page 4-2

Draft

Deleted reference to MTRR BIOS setting; obsolete.	"BIOS Settings" on page 4-5 and "Performance Settings and Management Tips" on page 4-21
HP-MPI and Scali now run over Verbs.	"Other MPIs" on page 5-2
ssh has limit on number of concurrent connections.	"Configuring for ssh" on page 4-25
Added new Troubleshooting issue related to removed network interface ipath_ether	"Outdated ipath_ether Configuration Setup Generates Error" on page D-8
New section in Troubleshooting.	"General Error Messages" on page D-25
New troubleshooting issue.	"Problem with Shell Special Characters and Wrapper Scripts" on page D-17
Renamed Performance and Management Tips section.	"Performance Settings and Management Tips" on page 4-21
Renamed and moved Checking Software Status section.	"Checking Cluster and Software Status" on page 4-29
Added default values, if used, to table of environment variables.	Table 5-7 on page 5-18
Clarified `mpihosts` file format	"`mpihosts` File Details" on page 5-13
Update version numbers on other MPIs.	"Other MPIs" on page 5-2
Renamed Multiprocessor Nodes to Process Allocation. This section now consolidates information about process allocation, and hardware and software contexts. The entire section has been updated.	"Process Allocation" on page 5-10
Added error messages about software contexts.	"Enabling and Disabling Software Context Sharing" on page 5-11
Clarified that the second example uses the same `mpihosts` file.	"Compile and Run An Example C Program" on page 5-3
Added new appendix.	"Write Combining" on page E-1
Updated output for `ipath_control`. Also took out information for `-d` option, since it is outdated.	"`ipath_control`" on page F-8
Updated output for modprobe.	"`modprobe`" on page F-11
Updated output for boardversion. Removed ***NOTE*** info about QHT7040, as it is obsolete. Corrected that it reports version of the chip architecture, not installed software.	"`boardversion`" on page F-16
Removed `ident` string for non-qlogic built kernel modules	"`ident`" on page F-6

Draft

Update `rpm` section	"`rpm`" on page F-13
Removed `/etc/sysconfig/ics_inic.cfg` entry from table. `/etc/infini-band/qlgc_vnic.cfg` is the current file to use. Modified entry for `/etc/infini-band/qlgc_vnic.cfg`.	Table F-7 on page F-18
Rearranged Driver configuration information to match *QLogic HCA and QLogic OFED Software Install Guide*.	"InfiniPath and OpenFabrics Driver Overview" on page 4-5, "OpenFabrics Drivers and Services Configuration and Startup" on page 4-6, "Managing the InfiniPath Driver" on page 4-18
Add new section about `ibv_devinfo`.	"`ibv_devinfo`" on page 4-30
Updated output from programs.	"Checking Cluster and Software Status" on page 4-29
Changed MPI over InfiniBand to Ohio State University MVAPICH (1.1).	"Other MPIs" on page 5-2
Deleted OMP_NUM_THREADS from table. QLogic MPI does not use it. Also add information about it in MPI chapter.	Table 5-7 and "QLogic MPI and Hybrid MPI/OpenMP Applications" on page 5-24
Modified Rebuild or Reinstall Drivers if Different Kernel Installed section.	"Rebuild or Reinstall Drivers if Different Kernel Installed" on page D-3
New Table Notes.	Table 6-1 on page 6-1
Updated content to remove `ipath_ether` and change to "the OFED ULPs, such as IPoIB"	"Performance Warning if `ib_ipath` Shares Interrupts with `eth0`" on page D-12

Pathname changed from:
`sys/bus/pci/drivers/ib_ipath/<number>/`
to:
`/sys/class/infiniband/ipath0/device/`
	F Useful Programs and Files

Updated performance numbers for Benchmarks One and Two	B Benchmark Programs
Added new table for hardware and software contexts on each adapter.	Table 5-6
Added ***NOTE***. TotalView debugger can be used with Open MPI in this release.	"Debugging MPI Programs" on page 5-25
Removed this sentence: "This issue will be fixed in the next release"	"Large Message Receive Side Bandwidth Varies with Socket Affinity on Opteron Systems" on page D-10

Draft

Notes

Draft

Table of Contents

Draft

Draft

Draft

Draft

B Benchmark Programs

C Integration with a Batch Queuing System

D Troubleshooting

Draft

Draft

F Useful Programs and Files

G Recommended Reading

Draft

List of Figures

List of Tables

Draft

1 Introduction

This section describes the objectives, intended audience, and organization of the *QLogic HCA and QLogic OFED Software User Guide*.

The *QLogic HCA and QLogic OFED Software User Guide* shows end users how to use their InifiniPath cluster. End users include both the cluster administrator and the Message-Passing Interface (MPI) application programmers, who have different but overlapping interests in the details of the technology.

For specific instructions about installing the QLogic QLE7140, QLE7240, and QLE7280 PCI Express™ (PCIe) adapters, the QHT7140/QHT7040 HyperTransport eXpansion (HTX™) adapters, and the initial installation of the InfiniPath Software, see the *QLogic HCA and QLogic OFED Software Install Guide*.

Who Should Read this Guide

This guide is intended for end users responsible for administration of an InfiniPath cluster network as well as for end users who want to use that cluster.

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

How this Guide is Organized

The *QLogic HCA and QLogic OFED Software User Guide* is organized into these sections:

- Section 1, Introduction
- Section 2, Feature Overview, lists the features for this release, the supported QLogic adapter models, the supported distributions and kernels, and a list of the software components.
- Section 3, Step-by-Step Cluster Setup and MPI Usage Checklists, describes how to setup your cluster to run high-performance MPI jobs.
- Section 4, InfiniPath Cluster Setup and Administration, describes the lower levels of the supplied InfiniPath software. This section is of interest to an InfiniPath cluster administrator.

Draft

- Section 5, Using QLogic MPI, helps the MPI programmer make the best use of the QLogic MPI implementation. Examples are provided for compiling and running MPI programs.
- Section 6, Using Other MPIs, gives examples for compiling and running MPI programs with other MPI implementations.
- Appendix A, `mpirun` Options Summary
- Appendix B, Benchmark Programs
- Appendix C, Integration with a Batch Queuing System
- Appendix D, Troubleshooting, provides information for troubleshooting installation, cluster administration, and MPI
- Appendix E, Write Combining
- Appendix F, Useful Programs and Files, contains many useful programs and files for debugging, as well as commands for common tasks.
- Appendix G, Recommended Reading
- Glossary defines the technical terms used in this document.
- Index lists major subjects and concepts with page numbers for easy reference.

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

Overview

The material in this documentation pertains to a QLogic OFED *cluster*. A cluster is defined as a collection of nodes, each attached to an InfiniBand™-based fabric through the QLogic interconnect. The nodes are Linux-based computers, each having up to 16 processors.

The QLogic HCAs are InfiniBand 4X. The Double Data Rate (DDR) QLE7240 and QLE7280 adapters have a raw data rate of 20Gbps (data rate of 16Gbps). For the Single Data Rate (SDR) adapters, the QLE7140 and QHT7140, the raw data rate is 10Gbps (data rate of 8Gbps). The QLE7240 and QLE7280 can also run in SDR mode.

The QLogic adapters utilize standard, off-the-shelf InfiniBand 4X switches and cabling. The QLogic interconnect is designed to work with all InfiniBand-compliant switches.

Draft

NOTE:

If you are using the QLE7240 or QLE7280, and want to use DDR mode, then DDR-capable switches must be used.

QLogic OFED OpenFabrics software is interoperable with other vendors' InfiniBand Host Channel Adapters (HCAs) running compatible OpenFabrics releases. There are several options for subnet management in your cluster:

- Use the embedded Subnet Manager (SM) in one or more managed switches supplied by your InfiniBand switch vendor.
- Use a host-based Subnet Manager. QLogic provides one, HSM, as a part of the InfiniBand fabric Suite download.
- Use the Open source Subnet Manager (OpenSM) component of OpenFabrics.

Interoperability

QLogic InfiniPath participates in the standard InfiniBand subnet management protocols for configuration and monitoring. Note that:

- InfiniPath OpenFabrics (including Internet Protocol over InfiniBand (IPoIB)) is interoperable with other vendors' InfiniBand HCAs running compatible OpenFabrics releases.
- The QLogic MPI stack is not interoperable with other InfiniBand HCAs and Target Channel Adapters (TCAs). Instead, it uses an InfiniBand-compliant, vendor-specific protocol that is highly optimized for QLogic MPI and for MPI over verbs.

NOTE:

See the OpenFabrics web site at www.openfabrics.org for more information on the OpenFabrics Alliance.

Draft

Conventions Used in this Guide

This guide uses the typographical conventions listed in Table 1-1.

Table 1-1. 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 and concepts, as well as for documentation names/titles.
`user input`	Bold fixed-space font is used for literal items in commands or constructs that you type.
`$`	Indicates a command line prompt.
`#`	Indicates a command line prompt as root.
[ ]	Brackets enclose optional elements of a command or program construct.
...	Ellipses indicate that a preceding element can be repeated.
>	A right caret identifies the cascading path of menu commands used in a procedure.
QLogic OFED 1.4	The current version number of the software is included within this documentation.
NOTE:	Indicates important information.

Documentation

The product documentation includes:

- *The QLogic HCA and QLogic OFED Software Install Guide*
- *The QLogic HCA and QLogic OFED Software User Guide*
- The *InfiniBand Software Installation Guide* (for information on QLogic InfiniBand Fabric Suite)
- The *OFED+ Users Guid*e (for information on QLogic VNIC and QLogic SRP)
- Release Notes

Draft

- Quick Start Guide
- Readme file

Contact Information

Support Headquarters	QLogic Corporation 4601 Dean Lakes Blvd Shakopee, MN 55379 USA
QLogic Web Site	www.qlogic.com
Technical Support Web Site	support.qlogic.com
Technical Support Email	support@qlogic.com
Technical Training Email	tech.training@qlogic.com
Additional contact information is available from the Contact Support area of the Technical Support Web Site.	

Draft

Notes

Draft

2 Feature Overview

This section contains the features for this release, the supported QLogic adapter models, supported distributions and kernels, and a list of the software components.

Features

The QLogic OFED 1.4 software release contains the complete OFED 1.4, plus additional QLogic improvements, including an enhanced QLogic HCA driver. The InfiniPath 2.3 components (libraries, QLogic MPI/PSM, and utilities) are also included. QLogic also supplies MVAPICH and OpenMPI compiled with newer versions of each of four different compilers (GCC, PGI, Intel and PathScale).

The following features and enhancements are included in the QLogic OFED 1.4 release:

- Installation improvements. Provides a single software load for InfiniBand HCAs from QLogic and other vendors supported by OFED. the software is available packaged in the following ways:
  - Text User Interface (TUI) installer available (with the QLogicIB-Basic* download). TUI is used for install on smaller clusters. Software can be installed either standalone or via FastFabric (if the QLogic InfiniBand Fabric Suite is purchased).
  - Software packaged for use with `rpm` install method.
  - A subset of the software (the accelerated MPI stack, precompiled versions of MVAPICH and Open MPI, and other user-level tools) can be installed on top of stock OFED or on an IB-enabled distribution.
  - Software packaged for Rocks installation method.
  - Software packaged for Platform OCS installation method
- Write-combining (WC) mappings for the PIO buffers is now configured by default using the x86 Page Attribute Table (PAT) mechanism.

Draft

- MVAPICH and OpenMPI compiled with newer versions of each of four different compilers (GCC, PGI, Intel and PathScale) are available.
- The QLogic InfiniBand Fabric Suite (IFS) is available separately for purchase. It includes FastFabric, the QLogic Host Subnet Manager (HSM), and the Fabric Viewer, and the InfiniServ Host Software.The QLogic OFED 1.4 software is supported by IFS.
- Support for newer compiler versions (PathScale 3.x, PGI 7.x, PGI 8.x, Intel 10.x, Intel 11.x)
- Support for newer Linux distributions, including RHEL 4 U7
- Performance enhancements and bug fixes

Other Changes

- `ipath_ether` Ethernet emulation has been removed; IPoIB-CM may be used instead.
- The `/etc/init.d/infinipath` command to start the InfiniPath service has been replaced by the `/etc/init.d/openibd` command.
- The `infinipath-kernel` RPM no longer exists: it has been integrated into the `kernel-ib` RPM.

Continued Support

- Multiple high-performance native PSM Message Passing Interface (MPI) implementations. (PSM is QLogic's accelerated library for high performance MPIs). In addition to QLogic MPI, the currently supported MPI implementations are HP-MPI, Open MPI, MVAPICH, and Scali (Platform). Open MPI provides MPI-2 functionality, including one-sided operations and dynamic processes. These all offer the same high performance as QLogic MPI.
- Dual PCIe QLogic adapters per node.
- QLogic MPI supports running exclusively on a single node without the installation of the HCA hardware.
- 4K Maximum Transfer Unit (MTU) is supported and is on by default. To take advantage of 4KB MTU, use a switch that supports 4KB MTU. QLogic also supports 2KB switches, and 4KB MTU switches configured for 2KB MTU. QLogic switches with firmware version 4.2.x or later are recommended.

Draft

This version of the QLogic OFED software provides support for all of the QLogic HCAs in Table 2-1.

Table 2-1. QLogic Adapter Model Numbers

QLogic Model Number	Description
QHT7040	Single port 10Gbps SDR 4X InfiniBand to HTX adapter. For systems with HTX expansion slots.
QHT7140[a]	Single port 10Gbps SDR 4X InfiniBand to HTX adapter. For systems with HTX expansion slots.
QLE7140	Single port 10Gbps SDR 4X InfiniBand to PCI Express x8 adapter. Supported on systems with PCI Express (PCIe) x8 or x16 slots.
QLE7240	Single port 20Gbps DDR 4X InfiniBand to PCI Express x8 adapter. Supported on systems with PCI Express x8 or x16 slots.
QLE7280	Single port 20Gbps DDR 4X InfiniBand to PCI Express x16 adapter. Supported on systems with PCI Express x16 slots. The QLE7280 is backward compatible; it can also be used with PCIe adapters that connect to x8 slots.

Table Notes

PCIe is Gen 1

[a] The QHT7140 has a smaller form factor than the QHT7040, but is otherwise the same. Throughout this document, the QHT7040 and QHT7140 will be collectively referred to as the *QHT7140* unless otherwise noted.

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

- The high performance computing page at www.qlogic.com/Products/HPC_products_landingpage.aspx
- The InfiniBand HCA page at www.qlogic.com/Products/HPC_products_infipathhcas.aspx

Supported Distributions and Kernels

The QLogic interconnect runs on AMD™ Opteron™ and 64-bit Intel Xeon systems running Linux®. The currently supported distributions and associated Linux kernel versions for InfiniPath and OpenFabrics are listed in Table 2-2.

The kernels are the ones that shipped with the distributions. All are for the x86_64 architecture.

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

Distribution	InfiniPath/OpenFabrics Supported Kernels
Red Hat® Enterprise Linux® (RHEL) 4.5	2.6.9-55 (U5),
RHEL 4.6	2.6.9-67 (U6)
RHEL 4.7	2.6.9-78 (U7)
CentOS 4.5	2.6.9.55
CentOS 4.6	2.6.9-67
CentOS 4.7	2.6.9-78
Scientific Linux 4.5	2.6.9.55
Scientific Linux 4.6	2.6.9-67
Scientific Linux 4.7	2.6.9-78
Red Hat Enterprise Linux 5.1 (RHEL 5.1)	2.6.18-53, 2.6.18-92
RHEL 5.2	2.6.18-92
CentOS 5.1	2.6.18-53, 2.6.18-92
CentOS 5.2	2.6.18-92
Scientific Linux 5.1	22.6.18-53, 2.6.18-92
Scientific Linux 5.2	2.6.18-92
SUSE® Linux Enterprise Server 10 SP 1	2.6.16.46
SUSE® Linux Enterprise Server 10 SP 2	2.6.16.60

NOTE:

Support for RHEL4 U4 and SLES 10.0 has been removed.

Compiler Support

QLogic MPI supports use of a number of compilers. These include:

- GNU gcc 3.3.*x*, 3.4.*x*, 4.0, 4.1, 4.2.x, and 4.3.x compiler suites
- PathScale Compiler Suite 3.0, 3.1 and 3.2
- PGI 5.2, 6.0. 7.1, 7.2-4, and 8.0-3
- Intel 9.x, 10.1, and 11.0
- gfortran 4.1.x

Draft

The PathScale Compiler Suite Version 3.*x* is now supported on systems that have the GNU 4.0 and 4.1 compilers and compiler environment (header files and libraries).

Software Components

This release includes all of OFED 1.4 with enhancements (QLogic OFED 1.4), including a new version of the VNIC tools and driver, and support for the QHT7xxx and QLE7xxx adapters. The software includes the QLogic InfiniPath HCA driver, libraries, QLogic MPI, Subnet Management Agent, and associated utilities. Included components are:

- InfiniPath driver
- InfiniPath libraries, InfiniPath utilities, configuration, and support tools, including `ipath_checkout`, `ipath_control`, `ipath_pkt_test`, and `ipathstats`
- QLogic MPI
- PSM support for accelerated MPI
- OpenMPI and MVAPICH (with PSM support) built with the GNU, PGI, PathScale, and Intel compilers, with corresponding `mpitests` and `mpi-selector`
- QLogic MPI benchmarks and utilities
- OpenFabrics protocols
- OpenFabrics libraries and utilities
- QLogic VNIC module
- FastFabric Enablement tools

This release provides support for the following protocols and transport services:

- IPoIB (TCP/IP networking in either Connected or Datagram mode)
- Sockets Direct Protocol (SDP)
- Open source Subnet Manager (OpenSM)
- Reliable Datagram Sockets (RDS)
- iSCSI Extensions for RDMA (iSER)

This release supports two versions of SCSI RDMA Protocol (SRP):

- OFED SRP
- QLogic SRP

No support is provided for Reliable Datagram (RD).

Draft

Notes

Draft

3 Step-by-Step Cluster Setup and MPI Usage Checklists

This section describes how to set up your cluster to run high-performance MPI jobs.

Cluster Setup

Perform the following tasks when setting up the cluster. These include BIOS, adapter and system settings.

1. Make sure that hardware and software installation and driver configuration has been completed according to the instructions in the *QLogic HCA and QLogic OFED Software Install Guide*. To minimize management problems, the compute nodes of the cluster must have very similar hardware configurations and identical software installations. See “Homogeneous Nodes” on page 4-22 for more information.
2. Check that the BIOS is set properly. See “BIOS Settings” on page 4-5.
3. Adjust settings, including setting appropriate MTU size. See “Adapter and Other Settings” on page 4-22 for more information.
4. Remove unneeded services. QLogic recommends turning `irqbalance` off. This is covered in “Remove Unneeded Services” on page 4-23.
5. Disable powersaving features. Instructions are given in “Disable Powersaving Features” on page 4-24.
6. Check other performance tuning settings. See “Performance Settings and Management Tips” on page 4-21.
7. If using Intel processors, turn off Hyper-Threading. See “Hyper-Threading” on page 4-24.
8. Set up the host environment to use `ssh`. Two methods are discussed in “Host Environment Setup for MPI” on page 4-24.
9. Verify the cluster setup. See “Checking Cluster and Software Status” on page 4-29. Intel cluster users can use the Intel Cluster Checker. See “The Intel Cluster Checker” on page 4-31.

Draft

Using MPI

1. Verify that the QLogic hardware and software has been installed on all the nodes you will be using, and that `ssh` is set up on your cluster (see all the steps in the checklist above).
2. Copy the examples to your working directory. See "Copy Examples" on page 5-3.
3. Make an `mpihosts` file that lists the nodes on which your programs will run. See "Create the `mpihosts` File" on page 5-3.
4. Compile the example C program using the default wrapper script `mpicc`. Use `mpirun` to run it. See "Compile and Run An Example C Program" on page 5-3.
5. Try the examples with other programming languages, C++, Fortran77, and Fortran90 in "The Examples Using Other Programming Languages" on page 5-5.
6. To test using other MPIs that run over PSM, such as MVAPICH, Open MPI, HP-MPI, Platform MPI, and Intel MPI, see "Using Other MPIs" on page 6-1.
7. To switch between multiple versions of Open MPI, MVAPICH, and QLogic MPI, use the `mpi-selector`. See "Managing Open MPI, MVAPICH and QLogic MPI with the `mpi-selector` Utility" on page 6-6.
8. Refer to "QLogic MPI Details" on page 5-5 for more information about QLogic MPI, and to "Performance Tuning" on page 5-21 to read more about runtime performance tuning.
9. Refer to "Using Other MPIs" on page 6-1 to learn about using other MPI implementations.

Draft

4 InfiniPath Cluster Setup and Administration

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

Introduction

The InfiniPath driver `ib_ipath`, Open source Subnet Manager (OpenSM), the protocol and Message-Passing Interface (MPI) support libraries, and other modules are components of the InfiniPath software. This software provides the foundation that supports the MPI implementation.

Figure 4-1 illustrates these relationships. Note that HP-MPI, Scali, MVAPICH, and Open MPI can run either over PSM or OpenFabrics User Verbs. The QLogic Virtual Network Interface Controller (VNIC) driver module is also illustrated in the figure.

Draft

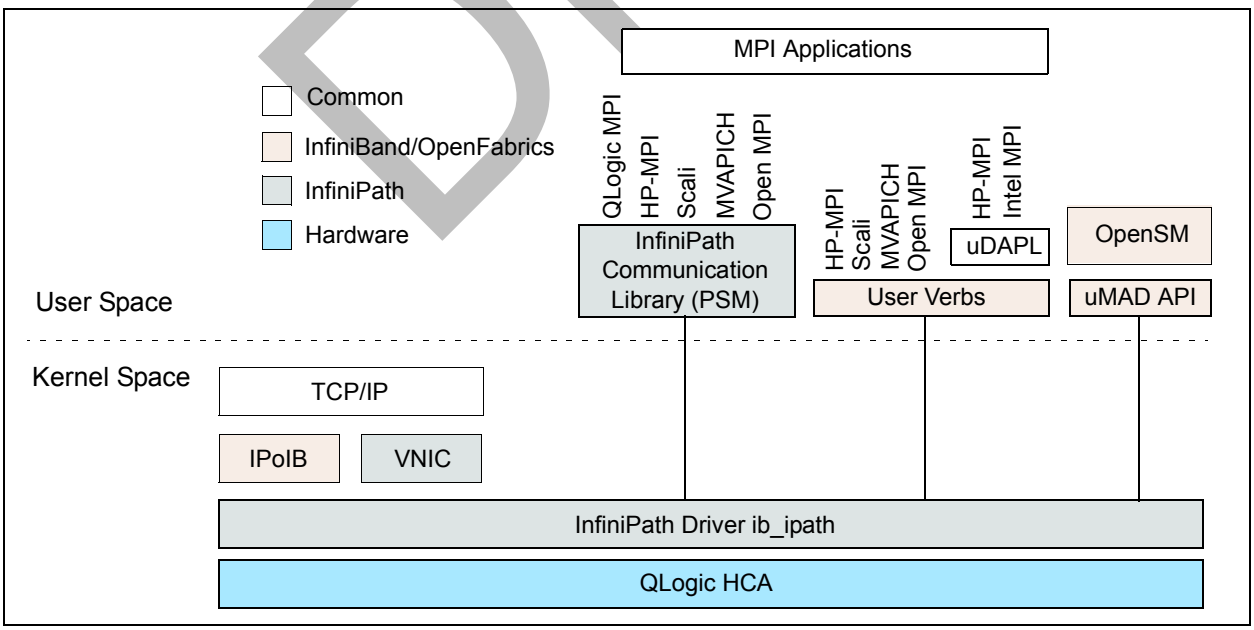

Figure 4-1. InfiniPath Software Structure

Installed Layout

The default installed layout for the InfiniPath software and QLogic-supplied MPIs is described here.

The 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 the source for several MPI benchmarks are in:

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

NOTE:

If QLogic MPI is installed in an alternate location, the argument passed to `--prefix (/usr/mpi/qlogic)` replaces the default `/usr` prefix. QLogic MPI binaries, documentation, and libraries are installed under that prefix. However, a few configuration files are installed in `/etc` regardless of the desired `--prefix`. The remaining InfiniPath libraries and tools stay their default installation location.

If you have installed the software into an alternate location, the $`MPICH_ROOT` environment variable needs to match `--prefix`.

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

```
/usr/bin
```

Documentation is found in:

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

Note that license information only is found in `usr/share/doc/infinipath`. InfiniPath user documentation can be found on the QLogic web site on the software download page for your distribution.

Configuration files are found in:

```
/etc/sysconfig
```

Init scripts are found in:

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

Draft

The InfiniPath driver modules in this release are installed in:

```
/lib/modules/$(uname -r)/updates/kernel/drivers/infiniband/hw/ipath
```

Most of the other OFED modules are installed under the `infiniband` subdirectory, above. Some others are installed under:

```
/lib/modules/$(uname -r)/updates/kernel/drivers/net
```

The RDS modules are installed under:

```
/lib/modules/$(uname -r)/updates/kernel/net/rds
```

QLogic-supplied OpenMPI and MVAPICH RPMs with PSM support and compiled with GCC, PathScale, PGI, and the Intel compilers are now installed in directories using this pattern:

```
/usr/mpi/<compiler>/<mpi>-<mpi_version>-qlc
```

For example:

```
/usr/mpi/gcc/openmpi-1.2.8-qlc
```

Memory Footprint

This section contains a preliminary guideline for estimating the memory footprint of the QLogic adapter on Linux x86_64 systems. Memory consumption is linear, based on system configuration. OpenFabrics support is under development and has not been fully characterized. Table 4-1 summarizes the guidelines.

Table 4-1. Memory Footprint of the QLogic Adapter on Linux x86_64 Systems

Adapter Component	Required/ Optional	Memory Footprint	Comment
InfiniPath driver	Required	9 MB	Includes accelerated IP support. Includes table space to support up to 1000 node systems. Clusters larger than 1000 nodes can also be configured.
MPI	Optional	68 MB per process + 264 bytes × num_remote_procs: 68 MB = 60 MB (base) + 512 × 2172 (sendbufs) + 1024×1K (misc allocations) + 6 MB (shared memory)	Several of these parameters (sendbufs, recvbufs and size of the shared memory region) are tunable if you want a reduced memory footprint.

Draft

Table 4-1. Memory Footprint of the QLogic Adapter on Linux x86_64 Systems (Continued)

Adapter Component	Required/ Optional	Memory Footprint	Comment
OpenFab-rics	Optional	1~6 MB + ~500 bytes per QP + TBD bytes per MR + ~500 bytes per EE + OpenFabrics stack from openfabrics.org (size not included in these guidelines)	This component has not been fully characterized at the time of publication.

The following paragraphs provide an example for a 1024 processor system:

- 1024 cores over 256 nodes (each node has 2 sockets with dual-core processors).
- One adapter per node
- Each core runs an MPI process, with the four 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 example breaks down to a memory footprint of 290 MB per node, as shown in Table 4-2.

Table 4-2. Memory Footprint, 290 MB per Node

Component	Footprint (in MB)	Breakdown
Driver	9	Per node
MPI	273	4×68 MB (MPI per process including shared memory) + 4×264×1020 (for 1020 remote ranks)
OpenFabrics	8	6 MB + 1024 × 200 KB per node

Draft

BIOS Settings

To achieve the best performance with QLogic adapters, you need to configure your BIOS with specific settings. 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.

This setting is required:

- Advanced Configuration and Power Interface (ACPI) needs to be enabled.

If ACPI has been disabled, it may result in initialization problems, as described in "InfiniPath Interrupts Not Working" on page D-3.

Some other BIOS settings can be adjusted for better adapter performance. See "Performance Settings and Management Tips" on page 4-21.

For specific instructions about BIOS settings, follow the hardware documentation that came with your system.

NOTE:

The x86 Page Attribute Table (PAT) mechanism that allocates write-combining (WC) mappings for the PIO buffers has been added and is now the default. This was previously a BIOS setting. For more information, see "Write Combining" on page E-1.

InfiniPath and OpenFabrics Driver Overview

The InfiniPath `ib_ipath` module provides low level QLogic hardware support, and is the base driver for both MPI/PSM programs, and general OpenFabrics protocols such as IPoIB and SDP. The driver also supplies the Subnet Management Agent (SMA) component.

Optional configurable OpenFabrics components and their default settings at startup are:

- IPoIB network interface. Required for TCP/IP networking for running Ethernet traffic over the InfiniPath link. It is not running until it is configured.
- VNIC. It is not running until it is configured.
- OpenSM. It is disabled on startup. You can either install it on only one node, or disable it on all nodes except where it will be used as an SM.

Draft

- SRP (OFED and QLogic modules). SRP is not running until the module is loaded and the SRP devices on the fabric have been discovered.
- MPI over uDAPL (can be used by Intel MPI or HP-MPI). IPoIB needs to be configured before MPI over uDAPL can be set up.

Other optional drivers can now be configured and enabled, as described in "OpenFabrics Drivers and Services Configuration and Startup" on page 4-6.

Complete information about starting, stopping, and restarting the InfiniPath services are in "Managing the InfiniPath Driver" on page 4-18.

OpenFabrics Drivers and Services Configuration and Startup

IPoIB, VNIC, OpenSM, SRP, and MPI over uDAPL configuration and startup is explained in more detail in the following sections.

Configuring the IPoIB Network Interface

The following instructions show you how to manually configure your OpenFabrics IPoIB network interface. This example assumes that you are using `sh` or `bash` as your shell, all required InfiniPath and OpenFabrics RPMs are installed, and your startup scripts have been run (either manually or at system boot).

For this example, the 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, the host being configured has the IP address 10.1.17.3, and DHCP is not used.

NOTE:

Instructions are only for this static IP address case. Configuration methods for using DHCP will be supplied in a later release.

1. Type the following commands (as root):

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

2. To verify the configuration, type:

```
# ifconfig ib0
```

Draft

The output from this command will 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:4096  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)
```

3. Type:

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

The output of the `ping` command will be similar to the following, 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.

4. Restart (as root) by typing:

```
# /etc/init.d/openibd restart
```

NOTE:

- The configuration must be repeated each time the system is rebooted.
- IPoIB-CM (Connected Mode) is enabled by default. The setting in `/etc/infiniband/openib.conf` is `SET_IPOIB_CM=yes`. To use datagram mode use `SET_IPOIB_CM=no`.

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 the cluster that will act as a subnet manager. You do not need to use OpenSM if any of your InfiniBand switches provide a subnet manager, or if you are running a host-based SM.

Draft

If you are using the Installer tool, you can set the OpenSM default behavior at the time of installation.

If you are using the `rpm` install method, note that after installing the `opensm` package, OpenSM is configured to be *off* after the next machine reboot. It only needs to be enabled on the node that acts as the subnet manager, so use the `chkconfig` command (as root) to enable it on the node where it is to run:

```
# chkconfig opensmd on
```

The command to disable it on reboot is:

```
# chkconfig opensmd off
```

You can start `opensmd` without rebooting your machine by typing:

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

You can stop `opensmd` again like this:

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

If you want to pass any arguments to the OpenSM program, modify the following file, and add the arguments to the `OPTIONS` variable:

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

For example:

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

For more information on OpenSM, see the OpenSM `man` pages, or look on the OpenFabrics web site.

SRP

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

In this release, two versions of SRP are available: QLogic SRP and OFED SRP. QLogic SRP is available as part of the QLogicIB-Basic*, Rocks Roll, and Platform OCS downloads.

It is not available as a part of the RPM downloads.

SRP has been tested on targets from Engenio™ (now LSI Logic®).

NOTE:

Before using SRP, the SRP targets must already be set up by your system administrator.

Draft

Using QLogic SRP

If you have installed QLogic SRP as part of the QLogicIB-Basic download, you will need to configure it according to the steps shown in the *QLogic ULP and Tools Reference Guide (OFED+ Users Guide).*

Using OFED SRP

To use OFED SRP, follow these steps:

1. Add the line `SRP_LOAD=yes` to the module list in `/etc/infiniband/openib.conf` to have it automatically loaded.
2. Discover the SRP devices on your fabric by running this command (as root):

   ```
   # ibsrpdm
   ```

   In the output, look for lines similar to these:

   ```
   GUID:      0002c90200402c04
   ID:        LSI Storage Systems SRP Driver 200400a0b8114527
   service entries: 1
   service[  0]: 200400a0b8114527 / SRP.T10:200400A0B8114527

   GUID:      0002c90200402c0c
   ID:        LSI Storage Systems SRP Driver 200500a0b8114527
           service entries: 1
   service[  0]: 200500a0b8114527 / SRP.T10:200500A0B8114527

   GUID:      21000001ff040bf6
   ID:        Data Direct Networks SRP Target System
           service entries: 1
   service[  0]: f60b04ff01000021 / SRP.T10:21000001ff040bf6
   ```

   Note that not all the output is shown here; key elements are expected to show the match in Step 3.
3. Choose the device you want to use, and run the command again with the `-c` option (as root):

   ```
   # ibsrpdm -c
   id_ext=200400A0B8114527,ioc_guid=0002c90200402c04,dgid=fe8000
   00000000000002c90200402c05,pkey=ffff,service_id=200400a0b8114
   527
   id_ext=200500A0B8114527,ioc_guid=0002c90200402c0c,dgid=fe8000
   00000000000002c90200402c0d,pkey=ffff,service_id=200500a0b8114
   527
   id_ext=21000001ff040bf6,ioc_guid=21000001ff040bf6,dgid=fe8000
   0000000000021000001ff040bf6,pkey=ffff,service_id=f60b04ff01000
   021
   ```

Draft

4. Find the result that corresponds to the target you want, and `echo` it into the `add_target` file:

```
# echo
"id_ext=21000001ff040bf6,ioc_guid=21000001ff040bf6,dgid=fe800
000000000021000001ff040bf6,pkey=ffff,service_id=f60b04ff0100
0021,initiator_ext=0000000000000001" >
/sys/class/infiniband_srp/srp-ipath0-1/add_target
```

5. You can look for the newly created devices in the `/proc/partitions` file. The file will look similar to this example (the partition names may vary):

```
# cat /proc/partitions
major minor  #blocks  name
8     64  142325760 sde
8     65  142319834 sde1
8     80   71162880 sdf
8     81   71159917 sdf1
8     96      20480 sdg
8     97      20479 sdg1
```

6. Create a mount point (as root) where you will mount the SRP device. For example:

```
# mkdir /mnt/targetname
# mount /dev/sde1 /mnt/targetname
```

NOTE:

Use `sde1` rather than `sde`. See the `mount(8)` man page for more information on creating mount points.

Configuring and Administering the VNIC Interface

The VirtualNIC (VNIC) Upper Layer Protocol (ULP) works in conjunction with firmware running on Virtual Input/Output (VIO) hardware such as the SilverStorm Ethernet Virtual I/O Controller (EVIC™) or the InfiniBand/Ethernet Bridge Module for IBM® BladeCenter®, providing virtual Ethernet connectivity.

The VNIC driver along with the QLogic EVIC's two 10 Gigabit ethernet ports, enables Infiniband clusters to connect to Ethernet networks. This driver also works with the earlier version of the I/O Controller, the VEx.

The QLogic VNIC driver creates virtual Ethernet interfaces and tunnels the Ethernet data to/from the EVIC over InfiniBand using an InfiniBand reliable connection.

The virtual Ethernet interface supports any Ethernet protocol. It operates normally like any other interface--`ping`, `ssh`, `scp`, `netperf` etc.

Draft

The VNIC interface must be configured before it can be used. To do so, perform the following steps:

1. Discover the EVIC/VEx Input/Output Controllers (IOCs) present on the fabric using `ib_qlgc_vnic_query`. For writing the configuration file, you will need information about the EVIC/VEx IOCs present on the fabric, such as their IOCGUID, IOCSTRING, etc. Use the `ib_qlgc_vnic_query` tool to get this information.

   When `ib_qlgc_vnic_query` is executed without any options, it displays detailed information about all the EVIC/VEx IOCs present on the fabric. Run it as root. For example:

```
# ib_qlgc_vnic_query
HCA No = 0, HCA = mlx4_0, Port = 1, Port GUID = 0x0002c903000010f9,
State = Active
    IO Unit Info:
        port LID:        0009
        port GID:        fe8000000000000000066a11de000070
        change ID:       0003
        max controllers: 0x02

    controller[  1]
        GUID:      00066a01de000070
        vendor ID: 00066a
        device ID: 000030
        IO class : 2000
        ID:        EVIC in Chassis 0x00066a00db00001e, Slot 1, Ioc 1
        service entries: 2
           service[  0]: 1000066a00000001 /
                         InfiniNIC.InfiniConSys.Control:01
           service[  1]: 1000066a00000101 /
                         InfiniNIC.InfiniConSys.Data:01

    IO Unit Info:
port LID:        000b
port GID:        fe8000000000000000066a21de000070
change ID:       0003
max controllers: 0x02
```

Draft

```
    controller[  2]
        GUID:      00066a02de000070
        vendor ID: 00066a
        device ID: 000030
        IO class : 2000
        ID:        EVIC in Chassis 0x00066a00db00001e, Slot 1, Ioc 2
        service entries: 2
           service[  0]: 1000066a00000002 /
                         InfiniNIC.InfiniConSys.Control:02
           service[  1]: 1000066a00000102 /
                        InfiniNIC.InfiniConSys.Data:02

HCA No = 0, HCA = mlx4_0, Port = 2, Port GUID = 0x0002c903000010fa,
State = Active
IO Unit Info:
        port LID:        0009
        port GID:        fe8000000000000000066a11de000070
        change ID:       0003
        max controllers: 0x02

    controller[  1]
        GUID:      00066a01de000070
        vendor ID: 00066a
        device ID: 000030
        IO class : 2000
        ID:        EVIC in Chassis 0x00066a00db00001e, Slot 1, Ioc 1
        service entries: 2
           service[  0]: 1000066a00000001 /
                         InfiniNIC.InfiniConSys.Control:01
           service[  1]: 1000066a00000101 /
                         InfiniNIC.InfiniConSys.Data:01

    IO Unit Info:
        port LID:        000b
        port GID:        fe8000000000000000066a21de000070
        change ID:       0003
        max controllers: 0x02
```

Draft

```
controller[  2]
    GUID:      00066a02de000070
    vendor ID: 00066a
    device ID: 000030
    IO class : 2000
    ID:        EVIC in Chassis 0x00066a00db00001e, Slot 1, Ioc 2
    service entries: 2
       service[  0]: 1000066a00000002 /
                     InfiniNIC.InfiniConSys.Control:02
       service[  1]: 1000066a00000102 /
                     InfiniNIC.InfiniConSys.Data:02
```

NOTE:

A VIO hardware card can contain up to six IOCs (and therefore up to six IOCGUIDs); one for each Ethernet port on the VIO hardware card. Each VIO hardware card contains a unique set of IOCGUIDs: (e.g., IOC 1 maps to Ethernet Port 1, IOC 2 maps to Ethernet Port 2, IOC 3 maps to Ethernet Port 3, etc.).

2. Create the VNIC interfaces using the configuration file:

   `/etc/infiniband/qlgc_vnic.cfg.`

   Look at the `qlgc_vnic.cfg.sample` file to see how VNIC configuration files are written. It can be found with the OFED documentation, or in the `qlgc_vnictools` subdirectory of the QLogicIB_Basic download. You can use this configuration file as the basis for creating a configuration file by replacing the Destination Global Identifier (DGID), IOCGUID, and IOCSTRING values with those of the EVIC/VEx IOCs present on your fabric.

   QLogic recommends using the DGID of the EVIC/VEx IOC, as it ensures the quickest startup of the VNIC service. When DGID is specified, the IOCGUID must also be specified. For more details, see the `qlgc_vnic.cfg` sample file.

3. Edit the VirtualNIC configuration file, `/etc/infiniband/qlgc_vnic.cfg`. For each IOC connection, add a CREATE block to the file using the following format:

```
{CREATE; NAME="eioc2";
PRIMARY={IOCGUID=0x66A0130000105; INSTANCE=0; PORT=1; }
SECONDARY={IOCGUID=0x66A013000010C; INSTANCE=0; PORT=2;}
}
```

Draft

NOTE:

The `qlgc_vnic.cfg` file is case and format sensitive.

a. Format 1: Defining an IOC using the IOCGUID. Use the following format to allow the host to connect to a specific VIO hardware card, regardless of which chassis and/or slot the VIO hardware card resides:

```
{CREATE;
NAME="eioc1";
IOCGUID=0x66A0137FFFFE7;}
The following is an example of VIO hardware failover:
{CREATE; NAME="eioc1";
PRIMARY={IOCGUID=0x66a01de000003; INSTANCE=1; PORT=1; }
SECONDARY={IOCGUID=0x66a02de000003; INSTANCE=1; PORT=1;}
}
```

NOTE:

Do not create EIOC names with similar character strings (e.g., eioc3 and eioc30). There is a limitation with certain Linux operating systems that cannot recognize the subtle differences. The result is that the user will be unable to ping across the network.

b. Format 2: Defining an IOC using the IOCSTRING. Defining the IOC using the IOCSTRING allows VIO hardware to be hot-swapped in and out of a specific slot. The host attempts to connect to the specified IOC (1, 2, or 3) on the VIO hardware that currently resides in the specified slot of the specified chassis. Use the following format to allow the host to connect to a VIO hardware that resides in a specific slot of a specific chassis:

```
{CREATE;
NAME="eioc1";
IOCSTRING="Chassis 0x00066A0005000001, Slot 1, IOC 1";
RX_CSUM=TRUE;
HEARTBEAT=100; }
```

Draft

NOTE:

The IOCSTRING field is a literal, case-sensitive string. Its syntax must be exactly in the format shown in the previous example, including the placement of commas. To reduce the likelihood of syntax error, use the command `ib_qlgc_vnic_query -es`. Note that the chassis serial number must match the chassis Ox (Hex) value. The slot serial number is specific to the line card as well.

Each CREATE block must specify a unique NAME. The NAME represents the Ethernet interface name that will be registered with the Linux operating system.

c. Format 3: Starting VNIC using DGID. Following is an example of a DGID and IOCGUID VNIC configuration. This configuration allows for the quickest start up of VNIC service:

```
{CREATE; NAME="eioc1";
DGID=0xfe8000000000000066a0258000001;IOCGUID=0x66a0130
000001;
}
This example uses DGID, IOCGUID and IOCSTRING:
{CREATE; NAME="eioc1";
DGID=0xfe8000000000000066a0258000001;
IOCGUID=0x66a0130000001;
IOCSTRING="Chassis 0x00066A00010003F2, Slot 1, IOC 1";
}
```

4. Create VirtualNIC interface configuration files. For each Ethernet interface defined in the `/etc/sysconfig/qlgc_vnic.cfg` file, create an interface configuration file, `/etc/sysconfig/network-scripts/ifcfg-<NAME>` (or `/etc/sysconfig/network/ifcfg-<NAME>` on Linux 2.6 kernels), where <NAME> is the value of the NAME field specified in the CREATE block.

   Following is an example of `ifcfg-eiocx` setup for Red Hat systems:

```
DEVICE=eioc1
BOOTPROTO=static
IPADDR=172.26.48.132
BROADCAST=172.26.63.130
NETMASK=255.255.240.0
NETWORK=172.26.48.0
ONBOOT=yes
TYPE=Ethernet
```

Draft

Following is an example of `ifcfg-eiocx` setup for SuSE and SLES systems:

```
BOOTPROTO='static'
IPADDR='172.26.48.130'
BROADCAST='172.26.63.255'
NETMASK='255.255.240.0'
NETWORK='172.26.48.0'
STARTMODE='hotplug'
TYPE='Ethernet'
```

5. Start the QLogic VNIC driver and the QLogic VNIC interfaces. Once you have created a configuration file, you can start the VNIC driver and create the VNIC interfaces specified in the configuration file by running the following command (as root):

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

You can stop the VNIC driver and bring down the VNIC interfaces by running the following command:

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

To restart the QLogic VNIC driver, run the following command:

```
# /etc/init.d/qlgc_vnic restart
```

If you have not started the InfiniBand network stack (InfiniPath or OFED), then running the `/etc/init.d/qlgc_vnic start` command also starts the InfiniBand network stack, since the QLogic VNIC service requires the InfiniBand stack.

If you start the InfiniBand network stack separately, then the correct starting order is:

- Start the InfiniBand stack.
- Start QLogic VNIC service.

For example, if you use InfiniPath, the correct order of starting is:

```
# /etc/init.d/openibd start
# /etc/init.d/qlgc_vnic start
```

Correct stopping order is:

- Stop QLogic VNIC service.
- Stop the InfiniBand stack.

For example, if you use InfiniPath, the correct stopping order is:

```
# /etc/init.d/qlgc_vnic stop
# /etc/init.d/openibd stop
```

Draft

If you try to stop the InfiniBand stack when the QLogic VNIC service is running, an error message displays, indicating that some of the modules of the InfiniBand stack are in use by the QLogic VNIC service. Also, any QLogic VNIC interfaces that you created are removed (because stopping the InfiniBand network stack unloads the HCA driver, which is required for the VNIC interfaces to be present).

In this case, do the following:

- Stop the QLogic VNIC service with **/etc/init.d/**qlgc_vnic stop.
- Stop the InfiniBand stack again.

If you want to restart the QLogic VNIC interfaces, run the following command:

```
# /etc/init.d/qlgc_vnic restart
```

You can get information about the QLogic VNIC interfaces by using the following script (as root):

```
# ib_qlgc_vnic_info
```

This information is collected from the /sys/class/infiniband_qlgc_vnic/interfaces/ directory, under which there is a separate directory corresponding to each VNIC interface.

VNIC interfaces can be deleted by writing the name of the interface to the /sys/class/infiniband_qlgc_vnic/interfaces/delete_vnic file. For example, to delete interface veth0, run the following command (as root):

```
# echo -n veth0 >
/sys/class/infiniband_qlgc_vnic/interfaces/delete_vnic
```

More information for configuration, starting and stopping the interface, and basic troubleshooting is found in the QLogic *OFED+ User Guide*.

Other Configuration: Changing the MTU Size

The Maximum Transfer Unit MTU is set to 4K and enabled in the driver by default.

To see the current MTU size, and the maximum supported by the adapter, use the command:

```
$ ibv_devinfo
```

To change the driver default back to 2K MTU, add this line (as root) into /etc/modprobe.conf (or /etc/modprobe.conf.local):

```
options ib_ipath mtu4096=0
```

Restart the driver as described in “Managing the InfiniPath Driver” on page 4-18.

Draft

NOTE:

To use 4K MTU, set the switch to have the same 4K default. If you are using QLogic switches the following will apply:

For the Externally Managed 9024, use 4.2.2.0.3 firmware (9024DDR4KMTU_firmware.emfw) for the 9024 EM. This has the 4K MTU default, for use on fabrics where 4K MTU is required. If 4K MTU support is not required, then the 4.2.2.0.2 DDR *.emfw file should be used for DDR externally-managed switches. Use FastFabric to load the firmware on all the 9024s on the fabric.

For the 9000 Chassis, use the most recent 9000 code 4.2.4.0.1. The 4K MTU support is in 9000 Chassis version 4.2.1.0.2 and later. For the 9000 chassis, when the FastFabric 4.3 (or later) chassis setup tool is used, the user is asked what MTU they want. FastFabric (FF) can then set that MTU in all the 9000 Internally managed switches. The change will take effect on next reboot. Alternatively, for the Internally Managed 9000s, the `ismChassisSetMtu` command-line interface (CLI) command can be used. This should be executed on every switch and both hemispheres of the 9240s.

For reference, see the FastFabric Users Guide Version 4.3 and the SIlverStorm 9000 CLI Reference Guide Version 4.2. Both are available from the QLogic web site.

For other switches, see the vendors' documentation.

Managing the InfiniPath Driver

The startup script for `ib_ipath` is installed automatically as part of the software installation, and normally does not need to be changed. It runs as a system service.

The primary configuration file for the InfiniPath driver `ib_ipath` and other modules and associated daemons, is:

```
/etc/infiniband/openib.conf
```

Normally, this configuration file is set up correctly at installation and the drivers are loaded automatically during system boot once the RPMs have been installed. However, the `ib_ipath` driver has several configuration variables that set reserved buffers for the software, define events to create trace records, and set the debug level.

If you are upgrading, your existing configuration files will not be overwritten.

Draft

The device files are:

```
/dev/ipath
/dev/ipath0, /dev/ipath1, ...
```

The numbered device files allow access to a specific InfiniPath unit.

See the `ib_ipath` man page for more details.

Configure InfiniPath Driver State

Use the following commands to check or configure the state. These methods will not reboot the system.

To check the configuration state, use this command. You do not need to be root:

```
$ chkconfig --list openibd
```

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

```
# chkconfig openibd on 2345
```

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

```
# chkconfig openibd off
```

NOTE:

This command does not stop and unload the driver if the driver is already loaded.

Start, Stop or Restart InfiniPath

Restart the software if you install a new InfiniPath release, change driver options, or do manual testing.

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

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

This method will not reboot the system. The following set of commands shows how to use this script. Note the following:

- If OpenSM is configured and running, it must be stopped before the `openibd stop` command, and must be started after the `openibd start` command. Omit the commands to start/stop `opensmd` if you are not running it on that node.

Draft

The sequence of commands to restart the driver are as follows.

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

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

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

```
# /etc/init.d/opensmd stop
# /etc/init.d/openibd restart
# /etc/init.d/opensmd start
```

NOTE:

Stopping or restarting openibd terminates any QLogic MPI processes, as well as any OpenFabrics processes that are running at the time.

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

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

When you need to determine which InfiniPath and OpenFabrics modules are running, use the following command. You do not need to be root:

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

Unloading the Driver/Modules Manually

You can also unload the driver/modules manually without using /etc/init.d/openibd. Use the following series of commands (as root):

```
# umount /ipathfs
# fuser -k /dev/ipath* /dev/infiniband/*
# lsmod | egrep '^ib_|^rdma_|^iw_' | xargs modprobe -r
```

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). The filesystem is unmounted when the InfiniPath script is invoked with the stop option (e.g. at system shutdown).

Draft

Here is a sample layout of a system with two cards:

```
/ipathfs
/ipathfs/00
/ipathfs/00/flash
/ipathfs/00/atomic_counters
/ipathfs/01
/ipathfs/01/flash
/ipathfs/01/atomic_counters
/ipathfs/atomic_stats
```

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

- `atomic_counters`
- `flash`

The `atomic_counters` file contains counters for the device, for example, interrupts received, bytes and packets in and out, etc. The `flash` file is an interface for internal diagnostic commands.

More Information on Configuring and Loading Drivers

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

Performance Settings and Management Tips

The following sections provide suggestions for improving performance and simplifying cluster management. Many of these settings will be done by the system administrator. User level runtime performance settings are shown in “Performance Tuning” on page 5-21.

Draft

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 can also cause problems. Old and new libraries must 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:

- `ident` (see "`ident`" on page F-6)
- `ipathbug-helper` (see "`ipathbug-helper`" on page F-6)
- `ipath_checkout` (see "`ipath_checkout`" on page F-7)
- `ipath_control` (see "`ipath_control`" on page F-8)
- `mpirun` (see "`mpirun`" on page F-12)
- `rpm` (see "`rpm`" on page F-13)
- `strings` (see "`strings`" on page F-13)

NOTE:

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

Adapter and Other Settings

The following adapter and other settings can be adjusted for better performance.

- **Use `taskset` to tune CPU affinity on Opteron systems with the QLE7240, QLE7280, and QLE7140.** Latency will be slightly lower for the Opteron socket that is closest to the PCI Express bridge. On some chipsets, bandwidth may be higher on this socket. See "Performance Tuning" on page 5-21 for more information on `taskset`. Also see the `taskset(1)` man page.
- **Use an IB MTU of 4096 bytes instead of 2048 bytes, if available, with the QLE7240, QLE7280, and QLE7140.** 4K MTU is enabled in the InfiniPath driver by default. A switch that supports and is configured for 4KB MTU is required to use this setting. To change this setting, see "Other Configuration: Changing the MTU Size" on page 4-17.
- **Use a PCIe Max Read Request size of at least 512 bytes with the QLE7240 and QLE7280.** QLE7240 and QLE7280 adapters can support sizes from 128 bytes to 4096 byte in powers of two. This value is typically set by the BIOS.
- **Use PCIe Max Payload size of 256, where available, with the QLE7240 and QLE7280.** The QLE7240 and QLE7280 adapters can support 128, 256, or 512 bytes. This value is typically set by the BIOS as the minimum value supported both by the PCIe card and the PCIe root complex.

Draft

- **Make sure that write combining is enabled.** The x86 Page Attribute Table (PAT) mechanism that allocates write-combining (WC) mappings for the PIO buffers has been added and is now the default. If PAT is unavailable or PAT initialization fails for some reason, the code will generate a message in the log and fall back to the MTRR mechanism. See E Write Combining for more information.
- **Check the PCIe bus width.** If slots have a smaller electrical width than mechanical width, lower than expected performance may occur. Use this command to check PCIe Bus width:

```
$ ipath_control -iv
```

This will also show link speed.

Remove Unneeded Services

The cluster administrator can enhance application performance by minimizing 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 Network File System (NFS) or yellow pages (yp) passwords:

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

Draft

To watch for disk problems:

- `smartd`
- `readahead`

The service comprising the InfiniPath driver and SMA:

- `openibd`

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

If your system is running the daemon `irqbalance`, QLogic recommends that you turn it off. Disabling `irqbalance` will enable more consistent performance with programs that use interrupts. Use this command:

```
# /sbin/chkconfig irqbalance off
```

See "Erratic Performance" on page D-11 for more information.

Disable Powersaving Features

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

For RHEL4, RHEL5, and RHEL6, run this command as root:

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

For SLES 10, run this command as root:

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

After running either of these commands, reboot the system for the changes to take effect.

Hyper-Threading

If you are using Intel® Netburst Processors that support Hyper-Threading, QLogic recommends turning off Hyper-Threading in the BIOS, which will provide more consistent performance. You can check and adjust this setting using the BIOS Setup utility. For specific instructions, follow the hardware documentation that came with your system.

Host Environment Setup for MPI

After the InfiniPath software and the GNU (GCC) compilers have been installed on all the nodes, the host environment can be set up for running MPI programs.

Draft

Configuring for ssh

Running MPI programs with the command `mpirun` on an InfiniPath cluster depends, by default, on secure shell `ssh` to launch node programs on the nodes. In QLogic MPI, `mpirun` uses the secure shell command `ssh` to start instances of the given MPI program on the remote compute nodes without the need for interactive password entry on every node.

To use `ssh`, you must have generated Rivest, Shamir, Adleman (RSA) or Digital Signal Algorithm (DSA) keys, public and private. The public keys must be distributed and stored on all the compute nodes so that connections to the remote machines can be established without supplying a password.

You or your administrator need to properly set up the `ssh` keys and associated files on the cluster. There are two methods for setting up ssh on your cluster. The first method, the `shosts.equiv` mechanism, is typically set up by the cluster administrator. The second method, using `ssh-agent`, is more easily accomplished by an individual user.

NOTE:

- `rsh` can be used instead of `ssh`. To use `rsh`, set the environment variable `MPI_SHELL=rsh`. See "Environment Variables" on page 5-18 for information on setting environment variables. Also see "Shell Options" on page A-5 for information on setting shell options in `mpirun`.
- `rsh` has a limit on the number of concurrent connections it can have, typically 255, which may limit its use on larger clusters.

Configuring `ssh` and `sshd` Using `shosts.equiv`

This section describes how the cluster administrator can set up `ssh` and `sshd` through the `shosts.equiv` mechanism. This method is recommended, provided that your cluster is behind a firewall and accessible only to trusted users.

"Configuring for `ssh` Using `ssh-agent`" on page 4-27 shows how an individual user can accomplish the same thing using `ssh-agent`.

The example in this section 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 end users have accounts with the same account name on the front end and on each node, by using Network Information Service (NIS) or another means of distributing the password file.
- The front end used here is called `ip-fe`.

Draft

- Root or superuser access is required on `ip-fe` and on each node 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.

How to use `shosts.equiv` to configure `ssg` and `sshd`:

1. On the system `ip-fe` (the front end node), change the `/etc/ssh/ssh_config` file to allow host-based authentication. Specifically, this file must contain the following four lines, all set to `yes`. If the lines are already there 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. Add the line:

   ```
   ip-fe
   ```

   Change the file to mode 600 when you are finished editing.

3. On each of the InfiniPath node systems, create or edit the file `/etc/ssh/ssh_known_hosts`. You will 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+1nwo4LeTVW
   pCD1QsvrYRmpsfwpzYLXiSJdZSA8hfePWmMfrkvAAk4ueN8L3ZT4QfCTwqvHV
   vSctpibf8n
   aUmzloovBndOX9TIHyP/Ljfzzep4wL17+5hr1AHXldzrmgeEKp6ect1wxAAAA
   FQDR56dAKFA4WgAiRmUJailtLFp8swAAAIBB1yrhF5P0jO+vpSnZrvrHa0Ok+
   Y9apeJp3sessee30NlqKbJqWj5DOoRejr2VfTxZROf8LKuOY8tD6I59I0vlcQ
   812E5iw1GCZfNefBmWbegWVKFwGlNbqBnZK7kDRLSOKQtuhYbGPcrVlSjuVps
   fWEju64FTqKEetA8l8QEgAAAIBNtPDDwdmXRvDyc0gvAm6lPOIsRLmgmdgKXT
   GOZUZ0zwxSL7GP1nEyFk9wAxCrXv3xPKxQaezQKs+KL95FouJvJ4qrSxxHdd1
   NYNR0DavEBVQgCaspgWvWQ8cL
   0aUQmTbggLrtD9zETVU5PCgRlQL6I3Y5sCCHuO7/UvTH9nneCg==
   ```

   Change the file to mode 600 when you are finished editing.

Draft

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

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

5. After creating or editing the 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 restart only on idle nodes. Type the following (as root) to notify `sshd` to use the new configuration files:

```
# killall -HUP sshd
```

NOTE:

This command terminates 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 end user should be able to login to the `ip-fe` front end system and use `ssh` to login to any InfiniPath node without being prompted for a password or pass phrase.

Configuring for `ssh` Using `ssh-agent`

The `ssh-agent`, a daemon that caches decrypted private keys, can be used to store the keys. Use `ssh-add` to add your private keys to `ssh-agent`'s cache. When `ssh` establishes a new connection, it communicates with `ssh-agent` to acquire these keys, rather than prompting you for a passphrase.

The process is described 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. The following command tells `ssh` that your key pair should let you in:

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

Draft

Edit the ~/.ssh/config file so that it reads like this:

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

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

4. Follow your administrator's cluster policy for setting up ssh-agent on the machine where you will be running ssh commands. Alternatively, you can start the ssh-agent by adding the following line to your ~/.bash_profile (or equivalent in another shell):

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

   Use back quotes rather than single quotes. Programs started in your login shell can then locate the 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).

Process Limitation with ssh

Process limitation with ssh is primarily an issue when using the mpirun option -distributed=off. The default setting is now -distributed=on; therefore, in most cases, ssh process limitations will not be encountered. This limitation for the -distributed=off case is described in the following paragraph. See "Process Limitation with ssh" on page D-20 for an example of an error message associated with this limitation.

MPI jobs that use more than 10 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 must increase the value of MaxStartups in your /etc/ssh/sshd_config file.

Draft

Checking Cluster and Software Status

ipath_control

InfiniBand status, IB link speed, and PCIe bus width can be checked by running the program `ipath_control`. Sample usage and output are as follows:

```
$ ipath_control -iv
$Id: QLogic OFED Release 1.4 $ $Date: 2009-03-10-10:15 $
 0: Version: ChipABI 2.0, InfiniPath_QLE7280, InfiniPath1 5.2,
    PCI 2, SW Compat 2
 0: Status: 0xe1 Initted Present IB_link_up IB_configured
 0: LID=0x1f MLID=0xc042 GUID=00:11:75:00:00:ff:89:a6 Serial:
    AIB0810A30297
 0: HRTBT:Auto RX_polarity_invert:Auto RX_lane_reversal: Auto
 0: LinkWidth:4X of 1X|4X Speed:DDR of SDR|DDR
 0: LocalBus: PCIe,2500MHz,x16
```

ibstatus

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

```
$ ibstatus
    Infiniband device 'ipath0' port 1 status:
    default gid:     fe80:0000:0000:0000:0011:7500:00ff:89a6
    base lid:        0x1f
    sm lid:          0x1
    state:           4: ACTIVE
    phys state:      5: LinkUp
    rate:            20 Gb/sec (4X DDR)
```

Draft

ibv_devinfo

ibv_devinfo queries RDMA devices. Use the -v option to see more information. Sample usage:

```
$ ibv_devinfo
hca_id: ipath0
        fw_ver:                         0.0.0
        node_guid:                      0011:7500:00ff:89a6
        sys_image_guid:                 0011:7500:00ff:89a6
        vendor_id:                      0x1175
        vendor_part_id:                 29216
        hw_ver:                         0x2
        board_id:                       InfiniPath_QLE7280
        phys_port_cnt:                  1
                port:   1
                        state:                  PORT_ACTIVE (4)
                        max_mtu:                4096 (5)
                        active_mtu:             4096 (5)
                        sm_lid:                 1
                        port_lid:               31
                        port_lmc:               0x00
```

ipath_checkout

ipath_checkout is a bash script that verifies that the installation is correct and that all the nodes of the network are functioning and mutually connected by the InfiniPath fabric. It must be run on a front end node, and requires specification of a nodefile. For example:

```
$ ipath_checkout [options] nodefile
```

The nodefile lists the hostnames of the nodes of the cluster, one hostname per line. The format of nodefile is as follows:

```
hostname1
hostname2
...
```

For more information on these programs, see “ipath_control” on page F-8, “ibstatus” on page F-4, and “ipath_checkout” on page F-7.

Draft

The Intel Cluster Checker

The Intel Cluster Checker checks and verifies Intel cluster ready certified clusters. To run the Intel Cluster Checker, create or edit (as root) the `/etc/security/limits.conf` file. The file will contain a single line:

```
* soft memlock 131072
```

The asterisk (`*`) must be in the first column of the line.

This is the same value that is set by the `ulimit -l 131072` command in `/etc/initscript`, which is created or modified when the `infinipath` RPM is installed.

For more information about the Intel Cluster Checker, see:

http://softwarecommunity.intel.com/articles/eng/1316.htm

Draft

Notes

Draft

5 Using QLogic MPI

This section provides information on using the QLogic Message-Passing Interface (MPI). Examples are provided for setting up the user environment, and for compiling and running MPI programs.

Introduction

The Message Passing Interface (MPI) standard 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.

QLogic MPI

QLogic's implementation of the MPI standard is derived from the MPICH reference implementation version 1.2.7. The QLogic MPI (InfiniPath) libraries have been highly tuned for the QLogic Interconnect, and will not run over other interconnects.

QLogic 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, QLogic MPI includes only the MPI-IO features implemented in ROMIO version 126 and the generalized MPI_All to allow communication exchange.

The QLogic MPI implementation in this release supports hybrid MPI/OpenMP and other multi-threaded programs, as long as only one thread uses MPI. For more information, see "QLogic MPI and Hybrid MPI/OpenMP Applications" on page 5-24.

PSM

The PSM InfiniPath Messaging API, or PSM API, is QLogic's low-level user-level communications interface for the InfiniPath family of products. Other than using some environment variables with the PSM prefix, MPI users typically need not interact directly with PSM. The PSM environment variables apply to other MPI implementations as long as the environment with the PSM variables is correctly forwarded. See "Environment Variables" on page 5-18 for a summary of the commonly used environment variables.

Draft

For more information on PSM, email QLogic at support@qlogic.com.

Other MPIs

Other high-performance MPIs, such as HP-MPI version 2.3, Open MPI (release 1.2.8), Ohio State University MVAPICH (1.1), and Scali 5.6.4 (Platform) MPI, have been ported to the PSM interface.

Open MPI, MVAPICH, HP-MPI, and Scali also run over IB Verbs (the Open Fabrics Alliance API which provides support for user level Upper Layer Protocols like MPI). Intel MPI, although not ported to the PSM interface, is supported over uDAPL, which uses IB Verbs. For more information, see "Using Other MPIs" on page 6-1.

Linux File I/O in MPI Programs

MPI node programs are Linux programs, which can execute file I/O operations to local or remote files in the usual ways, through APIs of the language in use. Remote files are accessed via a 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 can 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 ways 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. Alternately, the shared data files can be accessed directly by each node program. In this case, the shared files are available through some network file support, such as NFS. Also, in this case, the application programmer is responsible for ensuring file consistency, either through proper use of file locking mechanisms offered by the operating system and the programming language, such as `fcntl` in C, or by using MPI synchronization operations.

MPI-IO with ROMIO

MPI-IO is the part of the MPI-2 standard, supporting collective and parallel file I/O operations. One advantage of using MPI-IO is that it can take care of managing file locks when file data is shared among nodes.

QLogic MPI includes ROMIO version 126, 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, QLogic MPI includes only the MPI-IO features implemented in ROMIO version 126 and the generalized MPI_All to allow communication exchange. See the ROMIO documentation at http://www.mcs.anl.gov/romio for details.

Draft

NFS, PanFS, and local (UFS) support is enabled.

Getting Started with MPI

This section shows how to compile and run some simple example programs that are included in the InfiniPath software product. Compiling and running these examples enables you to verify that QLogic MPI and its components have been properly installed on the cluster. See "QLogic MPI Troubleshooting" on page D-12 if you have problems compiling or running these examples.

These examples assume that your cluster's policy allows you to use the `mpirun` script directly, without having to submit the job to a batch queuing system.

Copy Examples

Start by copying the examples to your working directory:

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

Create the `mpihosts` File

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
...
```

More details on the `mpihosts` file can be found in "`mpihosts` File Details" on page 5-13.

Compile and Run An Example C Program

In this step you will compile and run your MPI program.

QLogic 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 the file.

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

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

By default, `mpicc` runs the GNU `gcc` compiler, and is used for both compiling and linking, the same function as the `gcc` command.

Draft

NOTE:

For information on using other compilers, see "To Use Another Compiler" on page 5-8.

Then, run the program 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
```

In this example, `./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. Here is an example with four processes, using the same two hosts in the `mpihosts` file:

```
$ 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
```

Generally, `mpirun` tries to distribute the specified number of processes evenly among the nodes listed in the `mpihosts` file. However, 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.

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

Details on alternate means of specifying the `mpihosts` file are provided in "`mpihosts` File Details" on page 5-13.

More information on the `mpirun` options are in "Using `mpirun`" on page 5-15, and "`mpirun` Options Summary" on page A-1. "Process Allocation" on page 5-10 explains how processes are allocated by using hardware and software contexts.

Draft

The Examples Using Other Programming Languages

This section gives similar examples for computing pi for Fortran77 and Fortran90. Fortran95 usage is similar to Fortran90. The C++ example uses the traditional "Hello, World" program. All programs are located in the same directory.

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

```
$ mpif77 -o fpi fpi.f
$ mpirun -np 2 -m mpihosts ./fpi
```

`pi3f90.f90` is a Fortran90 program that does the same computation. Compile and link, and run it as follows:

```
$ mpif90 -o pi3f90 pi3f90.f90
$ 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 and run it as follows:

```
$ mpicxx -o hello hello++.cc
$ 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 GNU compiler for the respective language and the linker. See "To Use Another Compiler" on page 5-8 for an example of how to use other compilers. The use of `mpirun` is the same for programs in all languages.

QLogic MPI Details

The following sections provide more details on the use of QLogic MPI. These sections assume that you are familiar with standard MPI. For more information, see the references in "References for MPI" on page G-1. This implementation includes the man pages from the MPICH implementation for the numerous MPI functions.

Draft

Use Wrapper Scripts for Compiling and Linking

The following 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.

Table 5-1. QLogic MPI Wrapper Scripts

Wrapper Script Name	Language
`mpicc`	C
`mpicxx`	C++
`mpif77`	Fortran 77
`mpif90`	Fortran 90
`mpif95`	Fortran 95

On x86_64, these scripts (by default) call the GNU compiler and linker. To use other compilers, see "To Use Another Compiler" on page 5-8.

These scripts all provide the command line options listed in Table 5-2.

Table 5-2. Command Line Options for Scripts

Command	Meaning
`-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

In addition, each of these scripts allow a command line option for specifying a different compiler/linker as an alternative to the GNU Compiler Collection (GCC). For more information, see "To Use Another Compiler" on page 5-8.

Most other command line options are passed on to the invoked compiler and linker. The GNU compiler and alternative compilers all admit numerous command line options. See the GCC compiler documentation and the man pages for `gcc` and `gfortran` for complete information on available options. See the corresponding documentation for any other compiler/linker you may call for its options. Man pages for `mpif90(1)`, `mpif77(1)`, `mpicc(1)`, and `mpiCC(1)` are available.

Configuring MPI Programs for QLogic MPI

When configuring an MPI program (generating header files and/or Makefiles) for QLogic MPI, you usually need to specify `mpicc`, `mpicxx`, and so on as the compiler, rather than `gcc`, `g++`, etc.

Specifying the compiler is typically done with commands similar to the following, assuming that you are using `sh` or `bash` as the 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. If you use `csh`, use commands similar to the following:

```
$ setenv CC mpicc
```

You may need to pass arguments to `configure` directly, for example:

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

You may also 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. QLogic recommends that the linker be specified as `mpicc`, `mpif90`, etc. in these cases. This specification automatically includes 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 is good practice to use the provided scripts.

Draft

To Use Another Compiler

QLogic MPI supports a number of compilers, in addition to the default GNU Compiler Collection (GCC) versions 3.3.*x*, 3.4.*x*, 4.0, and 4.1 and gfortran. These include the PathScale Compiler Suite 3.0 and 3.1; PGI 5.2, 6.0, and 7.1; and Intel 9.*x* and 10.1.

NOTE:

The GNU 4.*x* environment is supported in the PathScale Compiler Suite 3.*x* release. However, the 2.*x* PathScale compilers are not currently supported on SLES 10 systems that use the GNU 4.*x* compilers and compiler environment (header files and libraries). QLogic recommends installing the PathScale 3.1 release.

These compilers can be invoked on the command line by passing options to the wrapper scripts. Command line options override environment variables, if set.

The following tables show the options for each of the compilers.

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 upon which it operates.

Table 5-3. PathScale Compiler Suite

Compiler	Command
C	**`mpicc -cc=pathcc .....`**
C++	`mpicc -CC=pathCC .....`
Fortran77	**`mpif77 -fc=pathf95 .....`**
Fortran90/95	**`mpif90 -f90=pathf95 .....`** **`mpif90 -f95=pathf95 .....`** Note: pathf95 invokes the Fortran77, Fortran90 and Fortran95 compilers.

Table 5-4. Portland Group (PGI)

Compiler	Command
C	**`mpicc -cc=pgcc .....`**
C++	`mpicc -CC=pgCC`
Fortran77	**`mpif77 -fc=pgf77 .....`**

Draft

Table 5-4. Portland Group (PGI) (Continued)

Compiler	Command
Fortran90/95	**mpif90 -f90=pgf90 .....** **mpif95 -f95=pgf95 .....**

Table 5-5. Intel

Compiler	Command
C	$ **mpicc -cc=icc .....**
C++	$ mpicc -CC=icpc
Fortran77	$ **mpif77 -fc=ifort .....**
Fortran90/95	$ **mpif90 -f90=ifort .....** $ **mpif95 -f95=ifort .....**

Also, use `mpif77, mpif90,` or `mpif95` for linking; otherwise, `.true.` may have the wrong value.

If you are not using the provided scripts for linking, link a sample program using the `-show` option as a test (without the actual build) to see what libraries to add to your link line. Some examples of the using the PGI compilers 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
```

Compiler and Linker Variables

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

Draft

When both the compiler and linker variables are set, and they do not match for the compiler you are using, the MPI program may fail to link; or, if it links, it may not execute correctly. For a sample error message, see “Compiler/Linker Mismatch” on page D-15.

Process Allocation

Normally MPI jobs are run with each node program (process) being associated with a dedicated QLogic HCA *hardware context,* which is mapped to a CPU.

If the number of node programs is greater than the available number of hardware contexts, *software context sharing* increases the number of node programs that can be run. Each adapter supports 4 software contexts per hardware context, so up to 4 node programs (from the same MPI job) can share that hardware context. There is a small additional overhead for each shared context.

Table 5-6 shows the maximum number of contexts available for each adapter.

Table 5-6. Available Hardware and Software Contexts

HCA	Available Hardware Contexts (same as number of supported CPUs)	Available Contexts When Software Context Sharing is Enabled
QLE7140	4	16
QHT7140	8	32
QLE7240	16	64
QLE7280	16	64

The default hardware context/CPU mappings can be changed on the QLE7240 and QLE7280. See “InfiniPath Hardware Contexts on the QLE7240 and QLE7280” on page 5-11 for more details.

Context sharing is enabled by default. Behavior of the system when context sharing is enabled or disabled is described in “Enabling and Disabling Software Context Sharing” on page 5-11.

When running a job in a batch system environment where multiple jobs may be running simultaneously, it is useful to restrict the number of InfiniPath contexts that are made available on each node of an MPI.See “Restricting InfiniPath Hardware Contexts in a Batch Environment” on page 5-12

Errors that may occur with context sharing are covered in “Context Sharing Error Messages” on page 5-13.

Draft

There are multiple ways of specifying how processes are allocated. You can use the `mpihosts` file, the `-np` and `-ppn` options with `mpirun`, and the `MPI_NPROCS` and `PSM_SHAREDCONTEXTS_MAX` environment variables. How these all are set are covered later in the document.

InfiniPath Hardware Contexts on the QLE7240 and QLE7280

On the QLE7240 and QLE7280 adapters, adapter receive resources are statically partitioned across the InfiniPath contexts according to the number of InfiniPath contexts enabled. The following defaults are automatically set according to the number of online CPUs in the node:

For four or less CPUs: 5 (4 + 1 for kernel)

For five to eight CPUs: 9 (8 + 1 for kernel)

For nine or more CPUs: 17 (16 + 1 for kernel)

Performance can be improved in some cases by disabling InfiniPath hardware contexts when they are not required so that the resources can be partitioned more effectively.

To disable this behavior, explicitly configure for the number you want to use with the `cfgports` module parameter in the file `/etc/modprobe.conf` (or `/etc/modprobe.conf.local` on SLES).

The maximum that can be set is 17 (16 + 1 for the kernel).

The driver must be restarted if this default is changed. See “Managing the InfiniPath Driver” on page 4-18.

NOTE:

In rare cases, setting contexts automatically on the QLE7240 and QLE7280 can lead to sub-optimal performance where one or more InfiniPath hardware contexts have been disabled and a job is run that requires software context sharing. Since the algorithm ensures that there is at least one InfiniPath context per online CPU, this case occurs only if the CPUs are over-subscribed with processes (which is not normally recommended). In this case, it is best to override the default to use as many InfiniPath contexts as are available, which minimizes the amount of software context sharing required.

Enabling and Disabling Software Context Sharing

By default, context sharing is enabled; it can also be specifically disabled.

Draft

Context Sharing Enabled: The MPI library provides PSM the local process layout so that InfiniPath contexts available on each node can be shared if necessary; for example, when running more node programs than contexts. All PSM jobs assume that they can make use of all available InfiniPath contexts to satisfy the job requirement and try to give a context to each process.

When context sharing is enabled on a system with multiple QLogic HCA (InfiniPath) boards (units) and the `IPATH_UNIT` environment variable is set, the number of InfiniPath contexts made available to MPI jobs is restricted to the number of contexts available on that unit. When multiple InfiniPath devices are present, it restricts the use to a specific InfiniPath unit. By default, all configured units are used in round robin order.

Context Sharing Disabled: Each node program tries to obtain exclusive access to an InfiniPath hardware context. If no hardware contexts are available, the job aborts.

To explicitly disable context sharing, set this environment variable in one of the two following ways:

```
PSM_SHAREDCONTEXTS=0
PSM_SHAREDCONTEXTS=NO
```

The default value of `PSM_SHAREDCONTEXTS` is `1` (enabled).

Restricting InfiniPath Hardware Contexts in a Batch Environment

If required for resource sharing between multiple jobs in batch systems, you can restrict the number of InfiniPath hardware contexts that are made available on each node of an MPI job by setting that number in the `PSM_SHAREDCONTEXTS_MAX` environment variable.

For example, if running two different jobs on nodes using the QLE7140, you could set `PSM_SHAREDCONTEXTS_MAX` to 2 instead of the default 4. Each job would then have at most two of the four available hardware contexts. Both of the jobs that wish to share a node would have to set `PSM_SHAREDCONTEXTS_MAX=2` on that node before sharing begins.

However, note that setting `PSM_SHAREDCONTEXTS_MAX=2` as a clusterwide default would unnecessarily penalize nodes that are dedicated to running single jobs. So a per-node setting, or some level of coordination with the job scheduler with setting the environment variable is recommended.

If some nodes have more cores than others, then the setting must be adjusted properly for the number of cores on each node.

Additionally, you can explicitly configure for the number of contexts you want to use with the `cfgports` module parameter. This will override the default settings (on the QLE7240 and QLE7280) based on the number of CPUs present on each node. See “InfiniPath Hardware Contexts on the QLE7240 and QLE7280” on page 5-11.

Draft

Context Sharing Error Messages

The error message when the context limit is exceeded is:

```
No free InfiniPath contexts available on /dev/ipath
```

This message appears when the application starts.

Error messages related to contexts may also be generated by `ipath_checkout` or `mpirun`.

```
PSM found 0 available contexts on InfiniPath device
```

The most likely cause is that the cluster has processes using all the available PSM contexts. Clean up these processes before restarting the job.

Running in Shared Memory Mode

QLogic MPI supports running exclusively in shared memory mode; no QLogic HCA is required for this mode of operation. This mode is used for running applications on a single node rather than on a cluster of nodes.

To enable shared memory mode, use either a single node in the `mpihosts` file or use these options with `mpirun`:

```
$ mpirun -np=<N> -ppn=<N>
```

`<N>` needs to be equal in both cases.

NOTE:

For this release, `<N>` must be ≤ 64.

When you are using a non-QLogic MPI that uses the InfiniPath PSM layer, ensure that the parallel job is contained on a single node and set the `PSM_DEVICES` environment variable:

```
PSM_DEVICES="shm,self"
```

If you are using QLogic MPI, you do not need to set this environment variable; it is set automatically if `np == ppn`.

When running on a single node with QLogic MPI, no HCA hardware is required if `-disable-dev-check` is passed to `mpirun`.

`mpihosts` File Details

As noted in "Create the `mpihosts` File" on page 5-3, an `mpihosts` file (also called a *machines file, nodefile*, or *hostsfile*) has been created in your current working directory. This file names the nodes on which the node programs may run.

Draft

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 (number of processes to spawn in the mpirun command) 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. When not specified, the default value is one. The value of process_count determines how many node programs will be started on that host before using the next entry in the mpihosts file. When the full mpihosts file is processed, and there are additional processes requested, processing starts again at the start of the file.

NOTE:

To create an mpihosts file you can use the ibhosts program. It will generate a list of available nodes that are already connected to the switch.

There are several alternative ways of specifying the mpihosts file.

- First, as noted in "Compile and Run An Example C Program" on page 5-3, 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.

- When 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, the MPI job fails.

- In the absence of both the -m option and the MPIHOSTS environment variable, mpirun uses the file ./mpihosts, if it exists.

- 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 a job submission script that generates an appropriate mpihosts file.

Draft

Using `mpirun`

The script `mpirun` is a front end program that starts a parallel MPI job on a set of nodes in an InfiniPath cluster. `mpirun` may be run on any i386 or x86_64 machine inside or outside the cluster, as long as it is on a supported Linux distribution, and has TCP connectivity to all InfiniPath cluster machines to be used in a job.

The script 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.

NOTE:

The `mpi-frontend-*` RPM needs to be installed on all nodes that will be using `mpirun`. Alternatively, you can use the `mpirun` option `-distributed=off`, which requires that only the `mpi-frontend` RPM be installed on the node where mpirun is invoked. Using `-distributed=off` can have a negative impact on `mpirun`'s performance when running large-scale jobs. More specifically, this option increases the memory usage on the host from which `mpirun` is started and will slow down the job startup, since it will spawn MPI processes serially.

The general syntax is:

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

`program-name` is usually the pathname to the executable MPI program. When the MPI program resides in the current directory and the current directory is not in your search path, then *`program-name`* must begin with '`./`', for example:

`./program-name`

Unless you want to run only one instance of the program, use the `-np` option, for example:

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

This option spawns *`n`* instances of *`program-name`*. These instances are called *node programs*.

Generally, `mpirun` tries to distribute the specified number of processes evenly among the nodes listed in the `mpihosts` file. However, 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.

Draft

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

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

This option overrides the `:process_count` specifications, if any, in the lines of the `mpihosts` file. As a general rule, `mpirun` distributes the *n* node programs among the nodes without exceeding, on any node, the maximum number of instances specified by the `:process_count` option. The value of the `:process_count` 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 `mpihosts` file times the `-ppn` value, `mpirun` cycles back through the hostsfile, assigning additional node programs per host.

Typically, the number of node programs should not be larger than the number of processor cores, at least not for compute-bound programs.

This option specifies the number of processes to spawn. If this option is not set, then environment variable `MPI_NPROCS` is checked. If `MPI_NPROCS` is not set, the default is to determine the number of processes based on the number of hosts in the machinefile `-M` or the list of hosts `-H`.

```
-ppn processes-per-node
```

This option creates up to the specified number of processes per node.

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

The `-distributed=on|off` option has been added to `mpirun`. This option reduces overhead by enabling `mpirun` to start processes in parallel on multiple nodes. Initially, `mpirun` spawns one `mpirun` child per node from the root node, each of which in turn spawns the number of local processes for that particular node. Control the use of distributed `mpirun` job spawning mechanism with this option:

```
-distributed [=on|off]
```

The default is *on*. To change the default, put this option in the global `mpirun.defaults` file or a user-local file. See “Environment for Node Programs” on page 5-17 and “Environment Variables” on page 5-18 for details.

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

Draft

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

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 from which it was produced.

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, QLogic MPI supports two mechanisms to redirect `stdin`:

- When `mpirun` is run from the same node as MPI rank 0, all input piped to the `mpirun` command is redirected to rank 0.
- When `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 redirect a file as standard input to all nodes (or to a particular node) as specified by the `-stdin-target` option.

Environment for Node Programs

InfiniPath-related environment variables are propagated to node programs. These include environment variables that begin with the prefix `IPATH_`, `PSM_`, `MPI` or `LD_`. Some other variables (such as `HOME`) are set or propagated by `ssh(1)`.

NOTE:

The environment variable `LD_BIND_NOW` is not supported for QLogic MPI programs. Not all symbols referenced in the shared libraries can be resolved on all installations. (They provide a variety of compatible behaviors for different compilers, etc.) Therefore, the libraries are built to run in *lazy binding mode*; the dynamic linker evaluates and binds to symbols only when needed by the application in a given runtime environment.

`mpirun` checks for these environment variables in the shell where it is invoked, and then propagates them correctly. The environment on each node is whatever it would be for the user's login via `ssh`, unless you are using a Multi-Purpose Daemon (MPD) (see "MPD" on page 5-23).

Environment variables are specified in descending order, as follows:

1. Set in the default shell environment on a remote node, e.g., `~/.bashrc` or equivalents.
2. Set in `-rcfile`.

Draft

3. Set the current shell environment for the `mpirun` command.
4. If nothing has been set (none of the previous sets have been performed), the default value of the environment variable is used.

As noted in the above list, using an `mpirunrc` file overrides any environment variables already set by the user. You can set environment variables for the node programs with the `-rcfile` option of mpirun with the following command:

```
$ mpirun -np n -m mpihosts -rcfile mpirunrc program_name
```

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 when the node program starts.

The `.mpirunrc` command line cannot contain any interactive commands. It can contain commands that output on `stdout` or `stderr`.

There is a global options file 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.

Environment Variables

Table 5-7 contains a summary of the environment variables that are used by InfiniPath and mpirun.

Table 5-7. Environment Variables

Name	Description
`MPICH_ROOT`	This variable is used by `mpirun` to find the `mpirun-ipath-ssh` executable, set up `LD_LIBRARY_PATH`, and set up a prefix for all InfiniPath pathnames. This variable is used by the `--prefix` argument (or is the same as `--prefix`), if installing InfiniPath RPMs in an alternate location. **Default:** Unset
`IPATH_UNIT`	This variable is for context sharing. When multiple InfiniPath devices are present, restricts the use to a specific InfiniPath unit. By default, all configured units are used in round robin order. **Default:** Unset

Draft

Table 5-7. Environment Variables (Continued)

Name	Description
LD_LIBRARY_PATH	This variable specifies the path to the run-time library. It is often set in the .mpirunrc file. **Default:** Unset
MPICH_CC	This variable selects which compiler for mpicc, and others, to use.
MPICH_CCC	This variable selects which compiler for mpicxx, and others, to use.
MPICH_F90	This variable selects which compiler for mpif90, and others, to use.
MPIHOSTS	This variable sets the name of the machines (mpihosts) file. **Default:** Unset
MPI_NPROCS	This variable specifies the number of MPI processes to spawn. **Default:** Unset
MPI_SHELL	Specifies the name of the program to log into remote hosts. **Default:** ssh unless MPI_SHELL is defined.
PSM_DEVICES	Non-QLogic MPI users can set this variable to enable running in shared memory mode on a single node. This variable is automatically set for QLogic MPI. **Default:** PSM_DEVICES="self,ipath"
PSC_MPIRUN_DEFAULTS_PATH	This variable sets the path to a user-local mpirun defaults file. **Default:** /opt/infinipath/etc/mpirun.defaults
PSM_SHAREDCONTEXTS	This variable overrides automatic context sharing behavior. YES is equivalent to 1, below. **Default:** PSM_SHAREDCONTEXTS=1

DRAFT

Table 5-7. Environment Variables (Continued)

Name	Description
PSM_SHAREDCONTEXTS_MAX	This variable restricts the number of InfiniPath contexts that are made available on each node of an MPI job. **Default:** PSM_SHAREDCONTEXTS_MAX=8 (QHT7140) PSM_SHAREDCONTEXTS_MAX=4 (QLE7140) Up to 16 on (QLE7240 and QLE7280; set automatically based on number of CPUs on node)

Running Multiple Versions of InfiniPath or MPI

The variable MPICH_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 also sets up the LD_LIBRARY_PATH for new programs. Consequently, multiple versions of the InfiniPath software releases can be installed on some or all nodes, and QLogic MPI and other versions of MPI can be 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 installed the software into an alternate location using the --prefix option with rpm, --prefix would have been set to $MPICH_ROOT.

If MPICH_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 to avoid name conflicts with other MPI implementations.

Job Blocking in Case of Temporary InfiniBand Link Failures

By default, as controlled by mpirun's quiescence parameter -q, an MPI job is killed for quiescence in the event of an IB link failure (or unplugged cable). This quiescence timeout occurs under one of the following conditions:

- A remote rank's process cannot reply to out-of-band process checks.
- MPI is inactive on the IB link for more than 15 minutes.

Draft

To keep remote process checks but disable triggering quiescence for temporary IB link failures, use the `-disable-mpi-progress-check` option with a nonzero `-q` option. To disable quiescence triggering altogether, use `-q 0`. No matter how these options are used, link failures (temporary or other) are always logged to `syslog`.

If the link is down when the job starts and you want the job to continue blocking until the link comes up, use the `-t -1` option.

Performance Tuning

These methods may be used at runtime. Performance settings that are typically set by the System Administrator are listed in "Performance Settings and Management Tips" on page 4-21.

Use `sysctl` to Configure Kernel Parameters

`sysctl` modifies kernel parameters at runtime. You can use the following parameters to get better TCP/IPoIB performance; QLogic recommends that you use all of them. Run `/sbin/sysctl` (as root):

```
# sysctl net.ipv4.tcp_low_latency=0
# sysctl net.ipv4.tcp_timestamps=0
# sysctl net.ipv4.tcp_sack=0
# sysctl net.ipv4.tcp_rmem='4096 87380 8475988'
# sysctl net.ipv4.tcp_wmem='4096 65536 8454144'
# sysctl net.ipv4.tcp_mem='786432 1048576 1572864'
# sysctl net.ipv4.tcp_max_syn_backlog=3000
# sysctl net.core.rmem_max=8475988
# sysctl net.core.wmem_max=8454144
# sysctl net.core.rmem_default=524287
# sysctl net.core.wmem_default=524287
# sysctl net.core.optmem_max=524287
# sysctl net.core.netdev_max_backlog=3000
```

These settings will persist until the next reboot. If you want the settings to take effect on every reboot, add them at the end of the `/etc/sysctl.conf` file.

NOTE:

The `sysctl` tuning is not necessary for kernel versions 2.6.18 or higher.

Draft

CPU Affinity

InfiniPath attempts 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.

Use the `taskset` utility with `mpirun` to specify the mapping of MPI processes to logical processors. This combination makes the best use of available memory bandwidth or cache locality when running on dual-core Symmetric MultiProcessing (SMP) cluster nodes.

The following example uses the NASA Advanced Supercomputing (NAS) Parallel Benchmark's Multi-Grid (MG) 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 command forces the programs to run on CPUs 1 and 3. See the `taskset` man page for more information on usage.

To turn off CPU affinity, set the environment variable `IPATH_NO_CPUAFFINITY`. This environment variable is propagated to node programs by mpirun.

`mpirun` Tunable Options

There are some `mpirun` options that can be adjusted to optimize communication. The most important one is:

```
-long-len, -L [default: 64000]
```

This option determines the length of the message above which the rendezvous protocol (instead of the eager protocol) must be used. The default value for `-L` was chosen for optimal unidirectional communication. Applications that have this kind of traffic pattern benefit from this higher default value. Other values for `-L` are appropriate for different communication patterns and data size. For example, applications that have bidirectional traffic patterns may benefit from using a lower value. Experimentation is recommended.

Two other options that are useful are:

```
-long-len-shmem, -s [default: 16000]
```

Length of the message above which the rendezvous protocol (instead of the eager protocol) must be used for intra-node communications. This option is for messages going through shared memory. The InfiniPath rendezvous messaging protocol uses two-way handshake (with MPI synchronous send semantics) and receive-side DMA.

```
-rndv-window-size, -W [default: 262144]
```

Draft

When sending a large message using the rendezvous protocol, QLogic MPI splits it into a number of fragments at the *source* and recombines them at the *destination*. Each fragment is sent as a single rendezvous stage. This option specifies the maximum length of each fragment. The default is 262144 bytes.

For more information on tunable options, type:

```
$ mpirun -h
```

The complete list of options is contained in Appendix A.

MPD

The Multi-Purpose Daemon (MPD) is an alternative to `mpirun` for launching MPI jobs. It is described briefly in the following sections.

MPD Description

MPD was developed by Argonne National Laboratory (ANL) as part of the MPICH-2 system. While the ANL MPD had some advantages over the use of their `mpirun` (faster launching, better cleanup after crashes, better tolerance of node failures), the QLogic `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 more complex to use than `mpirun` because it requires starting a ring of MPD daemons on the nodes. Therefore, QLogic recommends using the normal `mpirun` mechanism for starting jobs, as described in the previous chapter. However, if you want to use MPD, it is included in the InfiniPath software.

Using MPD

To start an MPD environment, use the `mpdboot` program. You must provide `mpdboot` with a file that lists 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 into a new shell.

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

To use `rsh` instead of `ssh` with `mpdboot`, set the environment variable `MPD_RSH` to the pathname of the desired remote shell. For example:

```
MPD_RSH=`which rsh` mpdboot -n 16 -f hosts
```

Draft

To run an MPI program from within the MPD environment, use the `mpirun` command. You do not need to provide an `mpihosts` file or a count of CPUs; by default, `mpirun` uses 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-*.rpm` and `python` (available with your distribution) must be installed on all nodes. See the *QLogic HCA and QLogic OFED Software Install Guide* for more details on software installation.

QLogic MPI and Hybrid MPI/OpenMP Applications

QLogic MPI supports hybrid MPI/OpenMP applications, provided that MPI routines are called only by the master OpenMP thread. This application 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 must be compiled with both OpenMP and MPI code enabled. To do this, use the `-mp` flag on the `mpicc` compile line.

As mentioned previously, MPI routines can be called only 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, you may want 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. (`OMP_NUM_THREADS` is used by other compilers' OpenMP products, but is not a QLogic MPI environment variable.) Use this variable 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 four CPUs, running one MPI process and four OpenMP threads. In this case, `OMP_NUM_THREADS` is set to four. `OMP_NUM_THREADS` is on a per-node basis.

See "Environment for Node Programs" on page 5-17 for information on setting environment variables.

At the time of publication, the `MPI_THREAD_SERIALIZED` and `MPI_THREAD_MULTIPLE` models are not supported.

Draft

NOTE:

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

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.

MPI Errors

Almost all MPI routines (except `MPI_Wtime` and `MPI_Wtick`) return an error code; either 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 the `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 G for details on the MPI error codes.

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 for `mpirun`. See the man pages to `pathdb`, `gdb`, and `strace` for details. When running under a debugger, you get an xterm window on the front end machine for each node process. Therefore, you can control the different node processes as desired.

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

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

Draft

The following features of QLogic MPI facilitate debugging:

- Stack backtraces are provided for programs that crash.
- The `-debug` and `-debug-no-pause` options are provided for `mpirun`. These options 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. Be sure to 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. This option 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 “MPI Stats” on page D-31 for more information and a sample output listing.

NOTE:

The TotalView debugger can be used with the Open MPI supplied in this release. Consult the TotalView documentation for more information.

QLogic MPI Limitations

The current version of QLogic MPI has the following limitations:

- 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 be between -2^{31} and $2^{31}-1$, and the argument to `MPI_File_read_at` must be between 0 and $2^{32}-1$.

Draft

6 Using Other MPIs

This section provides information on using other MPI implementations.

Introduction

Support for multiple high-performance MPI implementations has been added. Most run over both PSM and OpenFabrics Verbs. Table 6-1 lists the supported MPI implementations.

Table 6-1. Other Supported MPI Implementations

MPI Implementation	Runs Over	Compiled With	Comments
Open MPI 1.2.8	PSM Verbs	GCC, PGI, PathScale, Intel GCC	Provides some MPI-2 functionality (one-sided operations and dynamic processes). Available as part of the QLogic download. Can be managed by `mpi-selector`.
MVAPICH version 1.1	PSM Verbs	GCC, PGI, PathScale, Intel GCC	Provides MPI-1 functionality Available as part of the QLogic download. Can be managed by `mpi-selector`.
HP-MPI 2.3	PSM Verbs	GCC (Default)	Provides some MPI-2 functionality (one-sided operations) Available for purchase from HP.
Platform (Scali) 5.6.4	PSM Verbs	GCC (Default)	Provides MPI-1 functionality Available for purchase from Platform.

Table 6-1. Other Supported MPI Implementations (Continued)

MPI Implementation	Runs Over	Compiled With	Comments
Intel MPI version 3.1	uDAPL	GCC (Default)	Provides MPI-1 functionality Available for purchase from Intel.

Table Notes

MVAPICH and Open MPI have been have been compiled for PSM to support the following versions of the compilers

- (GNU) gcc 4.1.0
- (PathScale) pathcc 3.0
- (PGI) pgcc 7.2-5
- (Intel) icc 11.0 (Version 11.0, Build 20081105, Package ID: l_cproc_p_11.0.074)

These MPI implementations run on multiple interconnects, and have their own mechanisms for selecting the interconnect on which each runs. Basic information about using these MPIs will be offered here. However, for more detailed information, see the documentation provided with the version of MPI that you want to use.

Installed Layout

By default, the MVAPICH and Open MPI MPIs are installed in this directory tree:

```
/usr/mpi/<$compiler>/<$mpi>-<mpi_version>
```

The QLogic-supplied MPIs precompiled with the GCC, PathScale, PGI, and the Intel compilers will also have `-qlc` appended after `<mpi_version>`.

For example:

```
/usr/mpi/gcc/openmpi-1.2.8-qlc
```

If a prefixed installation location is used, `/usr` will be replaced by `$prefix`.

The following examples assume that the default path for each MPI implementation to `mpirun` is:

```
/usr/mpi/<$compiler>/<$mpi>/bin/mpirun
```

Again, `/usr` may be replaced by `$prefix`. This path is sometimes referred to simply as `$mpi_home/bin/mpirun` in the following sections.

Consult the documentation for HP-MPI, Intel MPI, and Platform MPI for their default installation directories.

Draft

Open MPI

Open MPI is an open source MPI-2 implementation from the Open MPI Project. Pre-compiled versions of Open MPI Version 1.2.8 that run over PSM and are built with the GCC, PGI, PathScale, and Intel compilers are available with the QLogic download.

Open MPI that runs over Verbs and is precompiled with the GNU compiler is also available.

Open MPI can be managed with the `mpi-selector` utility, as described below in "Managing Open MPI, MVAPICH and QLogic MPI with the `mpi-selector` Utility" on page 6-6.

Installation

Follow the instructions in the *QLogic HCA and QLogic OFED Software Install Guide* for installing Open MPI.

Newer versions than the one supplied with this release can be installed after QLogic OFED 1.4 has already been installed; these may be downloaded from the Open MPI web site. Note that versions that are released after the QLogic OFED 1.4 release will not be supported.

Setup

If you use the `mpi-selector` tool the necessary setup will be performed for you. If you do not use this tool, you can put your Open MPI installation directory in your `PATH`:

add `<$mpi_home>/bin` to `PATH`

The `<$mpi_home>` is the directory path where the desired MPI was installed.

Compiling Open MPI Applications

As with QLogic MPI, it is recommended that you use the included wrapper scripts that invoke the underlying compiler.

Table 6-2. Open MPI Wrapper Scripts

Wrapper Script Name	Language
`mpicc`	C
`mpiCC, mpicxx, or mpic++`	C++
`mpif77`	Fortran 77
`mpif90`	Fortran 90

To compile your program in C:

```
$ mpicc mpi_app_name.c -o mpi_app_name
```

Draft

Running Open MPI Applications

By default the Open MPI shipped with the InfiniPath software stack will default to running over PSM once it is installed.

Here is an example of a simple `mpirun` command running with four processes:

```
$ mpirun -np 4 -machinefile mpihosts mpi_app_name
```

If you want to specify the PSM transport explicitly, you can add `--mca mtl psm` to the above command line.

To run over IB Verbs instead, use this `mpirun` command line:

```
$ mpirun -np 4 -machinefile mpihosts --mca btl sm --mca btl
openib,self --mca mtl ^psm mpi_app_name
```

These do the following:

`--mca btl sm`

Enables shared memory

`--mca btl openib, self`

Enables openib transport and communication to self

`--mca mtl ^psm`

Disables PSM transport

`btl` stands for "byte transport layer" and `mtl` for "matching transport layer".

The PSM transport works in terms of MPI messages, whereas the OpenIB transport works in terms of byte streams.

Alternatively, you can use Open MPI with a sockets transport running over IPoIB like this:

```
$ mpirun -np 4 -machinefile mpihosts --mca btl sm --mca btl
tcp,self --mca btl_tcp_if_exclude eth0 --mca btl_tcp_if_include
ib0 --mca mtl ^psm mpi_app_name
```

Note that `eth0` and `psm` are excluded, while `ib0` is included. These instructions may need to be adjusted for your interface names.

Note that in Open MPI, `machinefile` is also known as the `hostfile`.

Further Information on Open MPI

For more information about Open MPI, please see:

http://www.open-mpi.org/

http://www.open-mpi.org/faq

Draft

MVAPICH

Pre-compiled versions of MVAPICH 1.1 built with the GNU, PGI, PathScale, and Intel compilers, and that run over PSM, are available with the QLogic download.

MVAPICH that runs over Verbs and is pre-compiled with the GNU compiler is also available.

MVAPICH can be managed with the `mpi-selector` utility, as described below in "Managing Open MPI, MVAPICH and QLogic MPI with the `mpi-selector` Utility" on page 6-6.

Installation

Follow the instructions in the Install Guide for installing MVAPICH.

Newer versions than the one supplied with this release can be installed after QLogic OFED 1.4 has already been installed; these may be downloaded from the MVAPICH web site. Note that versions that are released after the QLogic OFED 1.4 release will not be supported.

Setup

To launch MPI jobs, the MVAPICH installation directory needs to be included in `PATH` and `LD_LIBRARY_PATH`:

If using `sh` for launching MPI jobs:

```
$ source /usr/mpi/<$compiler>/<$mpi>/bin/mpivars.sh
```

If using `csh` for launching MPI jobs:

```
$ source /usr/mpi/<$compiler>/<$mpi>/bin/mpivars.csh
```

Compiling MVAPICH Applications

As with QLogic MPI, it is recommended that you use the included wrapper scripts that invoke the underlying compiler.

Table 6-3. MVAPICH Wrapper Scripts

Wrapper Script Name	Language
`mpicc`	C
`mpiCC, mpicxx`	C++
`mpif77`	Fortran 77
`mpif90`	Fortran 90

To compile your program in C:

```
$ mpicc mpi_app_name.c -o mpi_app_name
```

Draft

To check the default configuration for the installation, check:

```
/usr/mpi/<$compiler>/<$mpi>/etc/mvapich.conf
```

Running MVAPICH Applications

By default the MVAPICH shipped with the InfiniPath software stack will default to running over PSM once it is installed.

Here is an example of a simple `mpirun` command running with four processes:

```
$ mpirun -np 4 -hostfile mpihosts mpi_app_name
```

Password-less `ssh` will be used unless the `-rsh` option is added to the command line above.

Further Information on MVAPICH

For more information about MVAPICH, please see:

http://mvapich.cse.ohio-state.edu/

Managing Open MPI, MVAPICH and QLogic MPI with the `mpi-selector` Utility

If multiple MPI implementations have been installed on your cluster, the `mpi-selector` can be used to switch between them. The MPIs that can be managed with the `mpi-selector` are:

- Open MPI
- MVAPICH
- MVAPICH2
- QLogic MPI

The `mpi-selector` is an OFED utility that is installed as a part of QLogic OFED 1.4. Its basic functions include:

- List available MPI implementations
- Set a default MPI to use (per-user or site-wide)
- Unset a default MPI to use (per-user or site-wide)
- Query the current default MPI in use

Usage is similar to this for listing and selecting an MPI:

```
$ mpi-selector --list
  mpi-1.2.3
  mpi-3.4.5
$ mpi-selector --set mpi-4.5.6
```

The new default will take place in the next shell that is started. See the `mpi-selector` man page for more information.

Draft

For QLogic MPI inter-operation with the `mpi-selector` utility, you must install all QLogic MPI RPMs using a prefixed installation. Once the `$prefix` for QLogic MPI has been determined, install the `qlogic-mpi-register` with this same `$prefix`, which registers QLogic MPI with `mpi-selector` utility and shows QLogic MPI as an available MPI implementation with the four different compilers. See the *QLogic HCA and QLogic OFED Software Install Guide* for information on prefixed installations.

The example shell scripts `mpivars.sh` and `mpivars.csh`, for registering with `mpi-selector`, are provided as part of the `mpi-devel` RPM in `$prefix/share/mpich/mpi-selector-{intel,gnu,pathscale,pgi}` directories.

For all non-GNU compilers that are installed outside standard Linux search paths, you must set up the paths so that compiler binaries and runtime libraries can be resolved. For example, you will need to set `LD_LIBRARY_PATH`, both in your local environment and in an rcfile (such as `.mpirunrc`, `.bashrc`, or `.cshrc`), which will be invoked on remote nodes. See "Environment for Node Programs" on page 5-17 and "Compiler and Linker Variables" on page 5-9 for information on setting up the environment and "Specifying the Run-time Library Path" on page D-16 for information on setting the run-time library path. Also see "Run Time Errors with Different MPI Implementations" on page D-18 for information on run time errors that may occur if there are MPI version mismatches.

NOTE:

The Intel-compiled versions require that the Intel compiler be installed and that paths to the Intel compiler runtime libraries be resolvable from the user's environment. The version used is Intel 10.1.012.

HP-MPI

HP-MPI Version 2.3 is Hewlett-Packard's high-performance implementation of the Message-Passing Interface (MPI) Standard, with full MPI-2 functionality.

Installation

Follow the instructions for download and installation from the HP web site.

Setup

No special setup is needed.

Draft

Compiling HP-MPI Applications

As with QLogic MPI, it is recommended that you use the included wrapper scripts that invoke the underlying compiler.

Table 6-4. HP-MPI Wrapper Scripts

Wrapper Script Name	Language
mpicc	C
mpiCC	C
mpi77	Fortran 77
mpif90	Fortran 90

To compile your program in C using the default compiler:

```
$ mpicc mpi_app_name.c -o mpi_app_name
```

Running HP-MPI Applications

Here is an example of a simple `mpirun` command running with four processes, over PSM:

```
$ mpirun -np 4 -hostfile mpihosts -PSM mpi_app_name
```

To run over IB Verbs, use:

```
$ mpirun -np 4 -hostfile mpihosts -IBV mpi_app_name
```

To run over TCP (which could be IPoIB if the hostfile is setup for IPoIB interfaces):

```
$ mpirun -np 4 -hostfile mpihosts -TCP mpi_app_name
```

Further Information on HP-MPI

For further information on HP-MPI, please consult:

http://www.hp.com/

Platform (Scali) MPI

Platform MPI was formerly known as Scali MPI Connect. The version tested with this release is 5.6.4.

Installation

Follow the instructions for downloading and installing Platform MPI from the Platform (Scali) web site.

Setup

When run over PSM, no special setup is needed.

Draft

If running over IB Verbs, Platform MPI needs to be told which IB adapter to use. This is achieved by creating the file `/opt/scali/etc/iba_params.conf` using a line such as:

```
hcadevice=ipath0
```

For a second InfiniPath card, `ipath1` would be used, and so on.

Compiling Platform MPI Applications

As with QLogic MPI, it is recommended that you use the included wrapper scripts that invoke the underlying compiler. They default to using gcc/g++/g77.

Table 6-5. Platform MPI Wrapper Scripts

Wrapper Script Name	Language
`mpicc`	C
`mpic++`	C++
`mpif77`	Fortran 77
`mpif90`	Fortran 90

To compile your program in C using the default compiler:

```
$ mpicc mpi_app_name.c -o mpi_app_name
```

To invoke another compiler, in this case PathScale, use the `-cc1` option:

```
$ mpicc -cc1 pathcc mpi_app_name.c -o mpi_app_name
```

Running Platform MPI Applications

Here is an example of a simple `mpirun` command running with four processes, over PSM:

```
$ mpirun -np 4 -machinefile mpihosts mpi_app_name
```

By default Platform MPI, once installed, will use the PSM transport. If you want to specify PSM explicitly, add `-networks infinipath` to the above command.

To run Scali MPI over IB Verbs, use:

```
$ mpirun -np 4 -machinefile mpihosts -networks ib,smp mpi_app_name
```

This indicates that `ib` is to be used for inter-node, and `smp` for intra-node communications.

To run over TCP (or IPoIB):

```
$ mpirun -np 4 -machinefile mpihosts -networks tcp,smp
mpi_app_name
```

Draft

Further Information on Platform MPI

For more information on using Platform MPI, please consult:

http://www.scali.com/

Intel MPI

Intel MPI Version 3.1 is the version tested with this release.

Installation

Follow the instructions for download and installation of Intel MPI from the Intel web site.

Setup

Intel MPI can be run over uDAPL, which uses IB Verbs. uDAPL is the user mode version of the Direct Access Provider Library (DAPL), and is provided as a part of the OFED packages. You will also have to have IPoIB configured.

Setup for Intel MPI is described below.

1. Make sure that DAPL 1.2 (not version 2.0) is installed on every node. In this release they are called `compat-dapl`. (Both versions are supplied with the OpenFabrics RPMs.) They can be installed either with the Installer with the QLogicIB-Basic package or with `rpm` with the QLogic OFED 1.4 RPM set. They look something like this:

   ```
   $ rpm -qa | grep compat-dapl
   compat-dapl-1.2.12-1.x86_64.rpm
   compat-dapl-debuginfo-1.2.12-1.x86_64.rpm
   compat-dapl-devel-1.2.12-1.x86_64.rpm
   compat-dapl-devel-static-1.2.12-1.x86_64.rpm
   compat-dapl-utils-1.2.12-1.x86_64.rpm
   ```

2. Check that you have a `/etc/dat.conf` file. It should be installed by the `dapl-` RPM. The file `dat.conf` contains a list of interface adapters supported by uDAPL service providers. In particular, it must contain mapping entries for OpenIB-cma for `dapl 1.2.x`, in a form similar to this (all on one line):

   ```
   OpenIB-cma u1.2 nonthreadsafe default libdaplcma.so.1 dapl.1.2
   "ib0 0" ""
   ```

3. On every node, type the following command (as root):

   ```
   # modprobe rdma_ucm
   ```

   To ensure that the module is loaded whenever the driver is loaded, add `RDMA_UCM_LOAD=yes` to the `/etc/infiniband/openib.conf` file. (Note that `rdma_cm` is also used, but it is loaded automatically.)

Draft

4. Bring up an IPoIB interface on every node, eg. ib0. See the instructions for configuring IPoIB for more details.

Intel MPI has different bin directories for 32-bit (bin) and 64-bit (bin64). 64-bit is the most commonly used.

To launch MPI jobs, the Intel installation directory needs to be included in PATH and LD_LIBRARY_PATH.

If using sh for launching MPI jobs:

```
$ source <$prefix>/bin64/mpivars.sh
```

If using csh for launching MPI jobs:

```
$ source <$prefix>/bin64/mpivars.csh
```

Substitute bin if using 32-bit.

Compiling Intel MPI Applications

As with QLogic MPI, it is recommended that you use the included wrapper scripts that invoke the underlying compiler. The default underlying compiler is GCC, including gfortran. Note that there are more compiler drivers (wrapper scripts) with Intel MPI than are listed here; check the Intel documentation for more information.

Table 6-6. Intel MPI Wrapper Scripts

Wrapper Script Name	Language
mpicc	C
mpiCC	C++
mpif77	Fortran 77
mpif90	Fortran 90
mpiicc	C (Uses Intel C compiler)
mpiicpc	C++ (Uses Intel C++ compiler)
mpiifort	Fortran 77/90 (Uses Intel Fortran compiler)

To compile your program in C using the default compiler:

```
$ mpicc mpi_app_name.c -o mpi_app_name
```

To use the Intel compiler wrappers (mpiicc, mpiicpc, mpiifort) the Intel compilers will need to be installed and resolvable from the user's environment.

Running Intel MPI Applications

Here is an example of a simple mpirun command running with four processes:

```
$ mpirun -np 4 -f mpihosts mpi_app_name
```

For more information, follow the Intel MPI instructions for usage of `mpdboot` and `mpirun`. Remember to use `-r ssh` with `mpdboot` if you use `ssh`. You will need to pass the following option to `mpirun` to select uDAPL:

```
-genv I_MPI_DEVICE rdma:OpenIB-cma
```

To help with debugging, you can also add this option to the Intel `mpirun` command:

```
-genv I_MPI_DEBUG 2
```

Further Information on Intel MPI

For more information on using Intel MPI, consult:

http://www.intel.com/

Improving Performance of Other MPIs Over IB Verbs

Performance of MPI applications when using an MPI implementation over IB Verbs can be improved by tuning the IB MTU size.

NOTE:

No manual tuning is necessary for PSM-based MPIs, since the PSM layer determines the largest possible IB MTU for each source/destination path.

The maximum supported MTU size of InfiniPath adapter cards are:

- QHT7140 = 2K
- QLE7140, QLE7240, QLE7280 = 4K

Support for 4K IB MTU requires switch support for 4K MTU. The method to set the IB MTU size varies by MPI implementation:

1. Open MPI defaults to the lower of either the IB MTU size or switch MTU size.
2. MVAPICH defaults to an IB MTU size of 1024 bytes. This can be over-ridden by setting an environment variable:

   ```
   $ export VIADEV_DEFAULT_MTU=MTU4096
   ```

   where possible values are `MTU256`, `MTU512`, `MTU1024`, `MTU2048` and `MTU4096`. This environment variable needs to be set for all processes in the MPI job. This can be done with `~/.bashrc` or use of `/usr/bin/env`.
3. HP-MPI over IB Verbs automatically determines the IB MTU size.
4. Platform (Scali) MPI defaults to an IB MTU of 1KB. This can be changed by adding a line to `/opt/scali/etc/iba_params.conf`:

   ```
   mtu=2048
   ```

Draft

It turns out that a value of 4096 is not allowed by the Scali software (as of Scali Connect 5.6.0), and in that case it uses the default value of 1024 bytes. This problem has been reported to support at Platform Inc. The largest value that can currently be used is 2048 bytes.

5. Intel MPI over uDAPL (which uses IB Verbs) automatically determines the IB MTU size.

Draft

Notes

Draft

A mpirun Options Summary

This section contains information that is in the mpirun man page.

mpirun Options Summary

This section summarizes the most commonly used options to mpirun. See the mpirun (1) man page for a more complete listing.

-mpd

This option is used after running mpdboot to start a daemon, rather than using the default ssh protocol to start jobs. See the mpdboot(1) man page for more information. None of the other mpirun options (with the exception of -h) are valid when using this option.

-ssh

This option uses the ssh program to start jobs, either directly or via distributed startup. This is the default.

Essential Options

-H, -hosts *hostlist*

Take the list of possible hosts to run on from the specified hostlist, which has precedence over the -machinefile option. The hostlist can be comma-delimited or quoted as a space-delimited list. The hostlist specification allows compressed representation of the form:

host-[01-02,04,06-08] is equivalent to

host-01,host-02,host-04,host-06,host-07,host-08

If the -np count is left unspecified, it will be adjusted to the number of hosts in the hostlist. If the -ppn count is specified, each host will receive as many processes.

-machinefile *filename*, -m *filename*

Machines (mpihosts) file, which contains the list of hosts to be used for this job. The default is $MPIHOSTS, then ./mpihosts, then ~/.mpihosts.

-nonmpi

This option runs a non-MPI program. The option is required if the node program makes no MPI calls. This option allows non-QLogic MPI applications use mpirun's parallel spawning mechanism.

Draft

-np *np*

This option specifies the number of processes to spawn. If this option is not set, then environment variable MPI_NPROCS is checked. If MPI_NPROCS is not set, the default is to determine the number of processes based on the number of hosts in the machinefile -M or the list of hosts -H.

-ppn *processes-per-node*

This option creates up to the specified number of processes per node.

By default, a limit will be enforced that depends on how many InfiniPath contexts are supported by the node (depends on the hardware type and the number of InfiniPath cards present).

InfiniPath context (port) sharing is supported, beginning with the InfiniPath 2.0 release. This feature allows running up to four times as many processes per node as was previously possible, with a small additional overhead for each shared context.

Context sharing is enabled automatically if needed. Use of the full number of available contexts is assumed. If you need to restrict the number of contexts, then use the environment variable PSM_SHAREDCONTEXTS_MAX to divide the available number of contexts.

You can override automatic context sharing behavior by using the environment variable PSM_SHAREDCONTEXTS. Setting this variable to zero disables context sharing, and jobs that require more than the available number of contexts cannot be run.

Setting this variable it to 1 (the default) causes context sharing to be enabled if needed.

-rcfile *node-shell-script*

This is the startup script for setting environment on nodes. Before starting node programs, mpirun checks to see if a file called .mpirunrc exists in the user's home directory. If it exists, it is sourced into the running remote shell .mpirunrc should be used to set paths, and other environment variables such as LD_LIBRARY_PATH.

Default: $HOME/.mpirunrc

Spawn Options

-distributed [=on|off]

Control use of the distributed mpirun job spawning mechanism. The default is on. If you want to change the default, put this option in the global mpirun.defaults file or a user-local file (see the environment variable PSC_MPIRUN_DEFAULTS_PATH for details). If the option appears more than once on the command line, the last setting controls the behavior.

Default: on.

Draft

Quiescence Options

-disable-mpi-progress-check

This option disables the MPI communication progress check without disabling the ping reply check.

If quiescence or a lack of ping reply is detected, the job and all compute processes are terminated.

-i, -ping-interval,*seconds*

Seconds to wait between ping packets to mpirun (if -q > 0).

Default: 60

-q, -quiescence-timeout,*seconds*

This option specifies the wait time (in seconds) for quiescence (absence of MPI communication or lack of ping reply) on the nodes. It is useful for detecting deadlocks. A value of zero disables quiescence detection.

Default: 900

Verbosity Options

-job-info

Print brief job startup and shutdown timing information.

-no-syslog

Don't send critical errors through syslog. By default, critical errors are sent to the console and through syslog.

-V, -verbose

This option prints diagnostic messages from mpirun itself. The verbose option is useful in troubleshooting.

Verbosity will also list the IPATH_* and PSM_* environment variable settings that affect MPI operation.

Startup Options

-I, -open-timeout *seconds*

This option tries for the number of *seconds* to open the InfiniPath device. If *seconds* is -1 (negative one), the node program waits indefinitely. Use this option to avoid having all queued jobs in a batch queue fail when a node fails for some reason, or is taken down for administrative purposes. The -t option is also normally set to -1.

-k, -kill-timeout *seconds*

This option indicates the time to wait for other ranks after the first rank exits.

Default: 60

Draft

-listen-addr <*hostname*>|<*IPv4*>

This option specifies which hostname (or IPv4 address) to listen on for incoming socket connections. It is useful for an mpirun front-end multihomed host. By default, mpirun assumes that ranks can independently resolve the hostname obtained on the head node with gethostname(2). To change the default, put this option in the global mpirun.defaults file or a user-local file.

-runscript

This is the script with which to run the node program.

-t, -timeout *seconds*

This option waits for specified time (in seconds) for each node to establish connection back to mpirun. If seconds is -1 (negative one), mpirun will wait indefinitely.

Default: 60

Stats Options

-M [=*stats_types*], -print-stats [=*stats_types*]

Statistics include minimum, maximum, and median values for message transmission protocols as well as more detailed information for expected and unexpected message reception. If the option is provided without an argument, stats_types is assumed to be mpi.

The following stats_types can be specified:

- mpi. Shows an MPI-level summary (expected, unexpected message)
- ipath. Shows a summary of InfiniPath interconnect communication
- p2p. Shows detailed per-MPI rank communication information
- counters. Shows low-level InfiniPath device counters
- devstats. Shows InfiniPath driver statistics
- all. Shows statistics for all stats_types

One or more statistics types can be specified by separating them with a comma. For example, -print-stats=ipath,counters displays InfiniPath communication protocol as well as low-level device counter statistics. For details, see “MPI Stats” on page D-31.

-statsfile *file-prefix*

This option specifies an alternate file to receive the output from the -print-stats option.

Default: stderr

-statsmode absolute|diffs

When asked to print process statistics with the -print-stats option, this option specifies if the printed statistics have the absolute values of QLogic HCA chip counters and registers or if they are differences between those values at the start and end of the process.

Draft

Default mode: diffs

Tuning Options

-L, -long-len *length*

This option determines the length of the message above which the rendezvous protocol must be used. The InfiniPath rendezvous messaging protocol uses two-way handshake (with MPI synchronous send semantics) and receive-side DMA.

Default: 64000

-N, -num-send-bufs *buffer-count*

QLogic MPI uses the specified number as the number of packets that can be sent without having to wait from an acknowledgement from the receiver. Each packet contains approximately 2048 bytes of user data.

Default: 512

-s,-long-len-shmem *length*

Length of the message above which the rendezvous protocol must be used for intra-node communications. The InfiniPath rendezvous messaging protocol uses two-way handshake (with MPI synchronous send semantics) and receive-side DMA.

Default: 16000

-W, -rndv-window-size *length*

When sending a large message using the rendezvous protocol, QLogic MPI splits the message into a number of fragments at the *source* and recombines them at the *destination*. Each fragment is sent as a single rendezvous stage. This option specifies the maximum length of each fragment.

Default: 262144 bytes

Shell Options

-shell *shell-name*

This option specifies the name of the program to use to log into remote hosts.

Default: `ssh`, unless `$MPI_SHELL` is defined.

-shellx *shell-name*

This option specifies the name of program to use to log into remote hosts with X11 forwarding. This option is useful when running with `-debug` or in xterm.

Default: `ssh`, unless `$MPI_SHELL_X` is defined.

Draft

Debug Options

-debug

Start all the processes under debugger, and waits for the user to set breakpoints and run the program. The gdb option is used by default, but can be overridden using the -debugger argument. Other supported debuggers are strace and the QLogic debugger pathdb.

-debug-no-pause

This option is similar to -debug, except that it does not pause at the beginning. The gdb option is used by default.

-debugger *gdb|pathdb|strace*

This option uses the specified debugger instead of the default gdb.

-display *X-server*

This option uses the specified *X* server for invoking remote xterms. (-debug, -debug-no-pause, and -in-xterm options use this value.)

Default: whatever is set in $DISPLAY

-in-xterm

This option runs each process in an xterm window. This is implied when -debug or -debug-no-pause is used.

Default: write to stdout with no stdin

-psc-debug-level *mask*

This option controls the verbosity of messages printed by the MPI and InfiniPath protocol layer. The default is 1, which displays error messages. A value of 3 displays short messaging details such as source, destination, size, etc. A value of 0xFF prints detailed information in a messaging layer for each message. Use this option with care, since too much verbosity will negatively affect application performance.

Default: 1

-xterm *xterm*

This option specifies which xterm to use.

Default: xterm

Format Options

-l, -label-output

This option labels each line of output on stdout and stderr with the rank of the MPI process that produces the output.

Draft

-y, -labelstyle string
This option specifies the label that is prefixed to error messages and statistics. Process rank is the default prefix. The label that is prefixed to each message can be specified as one of the following:

- %n. Hostname on which the node process executes
- %r. Rank of the node process
- %p. Process ID of the node process
- %L. LID (Infinipath Local IDentifier (LID) adapter identifier) of the node
- %P. Infinipath port of the node process
- %l. Local rank of the node process within a node
- %%. Percent sign

Other Options

-h -help
This option prints a summary of mpirun options, then exits.

-stdin *filename*
This option specifies the filename that must be fed as stdin to the node program.

Default: /dev/null

-stdin-target *0..np-1* | *-1*
This option specifies the process rank that must receive the file specified with the -stdin option. Negative one (-1) means all ranks.

Default: -1

-v, -version
This option prints the mpirun version, then exits.

-wdir *path-to-working_dir*
This option sets the working directory for the node program.

Default: -wdir *current-working-dir*

Draft

Notes

Draft

B Benchmark Programs

Several MPI performance measurement programs are installed from the `mpi-benchmark` RPM. This appendix describes these 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://mvapich.cse.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 following examples 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.

Benchmark 1: Measuring MPI Latency Between Two Nodes

In the MPI community, *latency for a message of given size* is 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. The term *latency*, alone without a qualifying message size, indicates the latency for a message of size zero. This latency represents the minimum overhead for sending messages, due to both software overhead and 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.

Draft

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 zero process initiates a series of sends and the rank one process echoes them back, using the blocking MPI send and receive calls for all operations. Half the time interval observed by the rank zero process for each such exchange is a measure of the latency for messages of that size, as previously defined. The program uses a loop, executing many such exchanges for each message size, to get an average. The program defers the timing until the message has been sent and received a number of times, to be sure that all the caches in the pipeline have been filled.

This benchmark always involves 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 ensure 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. In this case, 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               1.06
1               1.06
2               1.06
4               1.05
8               1.05
16              1.30
32              1.33
64              1.30
128             1.36
256             1.51
512             1.84
1024            2.47
2048            3.79
4096            4.99
8192            7.28
16384           11.75
32768           20.57
65536           58.28
131072          98.59
262144          164.68
524288          299.08
```

Draft

```
1048576          567.60
2097152          1104.50
4194304          2178.66
```

The first column displays the message size in bytes. The second column displays the average (one-way) latency in microseconds. This example is provided 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.

Benchmark 2: Measuring MPI Bandwidth Between Two Nodes

The `osu_bw` benchmark measures the maximum rate at which you can pump data between two nodes. This benchmark 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 function, and then returns a zero-length acknowledgement when all of the set data 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                 3.549325
2                 7.110873
4                 14.253841
8                 28.537989
16                42.613030
32                81.144290
64                177.331433
128               348.122982
256               643.742171
512               1055.355552
1024              1566.702234
2048              1807.872057
4096              1865.128035
8192              1891.649180
16384             1898.205188
```

Draft

```
32768           1888.039542
65536           1931.339589
131072          1942.417733
262144          1950.374843
524288          1954.286981
1048576         1956.301287
2097152         1957.351171
4194304         1957.810999
```

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

Benchmark 3: Messaging Rate Microbenchmarks

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

- The messaging rate and the bandwidth are reported.
- N/2 is dynamically calculated at the end of the run.
- You can 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. Here is an example of the type of output you will see when you run `mpi_multibw`:

```
$ mpirun -np 16 -ppn 8 ./mpi_multibw
```

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

```
# PathScale Modified OSU MPI Bandwidth Test
(OSU Version 2.2, PathScale $Revision$)
# Running on 8 procs per node (uni-directional traffic for each
process pair)
# Size          Aggregate Bandwidth (MB/s)      Messages/s
1               26.890668                       26890667.530474
2               53.692685                       26846342.327320
4               107.662814                      26915703.518342
8               214.526573                      26815821.579971
16              88.356173                       5522260.840754
32              168.514373                      5266074.141949
64              503.086611                      7860728.303972
```

Draft

```
128             921.257051                  7197320.710406
256             1588.793989                 6206226.519112
512             1716.731626                 3352991.457783
1024            1872.073401                 1828196.680564
2048            1928.774223                 941784.288727
4096            1928.763048                 470889.416123
8192            1921.127830                 234512.674597
16384           1919.122008                 117133.911629
32768           1898.415975                 57935.057817
65536           1953.063214                 29801.379615
131072          1956.731895                 14928.679615
262144          1957.544289                 7467.438845
524288          1957.952782                 3734.498562
1048576         1958.235791                 1867.519179
2097152         1958.333161                 933.806019
4194304         1958.400649                 466.919100
Searching for N/2 bandwidth.  Maximum Bandwidth of 1958.400649
MB/s...
Found N/2 bandwidth of 992.943275 MB/s at size 153 bytes
```

This microbenchmark is available and can be downloaded from the QLogic web site: http://www.qlogic.com

Benchmark 4: Measuring MPI Latency in Host Rings

The program `mpi_latency` measures latency in a ring of hosts. Its syntax is 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. 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 output 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 receive replies back in the other direction.

Draft

Notes

Draft

C 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 task of the cluster administrator is to allow users to submit MPI jobs through these batch queuing systems. Two methods are described in this document:

- Use `mpiexec` within the Portable Batch System (PBS) environment.
- Invoke a script, similar to `mpirun`, within the SLURM context to submit MPI jobs. A sample is provided in "Using SLURM for Batch Queuing" on page C-2.

Using `mpiexec` with PBS

`mpiexec` can be used as a replacement for `mpirun` within a Portable Batch System (PBS) cluster environment. The PBS software performs job scheduling.

For PBS-based batch systems, QLogic MPI processes can be spawned using the `mpiexec` utility distributed and maintained by the Ohio Supercomputer Center (OSC).

Starting with `mpiexec` version 0.84, MPI applications compiled and linked with QLogic MPI can use `mpiexec` and PBS's Task Manager (TM) interface to spawn and correctly terminate parallel jobs.

To download the latest version of `mpiexec`, go to:

http://www.osc.edu/~pw/mpiexec/

To build `mpiexec` for QLogic MPI and install it in `/usr/local`, do the following:

```
$ tar zxvf mpiexec-0.84.tgz
$ cd mpiexec-0.84
$ ./configure --enable-default-comm=mpich-psm && gmake all install
```

Draft

NOTE:

This level of support is specific to QLogic MPI, and not to other MPIs that currently support InfiniPath.

For more usage information, see the OSC `mpiexec` documentation.

For more information on PBS, go to:

http://www.pbsgridworks.com/

Using SLURM for Batch Queuing

The following is an example of the some of the functions that a batch queuing script might perform. The example is 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. The following script is called `batch_mpirun`:

```
#! /bin/sh
# Very simple example batch script for QLogic 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, the setup and the various script functions are discussed in more detail.

Draft

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 "Environment for Node Programs" on page 5-17 for more information.) 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 executed. For example:

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

After parsing the command line arguments, the next step of `batch_mpirun` is to request an allocation of *`n`* processors from the batch system. In SLURM, this uses 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` 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 number of processors to the requestor.

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, the script must determine which nodes the batch system has allocated, and how many processes can be started 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 for each process slot that SLURM has allocated. If SLURM has allocated two slots on one node, `hostname -s` is output twice for that node.

Draft

The `sort | uniq -c` component determines 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 can use. See "Console I/O in MPI Programs" on page 5-17 for more information.

Simple Process Management

At this point, your script has enough information to be able to run an MPI program. The next step is to start the program when the batch system is ready, 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
```

Clean Termination of MPI Processes

The InfiniPath software normally ensures clean termination of all MPI programs when a job ends, but in some rare circumstances an MPI process may remain alive, and potentially interfere with future MPI jobs. To avoid this problem, run a script before and after each batch job that 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 QLogic interconnect. The easiest way to do this is with the `fuser` command, which is normally installed in `/sbin`.

Run these commands as root to ensure that all processes are reported.

```
# /sbin/fuser -v /dev/ipath
/dev/ipath:     22648m 22651m
```

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

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

This command displays 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*
```

`lsof` can also take the same form:

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

Draft

The following command terminates all processes using the QLogic interconnect:

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

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

Note that hard and explicit program termination, such as `kill -9` on the `mpirun` Process ID (PID), may result in QLogic MPI being unable to guarantee that the `/dev/shm` shared memory file is properly removed. As many stale files accumulate on each node, an error message can appear at startup:

```
node023:6.Error creating shared memory object in shm_open(/dev/shm
may have stale shm files that need to be removed):
```

If this occurs, administrators should clean up all stale files by using this command:

```
# rm -rf /dev/shm/psm_shm.*
```

See "Error Creating Shared Memory Object" on page D-24 for more information.

Lock Enough Memory on Nodes when Using SLURM

This section is identical to information provided in "Lock Enough Memory on Nodes When Using a Batch Queuing System" on page D-23. It is repeated here for your convenience.

QLogic 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 128 MB, with the command `ulimit -l 131072`).

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 one of the following will be generated.

The following message displays during installation:

```
$ mpirun -np 2 -m ~/tmp/sm mpi_latency 1000 1000000
iqa-19:0.ipath_userinit: mmap of pio buffers at 100000 failed:
Resource temporarily unavailable
iqa-19:0.Driver initialization failure on /dev/ipath
iqa-20:1.ipath_userinit: mmap of pio buffers at 100000 failed:
Resource temporarily unavailable
iqa-20:1.Driver initialization failure on /dev/ipath
```

Draft

The following message displays after installation:

```
$ 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 similar to this displays if `ulimit -l` is less than 4096:

```
!!!ERROR!!! Lockable memory less than 4096KB on x nodes
```

To fix this error, install the `infinipath` RPM on the node, and reboot it to ensure that `/etc/initscript` is run.

Alternately, you can create your own `/etc/initscript` and set the `ulimit` there.

Draft

D Troubleshooting

This appendix describes some of the tools you can use to diagnose and fix problems. The following topics are discussed:

- Using LEDs to Check the State of the Adapter
- BIOS Settings
- Kernel and Initialization Issues
- OpenFabrics and InfiniPath Issues
- System Administration Troubleshooting
- Performance Issues
- QLogic MPI Troubleshooting

Troubleshooting information for hardware and software installation is found in the *InstallationGuideTitle*.

Using LEDs to Check the State of the Adapter

The LEDs function as link and data indicators once the InfiniPath software has been installed, the driver has been loaded, and the fabric is being actively managed by a subnet manager.

Table D-1 describes the LED states. The green LED indicates the physical link signal; the amber LED indicates the link. The green LED will normally illuminate first. The normal state is *Green On*, *Amber On*. The QLE7240 and QLE7280, have an additional state, as shown in Table D-1.

Table D-1. LED Link and Data Indicators

LED States	Indication
Green OFF Amber OFF	The switch is not powered up. The software is neither installed nor started. Loss of signal. Verify that the software is installed and configured with `ipath_control -i`. If correct, check both cable connectors.

Draft

Table D-1. LED Link and Data Indicators (Continued)

LED States	Indication
Green ON Amber OFF	Signal detected and the physical link is up. Ready to talk to SM to bring the link fully up. If this state persists, the SM may be missing or the link may not be configured. Use `ipath_control -i` to verify the software state. If all HCAs are in this state, then the SM is not running. Check the SM configuration, or install and run `opensmd`.
Green ON Amber ON	The link is configured, properly connected, and ready. Signal detected. Ready to talk to an SM to bring the link fully up. The link is configured. Properly connected and ready to receive data and link packets.
Green BLINKING (quickly) Amber ON	Indicates traffic
Green BLINKING[a] Amber BLINKING	Locates the adapter This feature is controlled by `ipath_control -b [On | Off]`

Table Notes

[a] This feature is available only on the QLE7240 and QLE7280 adapters

BIOS Settings

This section covers issues related to BIOS settings.The most important setting is:

- Advanced Configuration and Power Interface (ACPI). This setting must be enabled. If ACPI has been disabled, it may result in initialization problems, as described in "InfiniPath Interrupts Not Working" on page D-3.

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.

Issue with SuperMicro® H8DCE-HTe and QHT7040

The QLogic adapter may not be recognized at startup when using the Supermicro H8DCE-HT-e and the QHT7040 adapter. To fix this problem, set the operating system selector option in the BIOS for Linux. The option will look like:

```
OS Installation [Linux]
```

Draft

Kernel and Initialization Issues

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

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 display:

```
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 problem, install one of the appropriate supported Linux kernel versions as listed in "Supported Distributions and Kernels" on page 2-3, then reload the driver.

Rebuild or Reinstall Drivers if Different Kernel Installed

If you upgrade the kernel, then you must reboot and then rebuild or reinstall the InfiniPath kernel modules (drivers).

To rebuild the drivers, do the following (as root):

```
# cd /usr/src/qlogic_ib/kernel-ib-<version>
# ./make-install.sh
# /etc/init.d/openibd restart
```

An alternative method is to reinstall the InfiniPath kernel modules and then restart the InfiniPath service. Type (as root):

```
# rpm -U --replacepkgs kernel-ib-*
# /etc/init.d/openibd restart
```

InfiniPath Interrupts Not Working

The InfiniPath driver cannot configure the InfiniPath link to a usable state unless interrupts are working. Check for this problem with the command:

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

Normal output is similar to this:

```
           CPU0       CPU1
185:     364263          0   IO-APIC-level  ib_ipath
```

NOTE:

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

Draft

If there is no output at all, the driver initialization failed. For more information on driver problems, see “Driver Load Fails Due to Unsupported Kernel” on page D-3 or “InfiniPath ib_ipath Initialization Failure” on page D-5.

If the output is similar to one of these lines, then interrupts are not being delivered to the driver.

```
66: 0 0 PCI-MSI  ib_ipath
185:00IO-APIC-level  ib_ipath
```

The following message appears when driver has initialized successfully, but no interrupts are seen within 5 seconds.

```
ib_ipath 0000:82:00.0: No interrupts detected.
```

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

The possible causes of this problem are:

- Booting the Linux kernel with ACPI disabled on either 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 option, use this command:

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

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

To track down other initialization failures, see “InfiniPath ib_ipath Initialization Failure” on page D-5.

The program ipath_checkout can also help flag these kinds of problems. See “ipath_checkout” on page F-7 for more information.

Draft

OpenFabrics Load Errors if ib_ipath Driver Load Fails

When the ib_ipath driver fails to load, the other OpenFabrics drivers/modules will load and be shown by lsmod, but commands like ibstatus, ibv_devinfo, and ipath_control -i will fail as follows:

```
# ibstatus
Fatal error:  device '*': sys files not found
(/sys/class/infiniband/*/ports)
# ibv_devinfo
libibverbs: Fatal: couldn't read uverbs ABI version.
No IB devices found
# ipath_control -i
InfiniPath driver not loaded ?
No InfiniPath info available
```

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>:ipath_userinit: assign_port command failed: Network is
down
<nodename>:can't open /dev/ipath, network down
```

This will be followed by messages of this type after 60 seconds:

```
MPIRUN<node_where_started>: 1 rank has not yet exited 60 seconds
after rank 0 (node <nodename>) exited without reaching
MPI_Finalize().
MPIRUN<node_where_started>:Waiting at most another 60 seconds for
the remaining ranks to do a clean shutdown before terminating 1
node processes.
```

If this error appears, check to see if the InfiniPath driver is loaded by typing:

```
$ lsmod | grep ib_ipath
```

If no output is displayed, the driver did not load for some reason. In this case, try the following commands (as root):

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

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

Draft

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

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

This command may generate more than one screen of output.

Also, check the link status with the commands:

```
$ cat /sys/class/infiniband/ipath*/device/status_str
```

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

See also "status_str" on page F-16 and "ipath_checkout" on page F-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 appear 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
```

These messages may indicate that a cable is not connected, the switch is down, SM is not running, or that a hardware error occurred.

OpenFabrics and InfiniPath Issues

The following sections cover issues related to OpenFabrics (including OpenSM) and InfiniPath.

Stop OpenSM Before Stopping/Restarting InfiniPath

OpenSM must be stopped before stopping or restarting InfiniPath. Here is a sample command and the corresponding error messages:

```
# /etc/init.d/openibd 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]
```

Draft

Manual Shutdown or Restart May Hang if NFS in Use

If you are using NFS over IPoIB and use the manual `/etc/init.d/openibd stop` (or `restart`) command, the shutdown process may silently hang on the `fuser` command contained within the script. This is because `fuser` cannot traverse down the tree from the mount point once the mount point has disappeared. To remedy this problem, the `fuser` process itself need to be killed. Run the following command either as root or as the user who is running the `fuser` process:

```
# kill -9 fuser
```

The shutdown will continue.

This problem is not seen if the system is rebooted or if the filesystem has already been unmounted before stopping `infinipath`.

Load and Configure IPoIB Before Loading SDP

SDP generates Connection Refused errors if it is loaded before IPoIB has been loaded and configured. To solve the problem, load and configure IPoIB first.

Set `$IBPATH` for OpenFabrics Scripts

The environment variable `$IBPATH` must 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 the following 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, set the variable (if it has not been set already) to `/usr/bin`, as follows:

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

`ifconfig` Does Not Display Hardware Address Properly on RHEL4

The `ifconfig` command can verify IPoIB network interface configuration. However, ifconfig does not report the hardware address (`HWaddr`) properly on RHEL4 U4 machines. In the following example, all zeroes are returned:

```
# ifconfig ib0
ib0       Link encap:UNSPEC  HWaddr
00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
.
.
.
```

As a workaround, use this command to display the hardware address:

```
# ip addr
```

Draft

SDP Module Not Loading

If the settings for debug level and the zero copy threshold from InfiniPath release 2.0 are present in the release 2.2 `/etc/modprobe.conf` file (RHEL) or `/etc/modprobe.conf.local` (SLES) file, the SDP module may not load:

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

To solve the problem, remove this line.

`ibsrpdm` Command Hangs When Two HCAs are Installed but Only Unit 1 is Connected to the Switch

If multiple HCAs (unit 0 and unit 1) are installed, and only unit 1 is connected to the switch, the `ibsrpdm` (to set up an SRP target) command can hang. If unit 0 is connected, and unit 1 is disconnected, the problem does not occur.

When only unit 1 is connected to the switch, use the `-d` option with `ibsrpdm`, then, using the output from the ibsrpdm command, echo the new target info into `/sys/class/infiniband_srp/srp-ipath1-1/add_target`.

For example:

```
# ibsrpdm -d /dev/infiniband/umad1 -c
# echo \
id_ext=21000001ff040bf6,ioc_guid=21000001ff040bf6,dgid=fe800000000
0000021000001ff040bf6,pkey=ffff,service_id=f60b04ff01000021 >
/sys/class/infiniband_srp/srp-ipath1-1/add_target
```

Outdated ipath_ether Configuration Setup Generates Error

Ethernet emulation (`ipath_ether`) has been removed in this release, and, as a result, an error may be seen if the user still has an alias set previously by `modprobe.conf` (for example, `alias eth2 ipath_ether`).

When `ifconfig` or `ifup` are run, the error will look similar to this (assuming `ipath_ether` was used for `eth2`):

```
eth2: error fetching interface information: Device not found
```

To prevent the error message, remove the following files (assuming `ipath_ether` was used for `eth2`):

`/etc/sysconfig/network-scripts/ifcfg-eth`2 (for RHEL)

`/etc/sysconfig/network/ifcfg-eth-eth2` (for SLES)

QLogic recommends using the IP over InfiniBand protocol (IPoIB-CM), included in the standard OpenFabrics software releases, as a replacement for `ipath_ether`.

Draft

System Administration Troubleshooting

The following sections provide details on locating problems related to system administration.

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 to supply the necessary inter-node connectivity. Problems with these inter-switch (or intermediate) links are sometimes 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 notice this 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.

Two diagnostic tools, `ibhosts` and `ibtracert`, may also be helpful. The tool `ibhosts` lists all the InfiniBand (IB) nodes that the subnet manager recognizes. To check the InfiniBand path between two nodes, use the `ibtracert` command.

Performance Issues

The following sections discuss known performance issues.

Unexpected Low Bandwidth or Poor Latency

If MTRR mapping is used for write combining (instead of the PAT mechanism), the BIOS must be set to *Discrete* if there is 4GB or more memory in the system; it affects where the PCI, PCIe, and HyperTransport I/O Base Address Registers (BARs) are mapped. If there is 4GB or more memory in the system, and the MTRR mapping is not set to Discrete, the bandwidth will be very low (under 250 MB/s) on anything that normally runs near full bandwidth over the QHT7140 and QLE7140 adapters.

Since QLE7240 and QLE7280 adapters use SendDMA rather than PIO for larger messages, peak message bandwidth is no longer a symptom of this problem. In this case, it appears as poor latency with small (less than 8K) messages.

Draft

The exact symptoms can vary with BIOS, amount of memory, etc. When the driver starts, you may see these errors:

```
ib_ipath 0000:04:01.0: infinipath0: Performance problem: bandwidth
to PIO buffers is only 273 MiB/sec
infinipath: mtrr_add(feb00000,0x100000,WC,0) failed (-22)
infinipath: probe of 0000:04:01.0 failed with error -22
```

If you do not see any of these messages on your console, but suspect this problem, check the `/var/log/messages` file. Some systems suppress driver load messages but still output them to the log file.

To check the bandwidth, type:

```
$ ipath_pkt_test -B
```

When configured correctly, the QLE7140 and QLE7240 report in the range of 1150–1500 MB/s, while the QLE7280 reports in the range of 1950–3000 MB/s. The QHT7040/7140 adapters normally report in the range of 2300–2650 MB/s.

You can also use ipath_checkout (“ipath_checkout” on page F-7) to check for MTRR problems.

The `dmesg` program (“dmesg” on page F-3) can also be used for diagnostics.

Details on both the PAT and MTRR mechanisms, and how the options should be set are given in “Write Combining” on page E-1.

Large Message Receive Side Bandwidth Varies with Socket Affinity on Opteron Systems

On Opteron systems, when using the QLE7240 or QLE7280 in DDR mode, there is a receive side bandwidth bottleneck for CPUs that are not adjacent to the PCI Express root complex. This may cause performance to vary. The bottleneck is most obvious when using SendDMA with large messages on the farthest sockets. The best case for SendDMA is when both sender and receiver are on the closest sockets. Overall performance for PIO (and smaller messages) is better than with SendDMA.

MVAPICH Performance Issues

At the time of publication, MVAPICH over OpenFabrics over InfiniPath performance tuning has not been done. However, if MVAPICH on InfiniPath is configured to use PSM, performance comparable to QLogic MPI can be obtained.

Draft

Erratic Performance

Sometimes erratic performance is seen on applications that use interrupts. An example is inconsistent SDP latency when running a program such as `netperf`. This may be seen on AMD-based systems using the QLE7240 or QLE7280 adapters. If this happens, check to see if the program `irqbalance` is running. This program is a Linux daemon that distributes interrupts across processors. However, it may interfere with prior Interrupt ReQuest (IRQ) affinity settings, introducing timing anomalies. After stopping this process (as root), bind IRQ to a CPU for more consistent performance. First, stop `irqbalance`:

```
# /sbin/chkconfig irqbalance off
# /etc/init.d/irqbalance stop
```

Next, find the IRQ number and bind it to a CPU. The IRQ number can be found in one of two ways, depending upon the system used. Both methods are described in the following paragraphs.

NOTE:

Take care when cutting and pasting commands from PDF documents, as quotes are special characters and may not be translated correctly.

Method 1

Check to see if the IRQ number is found in `/proc/irq/`*`xxx`*, where *`xxx`* is the IRQ number in `/sys/class/infiniband/ipath*/device/irq`. Do this as root. For example:

```
# my_irq=`cat /sys/class/infiniband/ipath*/device/irq`
# ls /proc/irq
```

If `$my_irq` can be found under `/proc/irq/`, then type:

```
# echo 01 > /proc/irq/$my_irq/smp_affinity
```

Method 2

If the above command, `ls /proc/irq`, cannot find `$my_irq`, then use the following commands instead:

```
# my_irq=`cat /proc/interrupts|grep ib_ipath|awk \
'{print $1}'|sed -e 's/://'`
# echo 01 > /proc/irq/$my_irq/smp_affinity
```

This method is not the first choice because, on some systems, there may be two rows of `ib_ipath` output, and you will not know which of the two numbers to choose. However, if you cannot find `$my_irq` listed under `/proc/irq` (**Method 1**), this type of system most likely has only one line for `ib_ipath` listed in `/proc/interrupts`, so you can use **Method 2**.

Draft

Here is an example

```
# cat /sys/class/infiniband/ipath*/device/irq
98
# ls /proc/irq
0  10   11  13  15  233  4  50  7  8  90
1  106  12  14  2   3    5  58  66  74 9
```

(Note that you cannot find 98.)

```
# cat /proc/interrupts|grep ib_ipath|awk \
'{print $1}'|sed -e 's/://'
106
# echo 01 > /proc/irq/106/smp_affinity
```

Using the `echo` command immediately changes the processor affinity of an IRQ. However, two points need to be noted:

- The contents of the `smp_affinity` file may not reflect the expected values, even though the affinity change has taken place.
- If the driver is reloaded, the affinity assignment will revert to the default, so you will need to reset it to the desired value.

You can look at the stats in `/proc/interrupts` while the adapter is active to observe which CPU is fielding `ib_ipath` interrupts.

Performance Warning if `ib_ipath` Shares Interrupts with `eth0`

When `ib_ipath` shares interrupts with `eth0`, performance may be affected the OFED ULPs, such as IPoIB. A warning message appears in `syslog`, and also on the console or `tty` session where `/etc/init.d/openibd start` is run (if messages are set up to be displayed). Messages are in this form:

```
Nov  5 14:25:43 <nodename> infinipath: Shared interrupt will
affect performance: vector 169: devices eth0, ib_ipath
```

Check `/proc/interrupts`: "169" is in the first column, and "devices" are shown in the last column.

You can also contact your system vendor to see if the BIOS settings can be changed to avoid the problem.

QLogic MPI Troubleshooting

Problems specific to compiling and running MPI programs are described in the following sections.

Draft

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. In the following example, `mpirun` from release 2.1 is being used with a 2.2 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 Release2.3: Built on Wed Nov 6 17:28:58 PDT 2008
by mee
```

The following example is the error that occurs when `mpirun` from the 2.2 release is being used with the 2.1 libraries.

```
$ mpirun-ipath-ssh -np 2 -ppn 1 -m ~/tmp/idev osu_latency
MPIRUN: mpirun from the 2.3 software distribution requires all
node processes to be running 2.3 software. At least node <nodename>
uses non-2.3 MPI libraries
```

The following string means that either an incompatible non-QLogic `mpirun` binary has been found or that the binary is from an InfiniPath release prior to 2.3.

```
Found incompatible non-InfiniPath or pre-2.3
InfiniPath mpirun-ipath-ssh (exec=/usr/bin/mpirun-ipath-ssh)
```

Missing `mpirun` Executable

When the `mpirun` executable is missing, the following error appears:

```
Please install mpirun on <nodename> or provide a path to
mpirun-ipath-ssh
(not found in $MPICH_ROOT/bin, $PATH
or path/to/mpirun-ipath-ssh/on/the/head/node) or run with
mpirun -distributed=off
```

This error string means that an `mpirun` executable (`mpirun-ipath-ssh`) was not found on the computation nodes. Make sure that the `mpi-frontend-*` RPM is installed on all nodes that will use `mpirun`.

Draft

Resolving Hostname with Multi-Homed Head Node

By default, `mpirun` assumes that ranks can independently resolve the hostname obtained on the head node with `gethostname`. However, the hostname of a multi-homed head node may not resolve on the compute nodes. To address this problem, the following new option has been added to `mpirun`:

```
-listen-addr <hostname|IPv4>
```

This address will be forwarded to the ranks. To change the default, put this option in the global `mpirun.defaults` file or in a `user-local` file.

If the address on the frontend cannot be resolved, then a warning is sent to the console and to `syslog`. If you use the following command line, you may see messages similar to this:

```
% mpirun-ipath-ssh -np 2 -listen-addr foo -m ~/tmp/hostfile-idev
osu_bcast
MPIRUN.<nodename>: Warning: Couldn't resolve listen address 'foo'
on head node
(Unknown host), using it anyway...
MPIRUN.<nodename>: No node programs have connected within 60
seconds.
```

This message occurs if none of the ranks can connect back to the head node.

The following message may appear if some ranks cannot connect back:

```
MPIRUN.<nodename>: Not all node programs have connected within
60 seconds.
MPIRUN.<nodename>: No connection received from 1 node process on
node <nodename>
```

Cross-Compilation Issues

The GNU 4.*x* environment is supported in the PathScale Compiler Suite 3.*x* release.

However, the 2.*x* QLogic PathScale compilers are not currently supported on SLES 10 systems that use the GNU 4.*x* compilers and compiler environment (header files and libraries).

QLogic recommends installing the PathScale 3.1 release.

Draft

Compiler/Linker Mismatch

If the compiler and linker are not matching in C and C++ programs, the following error message appears:

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

Compiler Cannot 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 the following appear:

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

NOTE:

As noted in the Software Installation section of the *InstallationGuideTitle*, 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 ensure that the run-time library path is the same as the path that was used to compile the program.

The following examples show what compiler options to use for include files and libraries on the development nodes, and how to specify the 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 the examples in “Compiling on Development Nodes” on page D-16, it is assumed that the new locations are:

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

Draft

Compiling on Development Nodes

If the `mpi-devel-*` RPM is installed with the `--prefix /path/to/devel` option, then `mpicc`, etc. must be passed in `-I/path/to/devel/include` 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 needs 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 64 bit)
```

To find both the include files and the libraries with these non-standard locations, type:

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

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 compute nodes to include the path.
- Export the path in the `.mpirunrc` file.

These methods are explained in more detail in the following paragraphs.

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 ensures that the program will run using this path on any machine.

Draft

For the second option, change the file `/etc/ld.so.conf` on the compute nodes rather than using the `-Wl,-rpath,` option when compiling on the development node. It is assumed 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, add the following lines to the file `/etc/ld.so.conf`:

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

To make sure that the changes take effect, run (as root):

```
# /etc/ldconfig
```

The libraries can now be found by the runtime linker on the compute nodes. The advance to this method is that it works for all InfiniPath programs, without having to remember to change the compile/link lines.

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

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

See “Environment for Node Programs” on page 5-17 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.

Problem with Shell Special Characters and Wrapper Scripts

Care should be exercised in dealing with shell special characters, especially when using the `mpicc`, etc. wrapper scripts. These characters must be “escaped” to avoid the shell interpreting them.

For example, when compiling code using the `-D` compiler flag, `mpicc` (and other wrapper scripts) will fail if the defined variable contains a space, even when surrounded by double quotes. In the example below, the result of the `-show` option reveals what happens to the variable:

```
$ mpicc -show -DMYDEFINE="some value" test.c
gcc -c -DMYDEFINE=some value test.c
gcc -Wl,--export-dynamic,--allow-shlib-undefined test.o -lmpich
```

Draft

The shell strips off the double quotes before handing the arguments to the `mpicc` script, thus causing the problem. The workaround is to "escape" the double quotes and white space by using backslashes, so the shell does not process them. (Also note the single quote (‘) around the -D, since the scripts do an eval rather than directly invoking the underlying compiler.) Use this command instead:

```
$ mpicc -show -DMYDEFINE=\"some\ value\" test.c
gcc -c ‘-DMYDEFINE="some value"‘ test.c
gcc -Wl,--export-dynamic,--allow-shlib-undefined test.o -lmpich
```

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 provided in the following paragraphs.

In the following command, the HP-MPI version of mpirun is invoked by the full path name. However, the program `mpi_nxnlatbw` was compiled with the QLogic version of `mpicc`. The mismatch produces 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
```

Draft

In the next case, mpi_nxnlatbw.c is compiled with the HP-MPI version of mpicc, and given the name hpmpi-mpi_nxnlatbw, so that it is easy to see which version was used. However, it is run with the QLogic mpirun, which produces 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
```

The 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-2.3-5314.919_sles10_qlc
  ```

- 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.

Draft

See “Compiler Cannot Find Include, Module, or Library Files” on page D-15, “Compiling on Development Nodes” on page D-16, and “Specifying the Run-time Library Path” on page D-16 for additional information.

Process Limitation with `ssh`

MPI jobs that use more than eight 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, a message similar to this appears 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.

NOTE:

This limitation applies only if `-distributed=off` is specified. By default, with `-distributed=on`, you will not normally have this problem.

Number of Processes Exceeds `ulimit` for Number of Open Files

When users scale up the number of processes beyond the number of open files allowed by `ulimit`, `mpirun` will print an error message. The `ulimit` for the number of open files is typically 1024 on both RedHat and SLES. The message will look something like this:

```
MPIRUN.up001: Warning: ulimit for the number of open files is only
1024, but this mpirun request requires at least <number of files>
open files (sockets). The shell ulimit for open files needs to be
increased.
This is due to limit:
descriptors  1024
```

The `ulimit` can be increased; QLogic recommends an increase of approximately 20% over the number of CPUs. For example, in the case of 2048 CPUs, ulimit could be increased to 2500:

```
ulimit -n 2500
```

The `ulimit` only needs to be increased on the host where `mpirun` was started, unless the mode of operation allows `mpirun` from any node.

Draft

Using `MPI.mod` Files

`MPI.mod` (or `mpi.mod`) are the Fortran90/Fortran95 mpi modules files. These files 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 does not 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, create your own MPI module file and compile the application with it, rather than using 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 causes `/your/search/directory` to be checked before `/usr/include`. For example:

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

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

Extending MPI Modules

MPI implementations provide procedures that accept an argument having any data type, any precision, and any rank. However, it is not 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:

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

Draft

If you know that an argument can accept a data type that the MPI module does not explicitly allow, you can extend the interface for yourself. For example, the following program shows 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
```

Draft

This is equally applicable if the module `mpi` provides only a lower-rank interface and you want to add a higher-rank interface, for example, when the module explicitly provides for 1-D and 2-D integer arrays, but you need to pass a 3-D integer array. Add a higher-rank interface only under the following conditions:

- The module mpi provides an explicit Fortran 90 style interface for `mpi_bcast`. If the module `mpi` does not have this interface, the program uses 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 accepts 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.

Lock Enough Memory on Nodes When Using a Batch Queuing System

QLogic 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 128 MB, with the command `ulimit -l 131072`).

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 one of the following will be generated.

The following message displays during installation:

```
$ mpirun -np 2 -m ~/tmp/sm mpi_latency 1000 1000000
iqa-19:0.ipath_userinit: mmap of pio buffers at 100000 failed:
Resource temporarily unavailable
iqa-19:0.Driver initialization failure on /dev/ipath
iqa-20:1.ipath_userinit: mmap of pio buffers at 100000 failed:
Resource temporarily unavailable
iqa-20:1.Driver initialization failure on /dev/ipath
```

Draft

The following message displays after installation:

```
$ 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 similar to this displays if `ulimit -l` is less than 4096:

```
!!!ERROR!!! Lockable memory less than 4096KB on x nodes
```

To fix this error, install the `infinipath` RPM on the node, and reboot it to ensure that `/etc/initscript` is run.

Alternately, you can create your own `/etc/initscript` and set the `ulimit` there.

Error Creating Shared Memory Object

QLogic MPI (and PSM) use Linux's shared memory mapped files to share memory within a node. When an MPI job is started, a shared memory file is created on each node for all MPI ranks sharing memory on that one node. During job execution, the shared memory file remains in `/dev/shm`. At program exit, the file is removed automatically by the operating system when the QLogic MPI (InfiniPath) library properly exits. Also, as an additional backup in the sequence of commands invoked by `mpirun` during every MPI job launch, the file is explicitly removed at program termination.

However, under circumstances such as hard and explicit program termination (i.e. `kill -9` on the `mpirun` process PID), QLogic MPI cannot guarantee that the `/dev/shm` file is properly removed. As many stale files accumulate on each node, an error message like the following can appear at startup:

```
node023:6.Error creating shared memory object in shm_open(/dev/shm
may have stale shm files that need to be removed):
```

If this occurs, administrators should clean up all stale files by running this command (as root):

```
# rm -rf /dev/shm/psm_shm.*
```

You can also selectively identify stale files by using a combination of the `fuser`, `ps`, and `rm` commands (all files start with the `psm_shm` prefix). Once identified, you can issue `rm` commands on the stale files that you own.

Draft

NOTE:

It is important that `/dev/shm` be writable by all users, or else error messages like the ones in this section can be expected. Also, non-QLogic MPIs that use PSM may be more prone to stale shared memory files when processes are abnormally terminated.

`gdb` Gets SIG32 Signal Under `mpirun -debug` with the PSM Receive Progress Thread Enabled

When you run `mpirun -debug` and the PSM receive progress thread is enabled, `gdb` (the GNU debugger) reports the following error:

```
(gdb) run
Starting program: /usr/bin/osu_bcast < /dev/null [Thread debugging
using libthread_db enabled] [New Thread 46912501386816 (LWP
13100)] [New Thread 1084229984 (LWP 13103)] [New Thread 1094719840
(LWP 13104)]
Program received signal SIG32, Real-time event 32.
[Switching to Thread 1084229984 (LWP 22106)] 0x00000033807c0930 in
poll () from /lib64/libc.so.6
```

This signal is generated when the main thread cancels the progress thread. To fix this problem, disable the receive progress thread when debugging an MPI program. Add the following line to `$HOME/.mpirunrc`:

```
export PSM_RCVTHREAD=0
```

NOTE:

Remove the above line from `$HOME/.mpirunrc` after you debug an MPI program. If this line is not removed, the PSM receive progress thread will be permanently disabled. To check if the receive progress thread is enabled, look for output similar to the following when using the `mpirun -verbose` flag:

```
idev-17:0.env PSM_RCVTHREAD Recv thread flags
0 disables thread) => 0x1
```

The value `0x1` indicates that the receive thread is currently enabled. A value of `0x0` indicates that the receive thread is disabled.

General Error Messages

The following message may be generated by `ipath_checkout` or `mpirun`.

```
PSM found 0 available contexts on InfiniPath device
```

Draft

The most likely cause is that the cluster has processes using all the available PSM contexts.

Error Messages Generated by `mpirun`

The following sections describe the `mpirun` error messages. These messages are in one of these categories:

- Messages from the QLogic MPI (InfiniPath) library
- MPI messages
- Messages relating to the InfiniPath driver and InfiniBand links

Messages generated by `mpirun` follow this format:

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

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

Messages from the QLogic MPI (InfiniPath) Library

Messages from the QLogic MPI (InfiniPath) library appear in the `mpirun` output.

The following example contains rank values received during connection setup that were higher than the number of ranks (as indicated in the mpirun startup code):

```
sender rank rank is out of range (notification)
sender rank rank is out of range (ack)
```

The following are error messages, which indicate internal problems and must be reported to Technical Support.

```
unknown frame type type
[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 usually 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, try to reboot. If that does not work, call Technical Support.

The following message indicates a mismatch between the QLogic 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
```

Draft

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

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

```
assign_port command failed: str
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: str
can't stat infinipath device to determine type: str
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
mmap of egr bufs from h failed: str
mmap of pio buffers at %llx failed: str
mmap of pioavail registers (%llx) failed: str
mmap of rcvhdr q failed: str
mmap of user registers at %llx failed: str
userinit command failed: str
Failed to set close on exec for device: str
```

NOTE:

These messages should never occur. If they do, notify Technical Support.

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 error messages should rarely occur; they 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
```

In this case, `str` can give additional information about why the failure occurred.

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

```
Couldn’t connect to <IP> (LID=<lid>:<port>:<subport>). Time
elapsed 00:00:30. Still trying...
```

Draft

IP is the MPI rank's IP address, and <lid><port><subport> are the rank's lid, port, and subport.

If messages similar to the following display, it may mean that the program is 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 rendezvous, 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 QLogic hardware. This error indicates a failure of the program, not the hardware or driver.

MPI Messages

Some MPI error messages are issued from the parts of the code inherited from the MPICH implementation. See the MPICH documentation for message descriptions. This section discusses the error messages specific to the QLogic 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.

The following message is always followed by an abort. The processlabel is usually in the form of the host name followed by process rank:

```
processlabel Fatal Error in filename line_no: error_string
```

At the time of publication, the possible error_strings are:

```
Illegal label format character.
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.
Node table has inconsistent len! Hdr claims %d not %d
Timeout waiting %d seconds to receive peer node table from mpirun
```

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
```

Draft

The following indicates that 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>
```

The following indicates that 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.
```

The following indicates that node jobs have started, but one host could not 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>.<rank> Cannot connect to spawner on host %s port %d
within 60 seconds.
MPIRUN: Some node programs ended prematurely without connecting to
mpirun.
MPIRUN: No connection received from 1 node process on node
<nodename>
```

The following indicates that node jobs have started, but both hosts could not 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>.<rank> Cannot connect to spawner on host %s port %d
within 60 seconds.
6083.psc_skt_connect: Error connecting to socket: No route to host
<nodename>.<rank> Cannot connect to spawner on host %s port %d
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.
```

The following indicates that one program on one node died:

```
$ mpirun -np 2 -m ~/tmp/q mpi_latency 100000 1000000
MPIRUN: <nodename> node program unexpectedly quit: Exiting.
```

Draft

The `quiescence detected` message is printed when an MPI job is not making progress. The default timeout is 900 seconds. After this length of time, all the node processes are 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 code is an example of the type of message that displays 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 message should never occur. If it does, notify Technical Support:

```
Internal Error: NULL function/argument found:func_ptr(arg_ptr)
```

Driver and Link Error Messages Reported by MPI Programs

The following driver and link error messages are reported by MPI programs:

- When the InfiniBand link fails during a job, a message is reported once per occurrence. The message will be similar to:

  ```
  ipath_check_unit_status: IB Link is down
  ```

  This message occurs when a cable is disconnected, a switch is rebooted, or when there are other problems with the link. The job continues retrying until the quiescence interval expires. See the `mpirun -q` option for information on quiescence.

- If a hardware problem occurs, an error similar to this displays:

  ```
  infinipath: [error strings ] Hardware error
  ```

Draft

In this case, the MPI program terminates. The error string may provide additional information about the problem. To further determine the source of the problem, examine `syslog` on the node reporting the problem.

MPI Stats

Using the `-print-stats` option to mpirun provides a listing to stderr of various MPI statistics. Here is example output for the `-print-stats` option when used with an eight-rank run of the HPCC benchmark, using the following command:

```
$ mpirun -np 8 -ppn 1 -m machinefile -M ./hpcc
```

```
STATS: MPI Statistics Summary (max,min @ rank)
STATS: Eager count sent       (max=171.94K @ 0, min=170.10K @ 3, med=170.20K @ 5)
STATS: Eager bytes sent       (max=492.56M @ 5, min=491.35M @ 0, med=491.87M @ 1)
STATS: Rendezvous count sent  (max= 5735 @ 0, min= 5729 @ 3, med= 5731 @ 7)
STATS: Rendezvous bytes sent  (max= 1.21G @ 4, min= 1.20G @ 2, med= 1.21G @ 0)
STATS: Expected count received(max=173.18K @ 4, min=169.46K @ 1, med=172.71K @ 7)
STATS: Expected bytes received(max= 1.70G @ 1, min= 1.69G @ 2, med= 1.70G @ 7)
STATS: Unexpect count received(max= 6758 @ 0, min= 2996 @ 4, med= 3407 @ 2)
STATS: Unexpect bytes received(max= 1.48M @ 0, min=226.79K @ 5, med=899.08K @ 2)
```

By default, `-M` assumes `-M=mpi` and that the user wants only `mpi` level statistics. The man page shows various other low-level categories of statistics that are provided. Here is another example:

```
$ mpirun -np 8 -ppn 1 -m machinefile -M=mpi,ipath hpcc
```

```
STATS: MPI Statistics Summary (max,min @ rank)
STATS: Eager count sent       (max=171.94K @ 0, min=170.10K @ 3, med=170.22K @ 1)
STATS: Eager bytes sent       (max=492.56M @ 5, min=491.35M @ 0, med=491.87M @ 1)
STATS: Rendezvous count sent  (max= 5735 @ 0, min= 5729 @ 3, med= 5731 @ 7)
STATS: Rendezvous bytes sent  (max= 1.21G @ 4, min= 1.20G @ 2, med= 1.21G @ 0)
STATS: Expected count received(max=173.18K @ 4, min=169.46K @ 1, med=172.71K @ 7)
STATS: Expected bytes received(max= 1.70G @ 1, min= 1.69G @ 2, med= 1.70G @ 7)
STATS: Unexpect count received(max= 6758 @ 0, min= 2996 @ 4, med= 3407 @ 2)
STATS: Unexpect bytes received(max= 1.48M @ 0, min=226.79K @ 5, med=899.08K @ 2)
STATS: InfiniPath low-level protocol stats
STATS: pio busy count         (max=190.01K @ 0, min=155.60K @ 1, med=160.76K @ 5)
STATS: scb unavail exp count  (max= 9217 @ 0, min= 7437 @ 7, med= 7727 @ 4)
STATS: tid update count       (max=292.82K @ 6, min=290.59K @ 2, med=292.55K @ 4)
STATS: interrupt thread count (max=   941 @ 0, min=   335 @ 7, med=   439 @ 2)
STATS: interrupt thread success(max= 0.00 @ 3, min= 0.00 @ 1, med= 0.00 @ 0)
```

Statistics other than MPI-level statistics are fairly low level; most users will not understand them. Contact QLogic Technical Support for more information.

Message statistics are available for transmitted and received messages. In all cases, the MPI rank number responsible for a minimum or maximum value is reported with the relevant value. For application runs of at least three ranks, a median is also available.

Draft

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 *expected* or *unexpected* messages. Unexpected messages cause the MPI implementation to buffer the transmitted data until the receiver can produce a matching MPI receive buffer. Expected messages refer to the inverse case, which is 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 detailed information, use MPI profilers such as mpiP. For more information on mpiP, see:
http://mpip.sourceforge.net/

For information about the HPCC benchmark, see:
http://icl.cs.utk.edu/hpcc/

Draft

E Write Combining

Introduction

Write combining improves write bandwidth to the QLogic chip by writing multiple words in a single bus transaction (typically 64 bytes). Write combining applies only to x86_64 systems.

The x86 Page Attribute Table (PAT) mechanism that allocates write-combining (WC) mappings for the PIO buffers has been added and is now the default.

If PAT is unavailable or PAT initialization fails for some reason, the code will generate a message in the log and fall back to the Memory Type Range Registers (MTRR) mechanism.

If write combining is not working properly, lower than expected bandwidth may occur.

Instructions for checking write combining and for using PAT and MTRR are given below.

Verify Write Combining is Working

To see if write combining is working correctly and to check the bandwidth, run the following command:

```
$ ipath_pkt_test -B
```

With write combining enabled, the QLE7140 and QLE7240 report in the range of 1150–1500 MB/s; the QLE7280 reports in the range of 1950–3000 MB/s. The QHT7040/7140 adapters normally report in the range of 2300–2650 MB/s.

You can also use `ipath_checkout` (use option 5) to check bandwidth.

Although the PAT mechanism should work correctly by default, increased latency and low bandwidth may signal a problem. If so, the interconnect operates, but in a degraded performance mode, with latency increasing to several microseconds, and bandwidth decreasing to as little as 200 MB/s.

Upon driver startup, you may see these errors:

```
ib_ipath 0000:04:01.0: infinipath0: Performance problem: bandwidth
to PIO buffers is only 273 MiB/sec
.
```

Draft

.
.

If you do not see any of these messages on your console, but suspect this problem, check the `/var/log/messages` file. Some systems suppress driver load messages but still output them to the log file.

Methods for enabling and disabling the two write combining mechanisms are given below. There are no conflicts between the two methods.

PAT and Write Combining

This is the default mechanism for allocating write-combining (WC) mappings for the PIO buffers. It is set as a parameter in `/etc/modprobe.conf` (on Red Hat systems) or `/etc/modprobe.conf.local` (on SLES systems). This is the default:

```
option ib_ipath wc_pat=1
```

If PAT is unavailable or PAT initialization fails for some reason, the code will generate a message in the log and fall back to the Memory Type Range Registers (MTRR) mechanism. To use MTRR, disable PAT by setting this module parameter to 0 (as root):

```
option ib_ipath wc_pat=0
```

Then, revert to using the MTRR-only behavior by following one of the two suggestions in “MTRR Mapping and Write Combining” on page E-2, below.

The driver will need to be restarted after the changes have been made

NOTE:

There will be no write-combining entry in `/proc/mtrr` when using PAT.

MTRR Mapping and Write Combining

Two suggestions for properly enabling MTRR mapping for write combining are described below.

See the Troubleshooting section of the *QLogic HCA and QLogic OFED Software User Guide* for more details on a related performance issue.

Edit BIOS Settings to Fix MTRR Issues

You can edit the BIOS setting for MTRR Mapping. The BIOS setting looks similar to:

```
MTRR Mapping          [Discrete]
```

Draft

For systems with very large amounts of memory (32GB or more), it may also be necessary to adjust the BIOS setting for the *PCI hole granularity* to 2GB. This setting allows the memory to be mapped with fewer MTRRs, so that there will be one or more unused MTRRs for the InfiniPath driver.

Some BIOS' do not have the MTRR mapping option. It may have a different name, depending on the chipset, vendor, BIOS, or other factors. For example, it is sometimes referred to as *32 bit memory hole*. This setting must be enabled.

If there is no setting for MTRR mapping or 32 bit memory hole, and you have problems with degraded performance, contact your system or motherboard vendor and ask how to enable write combining.

Use the `ipath_mtrr` Script to Fix MTRR Issues

QLogic also provides a script, `ipath_mtrr`, which sets the MTRR registers, enabling maximum performance from the InfiniPath driver. This Python script is available as a part of the InfiniPath software download, and is contained in the `infinipath*` RPM. It is installed in `/bin`.

To diagnose the machine, run it with no arguments (as root):

```
# ipath_mtrr
```

The test results will list any problems, if they exist, and provide suggestions on what to do.

To fix the MTRR registers, use:

```
# ipath_mtrr -w
```

Restart the driver after fixing the registers.

This script needs to be run after each system reboot. It can be set to run automatically upon restart by adding this line in `/etc/sysconfig/infinipath`:

```
IPATH_MTRR_ACTIVE=1
```

See the `ipath_mtrr(8)` man page for more information on other options.

Draft

Notes

Draft

F Useful Programs and Files

The most useful programs and files for debugging, and commands for common tasks are presented in the following sections. Many of these programs and files have been discussed elsewhere in the documentation. This information is summarized and repeated here for your convenience.

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 QLogic boards
- Runtime and build environments
- `.o` files from different compilers
- Libraries
- Processor/link speeds
- PIO bandwidth
- MTUs

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 F-1. More details on `ipath_checkout` are in "`ipath_checkout`" on page F-7.

Restarting InfiniPath

When the driver status appears abnormal on any node, you can try restarting (as root):

```
# /etc/init.d/openibd restart
```

These two commands perform the same function as `restart`:

```
# /etc/init.d/openibd stop
# /etc/init.d/openibd start
```

Draft

Also check the /var/log/messages file for any abnormal activity.

Summary and Descriptions of Useful Programs

Useful programs are summarized in Table F-1. Names in blue text are linked to a corresponding section that provides further details. Check the man pages for more information on the programs.

Table F-1. Useful Programs

Program Name	Function
chkconfig	Checks the configuration state and enables/disables services, including drivers. Can be useful for checking homogeneity.
dmesg	Prints out bootup messages. Useful for checking for initialization problems.
ibhosts[a]	Checks that all hosts in the fabric are up and visible to the subnet manager and to each other
ibstatus[a]	Checks the status of InfiniBand devices when OpenFabrics is installed
ibtracert[a]	Determines the path that InfiniBand packets travel between two nodes
ibv_devinfo[a]	Lists information about InfiniBand devices in use. Use when OpenFabrics is enabled.
ident[b]	Identifies RCS keyword strings in files. Can check for dates, release versions, and other identifying information.
ipathbug-helper[c]	A shell script that gathers status and history information for use in analyzing InfiniPath problems
ipath_checkout[c]	A bash shell script that performs sanity testing on a cluster using QLogic hardware and InfiniPath software. When the program runs without errors, the node is properly configured.
ipath_control[c]	A shell script that manipulates various parameters for the InfiniPath driver. This script gathers the same information contained in boardversion, status_str, and version.
ipath_mtrr[c]	A Python script that sets the MTRR registers.
ipath_pkt_test[c]	Tests the InfiniBand link and bandwidth between two QLogic HCAs, or, using an InfiniBand loopback connector, tests within a single QLogic HCA

Table F-1. Useful Programs (Continued)

Program Name	Function
ipathstats[c]	Displays driver statistics and hardware counters, including performance and "error" (including status) counters
lsmod	Shows status of modules in the Linux kernel. Use to check whether drivers are loaded.
modprobe	Adds or removes modules from the Linux kernel.
mpi_stress	An MPI stress test program designed to load up an MPI interconnect with point-to-point messages while optionally checking for data integrity.
mpirun[d]	A front end program that starts an MPI job on an InfiniPath cluster. Use to check the origin of the drivers.
ps	Displays information on current active processes. Use to check whether all necessary processes have been started.
rpm	Package manager to install, query, verify, update, or erase software packages. Use to check the contents of a package.
strings[e]	Prints the strings of printable characters in a file. Useful for determining contents of non-text files such as date and version.

Table Notes

[a] These programs are contained in the OpenFabrics openib-diags RPM.

[b] Contained within the rcs RPM for your distribution.

[c] These programs are contained in the infinipath RPM. To use these programs, install the infinipath RPM on the node(s) where you install the mpi-frontend RPM.

[d] Contained in the QLogic mpi-frontend RPM.

[e] Contained within the binutils RPM for your distribution.

dmesg

dmesg prints out bootup messages. It is useful for checking for initialization problems. You can check to see if problems were detected during the driver and QLogic hardware initialization with the command:

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

This command may generate more than one screen of output.

Draft

ibhosts

This tool determines if all the hosts in your InfiniBand fabric are up and visible to the subnet manager and to each other. It is installed from the `openib-diag` RPM. Running `ibhosts` (as root) produces output similar to this when run from a node on the InfiniBand fabric:

```
# ibhosts
Ca : 0x0008f10001280000 ports 2 "Voltaire InfiniBand
              Fiber-Channel Router"
Ca : 0x0011750000ff9869 ports 1 "idev-11"
Ca : 0x0011750000ff9878 ports 1 "idev-05"
Ca : 0x0011750000ff985c ports 1 "idev-06"
Ca : 0x0011750000ff9873 ports 1 "idev-04"
```

ibstatus

This program displays basic information on the status of InfiniBand devices that are currently in use when OpenFabrics RPMs are installed. It is installed from the `openib-diag` RPM.

Following is a sample output for the SDR adapters:

```
$ 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)
```

Following is a sample output for the DDR adapters; note the difference in rate:

```
$ ibstatus
Infiniband device 'ipath0' port 1 status:
default gid:     fe80:0000:0000:0000:0011:7500:00ff:9608
base lid:        0xb
sm lid:          0x1
state:           4: ACTIVE
phys state:      5: LinkUp
rate:            20 Gb/sec (4X DDR)
```

Draft

ibtracert

The tool `ibtracert` determines the path that InfiniBand packets travel between two nodes. It is installed from the `openib-diag` RPM. The InfiniBand LIDs of the two nodes in this example are determined by using the `ipath_control -i` command on each node. The `ibtracert` tool produces output similar to the following when run (as root) from a node on the InfiniBand fabric:

```
# ibtracert 0xb9 0x9a
   From ca {0x0011750000ff9886} portnum 1 lid 0xb9-0xb9 "iqa-37"
   [1] -> switch port {0x0002c9010a19bea0}[1] lid 0x14-0x14
"MT47396 Infiniscale-III"
   [24] -> switch port {0x00066a0007000333}[8] lid 0xc-0xc
"SilverStorm 9120 GUID=0x00066a000200016c Leaf 6, Chip A"
   [6] -> switch port {0x0002c90000000000}[15] lid 0x9-0x9
"MT47396 Infiniscale-III"
   [7] -> ca port {0x0011750000ff9878}[1] lid 0x9a-0x9a "idev-05"
   To ca {0x0011750000ff9878} portnum 1 lid 0x9a-0x9a "idev-05"
```

ibv_devinfo

This program displays information about InfiniBand devices, including various kinds of identification and status data. It is installed from the `openib-diag` RPM. Use this program when OpenFabrics is enabled. `ibv_devinfo` queries RDMA devices. Use the -v option to see more information. Sample usage:

```
$ ibv_devinfo
hca_id: ipath0
        fw_ver:                         0.0.0
        node_guid:                      0011:7500:00ff:89a6
        sys_image_guid:                 0011:7500:00ff:89a6
        vendor_id:                      0x1175
        vendor_part_id:                 29216
        hw_ver:                         0x2
        board_id:                       InfiniPath_QLE7280
        phys_port_cnt:                  1
                port:   1
                        state:                  PORT_ACTIVE (4)
                        max_mtu:                4096 (5)
                        active_mtu:             4096 (5)
                        sm_lid:                 1
                        port_lid:               31
                        port_lmc:               0x00
```

Draft

ident

The ident strings are available in ib_ipath.ko. Running ident provides driver information similar to the following. For QLogic RPMs, it will look like:

```
$ ident
/lib/modules/$(uname-r)/updates/kernel/drivers/infiniband/hw/ipath
/ib_ipath.ko
/lib/modules/2.6.16.46-0.12-smp/updates/kernel/drivers/infiniband/
hw/ipath/ib_ipath.ko:
$Id: QLogic OFED Release 1.4 $
$Date: Fri Feb 27 16:14:31 PST 2009 $
$Id: QLogic OFED Release 1.4 $
$Date: Fri Feb 27 16:14:39 PST 2009 $
```

If the /lib/modules/$(uname -r)/updates directory is not present, then the driver in use is the one that comes with the core kernel. In this case, either the kernel-ib RPM is not installed or it is not configured for the current running kernel.

If the updates directory is present, but empty except for the subdirectory kernel, then an OFED install is probably being used, and the ident string will be empty. For example:

```
$ cd /lib/modules/$(uname -r)/updates
$ ls
kernel
$ cd kernel/drivers/infiniband/hw/ipath/
lib/modules/2.6.18-8.el5/updates/kernel/drivers/infiniband/hw/ipat
h
$ ident ib_ipath.ko
        ib_ipath.ko:
        ident warning: no id keywords in ib_ipath.ko
```

NOTE:

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

ipathbug-helper

The tool ipathbug-helper is useful for verifying homogeneity. It is installed from the infinipath RPM. Before contacting QLogic Technical Support, run this script on the head node of your cluster and the compute nodes that you suspect are having problems. Looking at the output often helps you to see the problem. Run ipathbug-helper on several nodes and examine the output for differences.

Draft

It is best to run `ipathbug-helper` with root privilege, since some of the queries it makes requires this level of privilege. There is also a `--verbose` parameter, which increases the amount of gathered information.

If you cannot see the problem, send the `stdout` output to your reseller, along with information on the version of the InfiniPath software you are using.

ipath_checkout

The `ipath_checkout` tool is a `bash` script that verifies that the installation is correct and that all the nodes of the network are functioning and mutually connected by the InfiniPath fabric. It is installed from the `infinipath` RPM. It must be run on a front end node, and requires specification of a nodefile. For example:

```
$ ipath_checkout [options] nodefile
```

The `nodefile` lists the hostnames of the nodes of the cluster, one hostname per line. The format of `nodefile` is as follows:

```
hostname1
hostname2
...
```

NOTE:

- The hostnames in the `nodefile` are Ethernet hostnames, not IPv4 addresses.
- To create a `nodefile` you can use the `ibhosts` program. It will generate a list of available nodes that are already connected to the switch.

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

1. Executes the `ping` command to all nodes to verify that they all are reachable from the front end.
2. Executes the `ssh` command to each node to verify correct configuration of `ssh`.
3. Gathers and analyzes system configuration from the nodes.
4. Gathers and analyzes RPMs installed on the nodes.
5. Verifies InfiniPath hardware and software status and configuration, including tests for link speed, PIO bandwidth (incorrect MTRR settings), and MTU size.
6. Verifies the ability to `mpirun` jobs on the nodes.
7. Runs a bandwidth and latency test on every pair of nodes and analyzes the results.

Draft

The options available with `ipath_checkout` are shown in Table F-2.

Table F-2. `ipath_checkout` Options

Command	Meaning
`-h, --help`	These options display help messages describing how a command is used.
`-v, --verbose -vv, --vverbose -vvv, --vvverbose`	These options specify three successively higher levels of detail in reporting test results. There are four levels of detail in all, including the case where none of these options are given.
`-c, --continue`	When this option is not specified, the test terminates when any test fails. When specified, the tests continue after a failure, with failing nodes excluded from subsequent tests.
`-k, --keep`	This option keeps intermediate files that were created while performing tests and compiling reports. Results are saved in a directory created by `mktemp` and named `infinipath_XXXXXX` or in the directory name given to `--workdir`.
`--workdir=DIR`	Use `DIR` to hold intermediate files created while running tests. `DIR` must not already exist.
`--run=LIST`	This option runs only the tests in `LIST`. See the seven tests listed previously. For example, `--run=123` will run only tests 1, 2, and 3.
`--skip=LIST`	This option skips the tests in `LIST`. See the seven tests listed previously. For example, `--skip=2457` will skip tests 2, 4, 5, and 7.
`-d, --debug`	This option turns on the `-x` and `-v` flags in `bash(1)`.

In most cases of failure, the script suggests recommended actions. Also refer to the `ipath_checkout` man page.

ipath_control

The `ipath_control` tool is a shell script that manipulates various parameters for the InfiniPath driver. It is installed from the `infinipath` RPM. Many of the parameters are used only when diagnosing problems, and may require special system configurations. Using these options may require restarting the driver or utility programs to recover from incorrect parameters.

Draft

Most of the functionality is accessed via the `/sys` filesystem. This shell script gathers the same information contained in these files:

```
/sys/class/infiniband/ipath0/device/boardversion
/sys/class/infiniband/ipath0/device/status_str
/sys/class/infiniband/ipath0/device/driver/version
```

These files are also documented in Table F-4 and Table F-5.

Other than the `-i` option, this script must be run with root permissions. See the man pages for `ipath_control` for more details.

Here is sample usage and output:

```
% ipath_control -i
$Id: QLogic OFED Release 1.4 $ $Date: Mon Feb 23 21:39:17 PST 2009
$
0: Version: ChipABI 2.0, InfiniPath_QLE7280, InfiniPath1 5.2, PCI
2, SW Compat 2
0: Status: 0xe1 Initted Present IB_link_up IB_configured
0: LID=0x8 MLID=0xc042 GUID=00:11:75:00:00:ff:8f:37 Serial:
AIB0807A28872
```

The `-i` option combined with the `-v` option is very useful for looking at the IB width/rate and PCIe lanes/rate:

```
% ipath_control -iv
$Id: QLogic OFED Release 1.4 $ $Date: Mon Feb 23 21:39:17 PST 2009
$
0: Version: ChipABI 2.0, InfiniPath_QLE7280, InfiniPath1 5.2, PCI
2, SW Compat 2
0: Status: 0xe1 Initted Present IB_link_up IB_configured
0: LID=0x8 MLID=0xc042 GUID=00:11:75:00:00:ff:8f:37 Serial:
AIB0807A28872
0: HRTBT:Auto RX_polarity_invert:Auto RX_lane_reversal: Auto
0: LinkWidth:4X of 1X|4X Speed:DDR of SDR|DDR
```

NOTE:

On the first line, `Release<version>` refers to the current software release. The second line contains chip architecture version information.

Another useful option blinks the LED on the InfiniPath adapter (QLE7240 and QLE7280 adapters). This is useful for finding an adapter within a cluster. Run the following as root:

```
# ipath_control -b [On|Off]
```

Draft

ipath_mtrr

NOTE:

Use `ipath_mtrr` if you are not using the default PAT mechanism to enable write combining.

MTRR is used by the InfiniPath driver to enable write combining to the QLogic on-chip transmit buffers. This option improves write bandwidth to the QLogic chip by writing multiple words in a single bus transaction (typically 64 bytes). This option applies only to x86_64 systems. It can often be set in the BIOS.

However, some BIOS' do not have the MTRR mapping option. It may have a different name, depending on the chipset, vendor, BIOS, or other factors. For example, it is sometimes referred to as *32 bit memory hole.* This setting must be enabled.

If there is no setting for MTRR mapping or 32 bit memory hole, contact your system or motherboard vendor and ask how to enable write combining.

You can check and adjust these BIOS settings using the BIOS Setup utility. For specific instructions, follow the hardware documentation that came with your system.

QLogic also provides a script, `ipath_mtrr`, which sets the MTRR registers, enabling maximum performance from the InfiniPath driver. This Python script is available as a part of the InfiniPath software download, and is contained in the `infinipath*` RPM. It is installed in `/bin`.

To diagnose the machine, run it with no arguments (as root):

```
# ipath_mtrr
```

The test results will list any problems, if they exist, and provide suggestions on what to do.

To fix the MTRR registers, use:

```
# ipath_mtrr -w
```

Restart the driver after fixing the registers.

This script needs to be run after each system reboot. It can be set to run automatically upon restart by adding this line in `/etc/sysconfig/infinipath`:

```
IPATH_MTRR_ACTIVE=1
```

See the `ipath_mtrr(8)` man page for more information on other options.

Draft

ipath_pkt_test

This program is installed from the `infinipath` RPM. Use `ipath_pkt_test` to do one of the following:

- Test the InfiniBand link and bandwidth between two InfiniPath HCAs.
- Using an InfiniBand loopback connector, test the link and bandwidth within a single InfiniPath HCA.

The `ipath_pkt_test` program 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).

Upon 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.

ipathstats

The `ipathstats` program is useful for diagnosing InfiniPath problems, particularly those that are performance related. It is installed from the `infinipath` RPM. It displays both driver statistics and hardware counters, including both performance and "error" (including status) counters.

Running `ipathstats -c 10`, for example, displays the number of packets and 32-bit words of data being transferred on a node in each 10-second interval. This output may show differences in traffic patterns on different nodes, or at different stages of execution. See the man page for more information.

lsmod

When you need to find which InfiniPath and OpenFabrics modules are running, type the following command:

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

modprobe

Use this program to load/unload the drivers. You can check to see if the driver has loaded by using this command:

```
# modprobe -v ib_ipath
```

The `-v` option typically only prints messages if there are problems.

The configuration file that `modprobe` uses is `/etc/modprobe.conf` (`/etc/modprobe.conf.local` on SLES). In this file, various options and naming aliases can be set.

Draft

mpirun

mpirun can be used to determine whether the program is being run against a QLogic or non-QLogic driver. It is installed from the mpi-frontend RPM. Sample commands and results are shown in the following paragraphs.

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
```

mpi_stress

This is an MPI stress test program designed to load up an MPI interconnect with point-to-point messages while optionally checking for data integrity. By default, it runs with all-to-all traffic patterns, optionally including oneself and one's local shared memory (shm) peers. It can also be set up with multi-dimensional grid traffic patterns; this can be parameterized to run rings, open 2D grids, closed 2D grids, cubic lattices, hypercubes, and so on.

Optionally, the message data can be randomized and checked using CRC checksums (strong but slow) or XOR checksums (weak but fast). The communication kernel is built out of non-blocking point-to-point calls to load up the interconnect. The program is not designed to exhaustively test out different MPI primitives.Performance metrics are displayed, but should be carefully interpreted in terms of the features enabled.

This is an MPI application and should be run under mpirun or its equivalent.

The following example runs 16 processes and a specified hosts file using the default options (all-to-all connectivity, 64 to 4MB messages in powers of two, one iteration, no data integrity checking):

```
$ mpirun -np 16 -m hosts mpi_stress
```

There are a number of options for mpi_stress; this one may be particularly useful:

-P Poison receive buffers at init and after each receive; pre-initialize with random data so that any parts that are not being correctly updated with received data can be observed later.

See the mpi_stress(1) man page for more information.

Draft

rpm

To check the contents of an installed RPM, use these commands:

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

The option `-q` queries. The option `--qa` queries all. To query a package that has not yet been installed, use the `-qpl` option.

strings

Use the `strings` command to determine the content of and extract text from a binary file.

The command `strings` can also be used. For example, the command:

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

produces this output:

```
$Date: 2009-02-26 12:05 Release2.3 InfiniPath $
```

NOTE:

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

Draft

Common Tasks and Commands

Table F-3 lists some common commands that help with administration and troubleshooting. Note that `mpirun` in `nonmpi` mode can perform a number of checks.

Table F-3. Common Tasks and Commands Summary

Function	Command

Check the system state	`ipath_checkout [options] hostsfile`
	`ipathbug-helper -m hostsfile \`
	`> ipath-info-allhosts`
	`mpirun -m hostsfile -ppn 1 \`
	`-np numhosts -nonmpi ipath_control -i`
	Also see the file:
	`/sys/class/infini-band/ipath*/device/status_str`
	where * is the unit number. This file provides information about the link state, possible cable/switch problems, and hardware errors.

Verify hosts via an Ethernet ping	`ipath_checkout --run=1 hostsfile`
Verify `ssh`	`ipath_checkout --run=2 hostsfile`
Show `uname -a` for all hosts	`mpirun -m hostsfile -ppn 1 \` `-np numhosts -nonmpi uname -a`

Reboot hosts	As root:
	`mpirun -m hostsfile -ppn 1 \`
	`-np numhosts -nonmpi reboot`

Run a command on all hosts	`mpirun -m hostsfile -ppn 1 \`
	`-np numhosts -nonmpi <command>`
	Examples:
	`mpirun -m hostsfile -ppn 1 \`
	`-np numhosts -nonmpi hostname`
	`mpirun -m hostsfile -ppn 1 \`
	`-np numhosts -nonmpi date`

Table F-3. Common Tasks and Commands Summary (Continued)

Function	Command

Copy a file to all hosts	Using bash:
	`$ for i in $( cat hostsfile )`
	`do`
	`scp <source> $i:<destination>`
	`done`

Summarize the fabric components	`ipathbug-helper -m hostsfile \` `> ipath-info-allhosts`

Show the status of host IB ports	`ipathbug-helper -m hostsfile \`
	`> ipath-info-allhosts`
	`mpirun -m hostsfile -ppn 1 \`
	`-np numhosts -nonmpi ipath_control -i`

Verify that the hosts see each other	`ipath_checkout --run=5 hostsfile`
Check MPI performance	`ipath_checkout --run=7 hostsfile`
Generate all hosts problem report information	`ipathbug-helper -m hostsfile \` `> ipath-info-allhosts`

Table Notes:

The " \ " indicates commands that are broken across multiple lines.

Summary and Descriptions of Useful Files

Useful files are summarized in Table F-4. Names in blue text are linked to a corresponding section that provides further details.

Table F-4. Useful Files

File Name	Function
`boardversion`	File that shows the version of the chip architecture
`status_str`	File that verifies that the InfiniPath software is loaded and functioning
`/var/log/messages`	Logfile to which various programs write messages. Tracks activity on your system
`version`	File that provides version information of installed software/drivers

Draft

boardversion

It is useful to keep track of the current version of the chip architecture. You can check the version by looking in this file:

```
/sys/class/infiniband/ipath0/device/boardversion
```

Example contents are:

```
ChipABI 2.0,InfiniPath_QLE7280,InfiniPath1 5.2,PCI 2,SW Compat 2
```

This information is useful for reporting problems to Technical Support.

NOTE:

This file returns information on which the form factor adapter is installed. The HTX low-profile form factor is referred to as the QHT7140. The PCIe half-height, short form factor is referred to as the QLE7140, QLE7240, or QLE7280.

status_str

Check the file `status_str` to verify that the InfiniPath software is loaded and functioning. The file is located here:

```
/sys/class/infiniband/ipath0/device/status_str
```

Table F-5 shows the possible contents of the file, with brief explanations of the entries.

Table F-5. `status_str` File Contents

File Contents	Description
`Initted`	The driver has loaded and successfully initialized the IBA6110 or IBA7220 ASIC.
`Present`	The IBA6110 or IBA7220 ASIC has been detected (but not initialized unless Initted is also present).
`IB_link_up`	The InfiniBand link has been configured and is in the active state; packets can be sent and received.
`IB_configured`	The InfiniBand link has been configured. It may or may not be up and usable.

Draft

Table F-5. `status_str` File Contents (Continued)

File Contents	Description
`NOIBcable`	Unable to detect link present. This problem can be caused by one of the following problems with the QHT7140, QLE7140, QLE7240, or QLE7280 adapters: ■ No cable is plugged into the adapter. ■ The adapter is connected to something other than another InfiniBand device, or the connector is not fully seated. ■ The switch to which the adapter is connected is down.
`Fatal_Hardware_Error`	Check the system log (default is `/var/log/messages`) for more information, then call Technical Support.

This same directory contains other files with information related to status. These files are summarized in Table F-6.

Table F-6. Status—Other Files

File Name	Contents
`lid`	InfiniBand LID. The address on the InfiniBand fabric, similar conceptually to an IP address for TCP/IP. "Local" refers to it being unique only within a single InfiniBand fabric.
`mlid`	The Multicast Local ID (MLID), for InfiniBand multicast. Used for InfiniPath ether broadcasts, since InfiniBand has no concept of broadcast.
`guid`	The GUID for the InfiniPath chip, it is equivalent to a MAC address.
`nguid`	The number of GUIDs that are used. If `nguids == 2` and two chips are discovered, the first chip is assigned the requested GUID (from `eeprom`, or `ipath_sma`), and the second chip is assigned that GUID+1.
`serial`	The serial number of the QHT7140, QLE7140, QLE7240, or QLE7280 adapter.
`unit`	A unique number for each card or chip in a system.
`status`	The numeric version of the `status_str` file, described in Table F-5.

Draft

version

You can check the version of the installed InfiniPath software by looking in:

```
/sys/class/infiniband/ipath0/device/driver/version
```

QLogic-built drivers have contents similar to:

```
$Id: QLogic OFED Release 1.4$ $Date: Fri Feb 27 16:14:31 PST 2009 $
```

Non-QLogic-built drivers (in this case `kernel.org`) have contents similar to:

```
$Id: QLogic kernel.org driver $
```

Summary of Configuration Files

Table F-7 contains descriptions of the configuration and configuration template files used by the InfiniPath and OpenFabrics software.

Table F-7. Configuration Files

Configuration File Name	Description
`/etc/infiniband/qlgc_vnic.cfg`	VirtualNIC configuration file. Create this file after running `ib_qlgc_vnic_query` to get the information you need. This file was named `/etc/infini-band/qlogic_vnic.cfg` or `/etc/sysconfig/ics_inic.cfg` in previous releases. See the sample file `qlgc_vnic.cfg.sample` (described below) to see how it should be set up.
`/etc/modprobe.conf`	Specifies options for modules when added or removed by the `modprobe` command. Also used for creating aliases. PAT write-combing option is set here. For Red Hat systems.
`/etc/modprobe.conf.local`	Specifies options for modules when added or removed by the `modprobe` command. Also used for creating aliases. PAT write-combing option is set here. For SLES systems.
`/etc/infiniband/openib.conf`	The primary configuration file for Infini-Path, OFED modules, and other modules and associated daemons. Automatically loads additional modules or changes IPoIB transport type.

Draft

Table F-7. Configuration Files (Continued)

/etc/sysconfig/infinipath	Contains settings, including the one that sets the ipath_mtrr script to run on reboot.
/etc/sysconfig/network/ifcfg-<NAME>	Network configuration file for network interfaces When used for VNIC configuration, <NAME> is in the form eiocX, where X is the device number. There will be one interface configuration file for each interface defined in /etc/infiniband/qlgc_vnic.cfg. For SLES systems.
/etc/sysconfig/network-scripts/ifcfg-<NAME>	Network configuration file for network interfaces When used for VNIC configuration, <NAME> is in the form eiocX, where X is the device number. There will be one interface configuration file for each interface defined in /etc/infiniband/qlgc_vnic.cfg. For Red Hat systems.
Sample and Template Files	**Description**
qlgc_vnic.cfg.sample	Sample VNIC config file. It can be found with the OFED documentation, or in the qlgc_vnictools subdirectory of the QLogicIB_Basic download. It is also installed in /etc/infiniband.
/usr/share/doc/initscripts-*/sysconfig.txt	File that explains many of the entries in the configuration files For Red hat systems

Draft

Notes

Draft

G Recommended Reading

Reference material for further reading is provided in this appendix.

References for MPI

The MPI Standard specification documents are located at:
http://www.mpi-forum.org/docs

The MPICH implementation of MPI and its documentation are located at:
http://www-unix.mcs.anl.gov/mpi/mpich/

The ROMIO distribution and its documentation are located at:
http://www.mcs.anl.gov/romio

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

Reference and Source for SLURM

The open-source resource manager designed for Linux clusters is located at:
http://www.llnl.gov/linux/slurm/

InfiniBand

The InfiniBand specification can be found at the InfiniBand Trade Association site:
http://www.infinibandta.org/

OpenFabrics

Open InfiniBand Alliance, located at:
http://www.openfabrics.org

Draft

Clusters

Gropp, William, Ewing Lusk, and Thomas Sterling, *Beowulf Cluster Computing with Linux*, Second Edition, 2003, MIT Press, ISBN 0-262-69292-9

Networking

The Internet Frequently Asked Questions (FAQ) archives contain an extensive Request for Command (RFC) section. Numerous documents on networking and configuration can be found at:
http://www.faqs.org/rfcs/index.html

Rocks

Extensive documentation on installing Rocks and custom Rolls can be found at:
http://www.rocksclusters.org/

Other Software Packages

Environment Modules is a popular package to maintain multiple concurrent versions of software packages and is available from:
http://modules.sourceforge.net/

Draft

Glossary

A glossary is provided below for technical terms used in the documentation. Italicized terms in the definitions are defined in the glossary. If you are viewing this document as a PDF file, the blue terms are linked to the corresponding definition.

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

Stands 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.

connected mode

IPoIB runs in either connected mode (IPOIB-CM) or unreliable datagram (IPoIB-UD) mode. Connected mode uses the Reliable Connected (RC) protocol. IPoIB in connected mode achieves higher bandwidth because the RC protocol supports a larger MTU (typically at least 4MB) than the UD protocol (limited to the InfiniBand MTU).

context sharing

A method that allows MPI node programs to share QLogic InfiniPath hardware resources (contexts). With context sharing, up to four node programs (in the same MPI job) can share each available context.

DAPL

Stands for Direct Access Provider Library. This is the reference implementation for RDMA transports. Consists of both kernel mode (kDAPL) and user mode (uDAPL) versions.

development node

Same as build node

DHCP

Stands for Dynamic Host Configuration Protocol, a communications protocol for allocating IP addresses. DHCP also provides other basic networking information, such as router addresses and name servers.

Draft

EE

Stands for End to End

EEC

Stands for End to End Context

fabric

The InfiniBand interconnect infrastructure, consisting of a set of Host Channel Adapters (HCAs) (and possibly Target Channel Adapters (TCAs)) connected by switches, such that each end node can directly reach all other nodes.

front end node

The machine or machines that launch jobs.

funneled thread model

Only the main (master) thread may execute MPI calls. In QLogic MPI, hybrid MPI/OpenMP applications are supported, provided that the MPI routines are called only by the master OpenMP thread.

GID

Stands for Global Identifier. Used for routing between different InfiniBand subnets.

GUID

Stands for Globally Unique Identifier for the QLogic chip. GUID is equivalent to an Ethernet MAC address.

head node

Same as front end node.

HCA

Stands 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 daughter cards 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

Stands 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. IPoIB runs in either connected mode (IPoIB-CM) or unreliable datagram mode (IPoIB-UD).

iSER

Stands for iSCSI Extensions for RDMA. An upper layer protocol.

kDAPL

Stands for kernel Direct Access Provider Library. kDAPL is the kernel mode version of the DAPL protocol.

Draft

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

Stands 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

Stands for Media Access Control Address. It is a unique identifier attached to most forms of networking equipment.

machines file

Same as mpihosts file

MADs

Stands 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

Stands 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

Stands for Multicast Local ID for InfiniBand multicast. This is the identifier that a member of a multicast group uses for addressing messages to other members of the group.

MPD

Stands for Multi-Purpose Daemon. An alternative to `mpirun` to launch MPI jobs, it provides support for MPICH. Developed at Argonne National laboratory.

MPI

Stands 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 can be run. Also referred to as node file, hosts file, or machines file.

Draft

MR

Stands for Memory Region

MTRR

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

MTU

Stands 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 each 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.

multihomed head node

A host that has multiple IP addresses, usually assigned to a different interface and part of a different network. In the normal case, each active interface has a separate and unique IP address and a unique host name.

node file

Same as hosts file

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

Stands for Open source Subnet Manager. Provides provides basic functionality for subnet discovery and activation.

PAT

Stands for Page Attribute Table. Controls how areas of memory are cached. Similar to MTRR, except that it can be specified on a per-page basis,

PCIe

Stands for PCI Express. Based on PCI concepts and standards, PCIe uses a faster serial connection mechanism.

PSM

PSM is QLogic's low-level user level Application Programming Interface (API). QLogic MPI, as well as numerous other high performance MPI implementations, have been ported to the PSM interface.

QP

Stands for Queue Pair

RC

Stands for Reliable Connected. A transport mode used by InfiniBand.

Draft

RDMA

Stands 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

Stands for Red Hat Package Manager. A tool for packaging, installing, and managing software for Linux distributions.

SDP

Stands for Sockets Direct Protocol. An InfiniBand-specific upper layer protocol. It defines a standard wire protocol to support stream sockets networking over InfiniBand.

SRP

Stands for SCSI RDMA Protocol. The implementation of this protocol is under development for utilizing block storage devices over an InfiniBand fabric.

SM

Stands for Subnet Manager. A subnet contains a master subnet manager that is responsible for network initialization (topology discovery), configuration, and maintenance. The SM discovers and configures all the reachable nodes in the InfiniBand fabric. It discovers them at switch startup, and continues monitoring changes in physical network connectivity and topology. It is responsible for assigning Local IDentifiers, called LIDs, to the visible nodes. It also handles multicast group setup. When the network contains multiple managed switches, they negotiate among themselves as to which one controls SM. The SM communicates with the Subnet Management Agents (SMAs) that exist on all nodes in a cluster.

SMA

Stands 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

Connects 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

Stands for Target Channel Adapter. A TCA is a channel adapter for I/O nodes, such as shared storage devices.

TCP

Stands for Transmission Control Protocol. One of the core protocols of the Internet protocol suite. TCP is a transport mechanism that ensures that data arrives complete and in order.

TID

Stands for Token ID. A method of identifying a memory region. Part of the QLogic hardware.

UD

Stands for Unreliable Datagram. A transport protocol used by InfiniBand.

uDAPL

Stands for user Direct Access Provider Library. uDAPL is the user space implementation of the DAPL protocol.

Draft

unmanaged switch

A switch that does not have an active Subnet Manager (SM)

Verbs

In the InfiniBand Specification, Verbs are abstract descriptions of the functions needed to configure, manage, and operate an HCA. The OpenFabrics Alliance has created a User Level Verbs API, which provides support for user level Upper Layer Protocols like MPI, Cluster File Systems, OpenSM and other user level utilities. The OpenFabrics User Verbs libraries are available in the QLogic OFED and standard OFED packages as well as in Linux distributions.

VNIC

Stands for Virtual Network Interface Controller (or Card). VNIC is a device driver for a Virtual I/O Controller (VIC) card. It provides a standard Ethernet interface on a host on an InfiniBand fabric.

Draft

Index

Draft

D

E

Draft

F

G

H

I

Draft

K

L

Draft

M

Draft

N

O

Draft

P

Q

R

S

Draft

T

U

V

W

X

Draft

Y

Draft

Notes

Draft

Draft

Draft

QLOGIC®
The Ultimate in Performance

Corporate Headquarters QLogic Corporation 26650 Aliso Viejo Parkway Aliso Viejo, CA 92656 949.389.6000 **www.qlogic.com**

Europe Headquarters QLogic (UK) LTD. Quatro House Lyon Way, Frimley Camberley Surrey, GU16 7ER UK +44 (0) 1276 804 670

© 2005-2009 QLogic Corporation. Specifications are subject to change without notice. All rights reserved worldwide. QLogic and the QLogic logo are registered trademarks of QLogic Corporation. QLA, SANsurfer, InfiniPath, and SilverStorm are trademarks or registered trademarks of QLogic Corporation. AMD Opteron is a trademark of Advanced Microdevices Inc. BladeCenter and IBM are registered trademarks of International Business Machines Corporation. DataDirect Networks is a trademark of DataDirect Networks, Inc. EMCORE is a trademark of EMCORE Corporation. HTX is a trademark of the HyperTransport Technology Consortium. IBM and BladeCenter are registered trademarks of International Business Machines Corporation. InfiniBand is a trademark and service mark of the InfiniBand Trade Association. Intel is a registered trademark of Intel Corporation. Linux is a registered trademark of Linus Torvalds. LSI Logic and Engenio are trademarks or registered trademarks of LSI Logic Corporation. Lustre is a registered trademark of Cluster File Systems, Inc. Mellanox is a registered trademark and ConnectX is a trademark of Mellanox Technologies, Inc. PathScale is a trademark of PathScale LLC. PCI Express is a registered trademark of PCI-SIG Corporation. Red Hat and Enterprise Linux are registered trademarks of Red Hat, Inc. Supermicro is a registered trademark of Super Micro Computer Inc. SUSE is a registered trademark of Novell Inc. Zarlink is a trademark of Zarlink Semiconductor Inc. All other brand and product names are trademarks or registered trademarks of their respective owners. Information supplied by QLogic Corporation is believed to be accurate and reliable. QLogic Corporation assumes no responsibility for any errors in this brochure. QLogic Corporation reserves the right, without notice, to make changes in product design or specifications.