InfiniBand Guide

extreme computing

REFERENCE
86 A2 42FD 03

extreme computing

InfiniBand Guide

Software

August 2010

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

REFERENCE
86 A2 42FD 03

Trademarks and Acknowledgements

Table of Contents

List of Figures

List of Tables

Preface

Scope and Objectives

This guide describes how to install, manage, and optimize InfiniBand networks.

Intended Readers

This guide is for Administrators of **bull extreme computing** systems.

Prerequisites

Important
The Software Release Bulletin contains the latest information for your delivery. This should be read first. Contact your support representative for more information.

Note   The Bull Support Web site may be consulted for product information, documentation, downloads, updates and service offers:
http://support.bull.com

Highlighting

- Commands entered by the user are in a frame in 'Courier' font, as shown below:

```
mkdir /var/lib/newdir
```

- System messages displayed on the screen are in 'Courier New' font between 2 dotted lines, as shown below.

```
Enter the number for the path :
```

- Values to be entered in by the user are in 'Courier New', for example:

  `COM1`

- Commands, files, directories and other items whose names are predefined by the system are in '**Bold**', as shown below:

  The **/etc/sysconfig/**dump file.

- The use of *Italics* identifies publications, chapters, sections, figures, and tables that are referenced.

- < > identifies parameters to be supplied by the user, for example:

  `<node_name>`

WARNING
A Warning notice indicates an action that could cause damage to a program, device, system, or data.

Chapter 1. InfiniBand Cluster Environment

The management of **InfiniBand** networks involves many different hardware and software aspects. There are a large number of commands available in the OFED software stack, and not all of them are described in this chapter. The Administrator is strongly encouraged to read the man pages of the **InfiniBand-diags**, **OpenSM** and **ibutils** packages to identify which command is the most appropriate for the issue at hand.

See The documentation provided by the switch manufacturer.

Important **All commands in this manual which start with the hash (#) sign must be carried out as root.**

1.1 InfiniBand Software Packages

See The Bull documentation included with your delivery, with particular reference to the *Software Release Bulletin*, for details of the InfiniBand packages provided.

The following packages will be installed on your cluster:

• **kernel-ib**	**InfiniBand** updates for **udev** and boot time kernel module loading. **RPM** kernel-ib-<version>.rpm
• **Kernel-ib-scripts**	A collection of kernel scripts that provide an interface for the management of **InfiniBand** services. **RPM** kernel-ib-scripts-<version>.rpm
• **Ibutils**	Network and diagnostic tools. **RPM** ibutils-<version>.rpm
• **Infiniband-diags**	Diagnostic programs and scripts for the **InfiniBand** network. **RPM** infiniband-diags<version>.rpm

- libibcm

  A user space library that handles the majority of the low level work required to open an RDMA connection between two machines.

  RPM list
  libibcm-<version>.rpm

- libibcommon

  Common utility functions for the OFA diagnostic and management tools.

  RPM list
  libibcommon-<version>.rpm

- libibmad

  Low layer functions for use by diagnostic and management programs. These include MAD, SA, SMP, and other basic functions.

  RPMs
  libibmad-<version>.rpm

  (Optional development suite:
  libibmad-devel-<version>.rpm
  libibmad-static<version>.rpm)

- libibumad

  User MAD library functions which sit on top of the user MAD modules in the kernel. These are used by the InfiniBand diagnostic and management tools, including OpenSM.

  RPMs
  libibumad-<version>.rpm

  (Optional development suite:
  libibumad-static-<version>.rpm
  libibumad-devel-<version>.rpm)

- libibverbs

  A library that allows user space processes to use InfiniBand verbs as described in the *InfiniBand Architecture Specification* (see http://www.InfiniBandta.org/). This includes direct hardware access for fast path operations.

  RPMs
  libibverbs-1.1.1-<version>.rpm

  (Optional development suite:
  libibverbs-devel-<version>.rpm
  libibverbs-devel-static-<version>.rpm)

- libibverbs-utils

  Useful libibverbs sample programs, such as ibv_devinfo which is used to display information about InfiniBand devices.

  RPM
  libibverbs-utils-<version>.rpm

• librdmacm	librdmacm provides a user space Communication Management API. **RPMs** librdmacm-<version>.rpm (Optional development suite: librdmacm-devel-<version>.rpm)
• librdmacm-utils	RDMA connection management library test utilities. **RPM** librdmacm-utils-<version>.rpm
• libsdp	libsdp is an LD_PRELOAD-able library that can be used to configure applications to use InfiniBand Sockets Direct Protocol (SDP) instead of TCP sockets, transparently and without the need for recompilation. **RPMs** libsdp-<version>.rpm (Optional development suite: libsdp-devel-<version>.rpm)
• libmlx4	libmlx4 provides a device-specific user space driver for Mellanox ConnectX HCAs for use with the libibverbs library. **RPM** libmlx4-<version>.rpm
• libmthca	libmthca provides a device-specific user space driver for Mellanox HCAs (MT23108 InfiniHost and MT25208 InfiniHost III Ex) for use with the libibverbs library. **RPMs** libmthca-<version>.rpm

OpenSM Management Packages

A Subnet Manager must be running on one of the fabric nodes so that all machines are included in the fabric, Bull provides the **OpenSM** Subnet Manager for this purpose. See *Chapter 2* for details on how to configure **OpenSM**.

• OpenSM	OpenIB project's Subnet Manager for InfiniBand networks. The subnet manager runs as a system daemon on one of the machines in the InfiniBand fabric to manage the fabric's routing state. This package also contains various tools for diagnosing and testing InfiniBand networks. These can be used from any machine and do not need to run on the machine running the OpenSM daemon.

	RPMs opensm-<version>.rpm (Optional development suite: opensm-devel-<version>.rpm)
• opensm-libs	Shared OpenSM libraries for InfiniBand user space access. **RPM** opensm-libs-<version>.rpm
• opensm-static	Static version of OpenSM libraries. **RPM** opensm-static-<version>.rpm
• dapl	Library providing access to the DAT 1.2 and 2.0 APIs **RPM** dapl-<version>.rpm (Optional development suite: dapl-devel-<version>.rpm)

Optional InfiniBand packages

The following packages provide some useful utilities and tools. These are installed manually, as required:

• ibs	A tool used to analyze InfiniBand networks. **RPM** ibs-<version>.rpm
• mstflint	Includes a burning firmware tool for Mellanox manufactured HCA cards. **RPM** mstflint-<version>.rpm
• mft	Mellanox Firmware Tools (MFT) package (includes flint tool). **RPM** mft-<version>.rpm
• mthca_fw_update	Mellanox tool for updating HCA firmware. **RPM** mthca_fw_update-<version>.rpm

• ibsw_fw_update	Tool for updating switch firmware. **RPM** ibsw_fw_update-<version>.rpm

Important The **mft** and **ibsw_fw_update** packages must be installed manually. Search for the latest version of these RPMs in the sub-directories of the /release directory on the Management Node and then install them on the node by using the command:
`yum localinstall mft* ibsw_fw_updatet`

• ofed-docs	**Open Fabrics Enterprise Distribution** documentation. This package includes documentation files for **IButils**, **ipath**, **IPoIB**, **Mthca**, **OpenSM**, **SRP**, and **SDP**. **RPM** ofed-docs-<version>.rpm
• ofed-scripts	**Open Fabrics Enterprise Distribution** script for the **OFED** installation. **RPM** ofed-scripts-<version>.rpm
• perftest	InfiniBand Performance tests (uverbs microbenchmarks). **RPM** perftest-<version>.rpm
• qperf	Measure socket and **RDMA** performance. **RPM** qperf-<version>.rpm
• sdpnetstat	netstat for **SDP**. **RPM** sdpnetstat-<version>.rpm
• dapl-utils	Test suites used to validate the **dapl** library **APIs** and their operation. **RPM** dapl-utils-<version>.rpm

If necessary, use the command below to install these packages manually:

```
# yum install <package_name>
```

Use the command below to update a package:

```
# yum update <package_name>
```

See The *Software Release Bulletin* for details of any RPMs which may need to be installed manually for your system.

1.2 Check Firmware Version for Switches and Adapters

Check and update the firmware versions for the InfiniBand adapters and switches.

See The chapter on *Troubleshooting InfiniBand Networks* for more details on checking and updating the firmware for InfiniBand adapters and switches.

1.3 Installation results

The cluster should be configured as follows, after the installation of the **InfiniBand** RPMs delivered by Bull.

1. The InfiniBand kernel subsystem modules are in:
   /lib/modules/`uname -r`/updates/kernel/drivers/InfiniBand/
2. The InfiniBand kernel **mlx4** driver is in:
   /lib/modules/`uname -r`/updates/kernel/drivers/net/mlx4/
3. The **openibd** script, used to load and unload the **InfiniBand** software stack, is in the **/etc/init.d/** directory.
4. The **/etc/InfiniBand** directory has been created. This contains an **info** file with the **kernel-ib** compilation set-up and an **openib_<kernel release>.conf** file. The **openib_<kernel release>.conf** file contains the list of modules that are loaded when the **openibd** script is launched.
5. The **90-ib.rules** file is in the **/etc/udev/rules.d/** directory. This creates the dynamic **/dev/InfiniBand** directory which is required for device naming.
6. The **/etc/modprobe.conf** file has been updated to include the following lines:
   a. **alias ib<n> ib_ipoib** (for each ib<n> interface - see Chapter 2).
   b. **alias net-pf-27 ib_sdp** (for SDP - see Chapter 2).
7. The **opensmd** daemon is in the **/etc/init.d/** directory.
8. The InfiniBand Man pages will be installed in the **/usr/share/man/** directory.
9. InfiniBand **sysfs** entries are created - see below.

1.4 InfiniBand sysfs Entries

For each InfiniBand device, the InfiniBand driver creates the following files in the **/sys/class/InfiniBand/<device name>** directory.

Example

```
ls /sys/class/InfiniBand/mthca0
```

```
board_id  device  fw_ver  hca_type  hw_rev  node_desc  node_guid
node_type  ports  subsystem  sys_image_guid  uevent
```

node_type	Node type (CA, switch or router)
node_guid	Node GUID
sys_image_guid	System image GUID
board_id	Device PSID

The **Mellanox HCA** driver also creates the files:

hw_rev	Hardware revision number
fw_ver	Firmware version
hca_type	HCA type: "MT23108", "MT25208 (MT23108 compat mode)", or "MT25208"

In addition, a ports sub-directory is created with one sub-directory for each port. For example, if mthca0 is a 2-port HCA, there will be two directories:

```
ls /sys/class/InfiniBand/mthca0/ports/*
```

```
/sys/class/InfiniBand/mthca0/ports/1:
cap_mask  counters  gids  lid  lid_mask_count  phys_state  pkeys  rate
sm_lid  sm_sl  state
/sys/class/InfiniBand/mthca0/ports/2:
cap_mask  counters  gids  lid  lid_mask_count  phys_state  pkeys  rate
sm_lid  sm_sl  state
```

Note A switch will have a single 0 sub-directory for switch port 0; no sub-directories are created for other switch ports.

In each port subdirectory, the following files are created:

cap_mask	Port capability mask
lid	Port LID
lid_mask_count	Port LID mask count
rate	Port data rate (active width * active speed)
sm_lid	Subnet manager LID for port's subnet
sm_sl	Subnet manager Service Level Subnet
state	Port state (DOWN, INIT, ARMED, ACTIVE or ACTIVE_DEFER)
phys_state	Port physical state (Sleep, Polling, LinkUp, etc)

A counters subdirectory is created which includes the following files.

VL15_dropped
excessive_buffer_overrun_errors
link_downed
link_error_recovery
local_link_integrity_errors
port_rcv_constraint_errors
port_rcv_data
port_rcv_errors
port_rcv_packets
port_rcv_remote_physical_errors
port_rcv_switch_relay_errors
port_xmit_constraint_errors
port_xmit_data
port_xmit_discards
port_xmit_packets
symbol_error

Each of these files contains the value from the corresponding Performance Management PortCounters attribute for the port.

See The *Troubleshooting InfiniBand Networks* chapter for details of the InfiniBand port counters.

Chapter 2. InfiniBand Network Management

2.1 Services and Settings - all InfiniBand Clusters

This section describes the **InfiniBand** services and settings which apply to all InfiniBand clusters.

2.1.1 The OpenIB Stack

The **openibd** service is a script that loads the appropriate InfiniBand drivers listed in the **openibd** configuration file. This configuration file is located in the **/etc/InfiniBand** directory, and its generic name is **openib_<kernel release>.conf**.

The example below shows the **openib** settings for a **2.6.18-128.el5.Bull.2** kernel. It sets the **ONBOOT** option to **yes** so that the **openibd** service starts automatically at boot time.

```
cat /etc/InfiniBand/openib_2.6.18-128.el5.Bull.2.conf
```

```
# Start HCA driver upon boot
ONBOOT=yes
# Load UCM module
UCM_LOAD=no
# Load RDMA_CM module
RDMA_CM_LOAD=yes
# Load RDMA_UCM module
RDMA_UCM_LOAD=yes
# Increase ib_mad thread priority
RENICE_IB_MAD=no
# Load MTHCA
MTHCA_LOAD=yes
# Load MLX4 modules
MLX4_LOAD=yes
# Load IPoIB
IPOIB_LOAD=yes
# Enable IPoIB Connected Mode
SET_IPOIB_CM=yes
# Load SDP module
SDP_LOAD=yes
# Load SRP module
SRP_LOAD=no
# Load SRP Target module
SRPT_LOAD=no
# Load ISER module
ISER_LOAD=no
```

Details of the **OpenIB** module components are shown below:

Core modules

ib_addr	InfiniBand address translation
ib_core	Core kernel InfiniBand API

Hardware support:

mlx4_ib	**Mellanox ConnectX HCA** InfiniBand driver
mlx4_core	**Mellanox ConnectX HCA** low-level driver
ib_mthca	**Mellanox InfiniBand HCA** low-level driver

qlgc_vnic	QLogic virtual NIC (VNIC) driver
mlx4_en	Mellanox ConnectX HCA Ethernet driver

IP over IB modules:

ib_ipoib	IP-over-InfiniBand net driver
ipoib_helper	Container for ipoib neighbour destructor

Subnet manager related modules:

ib_mad	IB MAD API
ib_sa	Subnet administration query support
ib_umad	User space MAD packet access.

MPI related modules:

ib_uverbs	User space verbs access.

Connection managers:

ib_ucm	User space connection manager access
ib_cm	Connection manager
iw_cm	iWARP connection manager
rdma_cm	Generic RDMA connection manager
rdma_ucm	RDMA user space connection manager

Socket direct protocol:

ib_sdp	Socket Direct Protocol (SDP)

Storage:

ib_iser	iSER (iSCSI Extensions for RDMA) data mover
ib_srp	InfiniBand SCSI RDMA Protocol initiator
ib_srpt	InfiniBand SCSI RDMA Protocol target
iscsi_tcp	iSCSI/TCP data-path
libiscsi	iSCSI library functions
scsi_transport_iscsi	iSCSI transport interface

Use the following command to start the openibd service

```
# service openibd start
```

```
Loading HCA driver and Access Layer:                         [  OK  ]
Setting up InfiniBand network interfaces:
Bringing up interface ib0:                                   [  OK  ]
Bringing up interface ib1:                                   [  OK  ]
Setting up service network . . .                             [  done  ]
```

2.1.2 OpenSM

OpenSM is an InfiniBand compliant subnet manager and a subnet administration (SA) tool that runs on top of the OpenIB stack and initializes the InfiniBand hardware. There must be at least one subnet manager for each InfiniBand subnet.

The opensmd service is used to launch OpenSM using the command below.

```
# service opensmd start
```

```
Starting IB Subnet Manager.                                    [  OK  ]
```

Notes

- The subnet manager should not be started on all nodes: the number of subnet managers and their location depends on the cluster architecture and the choices made by the Administrator.
- The **OpenSM** service must be started on **InfiniBand** networks that do not include InfiniBand managed switches.

  InfiniBand managed switches include a subnet manager in their software stack. However, it is strongly recommended that the Bull **OpenSM** version, provided by the Bull InfiniBand packages, is used on these networks in order to ensure Bull technical support.

2.1.2.1 OpenSM Routing Algorithms

It is essential that the best routing algorithm is set for the **InfiniBand** network, otherwise there will be an impact on performance.

Because **InfiniBand** is a statically routed network, routing tables must be set for all **ASICs** within the fabric. **OpenSM** manages these routing tables using the routing algorithms that are available.

mportant **Depending on the topology of the physical network, some of these algorithms may work while others may not. Performance varies considerably according to the algorithm. The algorithm, its options, and its settings, should selected carefully.**

OpenSM supports the following routing algorithms:

Min-hop

Based on the minimum hops to each node where the path length is optimized.

Up/Down

Also based on the minimum hops to each node, but it is constrained to ranking rules. This algorithm should be chosen if the subnet is not a pure **Fat Tree** one, and deadlock may occur due to a loop in the subnet.

Fat-tree

This algorithm optimizes routing for congestion-free 'shift' communication patterns. It should be chosen if a subnet is a symmetrical fat tree of various types, not just K-ary-N-Trees: non-constant K, not fully staffed, any constant bisection bandwidth ratio. Similar to up/down, Fat-tree routing is constrained to ranking rules.

Lash

This algorithm uses the **InfiniBand** virtual layers to provide deadlock-free shortest-path routing while also distributing the paths between layers. Lash is an alternative deadlock-free topology-agnostic routing algorithm to the non-minimal up/down algorithm avoiding the use of a potentially congested root node.

Dor

This algorithm is based on the min-hop algorithm, but avoids port equalization except for redundant links between the same two switches. This provides deadlock free routes for hypercubes when the fabric is cabled as a hypercube and for meshes when cabled as a mesh.

OpenSM also supports a file method which can load routes from a table. This method implies that the hardware topology is 100% stable with no risk of hardware failure. It is mainly used for routing simulation purposes.

Note For optimal performance Bull recommends the Fat Tree routing algorithm. If the physical network is not Fat Tree then **Up/Down** should be used.

2.1.2.2 Configuring OpenSM

OpenSM can be configured either by using the **OpenSM** configuration file (**/etc/opensm/opensm.conf** by default) or by using the command line.
Command line settings take precedence over the options listed in the **OpenSM** configuration file. The default **opensm.conf** file only provides the mandatory options required by **OpenSM**. The complete configuration file is generated by using the command below:

```
opensm -c /etc/opensm/opensm.conf
```

This command dumps the current options and creates a complete **opensm.conf** configuration file in the **/etc/opensm/** directory. This configuration file is loaded by default, from that point on, whenever OpenSM is invoked.

Note If specific **OpenSM** related environment variables are exported, their values will be assigned to the corresponding options in the configuration file generated.

From version 3.3.0 onwards, the **OpenSM** options are dynamic. Therefore, it is not necessary to restart **OpenSM** for the new options to be taken into account. New options can be added to the **OpenSM** configuration file when **OpenSM** is running, and are read automatically without the need to restart the **OpenSM** daemon.

OpenSM Configuration file options and Command Line Settings

Both **OpenSM** configuration file options and command line settings are listed in this section and are separated by the / (slash) character: *config file option / command line setting*

daemon / -B
Run in daemon mode - **OpenSM** will run in the background.

guid / -g <GUID in hex>
This option specifies the local port **GUID** value that **OpenSM** should bind to. OpenSM may be only bound to a single port at a time. If **-g** is not specified, **OpenSM** tries to use the default port (first active port on the first adapter card found).

routing_engine / -R <engine name>
This option chooses the routing engine(s) to use instead of the default **min-hop** algorithm. Multiple routing engines can be specified, separated by commas, in the order that they should be tried. If an algorithm fails the next in line will be used. The supported engines are **minhop**, **updn**, **file**, ftree, **lash** and **dor**.

sm_priority / -p <priority>
This option specifies the subnet manager priority. This applies to handover cases, in which the master is chosen by priority and GUID. The priority ranges from 0 (default and lowest priority) to 15 (highest priority).

sweep_interval / -s <interval>
This option specifies the number of seconds between subnet sweeps. Specifying -s 0 disables sweeping. Without the -s flag OpenSM defaults to a sweep interval of 10 seconds.

log_file / -f <log file path>
This option defines the file name for the log file produced by OpenSM. Its default file name is /var/log/opensm.log. The -f flag is used to redirect the logs to stdout.

log_flags / -D <flags>
This option sets the verbosity level for the log. A flags field must follow the -D option. Setting or clearing a bit in the flags field enables or disables a specific log level as below:

```
BIT    LOG LEVEL ENABLED
----   -----------------
0x01 - ERROR (error messages)
0x02 - INFO (basic messages, low volume)
0x04 - VERBOSE (interesting stuff, moderate volume)
0x08 - DEBUG (diagnostic, high volume)
0x10 - FUNCS (function entry/exit, very high volume)
0x20 - FRAMES (dumps all SMP and GMP frames)
0x40 - ROUTING (dump FDB routing information)
0x80 - currently unused.
```

Without the -D option OpenSM defaults to ERROR + INFO (0x3). Specifying -D 0 disables all messages. Specifying -D 0xFF enables all messages (see -V). High verbosity levels may require that the transaction timeout is increased using the -t option.

log_max_size / –log-limit <size in MB>
Defines the size of the log file in MBs. When specified, the log file will be truncated upon reaching this limit.

console / –console <off/local/remote/socket>
This option brings up the OpenSM console (default off).

console_port / -console-port <port>
Specify an alternate telnet port for the socket console (default 10000).

sminfo_polling_timeout / (config file only)
Timeout in [msec] between two polls of the active master subnet manager.

polling_retry_number / (config file only)
Number of polls attempted until the remote SM is assumed to be dead.

sweep_on_trap / (config file only)
If TRUE every trap will cause a heavy sweep.

Note Successive identical traps (>10) are suppressed.

force_link_speed / (config file only)
Force PortInfo:LinkSpeedEnabled on switch ports. If set to 0, do not modify the PortInfo:LinkSpeedEnabled on the switch port. Otherwise, use the specified value for the PortInfo:LinkSpeedEnabled on the switch port.

The values permitted are:

```
1: 2.5 Gbps
3: 2.5 or 5.0 Gbps
5: 2.5 or 10.0 Gbps
7: 2.5 or 5.0 or 10.0 Gbps
2,4,6,8-14 Reserved
Default 15: set to PortInfo:LinkSpeedSupported
```

Fat-Tree Specific Options

root_guid_file / -a <root guid file>
Set the root nodes for the **Up/Down** or **Fat-Tree** routing algorithm to the **GUIDs** provided in the given file (one to a line). If the root **GUID** file is provided, the topology does not have to be pure Fat-tree, and it must comply with the following rules:

a. Tree rank must be between two and eight (inclusively)

b. All the Compute Nodes have to be at the same tree level (rank).

Note — Non-compute Node **CAs** are allowed to be at different tree level (rank)s.

cn_guid_file / -u <cn guid file>
Sets the Compute node for the Fat-Tree routing algorithm to the **GUIDs** listed in the file (one to a line).

io_guid_file / -G <io guid file>
Sets the I/O nodes for the Fat-Tree routing algorithm to the GUIDs listed in the file (one to a line). I/O nodes are allowed to use **max_reverse_hops** switches to improve connectivity.

max_reverse_hops / -H <max reverse hops>
Sets the maximum number of reverse hops an I/O node is allowed to make. A reverse hop is the use of a switch the wrong way around.

Routing between non Compute Nodes

The use of the **cn_guid_file** option allows non Compute Nodes to be at different levels in the fat tree. However, it is not guaranteed that the Fat- Tree algorithm will route between two non Compute Nodes.

In the diagram below, N1, N2 and N3 are non Compute Nodes. Although all the Compute Nodes have routes to and from them, there is not necessarily a route between N1, N2 and N3, as such routes require the use of at least one Switch the wrong way around, for example, data routed down instead of up.

```
  Spine1   Spine2    Spine 3
   / \     /  |  \    /   \
  /   \   /   |   \  /     \
 N1  Switch   N2  Switch    N3
      /|\          /|\
     / | \        / | \
    Going down to compute nodes
```

To solve this problem, a list of non Compute Nodes can be specified by using either the **-G** or **–io_guid_file** options. Theses nodes are allowed to use switches the wrong way around a specific number of times, as specified by the **-H** or **–max_reverse_hops** options.

With the appropriate **max_reverse_hops** and **io_guid_file** values, full connectivity can be assured for Fat Tree networks. In the diagram above, with a **max_reverse_hop** of 1, routes will be instantiated between **N1<->N2** and N2<->N3. With a **max_reverse_hops** value of 2, **N1**, **N2** and **N3** will all have routes between them.

Important Use the max_reverse_hops and io_guid_file options with extreme care. These options should never be used to connect nodes with high bandwidth traffic between them! They should only be used to allow connectivity for HA or similar purposes.
Also, having routes the other way around can in theory cause credit loops.

2.1.3 High Availability for OpenSM

The **InfiniBand** specification states that a subnet manager should be fault tolerant. Consequently, any **IBTA** compliant subnet manager should provide some kind of High Availability functionality.

2.1.3.1 Mode of operation

High Availability is built-in for **OpenSM** subnet managers, as long as there are two or more subnet managers within the same interconnect network, fault tolerance is achieved without the need for any additional configuration.

When two or more subnet managers are started in the same subnet, some kind of negotiation takes place between the subnet managers in order to determine which one will be elected as the master, the remaining subnet managers will be slaves. In the case of a failure of the master subnet manager, one of the slaves is elected as master. The state of the master subnet manager is monitored by the slave subnet managers through polling. Two options are included in the **OpenSM** configuration file to configure polling:

a. **sminfo_polling_timeout**: Timeout in msec between two polls of active master subnet manager.

b. **polling_retry_number**: Number of failing polls until the remote SM is assumed to be dead.

These options do not need to be set, but are modifable according to the requirements of High Availability.

Note This mechanism can use the **InfiniBand** network alone, i.e. it does not require a functional Ethernet network.

2.1.3.2 Architecture

Great care should be taken when choosing which nodes should be used to host the **OpenSM** subnet managers. The following points should be kept in mind:

- A high number of subnet managers improve fault tolerance.
- Subnet managers should be spread across the network to minimise the impact of hardware failures. For example, if all the subnet managers resided within the same leaf, a failure within this leaf would impact all the subnet managers.

- For performance reasons, it is not recommended to install OpenSM on Compute Nodes and I/O nodes (Lustre OSS/MDS, NFS servers, etc).
- OpenSM memory requirements are low: At most 300 MBs of RAM are required. However, depending on the debugging level configured, large log files can be generated on the nodes.
- A fast InfiniBand adapter is not required in order to run OpenSM. The related traffic is low.
- To ensure a consistent routing performance, the OpenSM configuration file should be identical on all nodes apart from the GUID field, which refers to the adapter each node.

2.1.3.3 Synchronising the OpenSM configuration files

The OpenSM configuration file is determined by the configuration of the cluster, and it is necessary to deploy it onto all the OpenSM hosts within the cluster, using one of the standard methords, such as NFS, NIS, rsync, etc.

Apart from the OpenSM configuration file itself, the other files that should be kept synchronised are those that correspond to the following entries in the OpenSM configuration file:

- partition_config_file
- port_prof_ignore_file
- root_guid_file
- cn_guid_file
- io_guid_file
- ids_guid_file
- guid_routing_order_file
- qos_policy_file
- prefix_routes_file

In conventional OpenSM setups, all these configuration files should be located in the /etc/opensm directory.

OpenSM Configuration File Location

To ensure the best routing options, and to benefit from QoS, the opensm.conf file must be reachable by each OpenSM subnet manager that is running, regardless of whether it is a master or a slave. If this is not the case, the routing scheme (or QoS) would change when an unconfigured OpenSM subnet manager becomes the master for the interconnect.

2.1.4 Voltaire Embedded subnet managers

Embedded subnet managers reside on the controller modules sold with managed Voltaire switches. In order to ensure High Availability, at least two managed Voltaire switches should be used. Alternatively, a managed switch with two controllers can also provide fault tolerance, although a complete switch failure could affect both embedded controllers, resulting in a subnet manager outage. It is therefore recommended to use two managed subnet managers instead of a single managed switch with two controllers.

2.2 Module Settings

This section summarises the main modules parameters dedicated to applications (QoS, ULP, etc.).

2.2.1 QoS

Activate Quality of Service within the **mlx4_core** driver (HCA Adapter) by adding the line below to the **/etc/modprobe.conf** file

```
# echo options mlx4_core enable_qos=1 >> /etc/modprobe.conf
```

2.2.2 IPoIB

For each IB port on the node, there should be one entry in the **/etc/modprobe.conf** file. Each entry describes the IP interface name and the corresponding kernel module to use. Example for 2 interfaces named ib0 and ib1:

```
cat /etc/modprobe.conf | grep ipoib
```

```
alias ib0 ib_ipoib
alias ib1 ib_ipoib
```

To set **IP over InfiniBand** parameters, add the following line to the **/etc/modpobe.conf** file:

```
options ib_ipoib parameter=<value>
```

```
modinfo ib_ipoib | awk -F 'parm:' '{print $2}'
```

```
  max_nonsrq_conn_qp:      Max number of connected-mode QPs per interface
(applied only if shared receive queue is not available) (int)
  set_nonsrq:              set to dictate working in none SRQ mode,
otherwise act according to device capabilities (int)
  mcast_debug_level:       Enable multicast debug tracing if > 0 (int)
  send_queue_size:         Number of descriptors in send queue (int)
  recv_queue_size:         Number of descriptors in receive queue (int)
  debug_level:             Enable debug tracing if > 0 (int)
```

2.2.3 Ethernet Over IB

A **ConnectX** InfiniBand adapter can be used in Ethernet mode. It requires **mlx4_en** modules which handle Ethernet specific functions and which plug into the **netdev** mid-layer. As a result of the high bandwidth possible with this adapter, changes should be made to the default settings for the network.

The following options should be added to the **/etc/sysctl.conf** file:

```
## MLX4_EN tuning parameters ##
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_sack = 0
net.core.netdev_max_backlog = 250000
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.core.rmem_default = 16777216
net.core.wmem_default = 16777216
net.core.optmem_max = 16777216
net.ipv4.tcp_mem = 16777216 16777216 16777216
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
## END MLX4_EN ##
```

After adding these parameters, run the following command:

```
# sysctl -a
```

2.2.4 SDP

SDP means **Socket Direct Protocol**. This line should be added to the **modprobe.conf** file to autoload **ib_sdp** when **SDP** is used:

```
cat /etc/modprobe.conf | grep sdp
```

```
alias net-pf-27 ib_sdp
```

Add the following line to the **/etc/modpobe.conf** file to set the **ib_sdp** parameters:

```
options ib_sdp parameter=<value>
```

The parameters available are listed below:

```
  rcvbuf_initial_size:      Receive buffer initial size in bytes. (int)
  rcvbuf_scale:             Receive buffer size scale factor. (int)
  top_mem_usage:            Top system wide sdp memory usage for recv (in MB).
(int)
  max_large_sockets:        Max number of large sockets (32k buffers). (int)
  sdp_keepalive_probes_sent:Total number of keepalive probes sent. (uint)
  debug_level:              Enable debug tracing if > 0. (int)
  data_debug_level:         Enable data path debug tracing if > 0. (int)
  send_poll_hit:            How many times send poll helped. (int)
  send_poll_miss:           How many times send poll missed. (int)
  recv_poll_hit:            How many times recv poll helped. (int)
  recv_poll_miss:           How many times recv poll missed. (int)
  send_poll:                How many times to poll send. (int)
  recv_poll:                How many times to poll recv. (int)
  send_poll_thresh:         Send message size thresh hold over which to start
polling. (int)
  sdp_keepalive_time:       Default idle time in seconds before keepalive probe
sent. (uint)
  sdp_zcopy_thresh:         Zero copy send threshold; 0=0ff. (int)
```

2.2.5 MTHCA

mthca is the low level driver implementation for some **Mellanox HCAs** (Infinihost:). To set the **mthca** parameters, add the following line to the **/etc/modpobe.conf** file:

```
options ib_mthca parameter=<value>
```

The parameters available are listed below:

```
  catas_reset_disable:     disable reset on catastrophic event if nonzero
(int)
  fw_cmd_doorbell:         post FW commands through doorbell page if nonzero
(and supported by FW) (int)
  debug_level:             Enable debug tracing if > 0 (int)
  msi_x:                   attempt to use MSI-X if nonzero (int)
  msi:                      attempt to use MSI if nonzero (deprecated, use
MSI-X instead) (int)
  tune_pci:                 increase PCI burst from the default set by BIOS if
nonzero (int)
  num_qp:                   maximum number of QPs per HCA (int)
  rdb_per_qp:               number of RDB buffers per QP (int)
  num_cq:                   maximum number of CQs per HCA (int)
  num_mcg:                  maximum number of multicast groups per HCA (int)
  num_mpt:                  maximum number of memory protection table entries
per HCA (int)
  num_mtt:                  maximum number of memory translation table
segments per HCA (int)
  num_udav:                 maximum number of UD address vectors per HCA (int)
  fmr_reserved_mtts:        number of memory translation table segments
reserved for FMR (int)
```

2.2.6 ConnectX MLX4

mlx4 modules are dedicated to **Mellanox ConnectX InfiniBand HCAs**. The **mlx4_core** module provides low-level functions, such as device initialization and firmware command processing. To set the **mlx4** parameters, add the following line to the **/etc/modpobe.conf** file:

```
options mlx4_core parameter=<value>
```

The parameters available are listed below:

```
  set_4k_mtu:               attempt to set 4K MTU to all ConnectX ports (int)
  debug_level:              Enable debug tracing if > 0 (int)
  block_loopback:           Block multicast loopback packets if > 0 (int)
  msi_x:                    attempt to use MSI-X if nonzero (int)
  log_num_qp:               log maximum number of QPs per HCA (int)
  log_num_srq:              log maximum number of SRQs per HCA (int)
  log_rdmarc_per_qp:        log number of RDMARC buffers per QP (int)
  log_num_cq:               log maximum number of CQs per HCA (int)
  log_num_mcg:              log maximum number of multicast groups per HCA (int)
  log_num_mpt:              log maximum number of memory protection table
entries per HCA (int)
  log_num_mtt:              log maximum number of memory translation table
segments per HCA (int)
  log_mtts_per_seg:         Log2 number of MTT entries per segment (1-5) (int)
  log_num_mac:              Log 2 Max number of MACs per ETH port (1-7) (int)
  log_num_vlan:             Log 2 Max number of VLANs per ETH port (0-7) (int)
  use_prio:                 Enable steering by VLAN priority on ETH ports (0/1,
default 0) (bool)
  enable_qos:               Enable Quality of Service support in the HCA
(default: off) (bool)
  internal_err_reset:       Reset device on internal errors if non-zero (default
1) (int)
```

2.3 Automated Boot Parameters

For automatic boot parameters you should first check your **openib_<kernel release>.conf** file, see section 2.3.1:
After that, set the corresponding run-level attributes, as below:

2.3.1 Openibd Service

By default, the service is automatically started when the node boots up (run levels 3 and 5):

```
chkconfig --list openibd
```

```
openibd          0:off    1:off    2:on     3:on     4:on     5:on     6:off
```

Use the command below to start the **openibd** service automatically:

```
# chkconfig openibd on
```

2.3.2 opensmd Service

By default, the service is automatically started when the node boots up (run levels 3 and 5):

```
chkconfig --list opensmd
```

```
opensmd          0:off  1:off  2:off  3:off  4:off   5:off   6:off
```

Use the command below to start the **opensmd** service automatically:

```
# chkconfig opensmd on
chkconfig --list opensmd
```

```
opensmd          0:off1:off2:on  3:on  4:on  5:on  6:off
```

If the subnet manager has started and is set up properly, a master subnet manager will be discovered:

```
sminfo
```

```
sminfo: sm lid 1 sm guid 0x2c903000262db, activity count 390772
priority 0 state 3 SMINFO_MASTER
```

2.3.3 IPoIB interface

As for any IP interface the default IP configuration of an **IPoIB** interface is set through a file in the **/etc/sysconfig/network-script/** directory.

To enable the **IPoIB** interface boot startup, add an entry called **ifcfg-<ib_interface>** to this directory. These files can are generated by using either the **config_ip** or **config_ipoib** command (the command available depends on the system).

Below is an example for the ib0 interface:

```
cat /etc/sysconfig/network-scripts/ifcfg-ib0
```

```
DEVICE=ib0
BOOTPROTO=static
IPADDR=10.12.0.1
NETMASK=255.255.0.0
ONBOOT=yes
```

If needed, you can trace or debug the **IPoIB** protocol using the following procedure:

1. Run the commands below:

```
# mount -t debugfs none /sys/kernel/debug
# cat /sys/kernel/debug/ipoib/ib0_path
```

Example Output:

```
GID: fe80:0:0:0:2:c903:2:5e83
  complete:    yes
  DLID:     0x0025
  SL:            0
  rate:         20 Gb/sec
GID: fe80:0:0:0:2:c903:2:5ea7
  complete:    yes
  DLID:     0x0023
  SL:            0
  rate:         20 Gb/sec
```

2. Run the command below:

```
# ip neigh show
```

Example Output:

```
10.11.0.8 dev eth0 lladdr 00:16:17:c3:9e:7c REACHABLE
10.11.0.119 dev eth0  FAILED
10.11.0.47 dev eth0 lladdr 00:15:17:70:51:68 REACHABLE
10.11.0.117 dev eth0 lladdr 00:30:48:c3:f7:02 STALE
10.11.0.6 dev eth0 lladdr 00:30:48:7f:2c:fa REACHABLE
10.11.0.211 dev eth0 lladdr 00:1b:54:1d:13:c0 REACHABLE
10.12.0.21 dev ib0 lladdr
80:00:00:48:fe:80:00:00:00:00:00:00:00:02:c9:03:00:02:5f:7b STALE
```

3. Run the command below:

```
# cat /sys/kernel/debug/ipoib/ib0_mcg
```

Example Output:

```
GID: ff12:401b:ffff:0:0:0:0:1
  created: 4734018661
  queuelen:        0
  complete:      yes
  send_only:      no
GID: ff12:401b:ffff:0:0:0:0:fb
  created: 4734018661
  queuelen:        0
  complete:      yes
  send_only:      no
```

2.4 Specific InfiniBand Settings

This section lists the settings for the following InfiniBand services.

- Quality of Service
- IP over InfiniBand
- Ethernet over InfiniBand
- Socket Direct Protocol

2.4.1 QoS

QoS controls the performance attributes of the network flow managed, within the bounds specified by the Administrator. QoS can attribute priority flow bandwith, or latency, to specific applications, such as MPI, IPoIB etc.

The main steps for the activatation of QoS are described, followed by more detailed configuration information, with examples.

2.4.1.1 Activating QoS

The following steps should be carried out.

1. Activate QoS within the **mlx4_core** driver by adding the line below to **/etc/modprobe.conf** file:

```
# echo options mlx4_core enable_qos=1 >> /etc/modprobe.conf
```

2. Configure the service levels in the **qos-policy** file:

```
cat /etc/opensm/qos-policy.conf
```

```
qos-levels
  qos-level
     name: DEFAULT
     sl: 0
  end-qos-level
  qos-level
     name: MPI
     sl: 1        #sl 1 for MPI stream
  end-qos-level
  qos-level
     name: Lustre
     sl: 2      #sl 2 for Lustre stream
  end-qos-level
end-qos-levels
qos-ulps
  default: 0
  ipoib: 3        #sl 3 for ipoib stream
end-qos-ulps
```

3. Set **qos** to **TRUE** in the **opensm.conf** file:

```
cat /etc/opensm/opensm.conf
```

```
[..]
#
# QoS OPTIONS
```

```
#
# Enable QoS setup
qos TRUE
```

4. Create a correspondence between SL and VL in the opensm.conf file:

```
cat /etc/opensm/opensm.conf
```

```
 [..]
 # QoS policy file to be used
 qos_policy_file /etc/opensm/qos-policy.conf

 # QoS default options
 qos_max_vls 15
 qos_high_limit 0
 qos_vlarb_high
0:4,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0,10:0,11:0,12:0,13:0,14:0
 qos_vlarb_low 0:0,1:64,2:128,3:64,1:64
 qos_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7

 # QoS CA options
 qos_ca_max_vls 15
 qos_ca_high_limit 0
 qos_ca_vlarb_high 0:0,1:0,2:0,3:0
 qos_ca_vlarb_low 0:1,1:64,2:128,3:64,1:64
 qos_ca_sl2vl 0,1,2,3,4,6,7,8,9,10,11,12,13,14,7,5

 # QoS Switch Port 0 options
 qos_sw0_max_vls 15
 qos_sw0_high_limit 0
 qos_sw0_vlarb_high
0:4,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0,10:0,11:0,12:0,13:0,14:0
 qos_sw0_vlarb_low 0:0,1:64,2:128,3:64,1:64
 qos_sw0_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7

 # QoS Switch external ports options
 qos_swe_max_vls 15
 qos_swe_high_limit 0
 qos_swe_vlarb_high
0:4,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0,10:0,11:0,12:0,13:0,14:0
 qos_swe_vlarb_low 0:0,1:64,2:128,3:64,1:64
 qos_swe_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7

 # QoS Router ports options
 qos_rtr_max_vls 15
 qos_rtr_high_limit 0
 qos_rtr_vlarb_high
0:4,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0,10:0,11:0,12:0,13:0,14:0
 qos_rtr_vlarb_low 0:0,1:64,2:128,3:64,1:64
 qos_rtr_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7
```

5. Restart the opensmd service:

```
# service opensmd restart
```

```
Stopping IB Subnet Manager..-.                              [  OK  ]
Starting IB Subnet Manager.                                 [  OK  ]
```

6. View QoS credits changes:

```
smpquery vlarb 1
```

```
# VLArbitration tables: Lid 1 port 0 LowCap 8 HighCap 8
```

```
 # Low priority VL Arbitration Table:
 VL    : |0x0 |0x1 |0x2 |0x3 |0x1 |0x0 |0x0 |0x0 |
 WEIGHT: |0x1 |0x40|0x80|0x40|0x40|0x0 |0x0 |0x0 |
 # High priority VL Arbitration Table:
 VL    : |0x0 |0x1 |0x2 |0x3 |0x0 |0x0 |0x0 |0x0 |
 WEIGHT: |0x0 |0x0 |0x0 |0x0 |0x0 |0x0 |0x0 |0x0 |
Additionnal informations are under
```

2.4.1.2 QoS policy file

The QoS policy file is used to specify the service level attributed, or differentiate flow management for ULPs settings, GUID origin or destination, etc. The policy file also specifies QoS partition management, and many other possibilities, as described below:

The QoS policy file is divided into 3 sub sections:

1. Port Group: a set of CAs, Routers or Switches that share the same settings.

   A Port Group might be a partition defined by the partition manager policy, a list of GUIDs, or a list of port names based on the port NodeDescription.

2. QoS-Levels Definition: This section defines the different sets of parameters for QoS that may be mapped to by a client. Each set includes SL and the following options: Max MTU, Max Rate and Packet Lifetime.

3. Matching Rules: A list of rules that match an incoming PathRecord/MultiPathRecord (PR/MPR) request to a QoS-Level.

   The rules are processed in order until a match is found, and this is then applied. Each rule is built out of a set of match expressions, which should all match for the rule to be applied. The matching expressions are defined for the following fields:

   a. SRC and DST to lists of port groups
   b. Service-ID to a list of Service-ID values or ranges
   c. QoS-Class to a list of QoS-Class values or ranges

QoS policy.conf file example

All sections of the policy file are optional except for the default QoS level, which is mandatory.

The following example shows all the possible options and keywords in the policy file and their syntax:

```
   # See the comments in the following example.
   # They explain different keywords and their meaning.
   #
   port-groups

       port-group # using port GUIDs
           name: Storage
           # "use" is just a description that is used for logging
           #  Other than that, it is just a comment
           use: SRP Targets
           port-guid: 0x10000000000001, 0x10000000000005-
0x1000000000FFFA
           port-guid: 0x1000000000FFFF
       end-port-group

       port-group
           name: Virtual Servers
           # The syntax of the port name is as follows:
```

```
        #   "node_description/Pnum".
        # node_description is compared to the NodeDescription of
        # the node, and "Pnum" is a port number on that node.
        port-name: vs1 HCA-1/P1, vs2 HCA-1/P1
    end-port-group

    # using partitions defined in the partition policy
    port-group
        name: Partitions
        partition: Part1
        pkey: 0x1234
    end-port-group

    # using node types: CA, ROUTER, SWITCH, SELF (for node that
    # runs SM)or ALL (for all the nodes in the subnet)
    port-group
        name: CAs and SM
        node-type: CA, SELF
    end-port-group

end-port-groups
qos-setup
    # This section of the policy file describes how to set up SL2VL
    # and VL. Arbitration tables on various nodes in the fabric.
    # However, this is not supported in OpenSM currently - the
    # section is parsed and ignored. SL2VL and VLArb tables should
    # be  configured in the OpenSM options file (by default -
    # /usr/local/etc/opensm/opensm.conf)
end-qos-setup

qos-levels

    # Having a QoS Level named "DEFAULT" is a must - it is applied
    # to PR/MPR requests that do not match any of the matching
    #rules.
    qos-level
        name: DEFAULT
        use: default QoS Level
        sl: 0
    end-qos-level

    # the whole set: SL, MTU-Limit, Rate-Limit, PKey, Packet
    #Lifetime
    qos-level
        name: WholeSet
        sl: 1
        mtu-limit: 4
        rate-limit: 5
        pkey: 0x1234
        packet-life: 8
    end-qos-level

end-qos-levels

# Match rules are scanned in order of their appereance in the
# policy file. First matched rule takes precedence.
qos-match-rules

    # matching by single criteria: QoS class
    qos-match-rule
        use: by QoS class
        qos-class: 7-9,11
        # Name of qos-level to apply to the matching PR/MPR
        qos-level-name: WholeSet
    end-qos-match-rule

    # show matching by destination group and service id
```

```
        qos-match-rule
            use: Storage targets
            destination: Storage
            service-id: 0x10000000000001, 0x10000000000008-
0x10000000000FFF
            qos-level-name: WholeSet
        end-qos-match-rule

        qos-match-rule
            source: Storage
            use: match by source group only
            qos-level-name: DEFAULT
        end-qos-match-rule

        qos-match-rule
            use: match by all parameters
            qos-class: 7-9,11
            source: Virtual Servers
            destination: Storage
            service-id: 0x0000000000010000-0x000000000001FFFF
            pkey: 0x0F00-0x0FFF
            qos-level-name: WholeSet
        end-qos-match-rule

    end-qos-match-rules
```

Note Some of these match rules may overlap, so in order to use the simplified QoS definition effectively, it is important to understand how each of the ULPs match.

IPoIB rules

The IPoIB query is matched with PKey. Default PKey for IPoIB partition is 0x7fff, so the following three match rules are equivalent:

```
    ipoib              : <SL>
    ipoib, pkey 0x7fff : <SL>
    any,   pkey 0x7fff : <SL>
```

SDP rules

SDP PR query is matched by the Service ID. The Service-ID for SDP is 0x000000000001PPPP, where PPPP = 4 hex digits for the remote TCP/IP port number for the connection. The following two match rules are equivalent:

```
   sdp                                                     : <SL>
   any, service-id 0x0000000000010000-0x000000000001ffff : <SL>
```

MPI rules

SL for MPI is manually configured by the MPI Administrator. OpenSM does not force a Service Level (SL) for the MPI traffic, and therfore only ULP does not appear in the qos-ulps section.

See http://www.openfabrics.org/ for more information on QoS and its service matching rules.

2.4.1.3 QoS OpenSM file

OpenSM options file has a set of QoS related configuration parameters that are used to configure SL2VL mapping and VL arbitration for the InfiniBand ports. These parameters include:

•	Max VLs	The maximum number of VLs that for a subnet.
•	High limit	The limit for High Priority components in the VL Arbitration table (IBA 7.6.9).
•	VLArb low table	Low priority VL Arbitration table (IBA 7.6.9) template.
•	VLArb high table	High priority VL Arbitration table (IBA 7.6.9) template.
•	SL2VL	SL2VL Mapping table (IBA 7.6.6) template. It is a list of VLs corresponding to SLs 0-15 (Note that VL15 used here means drop this SL).

There are separate QoS configuration parameters sets for various target types, CAs, routers, switch external ports, and port 0 enhanced switches. These parameters are prefixed by the **qos_<type>_** string.

The list of the currently supported sets is:

•	qos_ca_	QoS configuration parameters set for CAs.
•	qos_rtr_	Parameters set for routers.
•	qos_sw0_	Parameters set for switches' port 0.
•	qos_swe_	Parameters set for switches' external ports.

An example, including typical default values for CAs and switches' external ports, hard-coded in the OpenSM initialization is shown below:

```
   qos_ca_max_vls 15
   qos_ca_high_limit 0
   qos_ca_vlarb_high
0:4,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0,10:0,11:0,12:0,13:0,14:0
   qos_ca_vlarb_low
0:0,1:4,2:4,3:4,4:4,5:4,6:4,7:4,8:4,9:4,10:4,11:4,12:4,13:4,14:4
   qos_ca_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7

   qos_swe_max_vls 15
   qos_swe_high_limit 0
   qos_swe_vlarb_high
0:4,1:0,2:0,3:0,4:0,5:0,6:0,7:0,8:0,9:0,10:0,11:0,12:0,13:0,14:0
   qos_swe_vlarb_low
0:0,1:4,2:4,3:4,4:4,5:4,6:4,7:4,8:4,9:4,10:4,11:4,12:4,13:4,14:4
   qos_swe_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7
```

VL arbitration tables (both high and low) are lists of VL/Weight pairs. Each list entry contains a VL number (values from 0-14), and a weighting value (values 0-255), indicating the number of 64 byte units (credits) which may be transmitted from that VL when its turn in the arbitration occurs. A weight of 0 indicates that this entry should be skipped. If a list entry is programmed for VL15, or for a VL that is not supported, or is not currently configured by the port, the port may either skip that entry or send from any supported VL for that entry.

Note The same VL may be listed multiple times in the High or Low priority arbitration tables, and, further, it can be listed in both tables.

The limit of **high-priority VLArb** table (**qos_<type>_high_limit**) indicates the number of high-priority packets that can be transmitted. Specifically, the number of bytes that can be sent is **high_limit** times 4K bytes.

A **high_limit** value of 255 indicates that the byte limit is unbounded.

Note If the 255 value is used, low priority VLs may be starved.

A value of 0 indicates that only a single packet from the high-priority table may be sent before an opportunity is given to the low-priority table. Keep in mind that ports usually transmit packets of size equal to the **MTU**. For instance, for a **4KB MTU** a single packet will require 64 credits, so in order to achieve effective VL arbitration for packets of **4KB MTU**, the weighting values for each VL should be multiples of 64.

Below is an example of **SL2VL** and **VL** arbitration configuration on a subnet:

```
qos_ca_max_vls 15
qos_ca_high_limit 6
qos_ca_vlarb_high 0:4
qos_ca_vlarb_low 0:0,1:64,2:128,3:192,4:0,5:64,6:64,7:64
qos_ca_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7

qos_swe_max_vls 15
qos_swe_high_limit 6
qos_swe_vlarb_high 0:4
qos_swe_vlarb_low 0:0,1:64,2:128,3:192,4:0,5:64,6:64,7:64
qos_swe_sl2vl 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,7
```

In this example, there are 8 VLs (as is the case for **ConnectX** cards) configured on the subnet: **VL0** to **VL7**. **VL0** is defined as a high priority **VL**, and it is limited to 6 x 4KB = 24KB in a single transmission burst. This type of configuration suits VLs that need low latency and use small MTU when transmitting packets. The rest of the VLs are defined as low priority VLs with different weights, while VL4 is effectively turned off.

2.4.1.4 Verifying QoS Functionality

1. To verify the **QoS** settings attribution use the **smpquery** tool included in the **InfiniBand-diags** package.
2. To verify **QoS** settings application use **qperf** included in **qperf** package.
3. In general, HCAs cards have only 8 VLs:

```
smpquery portinfo 1 | grep VL
```

```
VLCap:..........................VL0-7
VLHighLimit:....................0
VLArbHighCap:...................8
VLArbLowCap:....................8
VLStallCount:...................0
OperVLs:........................VL0-7
```

4. Look at the corresponding **QoS SL** to **VL** mapping:

```
smpquery sl2vl 1
```

```
# SL2VL table: Lid 1
#                  SL: | 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|15|
ports: in  0, out  0: | 0| 1| 2| 3| 4| 6| 7| 0| 1| 2| 3| 4| 5| 6| 7| 5|
```

5. View the new QoS credit:

```
smpquery vlarb 1
```

```
# VLArbitration tables: Lid 1 port 0 LowCap 8 HighCap 8
# Low priority VL Arbitration Table:
VL    : |0x0 |0x1 |0x2 |0x3 |0x1 |0x0 |0x0 |0x0 |
WEIGHT: |0x1 |0x40|0x80|0x40|0x40|0x0 |0x0 |0x0 |
# High priority VL Arbitration Table:
VL    : |0x0 |0x1 |0x2 |0x3 |0x0 |0x0 |0x0 |0x0 |
WEIGHT: |0x0 |0x0 |0x0 |0x0 |0x0 |0x0 |0x0 |0x0 |
```

6. Test the QoS configuration with the **qperf** (qperf package) tool. Launch **qperf** on a remote server named **host1** and check the QoS efficiency on **host2**, using the command below:

```
qperf host1 -lp 19766 -sl 1 rc_rdma_write_bw & qperf host1 -lp 19764 -
sl 3 rc_rdma_write_bw
```

```
  rc_rdma_write_bw:
  rc_rdma_write_bw:
    bw  =  2.27 GB/sec
    bw  =  1.14 GB/sec
```

2.4.2 IPoIB

Carry out the following actions to use the **Internet Protocol over InfiniBand**.

2.4.2.1 Creating the IPoIB interfaces

Firstly, load the **ib_ipoib** module (if it is not already loaded):

```
# modprobe -v ib_ipoib
```

For each IB port on the node there should be an entry in the **/etc/modprobe.conf** file.

Example for 2 ports

```
alias ib0 ib_ipoib
alias ib1 ib_ipoib
```

These entries are generated automatically when **kernel-ib** is installed.

2.4.2.2 Setting a default IP address for an IPoIB interface

The default IP configuration of an IPoIB interface is set through a file in **/etc/sysconfig/network-script/** directory. These network configuration files can also be generated by using the **config_ip** or **config_ipoib** command (the command available depends on the system).

ib0 interface example

```
cat /etc/sysconfig/network-scripts/ifcfg-ib0
DEVICE=ib0
BOOTPROTO=static
IPADDR=192.168.0.215
NETMASK=255.255.255.0
ONBOOT=yes
```

IP over IB (IPoIB) can be used in two different modes:

- **Datagram**: In datagram mode, all packets are broadcast over the InfiniBand network. Although it may be useful for multicast operations, performance is low due to a limited MTU (2048). Note that multicast is not used by **MPI** or **Lustre**.
- **Connected**: In connected mode, point-to-point IB connections are created between the hosts for **IPoIB** transfers. This mode does not allow for broadcast operations. However, it allows for a large MTU (65520) and thus achieves a much higher bandwidth.

Note Within a given InfiniBand network, all IP over IB interfaces must use the same mode. The recommended mode for a Bull cluster is **connected**.

See The chapter on *InfiniBand Network Optimization* for more information on setting the **Datagram** and **Connected** modes for the **IPoIB** interfaces.

2.4.2.3 Using a specific pkey for an IPoIB interface

When the **IPoIB** driver is loaded, it creates one interface for each port with the **P_Key** at index 0. To create an interface with a different **P_Key**, write the desired **P_Key** into the **/sys/class/net/<intf name>/create_child** file for the main interface.

For example:

```
# echo 0x8001 > /sys/class/net/ib0/create_child
```

This will create an interface named ib0.8001 with P_Key 0x8001.
By default this interface is in **Datagram** mode, which implies a max **2K MTU** and as a consequence a very poor performance. Run the command below, where ib0.8001 is the name of your new interface, to fix this. This will change the IPoIB interface to connected mode:

```
# echo "connected" > /sys/class/net/ib0.8001/mode
```

Update the **MTU** to ensure better performance:

```
# ifconfig ib0.8001 mtu 65520
```

Remove the sub-interface, by using the **delete_child** file, command as below:

```
# echo 0x8001 > /sys/class/net/ib0/delete_child
```

A **pkey** specific **ipoib** interface can be configured automatically by adding the associated **/etc/sysconfig/network-scripts/ifcfg-ib<num_interface>.<num_pkey>**. An example of a **num_pkey** is ib0.8001.
OpenIB will automatically create and set the interface at boot time.

2.4.2.4 Verifying IPoIB Functionality

To check the **IPoIB** interface configuration, perform a **ping** command to a distant host (node). Firstly, check that the interface is reachable by using the **ifconfig** command:

```
ifconfig
```

```
ib0       Link encap:InfiniBand  HWaddr
80:00:00:48:FE:80:00:00:00:00:00:00:00:00:00:00:00:00:00:00
          inet addr:192.168.0.215  Bcast:192.168.0.255
Mask:255.255.255.0
          inet6 addr: fe80::202:c903:0:2071/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:65520  Metric:1
          RX packets:200 errors:0 dropped:0 overruns:0 frame:0
          TX packets:133 errors:0 dropped:47 overruns:0 carrier:0
          collisions:0 txqueuelen:256
          RX bytes:13020 (12.7 KiB)  TX bytes:8128 (7.9 KiB)
```

Ping your distant node with **192.168.0.214** IP address, for example.

```
ping 192.168.0.214
```

```
PING 192.168.0.214 (192.168.0.214) 56(84) bytes of data.
64 bytes from 192.168.0.214: icmp_seq=0 ttl=64 time=0.079 ms
64 bytes from 192.168.0.214: icmp_seq=1 ttl=64 time=0.044 ms
64 bytes from 192.168.0.214: icmp_seq=2 ttl=64 time=0.055 ms
```

2.4.3 Ethernet Over IB

Carry out the following actions In order to use the **Ethernet over InfiniBand** protocol.

Important This option is strictly available on Mellanox ConnectX cards and requires an OFED version >= 1.4 with ConnectX Firwmare >= 2.6.0

2.4.3.1 Creating Ethernet over IB interfaces

Firstlym load the **mlx4_en** module (if it is not already loaded):

```
# modprobe -v mlx4_en
```

By default both **ConnectX** ports are initialized as **InfiniBand** ports. If you wish to change the port type use the **connectx_port_config** script after the driver is loaded. Running the **/sbin/connectx_port_config -s** command will show the current port configuration for all ConnectX devices.

Possible port types are:

- **eth** Always Ethernet.
- **ib** Always InfiniBand.
- **auto** Link sensing mode - detect port type based on the attached network type.

If no link is detected, the driver will retry every few seconds to connect to the link. The port link type can be configured for each device in the system, at run time, by using the **/sbin/connectx_port_config** script.

```
# /sbin/connectx_port_config
```

```
 ConnectX PCI devices :
 |----------------------------|
 | 1             0000:07:00.0 |
 |----------------------------|
 Before port change:
 ib
 ib
 |----------------------------|
 | Possible port modes:       |
 | 1: InfiniBand              |
 | 2: Ethernet                |
 | 3: AutoSense               |
 |----------------------------|
 Make your changes
 Select mode for port 1 (1,2,3): 2

After port change:
 eth
 eth
```

Check that the changes have been put in place:

```
service openibd status
```

```
   [...]
   HCA driver loaded
   Configured MLX4_EN devices:
   eth2 eth3
   [..]
```

2.4.3.2 Setting a Default IP Address for an Ethernet over IB interface

The default IP configuration of an **Ethernet over IB** interface is set through a file in */etc/sysconfig/network-script/*

Here is an example for an **eth2** interface:

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

```
 DEVICE=eth2
 BOOTPROTO=static
 IPADDR=192.168.0.215
 NETMASK=255.255.255.0
 ONBOOT=yes
```

2.4.3.3 Verifying Ethernet over IB Functionality

To verify your Ethernet over IB interface configuration, perform a **ping** command to a distant host (node). First check that your interface is reachable using the **ifconfig** command:

```
ifconfig
```

```
 eth2      Link encap:Ethernet  HWaddr 00:02:C9:00:20:72
         inet addr:192.168.0.215  Bcast:192.168.0.255
Mask:255.255.255.0
         UP BROADCAST MULTICAST  MTU:1500  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:1000
         RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

Ping the distant node using the **192.168.0.214** IP address.

```
ping 192.168.0.214
```

```
PING 192.168.0.214 (192.168.0.214) 56(84) bytes of data.
64 bytes from 192.168.0.214: icmp_seq=0 ttl=64 time=0.079 ms
64 bytes from 192.168.0.214: icmp_seq=1 ttl=64 time=0.044 ms
64 bytes from 192.168.0.214: icmp_seq=2 ttl=64 time=0.055 ms
```

Note The port configuration is saved in the **/etc/InfiniBand/connectx.conf** file: The configuration saved is only restored when the drivers are restarted using the **/etc/init.d/openibd restart** command.

The following configurations are supported:

Port1 = eth Port2 = eth

Port1 = ib Port2 = ib

Port1 = auto Port2 = auto

Port1 = ib Port2 = eth

Port1 = ib Port2 = auto

Port1 = auto Port2 = eth

The following options are not supported:

Port1 = eth Port2 = ib

Port1 = eth Port2 = auto

Port1 = auto Port2 = ib

2.4.4 SDP

SDP defines a standard protocol over an **RDMA** fabric to support stream sockets (**SOCK_STREAM**) networks. **SDP** utilizes various **RDMA** network features for high-performance zero-copy data transfers. **SDP** is a pure wire-protocol level specification and does not go into any socket **API** or implementation specifics. The purpose of the **Sockets Direct Protocol** is to provide an **RDMA** accelerated alternative to the TCP IP protocol. The aim is to do this in a manner which is transparent to the application.

2.4.4.1 Creating SDP interfaces

First load the **ib_sdp** module, if it is not already loaded:

```
# modprobe -v ib_sdp
```

To use the sdp socket, add an entry in the **/etc/modprobe.conf** file.

```
cat /etc/modprobe.conf | grep ib_sdp
```

```
alias net-pf-27 ib_sdp
```

2.4.4.2 Setting a default IP address for an SDP interface

SDP uses the same interface as IPoIB.

See Section 2.4.2.2 for details on how to create a default IP address.

2.4.4.3 SDP configuration

Two environment variables are required to use a SDP socket:

LD_PRELOAD – This environment variable is used to preload libsdp.so and it should point to the libsdp.so library. The variable should be set by the System Administrator to /usr/lib64/libsdp.so.

LIBSDP_CONFIG_FILE – This environment variable is used to configure the policy for replacing TCP sockets with SDP sockets. By default it points to: /etc/libsdp.conf.

The libsdp.conf configuration (policy) file controls the automatic transparent replacement of TCP sockets with SDP sockets. Socket Stream selection could be done a specific port (destination port or listening port) or a programme name. Socket control statements in the libsdp.conf file allow the user to specify when libsdp should replace AF_INET/SOCK_STREAM sockets with AF_SDP/SOCK_STREAM sockets.

The use statement controls which type of sockets to open. The format of the use statement is as follows:

```
use <address-family> <role> <program-name|*> <address|*>:<port range|*>
```

The <address-family> field represents the family socket type:

• sdp	To specify when an SDP socket should be used
• tcp	To specify when an SDP socket should not be matched
• Both	To specify when both SDP and AF_INET sockets should be used

Note The semantics is different for server and client roles. For the server, it means that the server will be listening on both SDP and TCP sockets. For the client, the connect function will first attempt to use SDP, and will silently fall back to TCP if the SDP connection fails.

<role> can be one of:

• server or listen	To define the listening port address family
• client or connect	To define the connected port address family

<program-name|*> defines the program name the rule applies to (not including the path). For example, tcp would match on ttcp. If program-name is not provided (default), the statement matches all programs.

<address|*> This is either the local address to which the server binds, or the remote server address to which the client connects. The syntax for address matching is:

<IPv4 address>[/<prefix_length>]|*

Example

IPv4 address = [0-9]+\.[0-9]+\.[0-9]+\.[0-9]+ each sub number < 255

<port range> : start-port[-end-port] where port numbers are >0 and <65536

Note Rules are checked in the order that they are defined. So the first match wins. If no match is made, **libsdp** will default and use the **Both** rule (connect through SDP and fallback to TCP).

Examples

a. SDP used by clients connected to machines that belong to subnet 192.168.0.*

```
use sdp connect *        192.168.0.0/24:*
```

b. SDP used by **ttcp** when it connects to port 5001 of any machine

```
use sdp       listen    ttcp         *:5001
```

c. SSH connected through SDP and fallback to TCP to hosts on 11.2.6.* port 22

```
use both      connect *              11.2.6.0/24:22
```

d. TCP used for any program with name starting with ttcp* serving ports 22 to 26

```
use tcp       server    ttcp*        *:22-26
```

e. Listen on both TCP and SDP for any server that listens on port 8080

```
use both      server    *            *:8080
```

2.4.4.4 Verifying SDP Functionality

The **netperf** package can be used to test SDP performance. **Netperf** is available from http://www.netperf.org/netperf/DownloadNetperf.html.

1. Start the **netperf** server and force SDP to be used instead of TCP.

```
LD_PRELOAD=/usr/lib64/libsdp.so LIBSDP_CONFIG_FILE=/etc/libsdp.conf
netserver
```

```
       Starting netserver at port 12865
       Starting netserver at hostname 0.0.0.0 port 12865 and family
AF_UNSPEC
       host1#
```

2. Run the **netperf** client so that you force **SDP** to be used instead of TCP. The default test is the Bandwidth test.

```
LD_PRELOAD=/usr/lib64/libsdp.so LIBSDP_CONFIG_FILE=/etc/libsdp.conf
netperf -H 192.168.0.215 -t TCP_STREAM -c -C -- -m 65536
```

```
TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 11.4.17.6
(11.4.17.6) port 0 AF_INET
Recv     Send     Send                               Utilization         Service Demand
Socket  Socket     Message   Elapsed                  Send        Recv      Send    Recv
Size     Size     Size      Time        Throughput    local      remote    local   remote
bytes    bytes     bytes     secs.       10^6bits/s    % S         % S      us/KB   us/KB
87380    16384    65536     10.00         5872.60     19.41      17.12     2.166
```

Chapter 3. InfiniBand Tools

A series of InfiniBand tools is provided by the InfiniBand-diags and ibs packages. These check the network over the InfiniBand Fabric but also provide HCA/Switch state and configuration information.

6 types of tools are available:

Commands used to check the ocal InfiniBand status (HCA cards status)

ibstat	Display the status for the host adapters
ibstatus	Similar to ibstat but implemented as a script
ibaddr	Shows the lid range and default GID of the target (the default is the local port)

Commands used to scan IB network

Ibnetdiscover	Scan topology
smpquery	Formatted SMP query tool
saquery	Query the InfiniBand subnet administration attribute
perfquery	Dump (and optionally clear) the performance (including error) counters of the destination port
sminfo	Query the SMInfo attribute for a node
ibroute	Display the unicast and multicast forwarding tables for the switches
ibdiagnet	InfiniBand diagnostic net

Command used to change InfiniBand attributes

ibportstate	Manage the port (physical) state and link speed of an InfiniBand port

Commands used to view InfiniBand network errors

Ibcheckerrors	Check if the error counters of a port/node are within predefined thresholds
ibchecknet	Perform port/node/errors check on the subnet. ibnetdiscover output can be used as in input topology

Commands used to test IB network configuration

ibcheckport	Perform some basic tests on the specified port
ibchecknode	Perform some basic tests on the specified node
ibclearcounters	Clear port counters for the InfiniBand subnet

Others

ibtracert	Display unicast or multicast routes from source to destination
ibping	Ping/pong between InfiniBand nodes (currently using vendor MADs)
ibsysstat	Obtain basic information for remote nodes (hostname, cpus, memory, utilization)

smpdump	Simple solicited **SMP** query tool. Output is hex dump (unless requested otherwise, e.g. using **-s**)
ibswitches	Scan the net or use existing net topology file and list all switches
ibhosts	Scan the net or use existing net topology file and list all hosts

Various Troubleshooting Tasks can be carried out using the **OpenIB** Diagnostics tools. Some examples are shown below:

- Baseline topology checks with **ibnetdiscover** and compare the current topology to it.
- Run **ibchecknet** to scan the network for errors
- Use **perfquery** to obtain a more complete set of counters
- Use **smpquery** to check the speed of the links via the **portinfo** command and **LinkWidthActive** output value.
- Use **ibroute** and **ibtracert to** troubleshoot unicast and multicast issues

See The man page for each tool to obtain more details about it.

3.1 Standard tools options

Most **OpenIB** diagnostic tools use the following common flags. The exact list of supported flags per utility is found in the usage message and is shown using the **util_name -h** syntax.

Debugging flags

-d	Raise the InfiniBand debugging level, may be used several times (**-ddd** or **-d -d -d**)
-e	Show send and receive errors (timeouts and others)
-h	Show the usage message
-v	Increase the application verbosity level, may be used several times (**-vv** or **-v -v -v**)
-V	Show the version info

Addressing flags

-D	Use directed path address arguments. The path is a comma separated list of out ports
-G	Use the **GUID** address argument. In most case this is the **Port GUID**
-s <smlid>	Use **smlid** as the target lid for SM/SA queries

Other common flags

-C <ca_name>	Use the specified **ca_name**
-P <ca_port>	Use the specified **ca_port**
-t <timeout_ms>	Override the default timeout for the solicited MADs

When no InfiniBand device or port is specified, the port to be uses is selected by the following criteria:

a. The first port that is **ACTIVE**

b. If not found, the first port that is UP (physical link up)

If a port and/or CA name is specified, the user request is attempts to use this, and will fail if it is not possible.

3.2 ibstat

ibstat Query basic status of InfiniBand device(s)

ibstat is a binary tool which displays basic information obtained from the local InfiniBand driver. The output includes **LID**, **SMLID**, **port state**, **link width** active, and **port** physical state information. It is similar to the **ibstatus** utility but implemented as a binary rather than a script. It has options to list **CAs** and/or ports and displays more information than **ibstatus**.

Usage

ibstat [-d(ebug)] [-l(ist_of_cas)] [-s(hort)] [-p(ort_list)] [-V(ersion)] [-h] <ca_name> [portnum]

Options

-l, –list_of_cas	list all InfinBand devices
-s, –short	Short output
-p, –port_list	Show port list
ca_name	InfiniBand device name
portnum	Port number of InfiniBand device

Examples

```
ibstat
```

Display status of all ports on all InfiniBand devices.

Example output for ibstat

```
CA 'mlx4_0' (Mellanox ConnectX card)
          CA type: MT26428
          Number of ports: 2
          Firmware version: 2.6.0
          Hardware version: a0
          Node GUID: 0x0002c903000262a2
          System image GUID: 0x0002c903000262a5
          Port 1:
                 State: Active (Active Down)
                 Physical state: LinkUp (Linkup, Polling)
                 Rate: 40 (10Gb/s SDR, 20Gb/s DDR, 40Gb/s QDR)
                 Base lid: 14 (attribute port lid)
                 LMC: 0 (Lid Mask Control)
                 SM lid: 1 (which SM lid the port belong to)
                 Capability mask: 0x00000000
                 Port GUID: 0x0002c903000262a3 (unique port guid)
          Port 2:
                 State: Down
                 Physical state: Polling
                 Rate: 10
                 Base lid: 0
                 LMC: 0
                 SM lid: 0
                 Capability mask: 0x00000000
```

```
                Port GUID: 0x0002c903000262a4
```

```
ibstat -l
```

List all **InfiniBand** devices.

```
ibstat -p
```

Show port guids.

```
ibstat mthca0 2
```

Show status of port 2 for the switch **mthca0**.

3.3 ibnetdiscover

ibnetdiscover Discover the InfiniBand topology

ibnetdiscover examines the **InfiniBand** subnet and outputs a human readable topology file which includes **GUID**, node type, and port number, port LID and NodeDescription details . All nodes (and links) are displayed (full topology). Optionally, this utility can be used to list the nodes that are connected, by node type. The output is printed in a standard output unless a topology file is specified.

Usage

ibnetdiscover [-d(ebug)] [-e(rr_show)] [-v(erbose)] [-s(how)] [-l(ist)] [-g(rouping)] [-H(ca_list)] [-S(witch_list)] [-R(outer_list)] [-C ca_name] [-P ca_port] [-t(imeout) timeout_ms] [-V(ersion)] [–node-name-map <node-name-map>] [-p(orts)] [-h(elp)] [<topology-file>]

Options

-l, –list	List of connected nodes
-g, –grouping	Show grouping. Grouping correlates **InfiniBand** nodes by different vendor specific schemes. It may also show the switch external ports correspondence.
-H, –Hca_list	List of **CAs** that are connected
-S, –Switch_list	List of switches that are connected
-R, –Router_list	List of connected routers
-s, –show	Show more information
–node-name-map <node-name-map>	Specify a node name map. The node name map file maps GUIDs to more user-friendly names. See file format below.
-p, –ports:	Obtain a ports report which is a list of connected ports with relevant information (for example **LID**, **portnum**, **GUID**, **width**, **speed**, and **Node Description**).

Topology file format

The topology file format is human readable and largely intuitive. Most identifiers are given textual names like vendor ID (vendid), device ID (device ID), GUIDs of various types (**sysimgguid**, **caguid**, **switchguid**, etc.). PortGUIDs are shown in parentheses ().

Switch identifiers are shown on the switchguid line. CA and router ports identifiers are shown on the connectivity lines. The InfiniBand node is identified, and the number of ports and the node GUID are shown.

On the right of this line is a comment (#) followed by the Node Description in quotes. If the node is a switch, this line also states if the switch port 0 is base or enhanced, and the LID and LMC of port 0. Subsequent lines pertaining to this node show the connectivity. On the left is the port number of the current node. On the right is the peer node (node at other end of link). It is shown in quotes with the node type followed by the NodeGUID with the port number in square brackets.

Further on the right is a comment (#). The comment depends on the node type. If it it a switch node, it is followed by the Node Description in quotes and the LID of the peer node. If it is a CA or router node, it is followed by the local LID and LMC and then followed by the Node Description in quotes and the LID of the peer node. The active link width and speed are then appended to the end of this output line.

Topology file example

```
# Topology file:
## Max of 3 hops discovered
# Initiated from node 0008f10403960558 port 0008f10403960559

      Non-Chassis Nvendid=0x8f1
      devid=0x5a06
      sysimgguid=0x5442ba00003000
      switchguid=0x5442ba00003080(5442ba00003080)
      Switch  24 "S-005442ba00003080"      # "ISR9024 Voltaire" base port 0 lid 6 lmc 0
      [22]    "H-0008f10403961354"[1](8f10403961355)    # "MT23108 InfiniHost Mellanox
Technologies" lid 4 4xSDR
      [10]    "S-0008f10400410015"[1]      # "SW-6IB4 Voltaire" lid 3 4xSDR
      [8]     "H-0008f10403960558"[2](8f1040396055a)    # "MT23108 InfiniHost Mellanox
Technologies" lid 14 4xSDR
      [6]     "S-0008f10400410015"[3]      # "SW-6IB4 Voltaire" lid 3 4xSDR
      [12]    "H-0008f10403960558"[1](8f10403960559)    # "MT23108 InfiniHost Mellanox
Technologies" lid 10 4xSDR

      vendid=0x8f1
      devid=0x5a05
      switchguid=0x8f10400410015(8f10400410015)
      Switch  8 "S-0008f10400410015"       # "SW-6IB4 Voltaire" base port 0 lid 3 lmc 0
      [6]     "H-0008f10403960984"[1](8f10403960985)    # "MT23108 InfiniHost Mellanox
Technologies" lid 16 4xSDR
      [4]     "H-005442b100004900"[1](5442b100004901)   # "MT23108 InfiniHost Mellanox
Technologies" lid 12 4xSDR
      [1]     "S-005442ba00003080"[10]        # "ISR9024 Voltaire" lid 6 1xSDR
      [3]     "S-005442ba00003080"[6]         # "ISR9024 Voltaire" lid 6 4xSDR

      vendid=0x2c9
      devid=0x5a44
      caguid=0x8f10403960984
      Ca      2 "H-0008f10403960984"          # "MT23108 InfiniHost Mellanox Technologies"
      [1](8f10403960985)     "S-0008f10400410015"[6]    # lid 16 lmc 1 "SW-6IB4 Voltaire"
lid 3 4xSDR

      vendid=0x2c9
      devid=0x5a44
      caguid=0x5442b100004900
      Ca      2 "H-005442b100004900"         # "MT23108 InfiniHost Mellanox Technologies"
      [1](5442b100004901)     "S-0008f10400410015"[4]   # lid 12 lmc 1 "SW-6IB4 Voltaire"
lid 3 4xSDR

      vendid=0x2c9
      devid=0x5a44
      caguid=0x8f10403961354
      Ca      2 "H-0008f10403961354"         # "MT23108 InfiniHost Mellanox Technologies"
      [1](8f10403961355)     "S-005442ba00003080"[22]   # lid 4 lmc 1 "ISR9024 Voltaire"
lid 6 4xSDR

      vendid=0x2c9
      devid=0x5a44
      caguid=0x8f10403960558
      Ca      2 "H-0008f10403960558"         # "MT23108 InfiniHost Mellanox Technologies"
      [2](8f1040396055a)     "S-005442ba00003080"[8]    # lid 14 lmc 1 "ISR9024 Voltaire"
lid 6 4xSDR
      [1](8f10403960559)     "S-005442ba00003080"[12]   # lid 10 lmc 1 "ISR9024 Voltaire"
lid 6 1xSDR
```

When grouping is used, InfiniBand nodes are organized into chasses which are numbered. Nodes for which no chassis is detected are displayed *Non Chassis Nodes*. External ports are also shown on the connectivity lines.

Node map file format

The node name map is used to specify user friendly names for the nodes in the output. GUIDs are used to perform the lookup, for example:

#comment
 <guid> "<name>"

Example

```
# IB1
# Line cards
0x0008f104003f125c "IB1 (Rack 11 slot 1   ) ISR9288/ISR9096 Voltaire sLB-24D"
0x0008f104003f125d "IB1 (Rack 11 slot 1   ) ISR9288/ISR9096 Voltaire sLB-24D"
0x0008f104003f10d2 "IB1 (Rack 11 slot 2   ) ISR9288/ISR9096 Voltaire sLB-24D"
0x0008f104003f10d3 "IB1 (Rack 11 slot 2   ) ISR9288/ISR9096 Voltaire sLB-24D"
0x0008f104003f10bf "IB1 (Rack 11 slot 12  ) ISR9288/ISR9096 Voltaire sLB-24D"
# Spines
0x0008f10400400e2d "IB1 (Rack 11 spine 1   ) ISR9288 Voltaire sFB-12D"
0x0008f10400400e2e "IB1 (Rack 11 spine 1   ) ISR9288 Voltaire sFB-12D"
0x0008f10400400e2f "IB1 (Rack 11 spine 1   ) ISR9288 Voltaire sFB-12D"
0x0008f10400400e31 "IB1 (Rack 11 spine 2   ) ISR9288 Voltaire sFB-12D"
0x0008f10400400e32 "IB1 (Rack 11 spine 2   ) ISR9288 Voltaire sFB-12D"
# GUID   Node Name
0x0008f10400411a08 "SW1  (Rack  3) ISR9024 Voltaire 9024D"
0x0008f10400411a28 "SW2  (Rack  3) ISR9024 Voltaire 9024D"
0x0008f10400411a34 "SW3  (Rack  3) ISR9024 Voltaire 9024D"
0x0008f104004119d0 "SW4  (Rack  3) ISR9024 Voltaire 9024D"
```

ibnetdiscover examples

Obtain distant lid switch and port linked to HCA card information.

```
ibnetdiscover | grep inti17 (In this example HCA name in relation to
machine name)
```

```
[4] "H-0002c90300025e9e"[1](2c90300025e9f)     # "inti17 HCA-1" lid 12
4xQDR (lid 12 correspond to switch port lid)
```

```
Ca 2 "H-0002c90300025e9e"     # "inti17 HCA-1"
ibnetdiscover | grep "lid 12"
 [4] "H-0002c90300025e9e"[1](2c90300025e9f)     # "inti17 HCA-1" lid 12
4xQDR
 [1](2c90300025e9f)  "S-0002c902004047c0"[4]    # lid 12 lmc 0
"Infiniscale-IV Mellanox Technologies" lid 21 4xQDR
You could now execute query with <lid_switch> (21) and
<real_port_switch> (4)
```

View additional network information.

```
ibnetdiscover -s | more
```

```
DR path slid 0; dlid 0; 0,1 -> new remote switch {0002c90200404798}
portnum 0 lid 11-11"Infiniscale-IV Mellanox Technologies"
DR path slid 0; dlid 0; 0,1 -> processing switch {0002c90200404798}
portnum 0 lid 1-1"Infiniscale-IV Mellanox Technologies"
DR path slid 0; dlid 0; 0,1,2 -> new remote ca {0002c90300025e26}
portnum 1 lid 23-23"inti1 HCA-1"
DR path slid 0; dlid 0; 0,1,3 -> new remote switch {0002c902004044e0}
portnum 0 lid 17-17"Infiniscale-IV Mellanox Technologies"
DR path slid 0; dlid 0; 0,1,4 -> known remote switch
{0002c902004044e0} portnum 0 lid 17-17"Infiniscale-IV Mellanox
Technologies"
DR path slid 0; dlid 0; 0,1,5 -> new remote switch {0002c902004047c0}
portnum 0 lid 21-21"Infiniscale-IV Mellanox Technologies"
DR path slid 0; dlid 0; 0,1,5 -> processing switch {0002c902004047c0}
portnum 0 lid 0-0"Infiniscale-IV Mellanox Technologies"
DR path slid 0; dlid 0; 0,1,5,2 -> new remote ca {0002c90300025e82}
portnum 1 lid 6-6"inti15 HCA-1"
DR path slid 0; dlid 0; 0,1,5,3 -> new remote ca {0002c90300025f7a}
portnum 1 lid 5-5"inti14 HCA-1"
DR path slid 0; dlid 0; 0,1,5,4 -> new remote ca {0002c90300025e9e}
portnum 1 lid 12-12"inti17 HCA-1"
DR path slid 0; dlid 0; 0,1,5,6 -> new remote ca {0002c90300025f82}
portnum 1 lid 7-7"inti19 HCA-1"
DR path slid 0; dlid 0; 0,1,5,7 -> new remote ca {0002c90300025f72}
portnum 1 lid 4-4"inti18 HCA-1"
DR path slid 0; dlid 0; 0,1,3 -> processing switch {0002c902004044e0}
portnum 0 lid 0-0"Infiniscale-IV Mellanox Technologies"
DR path slid 0; dlid 0; 0,1,3,1 -> new remote ca {0002c90300025ea6}
portnum 1 lid 15-15"inti5 HCA-1"
DR path slid 0; dlid 0; 0,1,3,2 -> new remote ca {0002c90300025ea2}
portnum 1 lid 13-13"inti4 HCA-1"
DR path slid 0; dlid 0; 0,1,3,3 -> new remote ca {0002c90300026316}
portnum 1 lid 3-3"inti7 HCA-1"
DR path slid 0; dlid 0; 0,1,3,4 -> new remote ca {0002c903000262a2}
portnum 1 lid 14-14"MT25408 ConnectX Mellanox Technologies"
DR path slid 0; dlid 0; 0,1,3,7 -> new remote ca {0002c90300025eb2}
portnum 1 lid 16-16"inti11 HCA-1"
DR path slid 0; dlid 0; 0,1,3,8 -> new remote ca {0002c903000262ba}
portnum 1 lid 10-10"inti10 HCA-1"
DR path slid 0; dlid 0; 0,1,3,9 -> new remote ca {0002c90300025e9a}
portnum 1 lid 3963-3963"inti13 HCA-1"
```

3.4 smpquery

smpquery	Query **InfiniBand** subnet management attributes.
smpquery	Runs a basic subset of standard SMP queries including the following: **node info, node description, switch info, port info**. Fields are displayed in human readable format.

Usage

smpquery [-d(ebug)] [-e(rr_show)] [-v(erbose)] [-D(irect)] [-G(uid)] [-C ca_name] [-P ca_port] [-t(imeout) timeout_ms] [–node-name-map node-name-map] [-V(ersion)] [-h(elp)] <op> <dest dr_path|lid|guid> [op params]

Options

nodeinfo <addr>

nodedesc <addr>

portinfo <addr> [<portnum>] # default port is zero

switchinfo <addr>

pkeys <addr> [<portnum>]

sl2vl <addr> [<portnum>]

vlarb <addr> [<portnum>]

guids <addr>

–node-name-map <node-name-map>	Specify a node name map. The node name map file maps GUIDs to more user friendly names.

Examples

```
smpquery portinfo 3 1
```

portinfo by lid, with port modifier.

```
smpquery -G switchinfo 0x2C9000100D051 1
```

switchinfo by guid.

```
smpquery -D nodeinfo 0
```

nodeinfo by direct route.

```
smpquery -c nodeinfo 6 0,12
```

nodeinfo by combined route.

Smpquery Examples

Link properties between 2 ports.

```
      smpquery portinfo <lid_switch> <real_port_switch>
      smpquery portinfo 21 2
```

```
      # Port info: Lid 21 port 2
      Mkey:...........................0x0000000000000000
      GidPrefix:......................0x0000000000000000
      Lid:............................0
      SMLid:..........................0
      CapMask:........................0x0
      DiagCode:.......................0x0000
      MkeyLeasePeriod:................0
      LocalPort:......................11
      LinkWidthEnabled:................1X or 4X
      LinkWidthSupported:..............1X or 4X
      LinkWidthActive:.................4X
      LinkSpeedSupported:..............2.5 Gbps or 5.0 Gbps or 10.0
Gbps
      LinkState:.......................Active
      PhysLinkState:...................LinkUp
      LinkDownDefState:................Polling
      ProtectBits:.....................0
      LMC:.............................0
      LinkSpeedActive:.................10.0 Gbps
      LinkSpeedEnabled:................2.5 Gbps or 5.0 Gbps or 10.0
Gbps
      NeighborMTU:.....................2048
      SMSL:............................0
      VLCap:...........................VL0-7
      InitType:........................0x00
      VLHighLimit:.....................4
      VLArbHighCap:....................8
      VLArbLowCap:.....................8
      InitReply:.......................0x00
      MtuCap:..........................4096
      VLStallCount:....................7
      HoqLife:.........................16
      OperVLs:.........................VL0-7
      PartEnforceInb:..................1
      PartEnforceOutb:.................1
      FilterRawInb:....................0
      FilterRawOutb:...................0
      MkeyViolations:..................0
      PkeyViolations:..................0
      QkeyViolations:..................0
      GuidCap:.........................0
      ClientReregister:................0
      SubnetTimeout:...................0
      RespTimeVal:.....................0
      LocalPhysErr:....................8
      OverrunErr:......................8
      MaxCreditHint:...................85
      RoundTrip:.......................16777215
```

Check reference equipment corresponding to lid <lid>.

```
smpquery NodeDesc 14
```

```
      Node Description:.MT25408 ConnectX Mellanox Technologies
```

View QoS applied for lid <lid>.

```
smpquery VLArbitration 14
```

```
      # VLArbitration tables: Lid 14 port 0 LowCap 8 HighCap 8
      # Low priority VL Arbitration Table:
      VL    : |0x0 |0x1 |0x2 |0x3 |0x4 |0x5 |0x6 |0x7 |
      WEIGHT: |0x20|0x20|0x20|0x20|0x20|0x20|0x20|0x20|
      # High priority VL Arbitration Table:
      VL    : |0x0 |0x1 |0x2 |0x3 |0x4 |0x5 |0x6 |0x7 |
      WEIGHT: |0x0 |0x0 |0x0 |0x0 |0x0 |0x0 |0x0 |0x0
```

View virtual lanes distribution for lid <lid>.

```
smpquery SL2VLTable <lid>
smpquery SL2VLTable 14
```

```
# SL2VL table: Lid 14
#SL: | 0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|15|
ports: in  0, out  0: | 0| 1| 2| 3| 0| 1| 2| 3| 0| 1| 2| 3| 0| 1| 2| 3|
```

3.5 saquery

saquery Query **InfiniBand** subnet administration attributes.

saquery issues the SA query specified. Node records are queried by default.

Usage

saquery [-h] [-d] [-p] [-N] [–list | -D] [-S] [-I] [-L] [-l] [-G] [-O] [-U] [-c] [-s] [-g] [-m] [-x] [-C ca_name] [-P ca_port] [–smkey val] [-t(imeout <msec>] [–src-to-dst <src:dst>] [–sgid-to-dgid <sgid-dgid>] [–node-name-map <node-name-map>] [<name> | <lid> | <guid>]

Options

-p	Obtain PathRecord info
-N	Obtain NodeRecord info
–list | -D	Obtain NodeDescriptions of CAs only
-S	Obtain ServiceRecord info
-I	Obtain InformInfoRecord (subscription) info
-L	Return the Lids for the name specified
-l	Return the unique Lid for the name specified
-G	Return the Guids of the name specified
-O	Return the name for the Lid specified
-U	Return the name for the Guid specified
-c	Obtain the class port info for the SA
-s	Return the PortInfoRecords with isSM or isSMdisabled capability mask bit on
-g	Obtain multicast group info
-m	Obtain multicast member info. If a group is specified, limit the output to the group specified and print one line containing the GUID and node description for each entry only. Example: saquery -m 0xc000
-x	Obtain LinkRecord info
–src-to-dst	Obtain a PathRecord for **<src:dst>** where **src** and **dst** are either node names or LIDs
–sgid-to-dgid	Obtain a PathRecord for **sgid** to **dgid** where both **GIDs** are in an IPv6 format acceptable to inet_pton(3)
-C <ca_name>	Use the specified **ca_name**
-P <ca_port>	Use the specified **ca_port**
-d	Enable debugging
-h	Show help
–smkey <val>	Use **SM_Key** value for the query. Will be used with '*trusted*' queries only. If non-numeric value (e.g *x*) is specified then **saquery** will prompt for a value.

-t, -timeout <msec> Specify SA query response timeout in milliseconds. Default is 100 milliseconds. You may want to use this option if IB_TIMEOUT is indicated.

–node-name-map <node-name-map> Specify a node name map. The node name map file maps GUIDs to more user friendly names.

Supported query names (and aliases)

ClassPortInfo (CPI)

NodeRecord (NR) [lid]

PortInfoRecord (PIR) [[lid]/[port]]

SL2VLTableRecord (SL2VL) [[lid]/[in_port]/[out_port]]

PKeyTableRecord (PKTR) [[lid]/[port]/[block]]

VLArbitrationTableRecord (VLAR) [[lid]/[port]/[block]]

InformInfoRecord (IIR)

LinkRecord (LR) [[from_lid]/[from_port]] [[to_lid]/[to_port]]

ServiceRecord (SR)

PathRecord (PR)

MCMemberRecord (MCMR)

LFTRecord (LFTR) [[lid]/[block]]

MFTRecord (MFTR) [[mlid]/[position]/[block]]

saquery example

Obtain lid by HCA name on port 1

```
saquery -P1 -l 'zeus19 HCA-1'
```

```
176
```

3.6 perfquery

perfquery Query InfiniBand port counters

perfquery uses PerfMgt GMPs to obtain the PortCounters (basic performance and error counters), PortExtendedCounters, PortXmitDataSL, or PortRcvDataSL from the PMA at the node/port specified. Optionally, shows aggregated counters for all ports of a node. Also, optionally, reset after read, or reset counters only.

Notes In PortCounters, PortCountersExtended, PortXmitDataSL, and PortRcvDataSL, components that represent Data (e.g. PortXmitData and PortRcvData) indicate octets divided by 4 rather than just octets.

Inputting a port of 255 indicates an operation that is to be performed on all ports.

Usage

perfquery [-d(ebug)] [-G(uid)] [-x|–extended] [-X|–xmtsl] [-S|–rcvsl] [-a(ll_ports)] [-l(oop_ports)] [-r(eset_after_read)] [-R(eset_only)] [-C ca_name][-P ca_port] [-t(imeout) timeout_ms] [-V(ersion)] [-h(elp)] [<lid|guid> [[port] [reset_mask]]]

Options

-x, –extended	Show extended port counters rather than (basic) port counters. Note that extended port counters attribute is optional.
-X, –xmtsl	Show transmit data SL counter. This is an optional counter for QoS.
-S, –rcvsl	Show receive data SL counter. This is an optional counter for QoS.
-a, –all_ports	Show aggregated counters for all ports of the destination lid or reset all counters for all ports. If the destination lid does not support the AllPortSelect flag, all ports will be iterated through to emulate AllPortSelect behavior.
-l, –loop_ports	If all ports are selected by the user (either through the -a option or port 255) iterate through each port rather than doing than aggregate operation.
-r, –reset_after_read	Reset counters after they have been read
-R, –Reset_only	Reset counters only

Perfquery examples

```
perfquery
```

Read local port performance counters.

```
perfquery 32 1
```

Read performance counters from lid 32, port 1.

```
perfquery -x 32 1
```

Read extended performance counters from lid 32, port 1.

```
perfquery -a 32
```

Read perf counters from lid 32, all ports.

```
perfquery -r 32 1
```

Read performance counters and reset.

```
perfquery -x -r 32 1
```

Read extended performance counters and reset.

```
perfquery -R 0x20 1
```

Reset performance counters of port 1 only.

```
perfquery -x -R 0x20 1
```

Reset extended performance counters of port 1 only.

```
perfquery -R -a 32
```

Reset performance counters of all ports.

```
perfquery -R 32 2 0x0fff
```

Reset only error counters of port 2.

```
perfquery -R 32 2 0xf000
```

Reset only non-error counters of port 2.

Perfquery examples with output

Enumerate performances counter for a specific link.

```
perquery <lid_switch> <real_port_switch>
perfquery 21 7
```

```
# Port counters: Lid 21 port 7
PortSelect:......................7
CounterSelect:...................0x1b01
SymbolErrors:....................0
LinkRecovers:....................0
LinkDowned:......................0
RcvErrors:.......................0
RcvRemotePhysErrors:.............0
RcvSwRelayErrors:................0
XmtDiscards:.....................1
XmtConstraintErrors:.............0
RcvConstraintErrors:.............0
CounterSelect2:..................0
LinkIntegrityErrors:.............0
ExcBufOverrunErrors:.............0
VL15Dropped:.....................0
XmtData:.........................39504241
RcvData:.........................2826508
```

```
     XmtPkts:........................1240517
     RcvPkts:........................129734
     XmtWait:........................0
```

Print local card counter.

```
perquery -C <ca_name>
perfquery -C mlx4_0
```

```
     # Port counters: Lid 1 port 1
     PortSelect:......................1
     CounterSelect:...................0x1000
     SymbolErrors:....................0
     LinkRecovers:....................0
     LinkDowned:......................0
     RcvErrors:.......................0
     RcvRemotePhysErrors:.............0
     RcvSwRelayErrors:................0
     XmtDiscards:.....................0
     XmtConstraintErrors:.............0
     RcvConstraintErrors:.............0
     CounterSelect2:..................0
     LinkIntegrityErrors:.............0
     ExcBufOverrunErrors:.............0
     VL15Dropped:.....................0
     XmtData:.........................105785576
     RcvData:.........................69146308
     XmtPkts:.........................3356929
     RcvPkts:.........................2245972
     XmtWait:.........................0
```

Reset port performance statistics.

```
perfquery -R <lid_switch> <real_port_switch>
perfquery -R 21 7
perfquery -x 21 7
```

```
     # Port counters: Lid 21 port 7
     PortSelect:......................7
     CounterSelect:...................0x1b01
     PortXmitData:....................0
     PortRcvData:.....................0
     PortXmitPkts:....................0
     PortRcvPkts:.....................0
     PortUnicastXmitPkts:.............129862
     PortUnicastRcvPkts:..............129857
     PortMulticastXmitPkts:...........1112454
     PortMulticastRcvPkts:............35
```

See The chapter on *Troubleshooting InfiniBand Networks* for more details on the port counters.

3.7 sminfo

sminfo Query InfiniBand SMInfo attribute

Optionally set and display the output of a sminfo query in human readable format. The target Subnet Manager is the one listed by the local port info, or the SM specified by the optional SM lid or by the SM direct routed path.

Important Using sminfo for any purposes other then simple queries may be very dangerous, and may result in a malfunction of the target Subnet Manager.

Usage

sminfo [-d(ebug)] [-e(rr_show)] -s state -p prio -a activity [-D(irect)] [-G(uid)] [-C ca_name] [-P ca_port] [-t(imeout) timeout_ms] [-V(ersion)] [-h(elp)] sm_lid | sm_dr_path [modifier]

Options

-s Set SM state

—>0 - not active

—>1 - discovering

—>2 - standby

—>3 - master

-p Set priority (0-15)

-a Set activity count

Example

```
sminfo
```

sminfo for the local port.

Example Output

```
sminfo: sm lid 1 sm guid 0x2c9020024b8a5, activity count 129690
priority 0 state 3 SMINFO_MASTER
```

```
sminfo 32
```

Show **sminfo** for lid 32.

```
sminfo  -G 0x8f1040023
```

Show sminfo using **guid** address.

3.8 ibportstate

ibportstate Handle port (physical) state and link speed of an **InfiniBand** port.

ibportstate allows the port state and port physical state of an InfiniBand port to be checked (in addition to link width and speed being validated relative to the peer port when the port queried is a switch port), or a switch port to be disabled, enabled, or reset. It also allows the link speed enabled on any InfiniBand port to be adjusted.

Usage

ibportstate [-d(ebug)] [-e(rr_show)] [-v(erbose)] [-D(irect)] [-G(uid)] [-s smlid] [-V(ersion)] [-C ca_name] [-P ca_port] [-t(imeout) timeout_ms] [-h(elp)] <dest dr_path|lid|guid> <portnum> [<op>]

Options

The following port operations are supported: **enable**, **disable**, **reset**, **speed**, **query**. The default is **query**, **enable**, **disable**, and **reset** are only allowed on switch ports, an error is indicated if attempted on CA or router ports. The speed operation is allowed on any port.

The speed values are legal values for **PortInfo:LinkSpeedEnabled** (An error is indicated if **PortInfo:LinkSpeedSupported** does not support this setting).

Note Speed changes are not effected until the port goes through link renegotiation.

query also validates the port characteristics (link width and speed) based on the peer port. This checking is done when the port queried is a switch port, as it relies on combined routing (an initial **LID** route with directed routing to the peer) which can only be done on a switch. This peer port validation feature of the **query** operation requires **LID** routing to be functioning in the subnet.

Examples

Disable link between 2 ports:

```
ibportstate <lid_switch> <real_port_switch> disable
ibportstate 21 1 disable        # by lid
```

Enable link by GUID:

```
ibportstate -G 0x2C9000100D051 1 enable # By GUID
```

Enable a link between 2 ports.

```
ibportstate <lid_switch> <real_port_switch> reset
ibportstate 21 1 reset       # by lid
```

Get information for a specific route (lid or 0 if directed route):

```
ibportstate -D <lid_switch> <real_port_switch>
ibportstate -D 0 1      # (query) by direct route
```

Force speed for a specific link:

```
ibportstate <lid_switch> <real_port_switch> speed <rate>
ibportstate 21 1 speed 1 # by lid
```

If there is a hardware failure, the port will fail to negotiate the optimal speed or width. In this situation, it is useful to try and set the appropriate speed using the **ibportstate** tool.

The following speed identifiers can be used to set the link speed:

```
* 1: 2.5 Gbps
* 3: 2.5 or 5.0 Gbps
* 5: 2.5 or 10.0 Gbps
* 7: 2.5 or 5.0 or 10.0 Gbps
* 2,4,6,8-14: Reserved
* Default 15: set to PortInfo:LinkSpeedSupported
```

3.9 ibdiagnet

ibdiagnet InfiniBand diagnostic net.

ibdiagnet scans the fabric using the directed route packets and extracts all the available information regarding its connectivity and devices. It then produces the following files in the output directory (which is defined by the -o option (see below)).

Usage

ibdiagnet [-c <count>] [-v] [-r] [-o <out-dir>] [-t <topo-file>] [-s <sys-name>] [-i <dev-index>] [-p <port-num>] [-wt] [-pm] [-pc] [-P <<PM>=<Value>>] [-lw <1x|4x|12x>] [-ls <2.5|5|10>] [-skip <ibdiag_check/s>] [-load_db <db_file>]

Files

ibdiagnet.log	A dump of all the application reports generate according to the provided flags.
ibdiagnet.lst	List of all the nodes, ports and links in the fabric.
ibdiagnet.fdbs	A dump of the unicast forwarding tables of the fabric switches.
ibdiagnet.mcfdbs	A dump of the multicast forwarding tables of the fabric switches.
ibdiagnet.masks	When there are duplicate port/node GUIDs, these file include the map between masked GUID and real GUIDs.
ibdiagnet.sm	List of all the Subnet Managers (state and priority) in the fabric.
ibdiagnet.pm	A dump of the PM Counters values, of the fabric links.
ibdiagnet.pkey	A dump of the the existing partitions and their member host ports.
ibdiagnet.mcg	A dump of the multicast groups, their properties and member host ports.
ibdiagnet.db	A dump of the internal subnet database. This file can be loaded in later runs using the **-load_db** option.

In addition to generating the files above, the discovery phase also checks for duplicate node/port GUIDs in the **InfiniBand** fabric. If such an error is detected, it is displayed in the standard output. After the discovery phase is completed, directed route packets are sent multiple times (defined by the **-c** option) to detect possible problematic paths on which packets may be lost. Such paths are explored, and a report of the suspected bad links is displayed in the standard output.

After scanning the fabric, if the **-r** option is used, a full report of the fabric qualities is displayed. This report includes:

- Subnet Manage report
- Number of nodes and systems
- Hop-count information: maximal hop-count, an example path, and a hop-count histogram
- All CA-to-CA paths traced
- Credit loop report
- **mgid-mlid-HCAs** multicast group and report

- Partitions report
- IPoIB report

Note When the InfiniBand fabric includes only one Channel Adapter, then CA-to-CA paths are not reported. Furthermore, if a topology file is provided, ibdiagnet uses the names defined in it for the output reports.

Options

-c <count>	Minimum number of packets to be sent across each link (default = 10).
-r	Provide a report of the fabric qualities.
-t <topo-file>	Specify the topology file name.
-s <sys-name>	Specify the local system name. Meaningful only if a topology file is specified.
-i <dev-index>	Specify the index of the device of the port used to connect to the InfiniBand fabric (in case of multiple devices on the local system).
-p <port-num>	Specify the port number for the local device used to connect to the InfiniBand fabric.
-o <out-dir>	Specify the directory where the output files will be placed (default = /tmp)
-lw <1x|4x|12x>	Specify the expected link width
-ls <2.5|5|10>	Specify the expected link speed
-pm	Dump all the fabric links, pm Counters into **ibdiagnet.pm**
-pc:	Reset all the fabric links pmCounters
-P <PM=<Trash>>	If any of the provided Performance Monitor (counter) are greater then their provided value, print it to screen
-skip <skip-option(s)>	Skip the executions of the selected checks. Skip options (one or more can be specified): **dup_guids zero_guids pm logical_state part ipoib all**
-wt <file-name>>	Copy the topology discovered into the given file. This flag is useful if you later want to check for changes in the state of the fabric. A directory named **ibdiag_ibnl** is also created by this option, and holds the **IBNL** files required to load this topology. To use these files you will need to set the environment variable named **IBDM_IBNL_PATH** to that directory. The directory is located in **/tmp** or in the **output** directory provided by the **-o** flag. **-load_db <file-name>>**: Load subnet data from the given **.db** file, and skip subnet discovery stage.

Note Some of the checks require actual subnet discovery, and therefore would not run when load_db is specified. These checks are: Duplicated/zero guids, link state, SMs status.

-h|–help	Prints the help page information
-V|–version	Prints the version of the tool

–vars	Prints the tools environment variables and their values

Errors Codes

1 - Failed to fully discover the fabric
2 - Failed to parse command line options
3 - Failed to intract with IB fabric
4 - Failed to use local device or local port
5 - Failed to use Topology File
6 - Failed to load required Package

3.9.1 Examples of use of Ibdiagnet

Execute standard scanning:

```
ibdiagnet
```

```
Loading IBDIAGNET from: /usr/lib64/ibdiagnet1.2
-W- Topology file is not specified.
Reports regarding cluster links will use direct routes.
Loading IBDM from: /usr/lib64/ibdm1.2
-I- Using port 1 as the local port.
-I- Discovering ... 17 nodes (3 Switches & 14 CA-s) discovered.
-I---------------------------------------------------
-I- Bad Guids/LIDs Info
-I---------------------------------------------------
-I- No bad Guids were found
-I---------------------------------------------------
-I- Links With Logical State = INIT
-I---------------------------------------------------
-I- No bad Links (with logical state = INIT) were found
-I---------------------------------------------------
-I- PM Counters Info
-I---------------------------------------------------
-I- No illegal PM counters values were found
-I---------------------------------------------------
-I- Fabric Partitions Report (see ibdiagnet.pkey for a full hosts list)
-I---------------------------------------------------
-I-    PKey:0x0001 Hosts:14 full:14 partial:0
-I-    PKey:0x7fff Hosts:14 full:14 partial:0
-I---------------------------------------------------
-I- IPoIB Subnets Check
-I---------------------------------------------------
-I- Subnet: IPv4 PKey:0x7fff QKey:0x00000b1b MTU:2048Byte rate:10Gbps SL:0x00
-W- Suboptimal rate for group. Lowest member rate:40Gbps > group-rate:10Gbps
-I- Subnet: IPv4 PKey:0x0001 QKey:0x00000b1b MTU:2048Byte rate:10Gbps SL:0x00
-W- Suboptimal rate for group. Lowest member rate:40Gbps > group-rate:10Gbps
-I---------------------------------------------------
-I- Bad Links Info
-I---------------------------------------------------
-I- No bad link were found
----------------------------------------------------------------
-I- Stages Status Report:
    STAGE                                   Errors Warnings
    Bad GUIDs/LIDs Check                    0      0
    Link State Active Check                 0      0
    Performance Counters Report             0      0
    Partitions Check                        0      0
    IPoIB Subnets Check                     0      2
Please see /tmp/ibdiagnet.log for complete log
----------------------------------------------------------------
```

3.10 ibcheckerrors

ibcheckerrors   Validate IB subnet and report errors

ibcheckerrors is a script which uses a full topology file created by **ibnetdiscover** to scan the network to validate the connectivity and errors reported (from port counters).

Usage

ibcheckerrors [-h] [-b] [-v] [-N | -nocolor] [<topology-file> | -C ca_name -P ca_port -t(imeout) timeout_ms]

Options

-v	Increase the verbosity level
-b: brief mode	Reduce the output to show only if errors are present, not what they are.
-N | -nocolor	Use mono rather than color mode
-C <ca_name>	Use the specified **ca_name**
-P <ca_port>	Use the specified **ca_port**
-t <timeout_ms>	Override the default timeout for the solicited MADs

Example

Check network topology current status:

```
ibcheckerrors -b
```

```
Error check on lid 11 (Infiniscale-IV Mellanox Technologies) port all:  FAILED
Error check on lid 11 (Infiniscale-IV Mellanox Technologies) port 1:  FAILED
Error check on lid 21 (Infiniscale-IV Mellanox Technologies) port all:  FAILED
Error check on lid 21 (Infiniscale-IV Mellanox Technologies) port 4:  FAILED

## Summary: 17 nodes checked, 0 bad nodes found
##          34 ports checked, 2 ports have errors beyond threshold
```

3.11 ibchecknet

ibchecknet Validate InfiniBand subnet and report errors

ibchecknet is a script which uses a full topology file that was created by ibnetdiscover, and scans the network to validate the connectivity and reports errors (from port counters).

Usage

ibchecknet [-h] [-N | -nocolor] [<topology-file> | -C ca_name -P ca_port -t(imeout) timeout_ms]

Options

-N | -nocolor	Use mono rather than color mode
-C <ca_name>	Use the specified ca_name
-P <ca_port>	Use the specified ca_port
-t <timeout_ms>	Override the default timeout for the solicited MADs

Example

Check network topology current status:

```
ibchecknet
```

```
#warn: counter SymbolErrors = 65535  (threshold 10) lid 11 port 255
#warn: counter RcvSwRelayErrors = 110   (threshold 100) lid 11 port 255
Error check on lid 11 (Infiniscale-IV Mellanox Technologies) port all:  FAILED
#warn: Lid is not configured lid 11 port 5
#warn: SM Lid is not configured
Port check lid 11 port 5:  FAILED
# Checked Switch: nodeguid 0x0002c90200404798 with failure
#warn: Lid is not configured lid 11 port 4
#warn: SM Lid is not configured
Port check lid 11 port 4:  FAILED
#warn: Lid is not configured lid 11 port 3
#warn: SM Lid is not configured
Port check lid 11 port 3:  FAILED
#warn: Lid is not configured lid 11 port 2
#warn: SM Lid is not configured
Port check lid 11 port 2:  FAILED
#warn: counter RcvSwRelayErrors = 110   (threshold 100) lid 11 port 1
Error check on lid 11 (Infiniscale-IV Mellanox Technologies) port 1:  FAILED
#warn: Lid is not configured lid 11 port 1
#warn: SM Lid is not configured
Port check lid 11 port 1:  FAILED
#warn: counter SymbolErrors = 65535  (threshold 10) lid 21 port 255
#warn: counter LinkDowned = 255 (threshold 10) lid 21 port 255
#warn: counter RcvSwRelayErrors = 974   (threshold 100) lid 21 port 255
Error check on lid 21 (Infiniscale-IV Mellanox Technologies) port all:  FAILED
#warn: Lid is not configured lid 21 port 7
#warn: SM Lid is not configured
Port check lid 21 port 7:  FAILED
# Checked Switch: nodeguid 0x0002c902004047c0 with failure
#warn: Lid is not configured lid 21 port 6
#warn: SM Lid is not configured
Port check lid 21 port 6:  FAILED
#warn: counter RcvSwRelayErrors = 974   (threshold 100) lid 21 port 4
Error check on lid 21 (Infiniscale-IV Mellanox Technologies) port 4:  FAILED
#warn: Lid is not configured lid 21 port 4
#warn: SM Lid is not configured
```

```
Port check lid 21 port 4:  FAILED
#warn: Lid is not configured lid 21 port 3
#warn: SM Lid is not configured
Port check lid 21 port 3:  FAILED
#warn: Lid is not configured lid 21 port 2
#warn: SM Lid is not configured
Port check lid 21 port 2:  FAILED
#warn: Lid is not configured lid 21 port 11
#warn: SM Lid is not configured
Port check lid 21 port 11:  FAILED
#warn: Lid is not configured lid 17 port 9
#warn: SM Lid is not configured
Port check lid 17 port 9:  FAILED
# Checked Switch: nodeguid 0x0002c902004044e0 with failure
#warn: Lid is not configured lid 17 port 8
#warn: SM Lid is not configured
Port check lid 17 port 8:  FAILED
#warn: Lid is not configured lid 17 port 7
#warn: SM Lid is not configured
Port check lid 17 port 7:  FAILED
#warn: Lid is not configured lid 17 port 4
#warn: SM Lid is not configured
Port check lid 17 port 4:  FAILED
#warn: Lid is not configured lid 17 port 3
#warn: SM Lid is not configured
Port check lid 17 port 3:  FAILED
#warn: Lid is not configured lid 17 port 2
#warn: SM Lid is not configured
Port check lid 17 port 2:  FAILED
#warn: Lid is not configured lid 17 port 1
#warn: SM Lid is not configured
Port check lid 17 port 1:  FAILED
#warn: Lid is not configured lid 17 port 12
#warn: SM Lid is not configured
Port check lid 17 port 12:  FAILED
#warn: Lid is not configured lid 17 port 11
#warn: SM Lid is not configured
Port check lid 17 port 11:  FAILED
# Checking Ca: nodeguid 0x0002c90300025e9a
# Checking Ca: nodeguid 0x0002c903000262ba
# Checking Ca: nodeguid 0x0002c90300025eb2
# Checking Ca: nodeguid 0x0002c903000262a2
# Checking Ca: nodeguid 0x0002c90300026316
# Checking Ca: nodeguid 0x0002c90300025ea2
# Checking Ca: nodeguid 0x0002c90300025ea6
# Checking Ca: nodeguid 0x0002c90300025f72
# Checking Ca: nodeguid 0x0002c90300025f82
# Checking Ca: nodeguid 0x0002c90300025e9e
# Checking Ca: nodeguid 0x0002c90300025f7a
# Checking Ca: nodeguid 0x0002c90300025e82
# Checking Ca: nodeguid 0x0002c90300025e26
# Checking Ca: nodeguid 0x0002c903000262da
## Summary: 17 nodes checked, 0 bad nodes found
##          34 ports checked, 20 bad ports found
##          2 ports have errors beyond threshold
```

3.12 IBS

IBS is an InfiniBand diagnostic tool aimed at troubleshooting InfiniBand (IB) fabrics. It can also be used to configure InfiniBand switches and retrieve topology related information.

IBS can use up to five different data sources when checking a fabric:

- A network map that is supplied online by a managed Voltaire InfiniBand switch.

- A network map that has been retrieved from a managed Voltaire InfiniBand switch and saved to a file.
- A topology file produced by the ibnetdiscover program on a live system.
- A topology file produced by the ibnetdiscover program and saved to a file.
- The IBS database.

3.12.1 Synopsis

IBS supports the following options:

```
ibs -a <action> -S <datasource>
        [-s <switch>] [-n <networkmap>] [-c <counters>] [-i <topofile>]
        [-o <output>] [-l <lft>]
        [-x <port>]
        [-hvCNERX]
```

IBS obtains its data from a variety of sources within the cluster. Basically, there are two disctinct types of data: the topology information that describes how the equipment is interconnected, and the Traffic & Error Counters.

3.12.2 Topology data

In order to perform an action, IBS needs data (network map). Data can be provided by tools like smpquery, ibnetdiscover, an output switch xml file, etc. The following data source combinations, either offline (using the results of a command saved in a file) or online (running the command directly on a live system), of data sources are possible for IBS.

Note These data source combinations deal with the topology issues only and exclude the traffic and error counters.

3.12.2.1 Switch Network Map

IBS queries the switch specified by the -s option and retrieves (downloads) the topology information from the switch in XML format. This topology file is referred to as a *'network map'* using Voltaire terminology. This data source is typically used on a live system fitted with one of more Voltaire managed switch(es).

Advantage

This data source does not require the node on which IBS is invoked to be connected to the InfiniBand fabric.

Drawbacks

- The topology in the network map is often inaccurate regarding hostnames, and this prevents the retrieval of the localisation for these nodes from the cluster database. Many nodes are named after the type of HCA(s) they are fitted with, e.g. MT25218. It is advised to use the -N option to get proper hostnames in this case (this option requires the node on which IBS is invoked to be connected to the InfinBand fabric).
- The OpenSM Subnet Manager does not provide a network map, since the network map is generated by a proprietary Voltaire program. Consequently, this data source cannot be used in conjunction with OpenSM.

- The queried switch must be the subnet manager master. Querying any switch that is not running as subnet manager master will produce an empty network map. To determine which switch is running as the master Subnet Manager, the **sminfo** command should be used as follows:

```
sminfo
```

```
sminfo: sm lid 1 sm guid 0x8f1040041254a, activity count 544113
priority 3 state 3 SMINFO_MASTER
```

However, **sminfo** only indicates the GUID of the ASIC hosting the Subnet Manager and not its associated hostname. This is why the **ibnetdiscover** data source was introduced in **IBS** >= 0.3.X.

In the example below, the **NetworkMap.xml** and **PortCounters.csv** files are downloaded from the **switchname** switch and saved locally with the same names in the working directory:

```
ibs -S switch -s switchname -a topo
```

Note The **-n** flag allows the file name of the network map that is saved to disk to be specified, once downloaded from the specified switch. **IBS** will then read the topology information from this file. **IBS** only supports network maps generated by managed **Voltaire** switches, whose firmware versions is earlier than 5.0.

In the example below, **IBS** downloads the **NetworkMap.xml** file from the managed **switchname** switch and saves it as **/path/to/mynetworkmap.xml**. No portcounters data was requested and so the traffic and error counters are not available:

```
ibs -S switch -s switchname -n /path/to/mynetworkmap.xml -a topo
```

Note The managed Voltaire switches provide an XML based description of the InfiniBand fabric.

This **NetworkMap.xml** file can be viewed in a browser using the address: http://<switchname>/NetworkMap.xml

Alternatively, it can be downloaded by using the command:

```
wget http://<switchname>/NetworkMap.xml
```

3.12.2.2 xmlfile

This data source is also based upon a network map file, the difference being that instead of downloading the network map file from the **Voltaire** managed switch as is the case for the switch data source, it uses a network map that has already been downloaded from the switch. This mode is typically used for off-site analysis.

Note The network map file must be a valid one, i.e. downloaded from a switch running as the subnet manager master.

Advantages

This data source does not require the node on which IBS is invoked to be connected to the InfiniBand fabric.

Drawbacks

The topology in the network map is often inaccurate regarding hostnames, and this prevents the retrieval of the localisation for these nodes from the cluster database. Many nodes are named after the type of HCA(s) they are fitted with, e.g. MT25218. It is advised to use the -N option to get proper hostnames in this case (this option requires the node on which IBS is invoked to be connected to the InfinBand fabric, which may not apply for off-site analysis).

- The OpenSM Subnet Manager does not provide a network map, since the network map is generated by a proprietary Voltaire program. Consequently, this data source cannot be used in conjunction with OpenSM.

In the example below, no specific network map file, and therefore IBS defaults to NetworkMap.xml in the current working directory. No portcounters data was requested and so the traffic and error counters are not available:

```
ibs -S xmlfile -a topo
```

Note The -n flag specifies the file name of the network map that IBS should read its data from.

In the example below, IBS uses the /path/to/mynetworkmap.xml network map file. No portcounters data was requested and so the traffic and error counters are not available:

```
ibs -S xmlfile -n /path/to/mynetworkmap.xml -a topo
```

3.12.2.3 ibnetdiscover topology file

IBS invokes the ibnetdiscover program to retrieve the network topology. This program is part of the InfiniBand-diags (GPL) bundle.

Advantages

ibnetdiscover discovers the network topology automatically and does not need the end user to provide IBS with a switch hostname. This makes it easy to add to a crontab, as it does not require any hardcoded option for the switch hostname.
ibnetdiscover being GPL, one does not need the network to be fitted with a Voltaire managed switch.

Drawbacks

ibnetdiscover requires the node on which IBS is invoked to be connected to the InfiniBand fabric.

- IBS uses the grouping feature of ibnetdiscover, specified by the -g flag (this feature groups ASICs into boards, and boards into chassis). Older versions of ibnetdiscover did not have such as feature, or in some cases, it was not fully functional for OFED releases prior to 1.2.

In the example below, IBS invokes the ibnetdiscover program and saves its output to the default topology file name ibnetdiscover.out:

```
ibs -S ibnetdiscover -a errors
```

Note The -i flag allows specifies the file name of the topology file that is saved to disk, once it has been generated by the **ibnetdiscover** program. IBS will then read the topology information from this file.

In the example below, IBS generates the /path/to/mytopofile.out topology file. No portcounters data was requested and so the traffic and error counters are not available:

```
ibs -S ibnetdiscover -i path/to/mytopofile.out -a topo
```

3.12.2.4 topofile

This data source is also based upon an **ibnetdiscover** topology file, the difference being that instead of invoking **ibnetdiscover** on a live system, it uses the output that was generated by **ibnetdiscover** earlier on. This mode is typically used for off site analysis.

Advantages

ibnetdiscover being GPL, one does not need the network to be fitted with a Voltaire managed switch.

Drawbacks

- Requires the node on which **ibnetdiscover** was invoked to generate the topology file to be connected to the InfiniBand fabric.

IBS uses the grouping feature of **ibnetdiscover**, specified by the -g flag (this feature groups ASICs into boards, and boards into chassis). Older versions of **ibnetdiscover** did not include this feature, or in some cases, it was not fully functional for OFED releases prior to 1.2. Make sure that the topology file supplied to IBS was generated with this option enabled in the appropriate OFED environment.

In the example below, no specific topology file is specified, and IBS defaults to ibnetdiscover.out in the current working directory. No portcounters data was requested and so the traffic and error counters are not available:

```
ibs -S topofile -a topo
```

Note The -i flag allows specifies the file name of the topology file that is saved to disk, once it has been generated by the **ibnetdiscover** program. IBS will then read the topology information from this file.

In the example below, IBS generates the /path/to/mytopofile.out topology file. No portcounters data was requested and so the traffic and error counters are not available:

```
ibs -S ibnetdiscover -i path/to/mytopofile.out -a topo
```

3.12.3 Traffic and error counters

IBS can either get its traffic and error counters from the PortCounters.csv file or the **perfquery** program (both being mutually exclusive).

Important The InfiniBand specification defines the traffic port counters as 32 bit objects. However, as InfiniBand is a high bandwidth network, these 32 bit objects overflow almost immediately. Prior to any analysis, it is highly advisable to reset the port counters as described in the Monitoring InfiniBand Networks section in Chapter 4.

3.12.3.1 Using the PortCounters.csv file

The managed **Voltaire** switches provide the port counters as a comma separated value file which is easy to open as a spreadsheet for further analysis. This file is named **PortCounters.csv** and can be viewed by pointing a browser to: **http://<switchname>/PortCounters.csv**

Alternatively, it can be downloaded using the command below:

```
wget http://<switchname>/PortCounters.csv
```

Note The **PortCounters.csv** information is often out of date. Unless there is a good reason to use it (e.g. off site analysis), the **perfquery** mode is preferred for all running **InfiniBand** clusters.

IBS usage with the **-c** option specifies a specific port counters file:

```
ibs -S xmlfile -c /path/to/myportcounters.csv -a topo
```

3.12.3.2 Using perfquery

The **perfquery** tool uses the **InfiniBand** network and issues queries to the Subnet Manager to retrieve traffic and error counters.

```
perfquery 0x8 1
```

```
# Port counters: Lid 8 port 1
PortSelect:......................1
CounterSelect:...................0x0100
SymbolErrors:....................0
LinkRecovers:....................0
LinkDowned:......................0
RcvErrors:.......................0
RcvRemotePhysErrors:.............0
RcvSwRelayErrors:................0
XmtDiscards:.....................0
XmtConstraintErrors:.............0
RcvConstraintErrors:.............0
LinkIntegrityErrors:.............0
ExcBufOverrunErrors:.............0
VL15Dropped:.....................0
XmtBytes:........................6548782
RcvBytes:........................2809731637
XmtPkts:.........................98152
RcvPkts:.........................8435697
```

See Section 3.6 for more information on **perfquery**.

Use ibs with the **-E** option to query the Performance Manager directly to retrieve the traffic and error counters as follows:

```
ibs -S ibnetdiscover -E -a topo
```

Notes The performance manager must be enabled on the managed switch. Please read the switch manual for additional information.

The PortCounters.csv and perfquery sections are not applicable for the db data source, as IBS reads its data directly from the database in this situation. However, they are valid for all versions of IBS and all other data sources.

See The chapter on *Troubleshooting InfiniBand Networks* for more details on the traffic and error port counters.

3.12.4 Naming related issues

The NetworkMap.xml topology file is often inaccurate with reference to hostnames. It is strongly advised to use the smpquery based analysis whenever possible, i.e. when the node on which IBS is invoked is connected to a functioning InfiniBand network. To do so, proceed as follows:

```
ibs -S switch -s switchname -N -a topo
```

The -N option conflicts with the ibnetdiscover or topofile data sources. It does not make sense to use this option with these data sources because ibnetdiscover uses the same mechanism to retrieve the hostnames.

3.12.5 General purpose options

IBS supports the following general purpose options:

-h	Display usage information.
-v	Verbose mode.
-C	Disable coloured output to avoid clogging the screen with control characters on some terminals.

3.12.6 Actions

IBS performs the following actions, when they are specified by the -a flag. 0 is returned to indicate success for the action; any other value indicates a failure.

3.12.6.1 Discovery / Troubleshooting Related Actions

topo	Dump the network topology
bandwidth	Show the bandwidth figures
errors	Produce a short report detailing the faulty links

3.12.6.2 Switch Configuration Related Actions

config	Generate the instruction sequence needed to configure the hostname mapping in a managed switch (Voltaire switches with version 4.0 or later firmware only).

group	Generate the **group.csv** file needed to to configure the hostname mapping in a managed switch (**Voltaire** switches with version 4.0 or later firmware only).

3.12.6.3 Database Related Actions

dbcreate	Create an empty **IBS** database (**ibsdb**). Note that only the **postgres** user is allowed to create an empty database
dbdelete	Delete the **IBS** database. Note that only the **postgres** user is allowed to delete a database
dbpopulate	Populate the **IBS** database
dbupdate	Update the **IBS** database
dbupdatepc	Update the **IBS** database (portcounters only)

3.12.6.4 Utilities

availability	Show the interconnect availability. Note that this mode assumes that the **IBS** database has been created and populated.
showspines	Show the spines that are currently configured for the Subnet Manager.

3.12.6.5 Database Command Examples

Create a new IBS database (default name is ibsdb):

```
postgres@admin$ ibs -S db -a dbcreate
```

Populate this new database using the data supplied by the **ibnetdiscover** program:

```
ibs -S ibnetdiscover -a dbpopulate -E
```

Update the current database using the data supplied by the *ibnetdiscover* program:

```
ibs -S ibnetdiscover -a dbupdate -E
```

Once the **IBS** database has been created, the interconnect availability is displayed from the database hardware, as follows:

```
ibs -S db -a availability
```

Dump the fabric topology directly from the switch **iswu0c0-0**. The hostnames and traffic counters are retrieved using **OFED** tools:

```
ibs -S switch -a topo -s iswu0c0-0 -NE
```

Dump the fabric topology using the data stored in the **IBS** database.

```
ibs -S db -a topo
```

Dump the fabric topology using the local map file **test/NetworkMap.xml** and port counters **test/portcounters.csv**:

```
ibs -S xmlfile -n test/NetworkMap.xml -c test/portcounters.csv -a topo
```

Dump the fabric traffic and error counters using the data stored in the **IBS** database:

```
ibs -S db -a bandwidth
```

Show faulty links using the data stored in the **IBS** database:

```
ibs -S db -a errors
```

3.12.7 IBS output

As the **IBS** tool displays a wealth of information, it is advisable to use a 19 inches screen monitor with a small font. The performance of Windows terminals is too limited to display the **IBS** output properly.

3.12.7.1 Topology

A typical ibs topology output consists of:

- A switch header which details the description of the switch, its hostname, node **GUID**, lid and location.

A local/remote banner that separates the information seen on both ends of the cable. The **local** side corresponds to the end of the cable that is connected to the switch described in the switch header, while the **remote** side corresponds to the other end of the cable (may be a HCA or another switch). The separator is a double pipe (||) sign.
A connection header that lists all the information that is required to troubleshoot the InfiniBand network on both ends of the cable (GUIDs, LIDS, link width, link speed), as well as some localisation information to point the end user to the appropriate node or switch.

The cable that connects the switch to another piece of equipment should be considered as a black box with an input and an output: The data fed on one end of the cable is not necessarily the same as the data that comes out of it if the cable is faulty. This is why there are two types of errors, **local errors** and **remote errors**. Also, both piece of connected equipment do not necessarily see the same events on both ends of the cable: A switch that remains powered up while a node is rebooted sees the rebooted node generating errors, while the node being rebooted loses its connection to the **InfiniBand** fabric and does not see anything.

Typical output for a **Voltaire ISR 9024** switch:

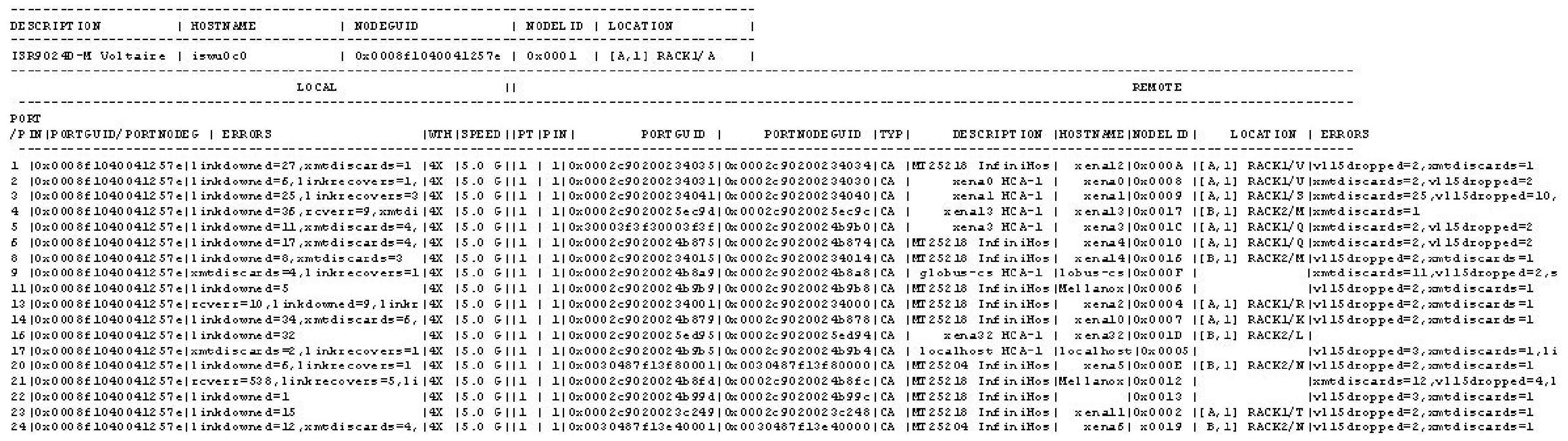
```
ibs -S ibnetdiscover -a topo -E
```

Chassis based switches are printed out in a fairly similar manner:

- A chassis header with the same fields that describe the switch chassis

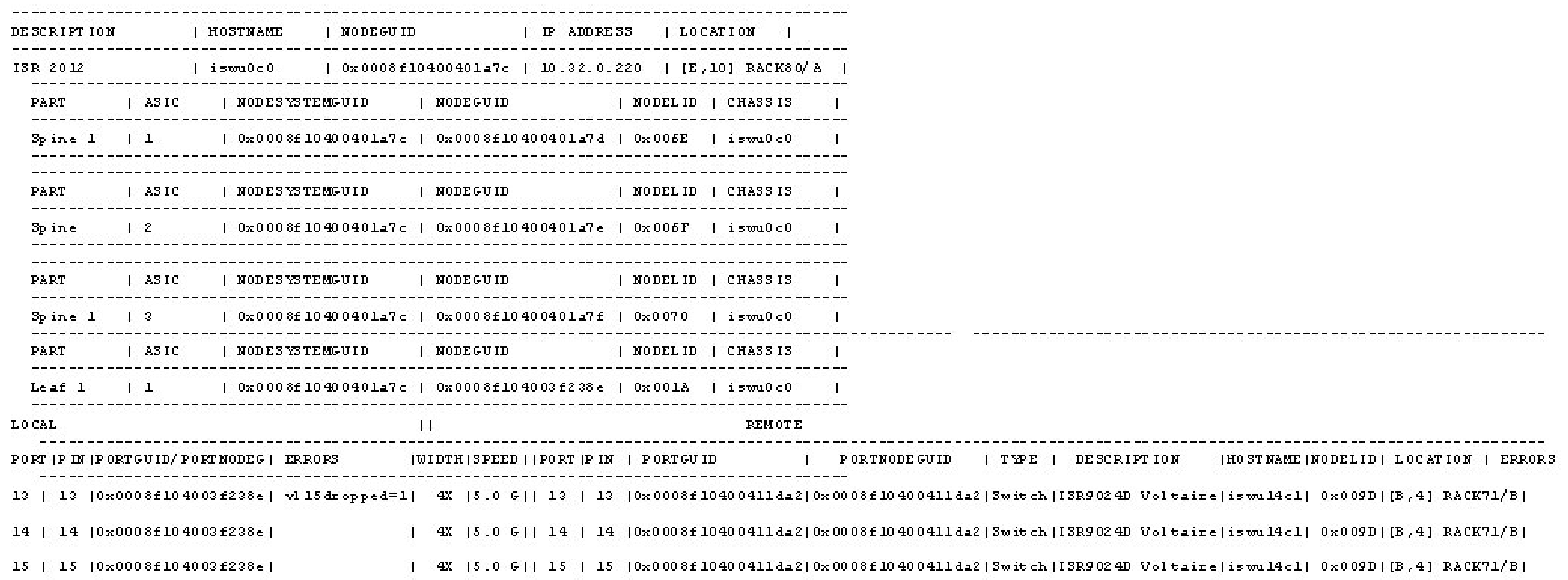
A Spine or Leaf ASIC header which describes the characteristics of each ASIC for each board fitted into the chassis. Note that the internal switch (midplane) connections are not displayed if they are not faulty.
The local/remote header is preserved.

Note The **port** field describes the socket port number that is printed on the board, while the **pin** field relates to the actual ASIC pin. When troubleshooting a system, the end user is interested in which port is faulty. However, the **OFED** tools cannot know how the ports were laid out by the manufacturer and use the actual pin number.



```
------------------------------------------------------------------------------------
DESCRIPTION          | HOSTNAME           | NODEGUID            | NODEL ID | LOCATION         |
------------------------------------------------------------------------------------
ISR9024D-M Voltaire  | iswu0c0            | 0x0008f1040041257e  | 0x0001   | [A,1] RACK1/A    |
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
                                 LOCAL                               ||                                                                       REMOTE
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
PORT
/PIN|PORTGUID/PORTNODEG | ERRORS                        |WTH|SPEED||PT|PIN|       PORTGUID  |     PORTNODEGUID |TYP|      DESCRIPTION |HOSTNAME|NODEL ID|    LOCATION  | ERRORS
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
1  |0x0008f1040041257e|linkdowned=27,xmtdiscards=1   |4X |5.0 G||1 |  1|0x0002c90200234035|0x0002c90200234034|CA |MT25218 InfiniHos|  xena12|0x000A |[A,1] RACK1/V|vl15dropped=2,xmtdiscards=1
2  |0x0008f1040041257e|linkdowned=6,linkrecovers=1,  |4X |5.0 G||1 |  1|0x0002c90200234031|0x0002c90200234030|CA |      xena0 HCA-1 |   xena0|0x0008 |[A,1] RACK1/U|xmtdiscards=2,vl15dropped=2
3  |0x0008f1040041257e|linkdowned=25,linkrecovers=3  |4X |5.0 G||1 |  1|0x0002c90200234041|0x0002c90200234040|CA |      xena1 HCA-1 |   xena1|0x0009 |[A,1] RACK1/S|xmtdiscards=25,vl15dropped=10,
4  |0x0008f1040041257e|linkdowned=36,rcverr=9,xmtdi  |4X |5.0 G||1 |  1|0x0002c9020025ec9d|0x0002c9020025ec9c|CA |     xena13 HCA-1 |  xena13|0x0017 |[B,1] RACK2/M|xmtdiscards=1
5  |0x0008f1040041257e|linkdowned=11,xmtdiscards=4,  |4X |5.0 G||1 |  1|0x30003f3f30003f3f|0x0002c9020024b9b0|CA |      xena3 HCA-1 |   xena3|0x001C |[A,1] RACK1/Q|xmtdiscards=2,vl15dropped=2
6  |0x0008f1040041257e|linkdowned=17,xmtdiscards=4,  |4X |5.0 G||1 |  1|0x0002c9020024b875|0x0002c9020024b874|CA |MT25218 InfiniHos|   xena4|0x0010 |[A,1] RACK1/Q|xmtdiscards=2,vl15dropped=2
8  |0x0008f1040041257e|linkdowned=8,xmtdiscards=3    |4X |5.0 G||1 |  1|0x0002c90200234015|0x0002c90200234014|CA |MT25218 InfiniHos|  xena14|0x0016 |[B,1] RACK2/M|vl15dropped=2,xmtdiscards=1
9  |0x0008f1040041257e|xmtdiscards=4,linkrecovers=1  |4X |5.0 G||1 |  1|0x0002c9020024b8a9|0x0002c9020024b8a8|CA | globus-cs HCA-1 |lobus-cs|0x000F |             |xmtdiscards=11,vl15dropped=2,s
11 |0x0008f1040041257e|linkdowned=5                  |4X |5.0 G||1 |  1|0x0002c9020024b9b9|0x0002c9020024b9b8|CA |MT25218 InfiniHos|Mellanox|0x0006 |             |vl15dropped=2,xmtdiscards=1
13 |0x0008f1040041257e|rcverr=10,linkdowned=9,linkr  |4X |5.0 G||1 |  1|0x0002c90200234001|0x0002c90200234000|CA |MT25218 InfiniHos|   xena2|0x0004 |[A,1] RACK1/R|vl15dropped=2,xmtdiscards=1
14 |0x0008f1040041257e|linkdowned=34,xmtdiscards=6,  |4X |5.0 G||1 |  1|0x0002c9020024b879|0x0002c9020024b878|CA |MT25218 InfiniHos|  xena10|0x0007 |[A,1] RACK1/K|vl15dropped=2,xmtdiscards=1
16 |0x0008f1040041257e|linkdowned=32                 |4X |5.0 G||1 |  1|0x0002c9020025ed95|0x0002c9020025ed94|CA |     xena32 HCA-1 |  xena32|0x001D |[B,1] RACK2/L|
17 |0x0008f1040041257e|xmtdiscards=2,linkrecovers=1  |4X |5.0 G||1 |  1|0x0002c9020024b9b5|0x0002c9020024b9b4|CA | localhost HCA-1 |localhost|0x0005|             |vl15dropped=3,xmtdiscards=1,li
20 |0x0008f1040041257e|linkdowned=6,linkrecovers=1   |4X |5.0 G||1 |  1|0x0030487f13f80001|0x0030487f13f80000|CA |MT25204 InfiniHos|   xena5|0x000E |[B,1] RACK2/N|vl15dropped=2,xmtdiscards=1
21 |0x0008f1040041257e|rcverr=538,linkrecovers=5,li  |4X |5.0 G||1 |  1|0x0002c9020024b8fd|0x0002c9020024b8fc|CA |MT25218 InfiniHos|Mellanox|0x0012 |             |xmtdiscards=12,vl15dropped=4,l
22 |0x0008f1040041257e|linkdowned=1                  |4X |5.0 G||1 |  1|0x0002c9020024b99d|0x0002c9020024b99c|CA |MT25218 InfiniHos|        |0x0013 |             |vl15dropped=3,xmtdiscards=1
23 |0x0008f1040041257e|linkdowned=15                 |4X |5.0 G||1 |  1|0x0002c9020023c249|0x0002c9020023c248|CA |MT25218 InfiniHos|  xena11|0x0002 |[A,1] RACK1/T|vl15dropped=2,xmtdiscards=1
24 |0x0008f1040041257e|linkdowned=12,xmtdiscards=4,  |4X |5.0 G||1 |  1|0x0030487f13e40001|0x0030487f13e40000|CA |MT25204 InfiniHos|   xena6| x0019 | B,1] RACK2/N|vl15dropped=2,xmtdiscards=1
```

Figure 3-1. Example of Topology Output for a ISR9024 switch

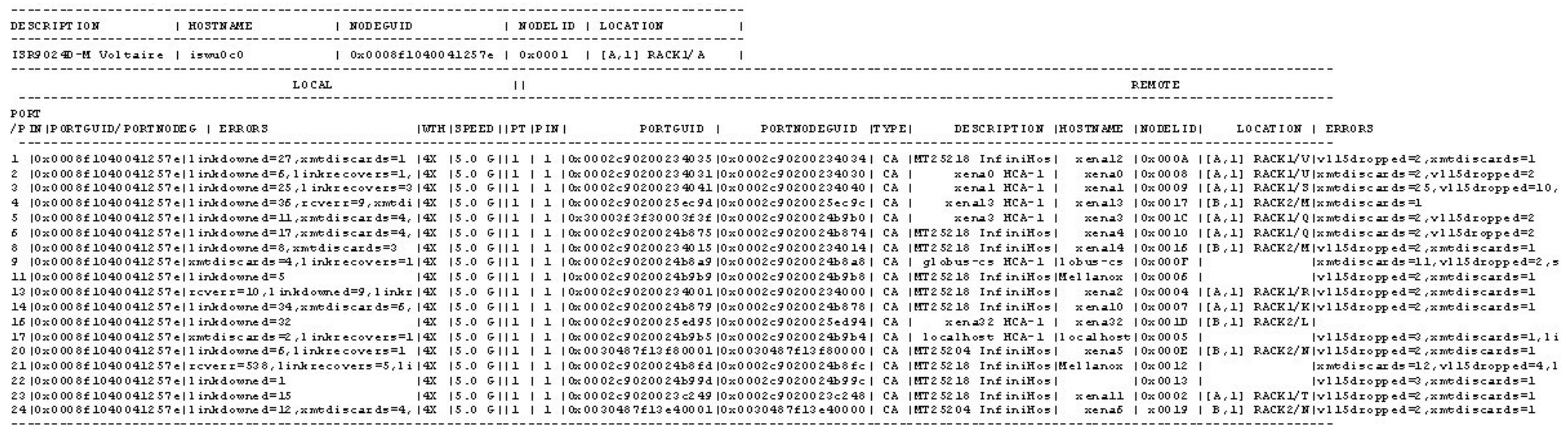


```
-------------------------------------------------------------------------------------------
DESCRIPTION        | HOSTNAME    | NODEGUID           | IP ADDRESS    | LOCATION    |
-------------------------------------------------------------------------------------------
ISR 2012           | iswu0c0     | 0x0008f10400401a7c | 10.32.0.220   | [E,10] RACK80/A  |
  -----------------------------------------------------------------------------------------
  PART       | ASIC     | NODESYSTEMGUID      | NODEGUID            | NODELID  | CHASSIS    |
  -----------------------------------------------------------------------------------------
  Spine 1    | 1        | 0x0008f10400401a7c  | 0x0008f10400401a7d  | 0x006E   | iswu0c0    |
  -----------------------------------------------------------------------------------------
  -----------------------------------------------------------------------------------------
  PART       | ASIC     | NODESYSTEMGUID      | NODEGUID            | NODELID  | CHASSIS    |
  -----------------------------------------------------------------------------------------
  Spine      | 2        | 0x0008f10400401a7c  | 0x0008f10400401a7e  | 0x006F   | iswu0c0    |
  -----------------------------------------------------------------------------------------
  -----------------------------------------------------------------------------------------
  PART       | ASIC     | NODESYSTEMGUID      | NODEGUID            | NODELID  | CHASSIS    |
  -----------------------------------------------------------------------------------------
  Spine 1    | 3        | 0x0008f10400401a7c  | 0x0008f10400401a7f  | 0x0070   | iswu0c0    |
  -----------------------------------------------------------------------------------------------------------   -------------------------------------------------------------
  PART       | ASIC     | NODESYSTEMGUID      | NODEGUID            | NODELID  | CHASSIS    |
  -----------------------------------------------------------------------------------------
  Leaf 1     | 1        | 0x0008f10400401a7c  | 0x0008f104003f238e  | 0x001A   | iswu0c0    |
  -----------------------------------------------------------------------------------------
LOCAL                                    ||                                       REMOTE
  ------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
PORT |PIN |PORTGUID/PORTNODEG| ERRORS         |WIDTH|SPEED||PORT |PIN  | PORTGUID           |   PORTNODEGUID     | TYPE |  DESCRIPTION     |HOSTNAME|NODELID| LOCATION  | ERRORS
  -----------------------------------------------------------------------------------------
13  | 13  |0x0008f104003f238e| vl15dropped=1|  4X  |5.0 G|| 13  | 13  |0x0008f10400411da2|0x0008f10400411da2|Switch|ISR9024D Voltaire|iswu14c1| 0x009D|[B,4] RACK71/B|

14  | 14  |0x0008f104003f238e|               |  4X  |5.0 G|| 14  | 14  |0x0008f10400411da2|0x0008f10400411da2|Switch|ISR9024D Voltaire|iswu14c1| 0x009D|[B,4] RACK71/B|

15  | 15  |0x0008f104003f238e|               |  4X  |5.0 G|| 15  | 15  |0x0008f10400411da2|0x0008f10400411da2|Switch|ISR9024D Voltaire|iswu14c1| 0x009D|[B,4] RACK71/B|
```

Figure 3-2. Example for chassis based switch



```
------------------------------------------------------------------------------------
DESCRIPTION          | HOSTNAME           | NODEGUID           | NODELID | LOCATION          |
------------------------------------------------------------------------------------
ISR9024D-M Voltaire | iswu0c0            | 0x0008f10400041257e | 0x0001  | [A,1] RACK1/A     |
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
                                 LOCAL                              ||                                                                        REMOTE
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
PORT
/PIN|PORTGUID/PORTNODEG | ERRORS                       |WTH|SPEED||PT|PIN|       PORTGUID  |    PORTNODEGUID  |TYPE|     DESCRIPTION |HOSTNAME |NODELID|    LOCATION | ERRORS
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
1  |0x0008f10400041257e|linkdowned=27,xmtdiscards=1  |4X  |5.0  G||1  |  1  |0x0002c90200234035|0x0002c90200234034| CA |MT25218 InfiniHos|  xena12 |0x000A |[A,1] RACK1/U|vl15dropped=2,xmtdiscards=1
2  |0x0008f10400041257e|linkdowned=6,linkrecovers=1, |4X  |5.0  G||1  |  1  |0x0002c90200234031|0x0002c90200234030| CA |      xena0 HCA-1 |   xena0 |0x0008 |[A,1] RACK1/U|xmtdiscards=2,vl15dropped=2
3  |0x0008f10400041257e|linkdowned=25,linkrecovers=3 |4X  |5.0  G||1  |  1  |0x0002c90200234041|0x0002c90200234040| CA |      xena1 HCA-1 |   xena1 |0x0009 |[A,1] RACK1/S|xmtdiscards=25,vl15dropped=10,
4  |0x0008f10400041257e|linkdowned=36,rcverr=9,xmtdi |4X  |5.0  G||1  |  1  |0x0002c9020025ec9d|0x0002c9020025ec9c| CA |     xena13 HCA-1 |  xena13 |0x0017 |[B,1] RACK2/M|xmtdiscards=1
5  |0x0008f10400041257e|linkdowned=11,xmtdiscards=4, |4X  |5.0  G||1  |  1  |0x30003f3f30003f3f|0x0002c9020024b9b0| CA |      xena3 HCA-1 |   xena3 |0x001C |[A,1] RACK1/Q|xmtdiscards=2,vl15dropped=2
6  |0x0008f10400041257e|linkdowned=17,xmtdiscards=4, |4X  |5.0  G||1  |  1  |0x0002c9020024b875|0x0002c9020024b874| CA |MT25218 InfiniHos|   xena4 |0x0010 |[A,1] RACK1/Q|xmtdiscards=2,vl15dropped=2
8  |0x0008f10400041257e|linkdowned=8,xmtdiscards=3   |4X  |5.0  G||1  |  1  |0x0002c90200234015|0x0002c90200234014| CA |MT25218 InfiniHos|  xena14 |0x0016 |[B,1] RACK2/M|vl15dropped=2,xmtdiscards=1
9  |0x0008f10400041257e|xmtdiscards=4,linkrecovers=1 |4X  |5.0  G||1  |  1  |0x0002c9020024b8a9|0x0002c9020024b8a8| CA | globus-cs HCA-1 |lobus-cs |0x000F |             |xmtdiscards=11,vl15dropped=2,s
11 |0x0008f10400041257e|linkdowned=5                 |4X  |5.0  G||1  |  1  |0x0002c9020024b9b9|0x0002c9020024b9b8| CA |MT25218 InfiniHos|Mellanox |0x0006 |             |vl15dropped=2,xmtdiscards=1
13 |0x0008f10400041257e|rcverr=10,linkdowned=9,linkr |4X  |5.0  G||1  |  1  |0x0002c90200234001|0x0002c90200234000| CA |MT25218 InfiniHos|   xena2 |0x0004 |[A,1] RACK1/R|vl15dropped=2,xmtdiscards=1
14 |0x0008f10400041257e|linkdowned=34,xmtdiscards=6, |4X  |5.0  G||1  |  1  |0x0002c9020024b879|0x0002c9020024b878| CA |MT25218 InfiniHos|  xena10 |0x0007 |[A,1] RACK1/K|vl15dropped=2,xmtdiscards=1
16 |0x0008f10400041257e|linkdowned=32                |4X  |5.0  G||1  |  1  |0x0002c9020025ed95|0x0002c9020025ed94| CA |     xena32 HCA-1 |  xena32 |0x001D |[B,1] RACK2/L|
17 |0x0008f10400041257e|xmtdiscards=2,linkrecovers=1 |4X  |5.0  G||1  |  1  |0x0002c9020024b9b5|0x0002c9020024b9b4| CA | localhost HCA-1 |localhost|0x0005 |             |vl15dropped=3,xmtdiscards=1,li
20 |0x0008f10400041257e|linkdowned=6,linkrecovers=1  |4X  |5.0  G||1  |  1  |0x00030487f13f80001|0x00030487f13f80000| CA |MT25204 InfiniHos|   xena5 |0x000E |[B,1] RACK2/N|vl15dropped=2,xmtdiscards=1
21 |0x0008f10400041257e|rcverr=538,linkrecovers=5,li |4X  |5.0  G||1  |  1  |0x0002c9020024b8fd|0x0002c9020024b8fc| CA |MT25218 InfiniHos|Mellanox |0x0012 |             |xmtdiscards=12,vl15dropped=4,l
22 |0x0008f10400041257e|linkdowned=1                 |4X  |5.0  G||1  |  1  |0x0002c9020024b99d|0x0002c9020024b99c| CA |MT25218 InfiniHos|         |0x0013 |             |vl15dropped=3,xmtdiscards=1
23 |0x0008f10400041257e|linkdowned=15                |4X  |5.0  G||1  |  1  |0x0002c9020023c249|0x0002c9020023c248| CA |MT25218 InfiniHos|  xena11 |0x0002 |[A,1] RACK1/T|vl15dropped=2,xmtdiscards=1
24 |0x0008f10400041257e|linkdowned=12,xmtdiscards=4, |4X  |5.0  G||1  |  1  |0x00030487f13e40001|0x00030487f13e40000| CA |MT25204 InfiniHos|   xena6 | x0019 | B,1] RACK2/N|vl15dropped=2,xmtdiscards=1
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Figure 3-3. Bandwidth output for Voltaire ISR 9024D-M Switch

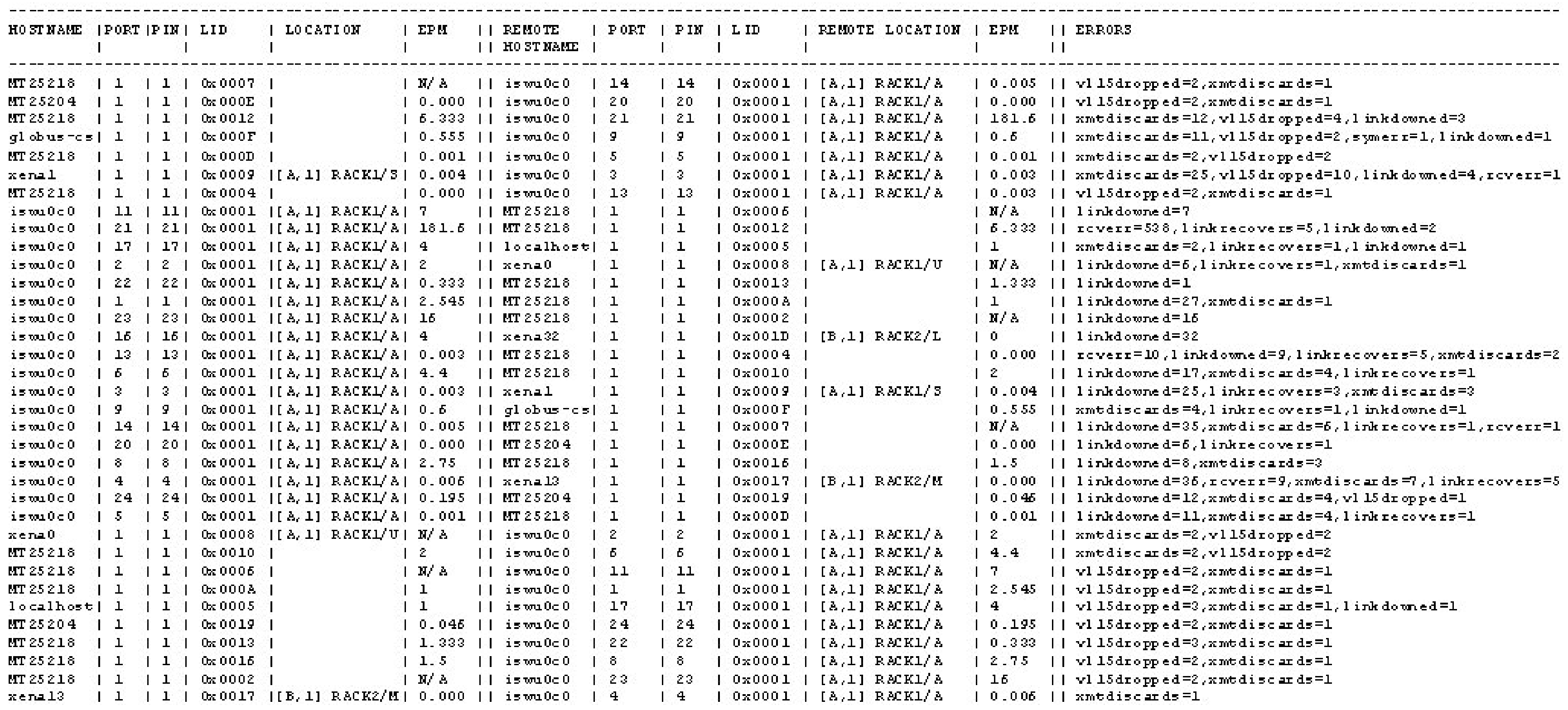
3.12.7.2 Bandwidth

The Bandwidth output is based upon the same layout as the topology output and is fairly self explanatory, see *Figure 3-3*:

3.12.7.3 Errors

When troubleshooting the InfiniBand network, the *errors* action is the most useful. It produces a list of all the ports that have encountered an error in the fabric.

Note The most interesting column is probably **EPM**. **EPM** stands for **Errors per Million** and describes the error rate for the link (the higher the error rate, the worst the link is). In the example below, port 21 of the switch *iswu0c0* definately needs to be fixed.



```
------------------------------------------------------------------------------------------------------------------------------------------------------------
HOSTNAME  |PORT |PIN| LID    | LOCATION    | EPM   || REMOTE    | PORT | PIN | LID    | REMOTE LOCATION | EPM   || ERRORS
          |         |        |             |       || HOSTNAME  |      |     |        |                 |       ||
------------------------------------------------------------------------------------------------------------------------------------------------------------
MT25218   | 1   | 1 | 0x0007 |             | N/A   || iswu0c0   | 14   | 14  | 0x0001 | [A,1] RACK1/A   | 0.005 || vl15dropped=2,xmtdiscards=1
MT25204   | 1   | 1 | 0x000E |             | 0.000 || iswu0c0   | 20   | 20  | 0x0001 | [A,1] RACK1/A   | 0.000 || vl15dropped=2,xmtdiscards=1
MT25218   | 1   | 1 | 0x0012 |             | 6.333 || iswu0c0   | 21   | 21  | 0x0001 | [A,1] RACK1/A   | 181.6 || xmtdiscards=12,vl15dropped=4,linkdowned=3
globus-cs | 1   | 1 | 0x000F |             | 0.555 || iswu0c0   | 9    | 9   | 0x0001 | [A,1] RACK1/A   | 0.6   || xmtdiscards=11,vl15dropped=2,symerr=1,linkdowned=1
MT25218   | 1   | 1 | 0x000D |             | 0.001 || iswu0c0   | 5    | 5   | 0x0001 | [A,1] RACK1/A   | 0.001 || xmtdiscards=2,vl15dropped=2
xena1     | 1   | 1 | 0x0009 |[A,1] RACK1/S| 0.004 || iswu0c0   | 3    | 3   | 0x0001 | [A,1] RACK1/A   | 0.003 || xmtdiscards=25,vl15dropped=10,linkdowned=4,rcverr=1
MT25218   | 1   | 1 | 0x0004 |             | 0.000 || iswu0c0   | 13   | 13  | 0x0001 | [A,1] RACK1/A   | 0.003 || vl15dropped=2,xmtdiscards=1
iswu0c0   | 11  | 11| 0x0001 |[A,1] RACK1/A| 7     || MT25218   | 1    | 1   | 0x0006 |                 | N/A   || linkdowned=7
iswu0c0   | 21  | 21| 0x0001 |[A,1] RACK1/A| 181.6 || MT25218   | 1    | 1   | 0x0012 |                 | 6.333 || rcverr=538,linkrecovers=5,linkdowned=2
iswu0c0   | 17  | 17| 0x0001 |[A,1] RACK1/A| 4     || localhost| 1    | 1   | 0x0005 |                 | 1     || xmtdiscards=2,linkrecovers=1,linkdowned=1
iswu0c0   | 2   | 2 | 0x0001 |[A,1] RACK1/A| 2     || xena0     | 1    | 1   | 0x0008 | [A,1] RACK1/U   | N/A   || linkdowned=6,linkrecovers=1,xmtdiscards=1
iswu0c0   | 22  | 22| 0x0001 |[A,1] RACK1/A| 0.333 || MT25218   | 1    | 1   | 0x0013 |                 | 1.333 || linkdowned=1
iswu0c0   | 1   | 1 | 0x0001 |[A,1] RACK1/A| 2.545 || MT25218   | 1    | 1   | 0x000A |                 | 1     || linkdowned=27,xmtdiscards=1
iswu0c0   | 23  | 23| 0x0001 |[A,1] RACK1/A| 16    || MT25218   | 1    | 1   | 0x0002 |                 | N/A   || linkdowned=16
iswu0c0   | 16  | 16| 0x0001 |[A,1] RACK1/A| 4     || xena32    | 1    | 1   | 0x001D | [B,1] RACK2/L   | 0     || linkdowned=32
iswu0c0   | 13  | 13| 0x0001 |[A,1] RACK1/A| 0.003 || MT25218   | 1    | 1   | 0x0004 |                 | 0.000 || rcverr=10,linkdowned=9,linkrecovers=5,xmtdiscards=2
iswu0c0   | 6   | 6 | 0x0001 |[A,1] RACK1/A| 4.4   || MT25218   | 1    | 1   | 0x0010 |                 | 2     || linkdowned=17,xmtdiscards=4,linkrecovers=1
iswu0c0   | 3   | 3 | 0x0001 |[A,1] RACK1/A| 0.003 || xena1     | 1    | 1   | 0x0009 | [A,1] RACK1/S   | 0.004 || linkdowned=25,linkrecovers=3,xmtdiscards=3
iswu0c0   | 9   | 9 | 0x0001 |[A,1] RACK1/A| 0.6   || globus-cs| 1    | 1   | 0x000F |                 | 0.555 || xmtdiscards=4,linkrecovers=1,linkdowned=1
iswu0c0   | 14  | 14| 0x0001 |[A,1] RACK1/A| 0.005 || MT25218   | 1    | 1   | 0x0007 |                 | N/A   || linkdowned=35,xmtdiscards=6,linkrecovers=1,rcverr=1
iswu0c0   | 20  | 20| 0x0001 |[A,1] RACK1/A| 0.000 || MT25204   | 1    | 1   | 0x000E |                 | 0.000 || linkdowned=6,linkrecovers=1
iswu0c0   | 8   | 8 | 0x0001 |[A,1] RACK1/A| 2.75  || MT25218   | 1    | 1   | 0x0016 |                 | 1.5   || linkdowned=8,xmtdiscards=3
iswu0c0   | 4   | 4 | 0x0001 |[A,1] RACK1/A| 0.006 || xena13    | 1    | 1   | 0x0017 | [B,1] RACK2/M   | 0.000 || linkdowned=36,rcverr=9,xmtdiscards=7,linkrecovers=5
iswu0c0   | 24  | 24| 0x0001 |[A,1] RACK1/A| 0.195 || MT25204   | 1    | 1   | 0x0019 |                 | 0.046 || linkdowned=12,xmtdiscards=4,vl15dropped=1
iswu0c0   | 5   | 5 | 0x0001 |[A,1] RACK1/A| 0.001 || MT25218   | 1    | 1   | 0x000D |                 | 0.001 || linkdowned=11,xmtdiscards=4,linkrecovers=1
xena0     | 1   | 1 | 0x0008 |[A,1] RACK1/U| N/A   || iswu0c0   | 2    | 2   | 0x0001 | [A,1] RACK1/A   | 2     || xmtdiscards=2,vl15dropped=2
MT25218   | 1   | 1 | 0x0010 |             | 2     || iswu0c0   | 6    | 6   | 0x0001 | [A,1] RACK1/A   | 4.4   || xmtdiscards=2,vl15dropped=2
MT25218   | 1   | 1 | 0x0006 |             | N/A   || iswu0c0   | 11   | 11  | 0x0001 | [A,1] RACK1/A   | 7     || vl15dropped=2,xmtdiscards=1
MT25218   | 1   | 1 | 0x000A |             | 1     || iswu0c0   | 1    | 1   | 0x0001 | [A,1] RACK1/A   | 2.545 || vl15dropped=2,xmtdiscards=1
localhost | 1   | 1 | 0x0005 |             | 1     || iswu0c0   | 17   | 17  | 0x0001 | [A,1] RACK1/A   | 4     || vl15dropped=3,xmtdiscards=1,linkdowned=1
MT25204   | 1   | 1 | 0x0019 |             | 0.046 || iswu0c0   | 24   | 24  | 0x0001 | [A,1] RACK1/A   | 0.195 || vl15dropped=2,xmtdiscards=1
MT25218   | 1   | 1 | 0x0013 |             | 1.333 || iswu0c0   | 22   | 22  | 0x0001 | [A,1] RACK1/A   | 0.333 || vl15dropped=3,xmtdiscards=1
MT25218   | 1   | 1 | 0x0016 |             | 1.5   || iswu0c0   | 8    | 8   | 0x0001 | [A,1] RACK1/A   | 2.75  || vl15dropped=2,xmtdiscards=1
MT25218   | 1   | 1 | 0x0002 |             | N/A   || iswu0c0   | 23   | 23  | 0x0001 | [A,1] RACK1/A   | 16    || vl15dropped=2,xmtdiscards=1
xena13    | 1   | 1 | 0x0017 |[B,1] RACK2/M| 0.000 || iswu0c0   | 4    | 4   | 0x0001 | [A,1] RACK1/A   | 0.006 || xmtdiscards=1
```

Figure 3-4. IBS Errors Action Ouput

Typica Error signatures

An error is rarely triggered on its own and errors usually come in bunches. When a node boots up, or when the OpenIBD service is restarted, we usually see patterns like these on the host (HCA) side of the cable:

```
xmtdiscards=2,vl15dropped=2
xmtdiscards=1,vl15dropped=2
xmtdiscards=2,vl15dropped=1
```

These errors can be ignored.

This appears as follows, as seen from the switch:

```
linkdowned=1
```

3.12.8 IBS Database

IBS can use the following data sources to populate or update its own database: **switch**, **xmlfile**, **ibnetdiscover** and **topofile**.

3.12.8.1 Browse data

Once data are stored in the ibs database, the fabric topology is easy to browse using a program such as **phpPgAdmin. IBS** can also use the data stored in its database to perform an action.

Advantages

- Easy to browse using phpPgAdmin: http://<admin node hostname>/phpPgAdmin
- Easy to retrieve data for scripting using psql.

Drawbacks

- Always need to make sure that the database is up to date

In the example below, both the topology and the port counters are read from the database:

```
ibs -S db -a topo
```

3.12.8.2 Keeping the IBS database up to date

In order to ensure that the data is always up to date, it is advisable to add the following line to the cron table (using the **crontab -e** command).
With the following setup, the traffic and error counters as well as the **InfiniBand** equipment stored in the **IBS** database will be refreshed every 10 minutes using the data supplied by the switch **iswu0c0-0**:

```
*/10 * * * * /usr/bin/ibs -s iswu0c0-0 -a dbupdate -vNE >>
/var/log/ibs.log 2>&1
```

Note The above example suffers from the Subnet Manager caveats. If for some reason, the switch iswu0c0-0 failed and another switch, say iswu0c0-1 was to become the new subnet manager master, the data fed into the database would be incorrect. When running on an InfiniBand network with multiple managed switches, the user needs to know which switch is running the subnet manager as master. This switch should always be provided as an argument of the -s flag. Assuming the data is refreshed by the cron daemon, if another switch was to become the subnet manager master, the data fed into the database would be incorrect when read from the slave switch hardcoded in the cron script.

A better solution would be to use the ibnetdiscover data source:

```
*/10 * * * * /usr/bin/ibs -S ibnetdiscover -a dbupdate -vNE >> /var/log/ibs.log 2>&1
```

Chapter 4. Troubleshooting InfiniBand Networks

The following topics are described:

Important All commands in this chapter which start with the hash (#) sign must be carried out as root.

4.1 Troubleshooting a non-functional InfiniBand Network

4.1.1 InfiniBand Hardware Checks

Carry out the following hardware checks if the **InfiniBand** network is not functioning:

- Make sure that the **InfiniBand** switches are powered up. Refer to the switch documentation for more details.
- Make sure that the each machine is powered up. Refer to the machine documentation for more details.
- Make sure that each machine is fitted with a functional **InfiniBand** adapter. If the adapter is not functional insert a new one following the instructions in the server documentation.
- Check that the **InfiniBand** adapter is supported for the architecture of the cluster.
- Once the card is fitted into the server, run the command below to check that it is visible:

```
lspci | grep InfiniBand
```

```
03:00.0 InfiniBand: Mellanox Technologies MT26428 [ConnectX IB QDR,
PCIe 2.0 5GT/s] (rev a0)
```

  If the card is not recognised, replace the faulty card by a card that is known to work properly, and run the above command again.

- Once the card is fitted into the server, make sure that it is connected properly by checking that the cable connectors are fully inserted into the ports.

Green LEDs, when lit, indicate that the physical link is up. Yellow LEDs, when lit, indicate that the logical link is up, and their blinking rate indicates link activity.

If a green LED is not lit, swap the cable with a cable that is known to work properly and test again. If it still fails, try the cable on another switch port. If it works, then the former switch port may be out of order or disabled by the system administrator.

For more information contact Bull Technical Support.

4.1.2 InfiniBand Software Installation Check

Make sure that following packages are installed:

- kernel-ib
- kernel-ib-scripts
- infiniband-diags
- libibmad
- libibumad
- opensm-libs

Use the command below to check that a given package is installed:

```
rpm -q <package name>
```

Depending on the interconnect architecture, the node may also be used as an **OpenSM** subnet manager. In this case, the **opensm** package should also be installed.

See *Chapter 1* in this manual and the **Software Release Bulletin** for more details.

4.1.3 Check the openibd service

The **openibd** service is a script that loads all the **InfiniBand** drivers listed in the **openibd** configuration file. This configuration file is located in the **/etc/infiniband** directory, and its generic name is **openib_<kernel release>.conf**.

The example below details the **openib** settings for a 2.6.18-128.el5.Bull.2 kernel configuration file.

```
cat /etc/infiniband/openib_2.6.18-128.el5.Bull.2.conf
```

```
# Start HCA driver upon boot
ONBOOT=yes
# Load UCM module
UCM_LOAD=no
# Load RDMA_CM module
RDMA_CM_LOAD=yes
# Load RDMA_UCM module
RDMA_UCM_LOAD=yes
# Increase ib_mad thread priority
RENICE_IB_MAD=no
# Load MTHCA
MTHCA_LOAD=yes
# Load MLX4 modules
MLX4_LOAD=yes
# Load IPoIB
IPOIB_LOAD=yes
# Enable IPoIB Connected Mode
SET_IPOIB_CM=yes
# Load SDP module
SDP_LOAD=yes
# Load SRP module
```

```
SRP_LOAD=no
# Load SRP Target module
SRPT_LOAD=no
# Load ISER module
ISER_LOAD=no
```

4.1.3.1 Modules and Drivers for InfiniBand networks

A range of modules and drivers are possible for **InfiniBand** networks, and include the following:

Core modules

• ib_addr	**InfiniBand** address translation
• ib_core	Core kernel InfiniBand API

Hardware support

• mlx4_ib	**Mellanox** ConnectX HCA InfiniBand driver
• mlx4_core	**Mellanox** ConnectX HCA low-level driver
• ib_mthca	**Mellanox** Infiniband HCA low-level driver
• qlgc_vnic	**QLogic** virtual NIC (VNIC) driver
• mlx4_en	**Mellanox** ConnectX HCA Ethernet driver

IP over IB modules

• ib_ipoib	IP-over-InfiniBand net driver
• ipoib_helper	Container for ipoib neighbour destructor

Subnet manager related modules

• ib_mad	IB MAD API
• ib_sa	Subnet administration query support
• ib_umad	Userspace MAD packet access

MPI related modules

• ib_uverbs	Userspace verbs access

Connection managers

• ib_ucm	Userspace connection manager access
• ib_cm	Connection manager
• iw_cm	iWARP connection manager
• rdma_cm	Generic RDMA connection manager
• rdma_ucm	RDMA userspace connection manager

Socket direct protocol

• ib_sdp	Socket direct protocol (SDP).

Storage

ib_iser	iSER (iSCSI Extensions for RDMA) datamover
ib_srp	InfiniBand SCSI RDMA Protocol initiator
ib_srpt	InfiniBand SCSI RDMA Protocol target
iscsi_tcp	iSCSI/TCP data-path
Libiscsi	iSCSI library functions
scsi_transport_iscsi	iSCSI transport interface

4.1.3.2 Starting the openibd service

1. Start the **openibd** service using the command below:

```
# service openibd start
```

This will give output similar to that below:

```
Loading HCA driver and Access Layer:                         [  OK  ]
Setting up InfiniBand network interfaces:
Bringing up interface ib0:                                   [  OK  ]
Bringing up interface ib1:                                   [  OK  ]
Setting up service network . . .                            [  done  ]
```

If **openibd** service fails to start, check that the configuration file is correct. If the configuration file is OK, then contact Bull Technical support for further instructions.

2. Also, please make sure that the service is automatically started when the node boots up (run levels 3 and 5 for full multi-user mode):

```
chkconfig --list openibd
```

This will give output similar to that below:

```
openibd         0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

3. Run the command below to ensure that the **openibd** service starts automatically:

```
# chkconfig openibd on
```

   a. If the **openibd** service fails to start, make sure that its version matches the version of the kernel that is running:

```
rpm -q kernel-ib
```

```
kernel-ib-1.3.2-2.6.18_128.el5.Bull.2.Bull.1
```

```
uname -r
```

```
2.6.18-128.el5.Bull.2
```

In the above example:

- The **kernel-ib** package version is 1.3.2
- The kernel for which the **kernel-ib** package was built is 2.6.18_128.el5.Bull.2

- The **kernel-ib** sub-version is Bull.1

  The kernel for which the **kernel-ib** package was built matches the running kernel version (2.6.18-128.el5.Bull.2).

b. Check that the **kernel-ib-scripts** version matches the **kernel-ib** version:

```
rpm -q kernel-ib-scripts
```

```
kernel-ib-scripts-1.3.2-Bull.1
```

In the above example:

- The **kernel-ib-scripts** package version is 1.3.2
- The **kernel-ib-scripts** sub-version is Bull.1

The **kernel-ib-scripts** version matches the version of the installed **kernel-ib** package.

c. After checking the **kernel**, **kernel-ib** and **kernel-ib-scripts** versions, make sure that the **openib** configuration file matches the version of the kernel that is running:

```
ls -l /etc/infiniband/openib_`uname -r`.conf
```

```
-rw-r--r-- 1 root root 481 Feb  5 18:13 /etc/infiniband/openib_2.6.18-
128.el5.Bull.2.conf
```

4.1.4 Check the InfiniBand Adapters

4.1.4.1 Mellanox Firmware Issues

Check that the **InfiniBand** adapters' firmware is up-to-date. The following packages are used to check the firmware version of the **InfiniBand** adapters.

- **mthca_fw_update**
- **mstflint**

Check the **InfiniBand** firmware version, by using the command below:

```
mthca_fw_upd  -t
```

```
mlx4_0: Firmware 2.6.0 is up to date
```

Alternatively, the current firmware version can be seen by running the command below:

```
cat /sys/class/infiniband/*/fw_ver
```

```
2.6.0
```

However, this command does not warn if the firmware version is not up to date.

If the firmware is not up to date, it should be upgraded as follows:

```
mthca_fw_upd -t
```

```
mlx4_0: Firmware 2.5.900 must be updated to 2.6.000
```

```
# mthca_fw_upd -u
```

```
Burn firmware 2.6.000 on mlx4_0
    Current FW version on flash:  2.5.900
    New FW version:               2.6.0
Burning first  FW image without signatures  - OK
Restoring first  signature                  - OK
Device(s)  mlx4_0 udpated, new firmware available after openibd
restart or next reboot
```

After upgrading the firmware, reboot the server to load the new firmware.

4.1.4.2 Voltaire Firmware issues

See See the **Mellanox** and **Voltaire** websites for more information on the **InfiniBand** adapter firmware versions.

4.1.4.3 Checking the physical status of the ports

Once the drivers listed in the **openib_<kernel-release>.conf** file have been loaded by the **openibd** service successfully, the physical state of the ports should be checked.

Note **InfiniBand** adapters may have several ports and depending on the cluster topology, and all the ports cannot be connected to a switch.

Use the **ibstat** command as follows to check the physical state of the adapter port(s):

```
ibstat
```

```
CA 'mlx4_0'
        CA type: MT26428
        Number of ports: 2
        Firmware version: 2.6.0
        Hardware version: a0
        Node GUID: 0x0002c903000262da
        System image GUID: 0x0002c903000262dd
        Port 1:
                State: Active
                Physical state: LinkUp
                Rate: 40
                Base lid: 1
                LMC: 0
                SM lid: 1
                Capability mask: 0x00000000
                Port GUID: 0x0002c903000262db
        Port 2:
                State: Down
                Physical state: Polling
                Rate: 10
                Base lid: 0
                LMC: 0
                SM lid: 0
                Capability mask: 0x00000000
                Port GUID: 0x0002c903000262dc
```

In the example above, we can see that the first port is physically connected (`LinkUp`) and that a Subnet manager is present in the **InfiniBand** network. The second port is not physically connected (`Polling`) and as a result cannot see a Subnet manager (`State: Down`).

Physical States of the Ports

The following physical states are possible:

• Sleep	Contact Bull support.
• Polling	
	– No cable is connected to the adapter port: try to reconnect the cable and check the LED status.
	– The cable is broken and needs to be replaced.
	– The port on the other end of the cable is disabled.
• Disabled	Enable the port using the **ibportstate** command
• PortConfigurationTraining	Wait for few seconds until the port changes state. If it does not change, check if the adapter and the switch that it is connected to are compatible.
• LinkUp	The port is physically connected. Please move to the next section.
• LinkErrorRecovery	This indicates that the cable is faulty. Replace the cable.

Alternatively, the **ibstatus** command can be used to check the adapter status:

```
ibstatus
```

```
Infiniband device 'mlx4_0' port 1 status:
        default gid:     fe80:0000:0000:0000:0002:c903:0002:62db
        base lid:        0x1
        sm lid:          0x1
        state:           4: ACTIVE
        phys state:      5: LinkUp
        rate:            40 Gb/sec (4X QDR)
Infiniband device 'mlx4_0' port 2 status:
        default gid:     fe80:0000:0000:0000:0002:c903:0002:62dc
        base lid:        0x0
        sm lid:          0x0
        state:           1: DOWN
        phys state:      2: Polling
        rate:            10 Gb/sec (4X)
cat /sys/class/infiniband/*/ports/*/state
4: ACTIVE
1: DOWN
```

4.1.5 Check the OpenSM Subnet Manager

Once the physical state for the port is set to `LinkUp`, the Subnet manager needs to be started for the **InfiniBand** network to function. In the example below, the physical state is `LinkUp`, but no Subnet manager is running (`State:  Initializing`):

```
ibstat
```

```
CA 'mlx4_0'
CA type: MT26428
Number of ports: 2
Firmware version: 2.6.0
Hardware version: a0
Node GUID: 0x0002c903000262da
System image GUID: 0x0002c903000262dd
Port 1:
   State: Initializing
   Physical state: LinkUp
   Rate: 40
   Base lid: 0
   LMC: 0
   SM lid: 0
   Capability mask: 0x00000000
   Port GUID: 0x0002c903000262db
Port 2:
   State: Down
   Physical state: Polling
   Rate: 10
   Base lid: 0
   LMC: 0
   SM lid: 0
   Capability mask: 0x00000000
   Port GUID: 0x0002c903000262dc
```

The absence of the subnet manager is confirmed by running the **sminfo** command:

```
sminfo
```

```
sminfo: iberror: failed: query
```

Two scenarios are possible here:

- The **InfiniBand** network includes **Voltaire** or **Mellanox** managed switches. Please refer to the switch manufacturer documentation to fix this problem.

- The **InfiniBand** network does not include managed switches. In this case, **OpenSM** should be installed on one or more nodes within the cluster. The number of Subnet managers and their location depends on the cluster architecture and the options selected by the customer.

If the Subnet manager package is not installed, install and configure **OpenSM**, as described in Chapter 2.

In the example below, it is assumed that the current node hosts a Subnet manager.

1. Once **OpenSM** is installed, check that the **opensm** service has started by running the command below:

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

```
opensm is stopped
```

2. If **opensm** is not running, start it as follows:

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

```
Starting IB Subnet Manager.                               [  OK  ]
```

3. If **OpenSM** fails to start, check that its configuration file is correct. If it is, contact Bull Technical Support for further instructions.
4. Check that the **opensmd** service starts automatically when the node boots up (run levels 3 and 5):

```
chkconfig --list opensmd
```

```
opensmd         0:off 1:off  2:off 3:off  4:off 5:off  6:off
```

5. Run the following commands to start the **opensmd** service automatically:

```
# chkconfig opensmd on
chkconfig --list opensmd
```

```
opensmd         0:off 1:off  2:on 3:on 4:on 5:on 6:off
```

6. If the subnet manager has started and is set up properly, a master subnet manager will be found:

```
sminfo
```

```
sminfo: sm lid 1 sm guid 0x2c903000262db, activity count 390772
priority 0 state 3 SMINFO_MASTER
```

7. Check that the port state is now active:

```
ibstat
```

```
CA 'mlx4_0'
CA type: MT26428
Number of ports: 2
Firmware version: 2.6.0
Hardware version: a0
Node GUID: 0x0002c903000262da
System image GUID: 0x0002c903000262dd
Port 1:
   State: Active
   Physical state: LinkUp
   Rate: 40
   Base lid: 1
   LMC: 0
   SM lid: 1
   Capability mask: 0x00000000
   Port GUID: 0x0002c903000262db
Port 2:
   State: Down
   Physical state: Polling
   Rate: 10
   Base lid: 0
   LMC: 0
   SM lid: 0
   Capability mask: 0x00000000
   Port GUID: 0x0002c903000262dc
```

8. Run the commands below to determine the name of the node hosting the subnet manager:

```
sminfo
```

```
sminfo: sm lid 1 sm guid 0x2c903000262db, activity count 390772
priority 0 state 3 SMINFO_MASTER
```

```
smpquery nodedesc -G 0x2c903000262db
```

```
Node Description:.....................inti0 HCA-1
```

4.1.6 Check the Switch firmware

Note The information in this section applies to non-managed switches only. Refer to the switch documentation if the switch is managed internally.

The following specific packages are required in order to check the(all) switch(s) firmware version.

- infiniband-diags
- mft
- ibsw_fw_update

The **mft** packages include the **flint** program that can be used to check the firmware version of the **InfiniBand** switches. **flint** uses an InfiniBand connection to retrieve and write data to the switch, and therefore does not require any **Ethernet** related setup.

ibsw_fw_update is a Bull tool which ensures that **mft** (with flint command for burning firmware) and **infiniband-diags** tools function correctly.

Run the command below to obtain the list of switches in an **InfiniBand** network:

```
ibswitches
```

```
Switch  : 0x0002c90200404798 ports 36 "Infiniscale-IV Mellanox
Technologies" base port 0 lid 11 lmc 0
Switch  : 0x0002c902004047c0 ports 36 "Infiniscale-IV Mellanox
Technologies" base port 0 lid 21 lmc 0
Switch  : 0x0002c902004044e0 ports 36 "Infiniscale-IV Mellanox
Technologies" base port 0 lid 17 lmc 0
```

In order to query the firmware information for the switch with **GUID** 0x0002c902004047c0, use the following command:

```
/usr/sbin/ibsw_fw_update -t -g 0x0002c902004047c0
```

```
Switch 0x0002c902004047c0: Firmware 7.1.0 must be updated to 07.02.00.
```

Update the firmware as follows:

```
/ibsw_fw_update]$ # usr/sbin/ibsw_fw_update -u -g 0x0002c902004047c0
```

```
Burn firmware 07.02.00 on "lid-0x0015" with guid 0x0002c902004047c0
   Current FW version on flash:  7.1.0
   New FW version:               7.2.0
Burning first  FW image without signatures  - OK
Restoring first  signature                  - OK
Switch with guid  "lid-0x0015" successfully updated.
1.Reset the switch as follows:
flint -d "lid-0x0015" swreset
```

```
2.View update:
flint -d "lid-0x0015" q
```

Important Before upgrading the firmware for a switch, check that the switch model is compatible with the new firmware. Upgrading firmware is a delicate procedure that requires extreme caution. Please contact Bull support for additional information.

Manual Procedure

a. Run the command below to see the firmware information for the switch, for example one with lid 11:

```
flint -d "lid-11" q
```

```
Image type:      FS2
FW Version:      7.1.0
Device ID:       48438
Chip Revision:   A0
Description:     Node             Sys image
GUIDs:           0002c90200404798 0002c9020040479b
Board ID:         (MT_0C20110003)
VSD:
PSID:            MT_0C20110003
```

b. If the firmware for an InfiniBand switch is not up to date, upgrade it as follows:

```
# flint -d "lid-11" -i fw-IS4-rel-7_2_000-MTS3600Q_A1.bin b
```

```
    Current FW version on flash:  7.1.0
    New FW version:               7.2.0
Burning second FW image without signatures  - OK
Restoring second signature                  - OK
```

c. After upgrading the firmware, reset the switch as follows:

```
# flint -d "lid-11" swreset
```

```
Resetting device lid-11 ...
```

d. Check that the new firmware is operational:

```
flint -d "lid-11" q
```

```
Image type:      FS2
FW Version:      7.2.0
Device ID:       48438
Chip Revision:   A0
Description:     Node             Sys image
GUIDs:           0002c90200404798 0002c9020040479b
Board ID:         (MT_0C20110003)
VSD:
PSID:            MT_0C20110003
```

See The **Mellanox** and **Voltaire** websites for more information on the **InfiniBand** switch firmware versions.

4.1.7 Check the IP over IB interfaces

Each InfiniBand port on the node should have a corresponding entry in **/etc/modprobe.conf** configuration file. The entry describes the IP interface name and the corresponding kernel module to use.

Example for 2 interfaces named **ib0** and **ib1**:

```
alias ib0 ib_ipoib
alias ib1 ib_ipoib
```

After adding the appropriate lines to **/etc/modprobe.conf** configuration file, run the following command:

```
# depmod -a
```

The **IPoIB** interface is configured by a file in the **/etc/sysconfig/network-script** directory which is generated using the **config_ip** or **config_ipoib** command (the command available depends on the system).

The default IP configuration for an **IPoIB** interface is set by a file in the **/etc/sysconfig/network-script** directory. These network configuration files can also be generated by using the **config_ip** or **config_ipoib** command.

ib0 interface example

```
cat /etc/sysconfig/network-scripts/ifcfg-ib0
DEVICE=ib0
BOOTPROTO=static
IPADDR=10.12.0.1
NETMASK=255.255.0.0
ONBOOT=yes
```

IP over IB (IPoIB) can be used in two different modes:

- **Datagram**: In datagram mode, all packets are broadcast over the InfiniBand network. Although it may be useful for multicast operations, performance is low due to a limited MTU (2048). Note that multicast is not used by **MPI** or **Lustre**.
- **Connected**: In connected mode, point-to-point IB connections are created between the hosts for **IPoIB** transfers. This mode does not allow for broadcast operations. However, it allows for a large MTU (65520) and thus achieves a much higher bandwidth.

Note Within a given InfiniBand network, all IP over IB interfaces must use the same mode. The recommended mode for a Bull cluster is **connected**.

See The chapter on *InfiniBand Network Optimization* for more information on setting the **Datagram** and **Connected** modes for the **IPoIB** interfaces.

4.1.8 Check the dapl configuration (for MPI applications only)

The **dapl** package is available on the **XIB** media. This package provides an implementation of the **DAT (Direct Access Transport) 1.2** and the **API 2.0** for the user space that is built to support **InfiniBand** network technology.

Configure the **/etc/dat.conf** file to obtain the **uDAPL RDMA** capabilities. If it does not exist, the **/etc/dat.conf** file should be created manually using the **create_daplconf.sh** script. This is done by running the following script:

```
#!/bin/sh set -x sysconfdir="/etc"
/sbin/ldconfig
if [ -e $sysconfdir/dat.conf ]; then
   sed -e '/OpenIB-.*/d' < $sysconfdir/dat.conf > /tmp/$$ofadapl
   mv /tmp/$$ofadapl $sysconfdir/dat.conf
fi

echo OpenIB-cma u1.2 nonthreadsafe default libdaplcma.so.1 dapl.1.2 '"ib0 0" ""'  >>
$sysconfdir/dat.conf
echo OpenIB-cma-1 u1.2 nonthreadsafe default libdaplcma.so.1 dapl.1.2 '"ib1 0" ""'  >>
$sysconfdir/dat.conf
echo OpenIB-mthca0-1 u1.2 nonthreadsafe default libdaplscm.so.1 dapl.1.2 '"mthca0 1" ""' >>
$sysconfdir/dat.conf
echo OpenIB-mthca0-2 u1.2 nonthreadsafe default libdaplscm.so.1 dapl.1.2 '"mthca0 2" ""' >>
$sysconfdir/dat.conf
echo OpenIB-mlx4_0-1 u1.2 nonthreadsafe default libdaplscm.so.1 dapl.1.2 '"mlx4_0 1" ""' >>
$sysconfdir/dat.conf
echo OpenIB-mlx4_0-2 u1.2 nonthreadsafe default libdaplscm.so.1 dapl.1.2 '"mlx4_0 2" ""' >>
$sysconfdir/dat.conf
echo OpenIB-ipath0-1 u1.2 nonthreadsafe default libdaplscm.so.1 dapl.1.2 '"ipath0 1" ""' >>
$sysconfdir/dat.conf
echo OpenIB-ipath0-2 u1.2 nonthreadsafe default libdaplscm.so.1 dapl.1.2 '"ipath0 2" ""' >>
$sysconfdir/dat.conf
echo OpenIB-ehca0-1 u1.2 nonthreadsafe default libdaplscm.so.1 dapl.1.2 '"ehca0 1" ""' >>
$sysconfdir/dat.conf
echo OpenIB-iwarp u1.2 nonthreadsafe default libdaplcma.so.1 dapl.1.2 '"eth2 0" ""'  >>
$sysconfdir/dat.conf
```

Edit and change the **dat.conf** file lines according to the specific details that apply to your cluster.

The following lines are included:

- The **IA** Name
- The **API** version of the library:
  [k|u]major.minor where **major** and **minor** are both integers in decimal format. User-level, examples: **u1.2**, and **u2.0**.
- Library setting for thread-safety: **[threadsafe|nonthreadsafe]**
- Default section setting: **[default|nondefault]**
- The **library image**, **version** included, to be loaded.
- The **vendor id** and version of **DAPL** provider: **id.major.minor** - **ia params**, IA specific parameters - **device name** and **port**
  platform params, (not used)

4.2 Monitoring InfiniBand Networks

See The chapter on *InfiniBand Network Optimization* for details on how to calculate InfiniBand bandwidth and the optimal data rates for the InfiniBand adapters.

4.2.1 InfiniBand Port Counters

Many factors can account for the lack of performance in an **InfiniBand** network. This section lists the counters that should be examined to pinpoint possible issues. The counters have to comply with the **InfiniBand** specifications, before they can be used to check the **InfiniBand** links. They are divided into two main categories: **Traffic Volume** and **Error** related counters.

4.2.2 InfiniBand Traffic Volume Counters

These counters apply to both **Voltaire** and **Mellanox** switches.

Counter	Description
PortXmitData	Total number of data octets, divided by 4, transmitted on all **VLs** from the port selected by **PortSelect**. This includes all octets between (and not including) the start of the packet delimiter and **VCRC**. It excludes all link packets.
PortRcvData	Total number of data octets, divided by 4, received on all VLs from the port selected by **PortSelect**. This includes all octets between (and not including) the start of the packet delimiter and VCRC. It excludes all link packets.
PortXmitPackets	Total number of packets, excluding link packets, transmitted on all VLs from the port.
PortRcvPackets	Total number of packets, excluding link packets, received on all VLs from the port.

Table 4-1. InfiniBand Traffic Volume Counters

4.2.3 InfiniBand Error counters

Counter	Description	Importance
SymbolError	Total number of symbol errors received on one or more Virtual Lanes.	Symbol errors badly affect performance and usually indicate a bad link.
LinkErrorRecovery	Total number of times the Port Training state machine has successfully completed the link error recovery process.	Link errors badly affect performance. If **SymbolErrors** are increasing quickly and this counter is increasing, it may indicate a bad link.

LinkDowned	Total number of times the Port Training state machine has failed the link error recovery process and downed the link	This counter is a true indication of the number of times that the port has gone down (usually for valid reasons). Correlate these errors with the operating system logs or the output of the *last* command checking for machine reboots.
PortRcvErrors	Total number of packets containing an error that were received by a port. These errors include: Local physical errors (CRC,VCRC, FCCRC and all physical errors that cause entry into the BAD PACKET or BAD PACKET DISCARD states of the packet receiver state machine) Malformed data packet errors Malformed link packet errors Packets discarded due to buffer overrun.	Port receive errors badly affect performance. This counter should not be increasing constantly, if so it probably indicates a bad link.
PortRcvRemotePhysicalErrors	Total number of packets marked with the EBP delimiter received by the port.	This indicates that a problem is occurring elsewhere in the fabric and that the port received a packet that was intentionally corrupted by another switch in the fabric. Since these errors are hard to find, it is advised to fix other errors first. These errors may disappear once other errors have been fixed.
PortRcvSwitchRelayErrors	Total number of packets received by the port, and discarded because they could not be forwarded by the switch relay. Reasons for this include: DLID mapping VL mapping Looping (output port = input port)	These errors do not affect performance and can be ignored.
PortXmitDiscards	Total number of outbound packets discarded by the port because the port is down or congested. Reasons for this include: Output port is in the inactive state. Packet length exceeded neighbor MTU. Switch lifetime limit exceeded. Switch HoQ (head of queue) limit exceeded.	Typically will not increase. If it does, it may indicate that HoQ or other parameter should be tweaked. Please contact Bull support for tuning recommendations and guidelines. Lustre can cause these errors. Large clusters require a customised HoQ. See the section on *Specific Settings for Large Clusters* in the *InfiniBand Network Optimizatio*n chapter.

PortXmitConstraintErrors	Total number of packets not transmitted by the port for the following reasons: FilterRawOutbound is true and packet is raw PartitionEnforcementOutbound is true and packet fails partition check, IP version check, or transport header version check.	These errors are linked to partitioning (pkeys) and should not increase. If it does, this may indicate that HoQ or another parameter should be tweaked. Please contact Bull support.
PortRcvConstraintErrors	Total number of packets received on the port that are discarded for the following reasons: FilterRawOutbound is true and packet is raw PartitionEnforcementOutbound is true and packet fails partition check, IP version check, or transport header version check.	These errors are linked to partitioning (pkeys) and should not increase. If it is, this may be an indicator that HoQ or another parameter should be tweaked. Please contact Bull support.
LocalLinkIntegrityErrors	The number of times that the frequency of packets containing local physical errors exceeded local_phy_errors.	This counter increasing in numbers usually indicates a bad link. Try to fix the bad link by reseating the connectors.
ExcessiveBufferOverrunErrors	The number of times that overrun_errors consecutive flow control update periods occurred with at least one overrun error in each period (see the Portinfo Table in the *InfiniBand Architecture Specification Volume 1, release 1.2.1*).	Typically will not increase. If it does, it may indicate that a parameter should be tweaked. Please contact Bull support.
VL15Dropped	Number of incoming VL15 packets dropped due to resource limitations on port selected by PortSelect (due to lack of buffers)	This counter increasing in small increments is not seen as a problem.

Table 4-2. InfiniBand Error Counters

4.3 Troubleshooting Link Speed and Width

The speed and width parameters are very important because they determine the level of performance that can be achieved through a link. The following tools can be used to check the link speed and width.

4.3.1 Subnet Manager Query (smpquery)

smpquery includes a subset of standard **SMP** query options which may be used to bring up information – in a human readable format - for different parts of a InfiniBand network including nodes, ports and switches.

Syntax

smpquery [options] <op> <dest_addr> [op_params]

The example below shows how to check the status of LID 26 / port 1:

```
smpquery portinfo 26 1
```

```
# Port info: Lid 26 port 1
Mkey:...........................0x0000000000000000
GidPrefix:......................0xfe80000000000000
Lid:............................26
SMLid:..........................26
CapMask:........................0x2510a6a
        IsSM
        IsTrapSupported
        IsAutomaticMigrationSupported
        IsSLMappingSupported
        IsLedInfoSupported
        IsSystemImageGUIDsupported
        IsCommunicatonManagementSupported
        IsVendorClassSupported
        IsCapabilityMaskNoticeSupported
        IsClientRegistrationSupported
DiagCode:.......................0x0000
MkeyLeasePeriod:................0
LocalPort:......................1
LinkWidthEnabled:...............1X or 4X
LinkWidthSupported:.............1X or 4X
LinkWidthActive:................4X
LinkSpeedSupported:.............2.5 Gbps or 5.0 Gbps
LinkState:......................Active
PhysLinkState:..................LinkUp
LinkDownDefState:...............Polling
ProtectBits:....................0
LMC:............................0
LinkSpeedActive:................5.0 Gbps
LinkSpeedEnabled:...............2.5 Gbps or 5.0 Gbps
NeighborMTU:....................2048
SMSL:...........................0
VLCap:..........................VL0-7
InitType:.......................0x00
VLHighLimit:....................0
VLArbHighCap:...................8
VLArbLowCap:....................8
InitReply:......................0x00
MtuCap:.........................2048
VLStallCount:...................7
HoqLife:........................31
OperVLs:........................VL0-7
PartEnforceInb:.................0
```

```
PartEnforceOutb:................0
FilterRawInb:...................0
FilterRawOutb:..................0
MkeyViolations:.................0
PkeyViolations:.................0
QkeyViolations:.................0
GuidCap:........................32
ClientReregister:...............0
SubnetTimeout:..................18
RespTimeVal:....................16
LocalPhysErr:...................8
OverrunErr:.....................8
MaxCreditHint:..................0
RoundTrip:......................0
```

4.3.2 ibportstate

ibportstate allows the port state and port physical state of an InfiniBand port to be queried, in addition to link width and speed being validated relative to the peer port when the port queried is a switch port. It can also be used to disable, enable, or reset a switch port. Finally, it also allows the link speed enabled on any InfiniBand port to be adjusted.

Syntax

ibportstate [options] <op> <dest_addr> <portnum> [op_params]

ibportstate can be used to check three complementary aspects of a link for a given attribute (speed or width):

- What the link supports (determined by the physical characteristics of the hardware within the cluster).
- What the link is enabled to do (determined by the software settings).
- What the link negotiated (determined by the previous two attributes and possible hardware issues between the two endpoints connected by the cable).

In all situations, the port (or link) will always try to negotiate the best possible setting.

The example below shows the port state of the adapter with LID 1 on port 1:

```
ibportstate 1 1
```

```
PortInfo:
# Port info: Lid 1 port 1
LinkState:......................Active
PhysLinkState:..................LinkUp
LinkWidthSupported:.............1X or 4X
LinkWidthEnabled:...............1X or 4X
LinkWidthActive:................4X
LinkSpeedSupported:.............2.5 Gbps or 5.0 Gbps or 10.0 Gbps
LinkSpeedEnabled:...............2.5 Gbps or 5.0 Gbps or 10.0 Gbps
LinkSpeedActive:................10.0 Gbps
```

In the above example, the supported, enabled and active speed and width parameters indicate that the best possible performance is in place. However, the error counters for this particular port should also be checked.

See Section *4.1.8* for full details of the InfiniBand port counters.

In the event of hardware failure, the port will fail to negotiate the optimal speed or width. In this case, it may be useful to try and set the appropriate speed using the **ibportstate** command.

The following speed identifiers can be used to set the link speed:

1	2.5 Gbps
3	2.5 or 5.0 Gbps
5	2.5 or 10.0 Gbps
7	2.5 or 5.0 or 10.0 Gbps
2, 4, 6, 8-14	Reserved
Default 15	Set to PortInfo:LinkSpeedSupported

Example

A port running in SDR mode on LID 0xB / port 15:

```
ibportstate 0xB 15
```

```
PortInfo:
# Port info: Lid 11 port 15
LinkState:......................Active
PhysLinkState:..................LinkUp
LinkWidthSupported:.............1X or 4X
LinkWidthEnabled:...............1X or 4X
LinkWidthActive:................4X
LinkSpeedSupported:.............2.5 Gbps or 5.0 Gbps or 10.0 Gbps
LinkSpeedEnabled:...............2.5 Gbps
LinkSpeedActive:................2.5 Gbps
```

Try to negotiate the best possible speed, as shown in the example below:

```
# ibportstate 0xB 15 speed 15
```

```
Initial PortInfo:
# Port info: Lid 11 port 15
LinkSpeedEnabled:...............2.5 Gbps
After PortInfo set:
# Port info: Lid 11 port 15
LinkSpeedEnabled:...............2.5 Gbps or 5.0 Gbps or 10.0 Gbps
```

However, setting the speed is not sufficient: The port needs to be reset for the changes to take effect:

```
smpquery portinfo  0xB 15 | grep LinkSpeedActive
```

```
LinkSpeedActive:................2.5 Gbps
```

```
# ibportstate 0xB 15 reset
```

```
Initial PortInfo:
# Port info: Lid 11 port 15
LinkState:......................Active
PhysLinkState:..................LinkUp
LinkWidthSupported:.............1X or 4X
LinkWidthEnabled:...............1X or 4X
LinkWidthActive:................4X
```

```
LinkSpeedSupported:..............2.5 Gbps or 5.0 Gbps or 10.0 Gbps
LinkSpeedEnabled:................2.5 Gbps or 5.0 Gbps or 10.0 Gbps
LinkSpeedActive:.................2.5 Gbps
After PortInfo set:
# Port info: Lid 11 port 15
LinkState:.......................Down
PhysLinkState:...................Disabled
After PortInfo set:
# Port info: Lid 11 port 15
LinkState:.......................Down
PhysLinkState:...................PortConfigurationTraining
```

The port now uses the best speed:

```
smpquery portinfo  0xB 15 | grep LinkSpeedActive
```

```
LinkSpeedActive:.................10.0 Gbps
```

4.3.3 ibcheckwidth

ibcheckwidth checks all nodes, to validate the bandwidth for links which are active.

```
ibcheckwidth
```

```
## Summary: 7 nodes checked, 0 bad nodes found
##          14 ports checked, 0 ports with 1x width in error found
```

4.3.4 ibcheckportwidth

ibcheckportwidth checks connectivity and the link width for a given port lid and will indicate the actual bandwidth being used by the port. This should be checked against the maximum which is possible. For example, if the port supports 4X bandwidth then this should be used. Similarly, if the adapter supports DDR then this should be used.

Syntax

ibcheckportwidth [-h] [-v] [-N | -nocolor] [-G] [-C ca_name] [-P ca_port] [-t(imeout) timeout_ms] <lid|guid> <port>

```
ibcheckportwidth -v 11 1
```

```
Port check lid 11 port 1:  OK
```

4.4 Troubleshooting Procedure for InfiniBand Hardware Errors

The procedure to use when troubleshooting **InfiniBand** networks to identify hardware errors is:

STEP 1 Clear the Error Counters

Because old errors may still influence the current error counters, the first thing to do while looking for hardware related errors is to reset the error counters in order to remove any old errors which may influence on the counter results. Run the command below to do this

```
ibclearcounters
```

```
## Summary: 7 nodes cleared 0 errors
```

STEP 2 Generate Traffic to test the Network

There are many different ways that traffic can be generated on an InfiniBand network, including:

- MPI
  - IMB
  - ISV software
- The file systems that use the **InfiniBand** interconnects
  - Lustre
  - NFS-RDMA
  - NFS configured to use **IP over IB**
- TCP benchmarks programs using **IP over IB**
  - tcpperf
  - iperf
  - netperf
- Low level **InfiniBand** diagnostics
  - ib_rdma_bw
  - ib_read_bw

ib_rdma_bw example

Because there are so many tests available, it is not feasible to list them all. This section shows how to generate traffic using the **ib_rdma_bw** program. This program is included in the **perftest** package and can be used to test the connection between two endpoints.

In this example, we generate a bidirectional flow between each node: *test_node* is used as the server and *good_node* as the client. *good_node* is known to be functioning correctly. Each side should operate with the same bandwidth rate average (2925 MB/sec in this case). If the bandwidth rate is symmetrical, the test_node link is validated.

Start the **ib_rdma_bw** server on *test_node*:

```
user@test_node$ while :; do ib_rdma_bw -b -s $((1<<17)) -n 10000; done
```

Start the **ib_rdma_bw** client on *good_node*:

```
user@good_node$ while :; do ib_rdma_bw -b -s $((1<<17)) -n 10000 test_node; sleep 1; done
```

The *test_node* output will look something like that below:

```
24046: | port=18515 | ib_port=1 | size=131072 | tx_depth=100 | iters=10000 | duplex=1 |
cma=0 |
24046: Local address:  LID 0xc0, QPN 0x1b0406, PSN 0x445b2e RKey 0xe003a01 VAddr
0x002aaaaaf02000
24046: Remote address: LID 0x108, QPN 0x30406, PSN 0x5a046b, RKey 0x1c003a00 VAddr
0x002aaaaaf06000
24046: Bandwidth peak (#0 to #9999): 2925.49 MB/sec
24046: Bandwidth average: 2925.49 MB/sec
24046: Service Demand peak (#0 to #9999): 887 cycles/KB
24046: Service Demand Avg  : 887 cycles/KB
24232: | port=18515 | ib_port=1 | size=131072 | tx_depth=100 | iters=10000 | duplex=1 |
cma=0 |
24232: Local address:  LID 0xc0, QPN 0x1c0406, PSN 0x8fe3ea RKey 0x16003a01 VAddr
0x002aaaaaf02000
24232: Remote address: LID 0x108, QPN 0x40406, PSN 0xbf9216, RKey 0x24003a00 VAddr
0x002aaaaaf06000
24232: Bandwidth peak (#0 to #9999): 2925.25 MB/sec
24232: Bandwidth average: 2925.25 MB/sec
24232: Service Demand peak (#0 to #9999): 888 cycles/KB
24232: Service Demand Avg  : 888 cycles/KB
```

The output on *good_node* looks as follows:

```
18844: | port=18515 | ib_port=1 | size=131072 | tx_depth=100 | iters=10000 | duplex=1 |
cma=0 |
18844: Local address:  LID 0x108, QPN 0x20406, PSN 0x9fc0fa RKey 0x14003a00 VAddr
0x002aaaaaf06000
18844: Remote address: LID 0xc0, QPN 0x1a0406, PSN 0xa372d, RKey 0x6003a01 VAddr
0x002aaaaaf02000
18844: Bandwidth peak (#0 to #9999): 2925.57 MB/sec
18844: Bandwidth average: 2925.56 MB/sec
18844: Service Demand peak (#0 to #9999): 890 cycles/KB
18844: Service Demand Avg  : 890 cycles/KB
18851: | port=18515 | ib_port=1 | size=131072 | tx_depth=100 | iters=10000 | duplex=1 |
cma=0 |
18851: Local address:  LID 0x108, QPN 0x30406, PSN 0x5a046b RKey 0x1c003a00 VAddr
0x002aaaaaf06000
18851: Remote address: LID 0xc0, QPN 0x1b0406, PSN 0x445b2e, RKey 0xe003a01 VAddr
0x002aaaaaf02000
18851: Bandwidth peak (#0 to #9999): 2925.97 MB/sec
18851: Bandwidth average: 2925.97 MB/sec
18851: Service Demand peak (#0 to #9999): 890 cycles/KB
18851: Service Demand Avg  : 890 cycles/KB
```

In the example above the bandwidth averages are the same, and therefore it is clear that the ports for the nodes are functioning correctly.

In order to diagnose an entire InfiniBand based fabric, it is advised to generate traffic using the All2All IMB functionality. Refer to the appropriate IMB and MPI documentation for details on how to do this.

STEP 3 Retrieve and Analyse the Error Counters

Use the IBS, perfquery and ibdiagnet tools, as described below, to analyse the error counters.

4.4.1 perfquery

perfquery uses Performance Management General Services Management Packets (**GMP**) to obtain the **PortCounters** (basic performance and error counters) from the Performance Management Attributes at the node specified.

Syntax

perfquery [options] [<lid|guid> [[port] [reset_mask]]]

It can be invoked as follows:

```
$ perfquery <lid> <port>
$ perfquery -G <guid> <port>
```

Example

Query the performance and error counters for the node with LID 1 and port 1

```
perfquery 1 1
```

```
# Port counters: Lid 1 port 1
PortSelect:.....................1
CounterSelect:..................0x1000
SymbolErrors:...................0
LinkRecovers:...................0
LinkDowned:.....................1
RcvErrors:......................23
RcvRemotePhysErrors:............0
RcvSwRelayErrors:...............0
XmtDiscards:....................225
XmtConstraintErrors:............0
RcvConstraintErrors:............0
CounterSelect2:.................0
LinkIntegrityErrors:............0
ExcBufOverrunErrors:............0
VL15Dropped:....................0
XmtData:........................3357395770
RcvData:........................170316190
XmtPkts:........................9949136
RcvPkts:........................2666084
XmtWait:........................456
```

In the above example, we can see that 225 packets were discarded. The **openib** service was probably restarted once (LinkDowned = 1), which caused 23 receive errors.

4.4.2 ibs

ibs is a program that is part of the **IBS** software package. This section describes the use of **ibs** to locate the errors.

See	Section *4.1.8* for full details of the InfiniBand port counters.

When troubleshooting the InfiniBand network, the **ibs** *errors* action is the most useful. It produces a list of all the ports that encountered an error in the fabric. The most interesting column is probably EPM. EPM stands for 'error per million' and describes the error rate on the link (the higher the error rate, the worst the link is). In the example below, port 21 of the switch *iswu0c0* definitely needs to be fixed!

```
ibs -S ibnetdiscover -a errors -vE
```

```
HOSTNAME                   | PORT  | PIN   | LID     | LOCATION           | EPM   || REMOTE HOSTNAME
| PORT  | PIN   | LID      | REMOTE LOCATION         | EPM   || ERRORS

------------------------------------------------------------------------------------------
MT25218                    | 1     | 1     | 0x0007  |                    | N/A   || iswu0c0
| 14    | 14    | 0x0001   | [A,1] RACK1/A           | 0.005 || vl15dropped=2,xmtdiscards=1
MT25204                    | 1     | 1     | 0x000E  |                    | 0.000 || iswu0c0
| 20    | 20    | 0x0001   | [A,1] RACK1/A           | 0.000 || vl15dropped=2,xmtdiscards=1
MT25218                    | 1     | 1     | 0x0012  |                    | 6.333 || iswu0c0
| 21    | 21    | 0x0001   | [A,1] RACK1/A           | 181.6 ||
xmtdiscards=12,vl15dropped=4,linkdowned=3
globus-cs                  | 1     | 1     | 0x000F  |                    | 0.555 || iswu0c0
| 9     | 9     | 0x0001   | [A,1] RACK1/A           | 0.6   ||
xmtdiscards=11,vl15dropped=2,symerr=1,linkdowned=1
MT25218                    | 1     | 1     | 0x000D  |                    | 0.001 || iswu0c0
| 5     | 5     | 0x0001   | [A,1] RACK1/A           | 0.001 || xmtdiscards=2,vl15dropped=2
xena1                      | 1     | 1     | 0x0009  | [A,1] RACK1/S      | 0.004 || iswu0c0
| 3     | 3     | 0x0001   | [A,1] RACK1/A           | 0.003 ||
xmtdiscards=25,vl15dropped=10,linkdowned=4,rcverr=1
MT25218                    | 1     | 1     | 0x0004  |                    | 0.000 || iswu0c0
| 13    | 13    | 0x0001   | [A,1] RACK1/A           | 0.003 || vl15dropped=2,xmtdiscards=1
iswu0c0                    | 11    | 11    | 0x0001  | [A,1] RACK1/A      | 7     || MT25218
| 1     | 1     | 0x0006   |                         | N/A   || linkdowned=7
iswu0c0                    | 21    | 21    | 0x0001  | [A,1] RACK1/A      | 181.6 || MT25218
| 1     | 1     | 0x0012   |                         | 6.333 ||
rcverr=538,linkrecovers=5,linkdowned=2
iswu0c0                    | 17    | 17    | 0x0001  | [A,1] RACK1/A      | 4     || localhost
| 1     | 1     | 0x0005   |                         | 1     ||
xmtdiscards=2,linkrecovers=1,linkdowned=1
iswu0c0                    | 2     | 2     | 0x0001  | [A,1] RACK1/A      | 2     || xena0
| 1     | 1     | 0x0008   | [A,1] RACK1/U           | N/A   ||
linkdowned=6,linkrecovers=1,xmtdiscards=1
iswu0c0                    | 22    | 22    | 0x0001  | [A,1] RACK1/A      | 0.333 || MT25218
| 1     | 1     | 0x0013   |                         | 1.333 || linkdowned=1
iswu0c0                    | 1     | 1     | 0x0001  | [A,1] RACK1/A      | 2.545 || MT25218
| 1     | 1     | 0x000A   |                         | 1     || linkdowned=27,xmtdiscards=1
iswu0c0                    | 23    | 23    | 0x0001  | [A,1] RACK1/A      | 16    || MT25218
| 1     | 1     | 0x0002   |                         | N/A   || linkdowned=16
iswu0c0                    | 16    | 16    | 0x0001  | [A,1] RACK1/A      | 4     || xena32
| 1     | 1     | 0x001D   | [B,1] RACK2/L           | 0     || linkdowned=32
iswu0c0                    | 13    | 13    | 0x0001  | [A,1] RACK1/A      | 0.003 || MT25218
| 1     | 1     | 0x0004   |                         | 0.000 ||
rcverr=10,linkdowned=9,linkrecovers=5,xmtdiscards=2
iswu0c0                    | 6     | 6     | 0x0001  | [A,1] RACK1/A      | 4.4   || MT25218
| 1     | 1     | 0x0010   |                         | 2     ||
linkdowned=17,xmtdiscards=4,linkrecovers=1
iswu0c0                    | 3     | 3     | 0x0001  | [A,1] RACK1/A      | 0.003 || xena1
| 1     | 1     | 0x0009   | [A,1] RACK1/S           | 0.004 ||
linkdowned=25,linkrecovers=3,xmtdiscards=3
iswu0c0                    | 9     | 9     | 0x0001  | [A,1] RACK1/A      | 0.6   || globus-cs
| 1     | 1     | 0x000F   |                         | 0.555 ||
xmtdiscards=4,linkrecovers=1,linkdowned=1
iswu0c0                    | 14    | 14    | 0x0001  | [A,1] RACK1/A      | 0.005 || MT25218
| 1     | 1     | 0x0007   |                         | N/A   ||
linkdowned=35,xmtdiscards=6,linkrecovers=1,rcverr=1
iswu0c0                    | 20    | 20    | 0x0001  | [A,1] RACK1/A      | 0.000 || MT25204
| 1     | 1     | 0x000E   |                         | 0.000 || linkdowned=6,linkrecovers=1
iswu0c0                    | 8     | 8     | 0x0001  | [A,1] RACK1/A      | 2.75  || MT25218
| 1     | 1     | 0x0016   |                         | 1.5   || linkdowned=8,xmtdiscards=3
iswu0c0                    | 4     | 4     | 0x0001  | [A,1] RACK1/A      | 0.006 || xena13
| 1     | 1     | 0x0017   | [B,1] RACK2/M           | 0.000 ||
linkdowned=36,rcverr=9,xmtdiscards=7,linkrecovers=5
iswu0c0                    | 24    | 24    | 0x0001  | [A,1] RACK1/A      | 0.195 || MT25204
| 1     | 1     | 0x0019   |                         | 0.046 ||
linkdowned=12,xmtdiscards=4,vl15dropped=1
iswu0c0                    | 5     | 5     | 0x0001  | [A,1] RACK1/A      | 0.001 || MT25218
| 1     | 1     | 0x000D   |                         | 0.001 ||
linkdowned=11,xmtdiscards=4,linkrecovers=1
xena0                      | 1     | 1     | 0x0008  | [A,1] RACK1/U      | N/A   || iswu0c0
| 2     | 2     | 0x0001   | [A,1] RACK1/A           | 2     || xmtdiscards=2,vl15dropped=2
MT25218                    | 1     | 1     | 0x0010  |                    | 2     || iswu0c0
| 6     | 6     | 0x0001   | [A,1] RACK1/A           | 4.4   || xmtdiscards=2,vl15dropped=2
```

```
MT25218                    | 1     | 1      | 0x0006   |                      | N/A   || iswu0c0
| 11     | 11     | 0x0001   | [A,1] RACK1/A            | 7     || vl15dropped=2,xmtdiscards=1
MT25218                    | 1     | 1      | 0x000A   |                      | 1     || iswu0c0
| 1      | 1      | 0x0001   | [A,1] RACK1/A            | 2.545 || vl15dropped=2,xmtdiscards=1
localhost                  | 1     | 1      | 0x0005   |                      | 1     || iswu0c0
| 17     | 17     | 0x0001   | [A,1] RACK1/A            | 4     ||
vl15dropped=3,xmtdiscards=1,linkdowned=1
MT25204                    | 1     | 1      | 0x0019   |                      | 0.046 || iswu0c0
| 24     | 24     | 0x0001   | [A,1] RACK1/A            | 0.195 || vl15dropped=2,xmtdiscards=1
MT25218                    | 1     | 1      | 0x0013   |                      | 1.333 || iswu0c0
| 22     | 22     | 0x0001   | [A,1] RACK1/A            | 0.333 || vl15dropped=3,xmtdiscards=1
MT25218                    | 1     | 1      | 0x0016   |                      | 1.5   || iswu0c0
| 8      | 8      | 0x0001   | [A,1] RACK1/A            | 2.75  || vl15dropped=2,xmtdiscards=1
MT25218                    | 1     | 1      | 0x0002   |                      | N/A   || iswu0c0
| 23     | 23     | 0x0001   | [A,1] RACK1/A            | 16    || vl15dropped=2,xmtdiscards=1
xena13                     | 1     | 1      | 0x0017   | [B,1] RACK2/M        | 0.000 || iswu0c0
| 4      | 4      | 0x0001   | [A,1] RACK1/A            | 0.006 || xmtdiscards=1
```

An error is rarely triggered on its own and usually come in groups. When a node boots up (or when the **openibd** service is restarted), we usually see patterns like these on the host (HCA) side of the cable:

```
xmtdiscards=2,vl15dropped=2
xmtdiscards=1,vl15dropped=2
xmtdiscards=2,vl15dropped=1
```

These errors can be ignored.

The signature looks as follows as seen from the switch:

```
linkdowned=1
```

4.4.3 ibdiagnet

ibdiagnet scans the fabric using directed route packets and extracts all the available information regarding its connectivity and devices. This tool performs a cluster-wide check of the **InfiniBand** fabric:

```
ibdiagnet
```

```
Loading IBDIAGNET from: /usr/lib64/ibdiagnet1.2
-W- Topology file is not specified.
    Reports regarding cluster links will use direct routes.
Loading IBDM from: /usr/lib64/ibdm1.2
-I- Using port 1 as the local port.
-I- Discovering ... 7 nodes (3 Switches & 4 CA-s) discovered.
-I---------------------------------------------------
-I- Bad Guids/LIDs Info
-I---------------------------------------------------
-I- No bad Guids were found
-I---------------------------------------------------
-I- Links With Logical State = INIT
-I---------------------------------------------------
-I- No bad Links (with logical state = INIT) were found
-I---------------------------------------------------
-I- PM Counters Info
-I---------------------------------------------------
-I- No illegal PM counters values were found
-I---------------------------------------------------
-I- Fabric Partitions Report (see ibdiagnet.pkey for a full hosts list)
-I---------------------------------------------------
-I-    PKey:0x0001 Hosts:4 full:4 partial:0
-I-    PKey:0x7fff Hosts:4 full:4 partial:0
-I---------------------------------------------------
```

```
-I- IPoIB Subnets Check
-I-----------------------------------------------
-I- Subnet: IPv4 PKey:0x7fff QKey:0x00000b1b MTU:2048Byte rate:10Gbps SL:0x00
-W- Suboptimal rate for group. Lowest member rate:40Gbps > group-rate:10Gbps
-I- Subnet: IPv4 PKey:0x0001 QKey:0x00000b1b MTU:2048Byte rate:10Gbps SL:0x00
-W- Suboptimal rate for group. Lowest member rate:40Gbps > group-rate:10Gbps
-I-----------------------------------------------
-I- Bad Links Info
-I-----------------------------------------------
-I- No bad link were found
----------------------------------------------------------------
-I- Stages Status Report:
    STAGE                                    Errors Warnings
    Bad GUIDs/LIDs Check                     0      0
    Link State Active Check                  0      0
    Performance Counters Report              0      0
    Partitions Check                         0      0
    IPoIB Subnets Check                      0      2
Please see /tmp/ibdiagnet.log for complete log
----------------------------------------------------------------
-I- Done. Run time was 1 seconds.
```

STEP 4 Fix the Error

Once the error has been identified the faulty piece of equipment should be replaced. This may involve one of the following operations:

a. Reseating the connectors.

b. Resetting the ports.

c. Replacing the faulty equipment (cable, switch, HCA, etc).

STEP 5 Repeat the Troubleshooting Procedure

Check that everything is OK by repeating STEPS 1 to 4 to confirm that the network is functioning correctly.

Chapter 5. InfiniBand Network Optimization

This chapter describes how to monitor the performance of an **InfiniBand** network, and some steps that can be taken to improve it. The optimal performance figures achievable are dependent on the hardware configuration (machines, InfiniBand adapter, cables, switches, etc.).

Important	All commands in this chapter which start with the hash (#) sign must be carried out as root.

5.1 Hardware Constraints

The optimal performance figures achievable are dependent on the hardware configuration (machines, InfiniBand adapter, cables, switches, etc.).

5.1.1 Calculating InfiniBand Bandwidth

Most **InfiniBand** cards use host **PCI Express** buses. The maximum bandwidth possible varies according to the type and version of this bus.

Note	PCI X adapters are not covered by this chapter.

PCI Express bus performance depends on two key factors:

- **PCI Express** bus speed:
  - **PCI Express Generation I** with a maximum transfer rate of 2.5 GT/s (Giga transfers per second).
  - **PCI Express Generation II** with a maximum transfer rate of 5.0 GT/s.
- Bus width. This parameter depends on the chipset used by the machine. Two widths are supported:
  - 8x.
  - 16x.

To calculate the maximum bandwidth that is achievable for a **PCI Express** adapter, the bus speed should be multiplied by the bus width and then multiplied by 0.8 to represent the 8/10 encoding scheme, with its 20 percent overhead.

Maximum theoretical PCI Express Bandwidth = Bus Width x Bus Speed x 0.8

Example for a Gen1 8x PCI Express bus

Link : Speed 2.5 Gb/s, Width x8

Maximum theoretical **PCI Express** bandwidth = 2.5 x 8 x 0.8 = 16 Gb/s or 2.0 GB/s

The **InfiniBand** protocol also uses an 8/10 encoding scheme when transmitting data on the wire. Consequently, the maximum speed for an **InfiniBand** link is:

Maximum IB bandwidth = Maximum theoretical PCI-Express bandwidth x 0.8

The maximum **InfiniBand** bandwidth for the example above is:

Maximum IB bandwidth = 2.5 x 8 x 0.8 x 0.8 = 12.8 Gb/s or 1.6 GB/s

5.1.2 InfiniBand Adapter Data Rates

Three data transfer rates are supported by InfiniBand Host Channel Adapters.

- SDR (Single Data Rate): 2.5 Gb/s
- DDR (Double Data Rate): 5.0 Gb/s
- QDR (Quad Data Rate): 10 Gb/s

InfiniBand cables are bidirectional and use 4 links in each direction. Consequently, each cable has a nominal bandwidth of 4X in each direction. Depending on the technology used, and taking into account the 8/10 encoding overhead, the maximum bandwidth figures possible in each direction for a cable, are:

- SDR running @ 4X: 2.5 Gb/s x 4 x 0.8 = 1.0 GB/s
- DDR running @ 4X: 5.0 Gb/s x 4 x 0.8 = 2.0 GB/s
- QDR running @ 4X: 10.0 Gb/s x 4 x 0.8 = 4.0 GB/s

5.2 Tuning InfiniBand Networks

Note The chapter on *InfiniBand Tools* for details on **InfiniBand** monitoring tools including the port counters.

Once the **InfiniBand** interconnect hardware has been checked and is known to be functioning correctly, the following software configurations should be put into place to ensure that the performance of the **InfiniBand** network is at its optimal level.

Important All commands in this section must be carried out as root.

5.2.1 Subnet Manager Configuration

Background

Assuming the **InfiniBand** interconnect hardware has been configured correctly, the key factor regarding performance for an **InfiniBand** network is the Subnet Manager routing algorithm. Because **InfiniBand** is statically routed (as opposed to dynamically routed), bad routing has a direct impact upon the overall network bandwidth.

In order to achieve connectivity between two end points, the Subnet Manager establishes routes between each **ASIC** in the fabric. In a perfect routing scheme, all physical links within a given level of a Fat Tree based network should be traversed by the same number of routes. However, this is not the case when the routing algorithm is badly configured: some links may not be used, hence forcing the corresponding routes to use the cables that remain available. This leads to resource contention in the network and consequently to lower bandwidth.

5.2.2 OpenSM Routing algorithms

OpenSM provides many different routing algorithms. These algorithms are described in the **OpenSM** section in the *InfiniBand Network Management* chapter. Assuming that the cluster interconnect network is a **Fat Tree** one, the best routing algorithm should also be a **Fat Tree** one.

Modify the **OpenSM** configuration file to set the Fat Tree routing algorithm:

```
routing_engine ftree
```

If the network is not pure Fat Tree, the following **OpenSM** parameters should also be set in the configuration file:

- **root_guid_file**: This parameter sets the **GUID**s for the root nodes (one to a line), for the Up/Down or Fat Tree routing algorithms. The **GUID**s listed correspond to the topmost switches in the interconnect network.
- **cn_guid_file**: This parameter sets the **GUID**s for the Compute Nodes, not root nodes, (one to a line), for the Up/Down or Fat Tree routing algorithms. The **GUID**s listed correspond to the Compute Node ports connected to the lowest switches in the interconnect network.

- **io_guid_file**: This parameter sets the **GUIDs** for the I/O nodes (one to a line), for the Fat-Tree routing algorithm. I/O nodes are allowed to use a reverse traffic flow between switches, to enable connectivity between the nodes.
- **max_reverse_hops**: This parameter sets the maximum number of hops against the flow.

If any of the **root_guid_file** or **cn_guid_file** parameters are incorrect, **OpenSM** will fail to use the Fat Tree routing algorithm and default to **min-hop**. This will result in lower performance.

Please contact Bull Technical Support for assistance regarding these options.

5.2.3 Internet Protocol over InfiniBand setup

IP over IB (**IPoIB**) can be used in two different modes:

- **Datagram**: In datagram mode, all packets are broadcast over the **InfiniBand** network. Although it may be useful for multicast operations, performance is low due to a limited MTU (2048). Note that multicast is not used by **MPI** or **Lustre**.
- **Connected**: In connected mode, point-to-point IB connections are created between the hosts for **IPoIB** transfers. This mode does not allow for broadcast operations. However, it allows for a large MTU (65520) and thus achieves a much higher bandwidth.

Notes
- Within a given **InfiniBand** network, all IP over IB interfaces in the same IP subnet **must** use the same mode.
- The recommended mode for a Bull cluster is **Connected**.

To force an **IPoIB** interface into datagram mode, use the command below:

```
# echo "datagram" > /sys/class/net/interface_name/mode
```

To force an **IPoIB** interface into connected mode, use the command below:

```
# echo "connected" > /sys/class/net/interface_name/mode
```

The default mode of operation (**Connected** or **Datagram**) for **IPoIB** interfaces is set in the **Openibd** service configuration file via the **SET_IPOIB_CM** parameter in the **/etc/infiniband/openib_`uname -r`.conf** configuration file.

Datagram Mode

Set the datagram mode as follows:

```
# Disable IPoIB Connected Mode
SET_IPOIB_CM=no
```

Connected Mode

Set the connected mode as follows:

```
# Enable IPoIB Connected Mode
SET_IPOIB_CM=yes
```

Other openibd Settings

Ensure that the **IP over IB** kernel module is loaded by default in the **/etc/infiniband/openib_`uname -r`.conf** configuration file. The **RDMA** connection manager should also be loaded to ensure that IP over IB is functional:

```
# Load IPoIB
IPOIB_LOAD=yes
# Load RDMA_CM module
RDMA_CM_LOAD=yes
```

Note The **openibd** service needs to be restarted so the modifications are taken into account.

MTU size

Usually, the larger the **MTU**, the higher the bandwidth achieved. However, if for some reason the default **IP over IB** interface **MTU** needs to be modified, the new value can be added to the **/etc/sysconfig/network-scripts/ifcfg-<interface name>** file as follows:

Example: ib0 interface with MTU=16384

```
cat /etc/sysconfig/network-scripts/ifcfg-ib0
```

```
 DEVICE=ib0
 BOOTPROTO=static
 IPADDR=10.12.0.1
 NETMASK=255.255.0.0
 ONBOOT=yes
 MTU=16384
```

The new **MTU** value will be used when the **IP over IB** interface is restarted. Alternatively, the **MTU** can be modified using the **ifconfig** command.

```
# ifconfig ib0 mtu 16384
```

Note If the interface is restarted, the **MTU** value will return to its default value.

5.2.4 Specific Settings for Large Clusters

For large clusters, it is necessary to tweak the parameters for HCA nodes and managed switches.

Node Settings

The following settings assume that the cluster is fitted with **ConnectX HCAs**. The procedure described below must be applied to all nodes within the cluster.

If the cluster exceeds 8192 cores but has less than 16384 cores, tweak **mlx4_core** kernel module parameters by increasing:

a. The number of queue pairs (qp)

b. The number of command queues (cq)

c. The switch lifetime value (SLV)

d. The head of queue (HoQ) parameter

To increase number of queue pairs add the following parameters to the **/etc/modprobe.conf** file:

```
options mlx4_core log_num_qp=17
options mlx4_core log_num_cq=17
```

Then run the following command:

```
# depmod -a
```

And restart **openibd** as follows:

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

Contact Bull Technical Support for specific settings if the cluster has more than 16384 cores.

HoQ /SLV managed switches settings

For large clusters, it is necessary to tweak the **HOQ** (Head of Queue) and **SLV** (Switch Lifetime Value) parameters.

HoQ : Sets the length of time a packet can remain at the head of a VL queue.

SLV : Sets the length of time a packet can remain active in the switch.

See The **Mellanox** and **Voltaire** websites for more information on how to change the settings for these parameters.

5.2.5 MLX4_EN Module Tuning Parameters

A **Mellanox ConnectX InfiniBand** adapter can be used in **Ethernet** mode. The default settings for the network must be changed to facilitate the high bandwidth possible for this adapter.

The following options should be added to the **/etc/sysctl.conf** file:

```
## MLX4_EN tuning parameters ##
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_sack = 0
net.core.netdev_max_backlog = 250000
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.core.rmem_default = 16777216
net.core.wmem_default = 16777216
net.core.optmem_max = 16777216
net.ipv4.tcp_mem = 16777216 16777216 16777216
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
## END MLX4_EN ##
```

After adding these parameters, run the following command:

```
# sysctl -a
```

5.3 More Information

InfiniBandTM Architecture Specification Volume 1 Release 1.2.1, 1727p. Available from the web site: http://www.InfiniBandta.org/content/pages.php?pg=technology_download

InfiniBandTM Architecture Specification Volume 2 Release 1.2.1, 834p. Available on web site: http://www.InfiniBandta.org/content/pages.php?pg=technology_download

Glossary and Acronyms

A

ABI	Application Binary Interface
ACL	Access Control List
ACT	Administration Configuration Tool
ANL	Argonne National Laboratory (MPICH2)
API	Application Programmer Interface
ARP	Address Resolution Protocol
ASIC	Application Specific Integrated Circuit

B

BAS	Bull Advanced Server
BIOS	Basic Input Output System
Blade	Thin server that is inserted in a blade chassis
BLACS	Basic Linear Algebra Communication Subprograms
BLAS	Basic Linear Algebra Subprograms
BMC	Baseboard Management Controller
BSBR	Bull System Backup Restore
BSM	Bull System Manager

C

CGI	Common Gateway Interface
CLI	Command Line Interface
ClusterDB	Cluster Database
CLM	Cluster Management
CMC	Chassis Management Controller
ConMan	A management tool, based on telnet, enabling access to all the consoles of the cluster.
Cron	A UNIX command for scheduling jobs to be executed sometime in the future. A cron is normally used to schedule a job that is executed periodically - for example, to send out a notice every morning. It is also a daemon process, meaning that it runs continuously, waiting for specific events to occur.
CUBLAS	CUDA™ BLAS
CUDA™	Compute Unified Device Architecture
CUFFT	CUDA™ Fast Fourier Transform

CVS	Concurrent Versions System
Cygwin	A Linux-like environment for Windows. Bull cluster management tools use Cygwin to provide SSH support on a Windows system, enabling command mode access.

D

DDN	Data Direct Networks
DDR	Double Data Rate
DHCP	Dynamic Host Configuration Protocol
DLID	Destination Local Identifier
DNS	Domain Name Server: A server that retains the addresses and routing information for TCP/IP LAN users.
DSO	Dynamic Shared Object

E

EBP	End Bad Packet Delimiter
ECT	Embedded Configuration Tool
EIP	Encapsulated IP
EPM	Errors per Million
EULA	End User License Agreement (Microsoft)

F

FDA	Fibre Disk Array
FFT	Fast Fourier Transform
FFTW	Fastest Fourier Transform in the West
FRU	Field Replaceable Unit
FTP	File Transfer Protocol

G

Ganglia	A distributed monitoring tool used to view information associated with a node, such as CPU load, memory consumption, and network load.
GCC	GNU C Compiler
GDB	Gnu Debugger
GFS	Global File System
GMP	GNU Multiprecision Library
GID	Group ID
GNU	GNU's Not Unix

GPL
General Public License

GPT
GUID Partition Table

Gratuitous ARP
A gratuitous ARP request is an Address Resolution Protocol request packet where the source and destination IP are both set to the IP of the machine issuing the packet and the destination MAC is the broadcast address `xx:xx:xx:xx:xx:xx`. Ordinarily, no reply packet will occur. Gratuitous ARP reply is a reply to which no request has been made.

GSL
GNU Scientific Library

GT/s
Giga transfers per second

GUI
Graphical User Interface

GUID
Globally Unique Identifier

H

HBA
Host Bus Adapter

HCA
Host Channel Adapter

HDD
Hard Disk Drive

HoQ
Head of Queue

HPC
High Performance Computing

Hyper-Threading
A technology that enables multi-threaded software applications to process threads in parallel, within each processor, resulting in increased utilization of processor resources.

I

IB
InfiniBand

IBTA
InfiniBand Trade Association

ICC
Intel C Compiler

IDE
Integrated Device Electronics

IFORT
Intel® Fortran Compiler

IMB
Intel MPI Benchmarks

INCA
Integrated Cluster Architecture:
Bull Blade platform

IOC
Input/Output Board Compact with 6 PCI Slots

IPMI
Intelligent Platform Management Interface

IPO
Interprocedural Optimization

IPoIB
Internet Protocol over InfiniBand

IPR
IP Router

iSM
Storage Manager (FDA storage systems)

ISV
Independent Software Vendor

K

KDC	Key Distribution Centre
KSIS	Utility for Image Building and Deployment
KVM	Keyboard Video Mouse (allows the keyboard, video monitor and mouse to be connected to the node)

L

LAN	Local Area Network
LAPACK	Linear Algebra PACKage
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Data Interchange Format: A plain text data interchange format to represent LDAP directory contents and update requests. LDIF conveys directory content as a set of records, one record for each object (or entry). It represents update requests, such as Add, Modify, Delete, and Rename, as a set of records, one record for each update request.
LKCD	Linux Kernel Crash Dump: A tool used to capture and analyze crash dumps.
LOV	Logical Object Volume
LSF	Load Sharing Facility
LUN	Logical Unit Number
LVM	Logical Volume Manager
LVS	Linux Virtual Server

M

MAC	Media Access Control (a unique identifier address attached to most forms of networking equipment).
MAD	Management Datagram
Managed Switch	A switch with no management interface and/or configuration options.
MDS	MetaData Server
MDT	MetaData Target
MFT	Mellanox Firmware Tools
MIB	Management Information Base
MKL	Maths Kernel Library
MPD	MPI Process Daemons
MPFR	C library for multiple-precision, floating-point computations
MPI	Message Passing Interface
MTBF	Mean Time Between Failures

MTU
Maximum Transmission Unit

N

Nagios
A tool used to monitor the services and resources of Bull HPC clusters.

NETCDF
Network Common Data Form

NFS
Network File System

NIC
Network Interface Card

NIS
Network Information Service

NS
NovaScale

NTP
Network Time Protocol

NUMA
Non Uniform Memory Access

NVRAM
Non Volatile Random Access Memory

O

OFA
Open Fabrics Alliance

OFED
Open Fabrics Enterprise Distribution

OPMA
Open Platform Management Architecture

OpenSM
Open Subnet Manager

OpenIB
Open InfiniBand

OpenSSH
Open Source implementation of the SSH protocol

OSC
Object Storage Client

OSS
Object Storage Server

OST
Object Storage Target

P

PAM
Platform Administration and Maintenance Software

PAPI
Performance Application Programming Interface

PBLAS
Parallel Basic Linear Algebra Subprograms

PBS
Portable Batch System

PCI
Peripheral Component Interconnect (Intel)

PDSH
Parallel Distributed Shell

PDU
Power Distribution Unit

PETSc
Portable, Extensible Toolkit for Scientific Computation

PGAPACK
Parallel Genetic Algorithm Package

PM	Performance Manager Platform Management
PMI	Process Management Interface
PMU	Performance Monitoring Unit
pNETCDF	Parallel NetCDF (Network Common Data Form)
PVFS	Parallel Virtual File System

Q

QDR	Quad Data Rate
QoS	Quality of Service: A set of rules which guarantee a defined level of quality in terms of transmission rates, error rates, and other characteristics for a network.

R

RAID	Redundant Array of Independent Disks
RDMA	Remote Direct Memory Access
ROM	Read Only Memory
RPC	Remote Procedure Call
RPM	RPM Package Manager
RSA	Rivest, Shamir and Adleman, the developers of the RSA public key cryptosystem

S

SA	Subnet Agent
SAFTE	SCSI Accessible Fault Tolerant Enclosures
SAN	Storage Area Network
SCALAPACK	SCALable Linear Algebra PACKage
SCSI	Small Computer System Interface
SCIPORT	Portable implementation of CRAY SCILIB
SDP	Socket Direct Protocol
SDPOIB	Sockets Direct Protocol over InfiniBand
SDR	Sensor Data Record Single Data Rate
SFP	Small Form-factor Pluggable transceiver - extractable optical or electrical transmitter/receiver module.
SEL	System Event Log
SIOH	Server Input/Output Hub
SIS	System Installation Suite

SL	Service Level
SL2VL	Service Level to Virtual Lane
SLURM	Simple Linux Utility for Resource Management – an open source, highly scalable cluster management and job scheduling system.
SM	Subnet Manager
SMP	Symmetric Multi Processing: The processing of programs by multiple processors that share a common operating system and memory.
SNMP	Simple Network Management Protocol
SOL	Serial Over LAN
SPOF	Single Point of Failure
SSH	Secure Shell
Syslog-ng	System Log New Generation

T

TCL	Tool Command Language
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
TGT	Ticket-Granting Ticket

U

UDP	User Datagram Protocol
UID	User ID
ULP	Upper Layer Protocol
USB	Universal Serial Bus
UTC	Coordinated Universal Time

V

VCRC	Variant Cyclic Redundancy Check
VDM	Voltaire Device Manager
VFM	Voltaire Fabric Manager
VGA	Video Graphic Adapter
VL	Virtual Lane
VLAN	Virtual Local Area Network
VNC	Virtual Network Computing: Used to enable access to Windows systems and Windows applications from the Bull NovaScale cluster management system.

W

WWPN	World–Wide Port Name

X

XFS	eXtended File System
XHPC	Xeon High Performance Computing
XIB	Xeon InfiniBand
XRC	Extended Reliable Connection: Included in Mellanox ConnectX HCAs for memory scalability

Index

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

REFERENCE
86 A2 42FD 03