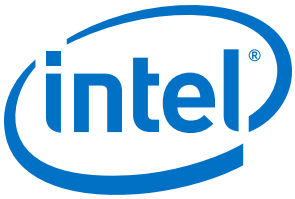


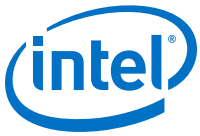
Intel® Omni-Path Fabric Performance Tuning

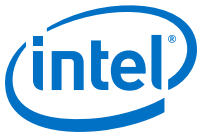
User Guide

Rev. 6.0

December 2016

Order No.: H93143-6.0



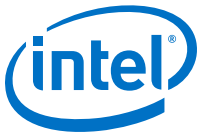


Revision History

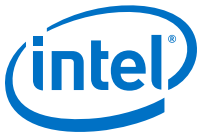
For the latest documentation, go to http://www.intel.com/omnipath/FabricSoftwarePublications.

Date	Revision	Description
December 2016	6.0	Updates to this document include: • Book title updated to include "Fabric". • Added new section " Cluster Configurator for Intel® Omni-Path Fabric" to Preface. • Updated "Performance Tuning Checklist". • Updated "Intel® Xeon Phi™ Product Family x200". • Updated "Tuning for LS-DYNA* Performance". • Added new section "Tuning for MPI Performance on Nodes with Intel® Xeon Phi™ Product Family x200". • Added new section "Tuning for Improved 1 KB to 8 KB Message Bandwidth at High Processes per Node". • Updated "IPoFabric Datagram (UD) Mode Configuration" • Added new section "TCP Parameter Tuning for IPoFabric Performance". • Updated "Driver IRQ Affinity Assignments".
August 2016	5.0	Document has been updated as follows: • Added new Section 1.1 "Performance Tuning Checklist". • Moved Appendix sections to new Section 4, "HFI1 Driver Module Parameters". All subsequent sections incremented appropriately. • Updated "MPI Performance". • Reorganized sections with more focus on procedures.
May 2016	4.0	Document has been updated as follows: • Added Section 8 "Driver IRQ Affinity Assignments". • Added Section 2.2 "Next-GenerationIntel® Xeon Phi™ Processor Family". • Clarified how to make driver parameter changes take effect in A.3 "Setting HFI1 Driver Parameters". • Added Section 4.4 "Tuning for LS-DYNA* Performance".
April 2016	3.0	Document has been updated as follows: • Combined Section 3.2 "Platform Settings" and Section 3.3 "Avoid acpi_pad Consuming CPU Resources" into Section 3.2.2 "Using the ACPI CPUfreq Driver and cpupower Governor". • Added Section 3.2.1 "Using the Intel® P-State Driver". • Updated Section 4.1 "Intel® MPI Library Settings". • Added Section 4.3 "Tuning for MPI Performance on Nodes with Intel® Xeon Phi™ CPUs". • Added Section 5.2 "Parallel File System Concurrency Improvement" • Added Section 5.3 "Lustre Parameter Tuning for Intel Omni-Path". • Added Section 7.2 "IPoFabric Datagram (UD) Mode Configuration". • Updated parameters in Section A.1 "Listing the Driver Parameters" and A.2 "Current Values of Module Parameters". • Clarified example in A.3 "Setting HFI1 Driver Parameters".
February 2016	2.0	Document has been updated as follows: • Clarified language in Section 3.2 "Platform Settings" and 5.1 "irqbalance".
November 2015	1.0	Document has been updated.

Contents



Tables



Preface

This manual is part of the documentation set for the Intel® Omni-Path Fabric (Intel® OP Fabric), which is an end-to-end solution consisting of Intel® Omni-Path Host Fabric Interfaces (HFIs), Intel® Omni-Path switches, and fabric management and development tools.

The Intel® OP Fabric delivers a platform for the next generation of High-Performance Computing (HPC) systems that is designed to cost-effectively meet the scale, density, and reliability requirements of large-scale HPC clusters.

Both the Intel® OP Fabric and standard InfiniBand* are able to send Internet Protocol (IP) traffic over the fabric, or *IPoFabric*. In this document, however, it is referred to as *IP over IB* or *IPoIB*. From a software point of view, IPoFabric and IPoIB behave the same way and, in fact, use the same `ib_ipoib` driver to send IP traffic over the ib0 and/or ib1 ports.

Intended Audience

The intended audience for the Intel® Omni-Path (Intel® OP) document set is network administrators and other qualified personnel.

Documentation Set

The complete end user publications set for the Intel® Omni-Path product includes the following items.

- Hardware Documents:
  - *Intel® Omni-Path Fabric Switches Hardware Installation Guide*
  - *Intel® Omni-Path Fabric Switches GUI User Guide*
  - *Intel® Omni-Path Fabric Switches Command Line Interface Reference Guide*
  - *Intel® Omni-Path Edge Switch Platform Configuration Reference Guide*
  - *Intel® Omni-Path Fabric Managed Switches Release Notes*
  - *Intel® Omni-Path Fabric Externally-Managed Switches Release Notes*
  - *Intel® Omni-Path Host Fabric Interface Installation Guide*
- Software Documents:
  - *Intel® Omni-Path Fabric Software Installation Guide*
  - *Intel® Omni-Path Fabric Suite Fabric Manager User Guide*
  - *Intel® Omni-Path Fabric Suite FastFabric User Guide*
  - *Intel® Omni-Path Fabric Host Software User Guide*
  - *Intel® Omni-Path Fabric Suite Fabric Manager GUI Online Help*
  - *Intel® Omni-Path Fabric Suite Fabric Manager GUI User Guide*

- — *Intel® Omni-Path Fabric Suite FastFabric Command Line Interface Reference Guide*
- — *Intel® Performance Scaled Messaging 2 (PSM2) Programmer's Guide*
- — *Intel® Omni-Path Fabric Performance Tuning User Guide*
- — *Intel® Omni-Path Host Fabric Interface Platform Configuration Reference Guide*
- — *Intel® Omni-Path Fabric Software Release Notes*
- — *Intel® Omni-Path Fabric Manager GUI Release Notes*
- — *Intel® Omni-Path Storage Router Design Guide*
- — *Building Lustre* Servers with Intel® Omni-Path Architecture Application Note*
- — *Intel® Omni-Path Fabric Staging Guide*

Documents are available at the following URLs:

- Intel® Omni-Path Switches Installation, User, and Reference Guides

  http://www.intel.com/omnipath/SwitchPublications
- Intel® Omni-Path Host Fabric Interface Installation, User, and Reference Guides (includes software documents)

  http://www.intel.com/omnipath/FabricSoftwarePublications
- Drivers and Software (including Release Notes)

  http://www.intel.com/omnipath/Downloads

Cluster Configurator for Intel® Omni-Path Fabric

The Cluster Configurator for Intel® Omni-Path Fabric is available at: http://www.intel.com/content/www/us/en/high-performance-computing-fabrics/omni-path-configurator.html.

This tool generates sample cluster configurations based on key cluster attributes, including a side-by-side comparison of up to four cluster configurations. The tool also generates parts lists and cluster diagrams.

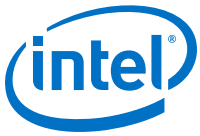
Documentation Conventions

The following conventions are standard for Intel® Omni-Path documentation:

- *Note:* provides additional information.
- ***Caution:*** indicates the presence of a hazard that has the potential of causing damage to data or equipment.
- ***Warning:*** indicates the presence of a hazard that has the potential of causing personal injury.
- Text in blue font indicates a hyperlink (jump) to a figure, table, or section in this guide. Links to websites are also shown in blue. For example:

  See License Agreements on page 9 for more information.

  For more information, visit www.intel.com.
- Text in **bold** font indicates user interface elements such as menu items, buttons, check boxes, key names, key strokes, or column headings. For example:



Click the **Start** button, point to **Programs**, point to **Accessories**, and then click **Command Prompt**.

Press **CTRL+P** and then press the **UP ARROW** key.

- Text in `Courier` font indicates a file name, directory path, or command line text. For example:

  Enter the following command: `sh ./install.bin`

- Text in *italics* indicates terms, emphasis, variables, or document titles. For example:

  Refer to *Intel® Omni-Path Fabric Software Installation Guide* for details.

  In this document, the term *chassis* refers to a managed switch.

Procedures and information may be marked with one of the following qualifications:

- **(Linux)** – Tasks are only applicable when Linux* is being used.
- **(Host)** – Tasks are only applicable when Intel® Omni-Path Fabric Host Software or Intel® Omni-Path Fabric Suite is being used on the hosts.
- **(Switch)** – Tasks are applicable only when Intel® Omni-Path Switches or Chassis are being used.
- Tasks that are generally applicable to all environments are not marked.

License Agreements

This software is provided under one or more license agreements. Please refer to the license agreement(s) provided with the software for specific detail. Do not install or use the software until you have carefully read and agree to the terms and conditions of the license agreement(s). By loading or using the software, you agree to the terms of the license agreement(s). If you do not wish to so agree, do not install or use the software.

Technical Support

Technical support for Intel® Omni-Path products is available 24 hours a day, 365 days a year. Please contact Intel Customer Support or visit www.intel.com for additional detail.

1.0 Introduction

The Intel® Omni-Path Architecture (Intel® OPA) is designed for excellent out-of-the-box performance. However, you can further tune the performance to better meet the needs of your system.

This document describes BIOS settings and parameters that have been shown to improve performance, or make performance more consistent, on Intel® Omni-Path Architecture. If you are interested in benchmarking the performance of your system, these tips may help you obtain better performance.

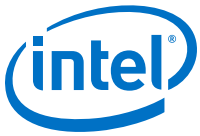
1.1 Performance Tuning Checklist

The table below provides a checklist for all the tuning instructions found in this User Guide.

Table 1. Performance Tuning Checklist

Task	Section
Set recommended BIOS settings	• Intel® Xeon® Processor E5 v3 and v4 Families • Intel® Xeon Phi™ Product Family x200
Distribute hardware interrupts across processors via irqbalance (Linux*)	irqbalance
Reduce CPU clock frequency to save power and cooling requirements	• Using the Intel® P-State Driver • Using the ACPI CPUfreq Driver and cpupower Governor
• Re-enable Intel P-State Driver	Using the Intel® P-State Driver
• Run CPU at its maximum non-Turbo frequency (Intel® P-State Driver).	Using the Intel® P-State Driver
• Run CPU at its maximum Turbo frequency (Intel® P-State Driver).	Using the Intel® P-State Driver
• Enable ACPI CPUfreq Driver and cpupower Governor	Using the ACPI CPUfreq Driver and cpupower Governor
• Set the CPU clock frequncy and power governor to reduce run-to-tun performance variations (acpi_cpufreq driver)	Using the ACPI CPUfreq Driver and cpupower Governor
• Run CPU at its maximum Turbo frequency (acpi_cpufreq driver)	Using the ACPI CPUfreq Driver and cpupower Governor
Blacklist acpi-pad to avoid consuming CPU resources (RHEL 7.1 or similar Linux distribution)	Avoid acpi_pad Consuming CPU Resources
Get a list of the HFI1 driver module parameters	Listing the Driver Parameters
Get a list of the current values for the module parameters	Current Values of Module Parameters
Set or change the HFI1 driver module parameters	Setting HFI1 Driver Parameters
Run MPIs over the PSM2 library	MPI Performance
Ensure you are using PSM2 library	Intel® MPI Library Settings

continued...



Task	Section
Reduce the MPI start-up time in large node-count runs when using Intel® MPI 5.x	Intel® MPI Library Settings
Pin the MPI benchmark process to a core on a socket	Intel® MPI Benchmarks (IMB) or OSU Micro Benchmarks (OMB)
Pin the MPI rank to a socket	Intel® MPI Benchmarks (IMB) or OSU Micro Benchmarks (OMB)
Set the eager_buffer_size for LS-DYNA*	Tuning for LS-DYNA* Performance
Set the user context per node (Intel® Xeon Phi™ Product Family x200)	Tuning for MPI Performance on Nodes with Intel® Xeon Phi™ Product Family x200
Set mtu_max to improve large-message MPI bandwidth (Intel® Xeon Phi™ Product Family x200)	Tuning for MPI Performance on Nodes with Intel® Xeon Phi™ Product Family x200
Set the MPI rank on one or more processors using taskset (Intel® Xeon Phi™ Product Family x200)	Tuning for MPI Performance on Nodes with Intel® Xeon Phi™ Product Family x200
Set the maximum number of user context for 288 MPI ranks (Intel® Xeon Phi™ Product Family x200)	Tuning for MPI Performance on Nodes with Intel® Xeon Phi™ Product Family x200
Set tuning for improved 1 KB to 8 KB message bandwidth at high processes per node	Tuning for Improved 1 KB to 8 KB Message Bandwidth at High Processes per Node
Turn on the Adaptive Cache Memcpy heuristic	Parallel File System Concurrency Improvement
Improve parallel file system scalability using kernel receive queues (krcvqs)	Parallel File System Concurrency Improvement
Accept defaults for Lustre* V2.8 and newer—no changes required	Lustre Parameter Tuning for Intel Omni-Path
Modify Lustre* settings, for versions earlier than 2.8	Lustre Parameter Tuning for Intel Omni-Path
Use perftest to measure latency and bandwidth.	Perftest
Select the 8KB MTU size in the `ib_write_bw` test	RDMA Performance
Enable the 8K MTU manually	IPoFabric Connected Mode Configuration
Enable the 8K MTU automatically	IPoFabric Connected Mode Configuration
Set Datagram (UD) mode	IPoFabric Datagram (UD) Mode Configuration
Increase the size of MTU in Datagram (UD) mode	IPoFabric Datagram (UD) Mode Configuration
Improve IPoFabric traffic performance through TCP Parameters	TCP Parameter Tuning for IPoFabric Performance
Run qperf test	qperf
Run iperf to improve IpoIB throughput	iperf
Identify affinity assignments for SDMA engines	Driver IRQ Affinity Assignments
Map MPI processes to SDMA engines	Driver IRQ Affinity Assignments

2.0 BIOS Settings

Setting the system BIOS is an important step in configuring a cluster to provide the best mix of application performance and power efficiency. In this chapter, we specify settings that can maximize the Intel® Omni-Path Fabric and application performance. Optimally, settings similar to these should be used during a cluster bring-up and validation phase in order to show that the fabric is performing as expected. For the long term, you may want to set the BIOS to provide more power savings, even though that will reduce overall application and fabric performance to some extent.

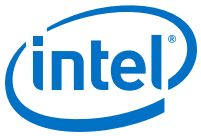
2.1 Intel® Xeon® Processor E5 v3 and v4 Families

The performance-relevant BIOS settings on a server with Intel® Xeon® Processor E5 V3 and V4 Family CPUs, recommended for all-around performance with an Intel® Omni-Path fabric, are shown in the table below:

Table 2. Recommended BIOS Settings for Intel® Xeon® Processor E5 v3 and v4 Families

BIOS Setting	Value
CPU Power and Performance Policy	Performance or Balanced Performance[1]
Workload Configuration	Balanced
Uncore Frequency Scaling	Enabled
Performance P-limit	Enabled
Enhanced Intel SpeedStep® Technology	Enabled
Intel Configurable TDP	Disabled
Intel® Turbo Boost Technology	Enabled
Intel® VT for Directed I/O (VT-d)	Disabled
Energy Efficient Turbo	Enabled
CPU C-State	Enabled
Processor C3	Disabled
Processor C6	Enabled
Intel® Hyper-Threading Technology	No recommendation (Test with your applications. to see if a benefit occurs.)
IOU Non-posted Prefetch	Disabled (where available)[2]
Cluster-on-Die	Disabled
Early Snoop	Disable
NUMA Optimized	Enable[3]
MaxPayloadSize	Auto or 256B[4]

continued...



BIOS Setting	Value
MaxReadReq	512B or 4096B[4]
Snoop Holdoff Count	9

Notes:
1. To get the most consistent Turbo mode performance for demanding workloads, set this to "Performance". Either Performance or Balanced Performance will result in good Intel® Omni-Path Fabric performance.
2. Available in the Intel® Xeon® Processor E5 v4 BIOS version R016.
3. Also known as `Memory.SocketInterleave=NUMA` in some BIOSes.
4. PCIe* Max Payload Size and Max Read Req settings are sometimes not in the BIOS. In that case, the Intel® OPA driver (hfi1) parameter `pcie_caps=0x51` setting (which implies setting *MaxPayLoad* to 256B and *MaxReadReq* to 4096B) can be made as described in HFI1 Driver Module Parameters on page 18.

2.2 Intel® Xeon Phi™ Product Family x200

For the Intel® Xeon Phi™ Product Family x200 CPUs, Intel recommends the following:

1. Install the latest BIOS version available from your vendor with these or similar settings.

Table 3. Recommended BIOS Settings for Intel® Xeon Phi™ Product Family x200

BIOS Setting	Value
Uncore setting: Cluster Mode	Customer-choice based on application[a]
Intel® VT for Directed I/O (VT-d)	Enabled[b]
Processor C6	Enabled
Snoop Holdoff Count	9

Notes:
a. Application-dependent, but "Quadrant" mode has shown some benefit to peak MPI message rate.
b. Intel® Xeon Phi™ Processors operate in X2Apic Mode, which requires that the Intel® VT for Directed I/O (VT-d) remain enabled. When verifying host settings via `verifyhosts.sh` script or `opaverifyhosts`, please do not select/include the vtd test.

2. Use the default settings, including Intel® Turbo Boost Technology= Enabled.
3. Enable Turbo speeds as specified in CPU Frequency Scaling Drivers on page 14.

3.0 Linux* Settings

Intel recommends the following settings to enable consistent performance measurements on the Linux* distributions supported with Intel® Omni-Path Fabric Host Software.

3.1 irqbalance

The purpose of irqbalance is to distribute hardware interrupts across processors on a multiprocessor system in order to increase performance. You set `--hintpolicy` to `exact` to work with the Receive and SDMA interrupt algorithms in the HFI1 driver.

To implement the `irqbalance` setting, perform the following steps.

1. Add a line to the `/etc/sysconfig/irqbalance` file, such as the line below:

```
IRQBALANCE_ARGS=--hintpolicy=exact
```

2. *After* the HFI1 driver is loaded, restart the `irqbalance` service:

```
/bin/systemctl restart irqbalance.service
```

3.2 CPU Frequency Scaling Drivers

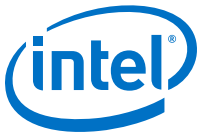
Methods for power saving on CPUs can impact performance inversely. By reducing the CPU clock frequency based on sustained demand and thermal conditions, CPUs reduce power consumption. This can result in substantial savings on power and cooling requirements. However, this can reduce the performance or make performance measurements more variable. Thermal conditions are not predictable, resulting in a run-to-run variation in CPU performance.

The default scaling driver in RHEL* 7 is the Intel P-State (`intel_pstate`). An alternative driver called the ACPI CPUfreq (`acpi_cpufreq`) is also available. Both have their advantages and disadvantages, but only one can be used. In this section we will describe how to use each driver for consistent, best-effort performance measurements. Setting your frequency scaling driver for maximum performance is advisable during cluster/fabric bring-up when trying to determine if all components of the cluster are performing up to their full capabilities.

For long-run operation of a production cluster/super-computer, settings other than those described in the following sections may be desired to scale up for performance when loaded, and to scale down for energy savings when idle.

3.2.1 Using the Intel® P-State Driver

The Intel® P-State Driver is the default driver for RHEL* 7, so no additional setup is required. This driver is recommended for use with Intel® Xeon Phi™ Processors, and may be used with Intel® Xeon® Processors as well.



If you have previously disabled the P-state driver, you must re-enable it.

To re-enable the P-state driver:

1. In `/etc/default/grub`, remove `intel_pstate=disable` from the GRUB_CMDLINE_LINUX command line.
2. Apply the change using:

```
     if [ -e /boot/efi/EFI/redhat/grub.cfg ]; then
    GRUB_CFG=/boot/efi/EFI/redhat/grub.cfg
  else if [ -e /boot/grub2/grub.cfg ]; then
       GRUB_CFG=/boot/grub2/grub.cfg
grub2-mkconfig -o $GRUB_CFG
```

3. Reboot.

Note: The Intel® P-State Driver works better with RHEL* 7.2 than with RHEL* 7.1. With RHEL* 7.1, users may experience difficultly "re-enabling" intel_pstate after disabling it. Also, with Intel® Xeon Phi™ Processor nodes, RHEL* 7.2 is recommended for stable intel_pstate operation.

To run the CPU at its maximum non-Turbo frequency (P1) without scaling to lower frequencies, as root set the minimum frequency to 100% as shown below:

```
echo 100 > /sys/devices/system/cpu/intel_pstate/min_perf_pct
```

To run the CPU at its maximum Turbo frequency, in the BIOS, set the following values:

- Set **Intel® Turbo Boost Technology ➢ Enabled**
- If it is in your BIOS, set **Advanced ➢ Advanced Power Management Configuration ➢ CPU P State Control ➢ Turbo Mode**
- Set `echo 0 > /sys/devices/system/cpu/intel_pstate/no_turbo`
- Set the cpufreq policy to "performance": `cpupower frequency-set -g performance`

  For information about the CPU frequency driver running and other frequency information, use the command:

  `cpupower frequency-info`

For more information on controlling and tuning the behavior of the Intel® P-State driver, please consult https://www.kernel.org/doc/Documentation/cpu-freq/intel-pstate.txt .

3.2.2 Using the ACPI CPUfreq Driver and cpupower Governor

Note: If you are satisfied with the behavior of your system when using the Intel® P-State driver, you do not need to set up the acpi_cpufreq driver.

The ACPI CPUfreq (acpi_cpufreq) driver, in conjunction with cpupower, can be used to set a consistent CPU clock rate on all CPU cores.

To enable the ACPI CPUfreq driver:

1. Disable intel_pstate in the kernel command line:

Edit `/etc/default/grub` by adding `intel_pstate=disable` to `GRUB_CMDLINE_LINUX`.

For example:

```
GRUB_CMDLINE_LINUX=vconsole.keymap=us console=tty0
vconsole.font=latarcyrheb-sun16 crashkernel=256M
console=ttyS0,115200 intel_pstate=disable
```

2. Apply the change using:

```
     if [ -e /boot/efi/EFI/redhat/grub.cfg ]; then
    GRUB_CFG=/boot/efi/EFI/redhat/grub.cfg
  else if [ -e /boot/grub2/grub.cfg ]; then
       GRUB_CFG=/boot/grub2/grub.cfg
grub2-mkconfig -o $GRUB_CFG
```

3. Reboot.

   When the system comes back up with `intel_pstate` disabled, the acpi_cpufreq driver is loaded. This allows CPU frequencies to be set using `cpupower`.

To reduce run-to-run performance variations, Intel recommends that you pin the CPU clock frequency to a specific value, and that you use the `Performance` setting of the CPU power governor.

To set the CPU clock frequency and power governor:

1. Set the clock frequency values and governor using the command line below.

```
 sudo cpupower -c all frequency-set --min <value> --max <value>
\ -g Performance
```

   Where *<value>* is a valid number and unit (GHz) for min and max settings. Note the values can be the same.

For example, the following command will set the frequency of all cores to a value of 2.6 GHz and `Performance` governor, when using the acpi-cpufreq driver.

```
sudo cpupower -c all frequency-set --min 2.6GHz --max 2.6GHz \
-g Performance
```

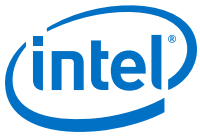
Note: The power savings will diminish and the heat dissipation will increase in the server chassis if the above scheme is used.

To get the maximum advantage from Turbo mode:

1. Ensure that Turbo mode is set to Enabled inn the BIOS (as recommended in BIOS Settings on page 12).
2. Set the frequencies appending "01" to the clock rate. This will enable the Turbo advantage.

For example, if running on an Intel® Xeon® Processor E5-2699 v3 (nominal 2.3 GHz clock rate), then the corresponding command option would be:

```
sudo cpupower -c all frequency-set --min 2.301GHz --max 2.301GHz \
-g Performance
```



3.3 Avoid acpi_pad Consuming CPU Resources

Note: This section is not applicable if you are using RHEL 7.2, SLES12, or a similarly new Linux* distribution.

acpi_pad is a module ACPI Processor Aggregator Driver. It handles high-core-count processor power management.

When running RHEL 7.1 or similar with HT disabled, the acpi_pad driver can cause a system to run acpi_pad and consume 100% of each core. A simple workaround is to keep the driver from loading at boot time by blacklisting acpi_pad.

To blacklist the acpi_pad driver, add the following line to `/etc/modprobe.d/blacklist.conf`:

```
blacklist acpi_pad
```

3.4 Transparent Huge Pages

Transparent Huge Pages is set to "always" enabled in RHEL 7.2 by default. Changing this setting to "never" will hurt large message bandwidth (above 64 MB) significantly. If the default is set, this file should show the following output:

```
$ cat /sys/kernel/mm/transparent_hugepage/enabled
[always] madvise never
```

If it does not, you can set it correctly this way, on all the nodes that don't have "always" set, use this command:

```
echo always > /sys/kernel/mm/transparent_hugepage/enabled
```

4.0 HFI1 Driver Module Parameters

Default settings for HFI1 Driver Module parameters currently achieve the best performance. However to further tune your performance, you can modify specific parameters as described in this document.

This chapter describes:

1. the list of HFI1 driver module parameters;
2. how to set them; and,
3. how to activate the changes.

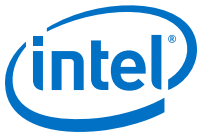
4.1 Listing the Driver Parameters

To get a listing and brief description of the HFI1 driver module parameters, enter the following command:

```
$ modinfo hfi1
```

Results:

```
filename:       /lib/modules/3.10.0-229.20.1.el7.x86_64/updates/
                drivers/infiniband/hw/hfi1/hfi1.ko
version:        0.10-158
description:    Intel Omni-Path Architecture driver
license:        Dual BSD/GPL
rhelversion:    7.1
srcversion:     59650D5AAB893A6594F25F8
alias:          pci:v00008086d000024F1sv*sd*bc*sc*i*
alias:          pci:v00008086d000024F0sv*sd*bc*sc*i*
depends:        ib_core,compat,ib_mad
vermagic:       3.10.0-229.20.1.el7.x86_64 SMP mod_unload modversions
parm:           lkey_table_size:LKEY table size in bits (2^n, 1 <= n
                <= 23) (uint)
parm:           max_pds:Maximum number of protection domains to
                support (uint)
parm:           max_ahs:Maximum number of address handles to support
                (uint)
parm:           max_cqes:Maximum number of completion queue entries
                to support (uint)
parm:           max_cqs:Maximum number of completion queues to
                support (uint)
parm:           max_qp_wrs:Maximum number of QP WRs to support (uint)
parm:           max_qps:Maximum number of QPs to support (uint)
parm:           max_sges:Maximum number of SGEs to support (uint)
parm:           max_mcast_grps:Maximum number of multicast groups to
                support (uint)
parm:           max_mcast_qp_attached:Maximum number of attached QPs
                to support (uint)
parm:           max_srqs:Maximum number of SRQs to support (uint)
parm:           max_srq_sges:Maximum number of SRQ SGEs to support
                (uint)
parm:           max_srq_wrs:Maximum number of SRQ WRs support (uint)
parm:           piothreshold:size used to determine sdma vs. pio
                (ushort)
```



```
parm:           sge_copy_mode:Verbs copy mode: 0 use memcpy, 1 use
                cacheless copy, 2 adapt based on WSS (uint)
parm:           wss_threshold:Percentage (1-100) of LLC to use as a
                threshold for a cacheless copy (uint)


parm:           wss_clean_period:Count of verbs copies before an
                entry in the page copy table is cleaned (uint)
parm:           sdma_comp_size:Size of User SDMA completion ring.
                Default: 128 (uint)
parm:           cache_size:Send and receive side cache size limit (in
                MB) (ulong)
parm:           sdma_descq_cnt:Number of SDMA descq entries (uint)
parm:           sdma_idle_cnt:sdma interrupt idle delay (ns,default
                250) (uint)
parm:           num_sdma:Set max number SDMA engines to use (uint)
parm:           desct_intr:Number of SDMA descriptor before interrupt
                (uint)
parm:           qp_table_size:QP table size (uint)
parm:           pcie_caps:Max PCIe tuning: Payload (0..3), ReadReq
                (4..7) (int)
parm:           aspm:PCIe ASPM: 0: disable, 1: enable, 2: dynamic
                (uint)
parm:           pcie_target:PCIe target speed (0 skip, 1-3 Gen1-3)
                (uint)
parm:           pcie_force:Force driver to do a PCIe firmware
                download even if already at target speed (uint)
parm:           pcie_retry:Driver will try this many times to reach
                requested speed (uint)
parm:           pcie_pset:PCIe Eq Pset value to use, range is 0-10
                (uint)
parm:           num_user_contexts:Set max number of user contexts to
                use (uint)
parm:           krcvqs:Array of the number of non-control kernel
                receive queues by VL (array of uint)
parm:           rcvarr_split:Percent of context's RcvArray entries
                used for Eager buffers (uint)
parm:           eager_buffer_size:Size of the eager buffers, default:
                2MB (uint)
parm:           rcvhdrcnt:Receive header queue count (default 2048)
                (uint)
parm:           hdrq_entsize:Size of header queue entries: 2 - 8B,
                16 - 64B (default), 32 - 128B (uint)
parm:           user_credit_return_threshold:Credit return threshold
                for user send contexts, return when unreturned
                credits passes this many blocks (in percent of
                allocated blocks, 0 is off) (uint)
parm:           max_mtu:Set max MTU bytes, default is 8192 (uint)
parm:           cu:Credit return units (uint)
parm:           prescan_rxq:Used to toggle rx prescan. Set to 1 to
                enable prescan (uint)
parm:           cap_mask:Bit mask of enabled/disabled HW features
parm:           kdeth_qp:Set the KDETH queue pair prefix (uint)
parm:           num_vls:Set number of Virtual Lanes to use (1-8)
                (uint)
parm:           rcv_intr_timeout:Receive interrupt mitigation timeout
                in ns (uint)
parm:           rcv_intr_count:Receive interrupt mitigation count
                (uint)
parm:           link_crc_mask:CRCs to use on the link (ushort)
parm:           loopback:Put into loopback mode (1 = serdes,
                3 = external cable (uint)
parm:           mem_affinity:Bitmask for memory affinity control:
                0 - device, 1 - process (uint)
```

4.2 Current Values of Module Parameters

To list the current values for the module parameters, run the following short script:

```
for x in /sys/module/hfi1/parameters/*; do echo $(basename $x)
       \ $(cat $x); done
```

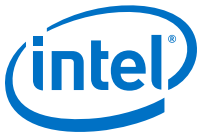
Output from script:

```
aspm 0
cache_size 256
cap_mask 0x4c09a00cb9a
cu 1
desct_intr 64
eager_buffer_size 2097152
hdrq_entsize 32
kdeth_qp 128
krcvqs 4
link_crc_mask 3
lkey_table_size 16
loopback 0
max_ahs 65535
max_cqes 196607
max_cqs 131071
max_mcast_grps 16384
max_mcast_qp_attached 16
max_mtu 8192
max_pds 65535
max_qps 16384
max_qp_wrs 16383
max_sges 96
max_srqs 1024
max_srq_sges 128
max_srq_wrs 131071
mem_affinity 0
num_sdma 0
num_user_contexts 28
num_vls 8
pcie_caps 0
pcie_force 0
pcie_pset 255
pcie_retry 5
pcie_target 3
piothreshold 256
prescan_rxq 0
qp_table_size 256
rcvarr_split 25
rcvhdrcnt 2048
rcv_intr_count 16
rcv_intr_timeout 840
sdma_comp_size 128
sdma_descq_cnt 2048
sdma_idle_cnt 250
sge_copy_mode 0
user_credit_return_threshold 33
wss_clean_period 256
wss_threshold 80
```

4.3 Setting HFI1 Driver Parameters

Note: The settings in this section are example only and not recommended for general use.

To set or change the HFI1 driver module parameters, *as root* perform the following:



1. Edit `hfi1.conf`.

   Example:

   ```
   $ cat /etc/modprobe.d/hfi1.conf
   options hfi1 pcie_caps=0x51 krcvqs=3
   ```

2. Determine if dracut needs to be run:
   - If the following sequence happens, run the dracut command as described in Step 3.
     a. At the start of boot, initramfs is all that is visible.
     b. The hfi1 driver is loaded while only the initramfs is visible.
     c. The `hfi1.conf` file **within the initramfs** is used.

     Note: If you are using Red Hat Linux*, dracut must be run.
   - If one of the following happens, then the dracut command is not needed. Skip to Step 4.
     — If you reload the driver while the OS is running, the initramfs is not used.
     — If the hfi1 driver is loaded after the initramfs stops being used, then the initramfs is not used.
3. Run the `/usr/bin/dracut -f` command to force `/etc/modprobe.d/hf1.conf` into the initramfs image.

   Note: With this dracut command, you may also need to add the following line to `/etc/dracut.conf`:

   ```
   install_items+="/etc/modprobe.d/hfi1.conf /etc/modprobe.d/hfi1.conf"
   ```

4. Reboot the system.

After the system comes up from the reboot, you can run the script listed in Current Values of Module Parameters on page 20 to see if your changes to `hfi1.conf` took effect.

The `krcvqs=3` setting only affects the first VL. To set a krcvqs value of 3 in case eight VLs were set in the Fabric Manager, the krcvqs values would look like:

```
options hfi1 krcvqs=3,3,3,3,3,3,3,3
```

5.0 MPI Performance

Message Passing Interface (MPI) libraries are a key type of middleware for building high-performance computing (HPC) applications. The Intel® Omni-Path Software package includes several builds of Open MPI and MVAPICH2. You can see the choices by looking at the two subdirectories located under `/usr/mpi` using the '`ls -r`' command as shown below:

```
/usr/mpi] $ ls -r *
intel:
openmpi-1.10.2-hfi  mvapich2-2.1-hfi
gcc:
openmpi-1.10.2-hfi  openmpi-1.10.2  mvapich2-2.1-hfi  mvapich2-2.1
```

The '**intel**' and '**gcc**' directories show what compiler suite is used to build the MPI library: Intel Parallel Studio (including the Intel C++ Compiler) or the Gnu Compiler Collection (GCC), respectively.

For best performance, run MPIs over the PSM2 library included with Intel® Omni-Path Fabric Host Software.

To run MPIs over the PSM2 library:

1. Use the MPIs with '**hfi**' in their name.
2. Source a '`mpivars.sh`' file from the `bin` directory of one of the MPIs from your Linux* shell's startup scripts.

   For example, include the following statement in a startup script such as `.bashrc`:

   ```
   source /usr/mpi/gcc/openmpi-1.10.2-hfi/bin/mpivars.sh
   ```

   This will set the PATH and LD_LIBRARY_PATH and MANPATH variables for this MPI version.
3. Use the options in your `mpirun` command to specify the use of PSM2.

   For example, to use the Open MPI version indicated in Step 2, use:

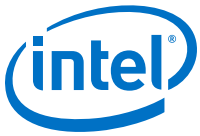
   ```
   mpirun -mca pml cm -mca mtl psm2 …
   ```

   Note: If you use the MVAPICH2 library with 'hfi' in the name of its root directory (such as `/usr/mpi/intel/mvapich2-2.1-hfi`), then no special `mpirun` options are needed to use the PSM2 library.

5.1 Intel® MPI Library Settings

To ensure you are using the PSM2 library, use one of the following `mprun`/`mpiexec` options:

- `-PSM2`



Uses the Intel® MPI Library's shared memory communications

- `-genv I_MPI_FABRICS shm:tmi`

  Uses the Intel® MPI Library's shared memory communications

- `-genv I_MPI_FABRICS tmi`

  Uses PSM2 for shared memory within a node

In general, `-PSM2` or `shm:tmi` performs the best overall; but this is application-dependent.

To reduce the MPI start-up time in large node-count runs when using Intel® MPI 5.x, one of the `mpirun` options should be:

```
-genv I_MPI_HYDRA_PMI_CONNECT alltoall
```

Note: If you are using Intel® MPI 2017 or a later version, the above options are part of the default and need not be specified.

5.2 Intel® MPI Benchmarks (IMB) or OSU Micro Benchmarks (OMB)

To pin the benchmark process to a core on a socket:

When running the MPI microbenchmarks with one process per node, and with any MPI, it is important to pin the benchmark process to a core on the socket that is local to the PCIe slot occupied by the Host Fabric Interface (HFI).

For example, if the PCIe slot with the HFI is connected to socket 1 (of 0 and 1), then you could use "`taskset -c N ./IMB-MPI1`" to pin the executable to a core N on socket 1.

To pin the MPI rank to a socket:

When using Intel® MPI Library 5.1, or newer version, for running the OSU Micro Benchmarks or Intel® MPI Benchmarks with one MPI rank active per node, you need to pin the rank to the socket closest to the HFI for best performance.

For example, a typical case is where the HFI is connected by PCI to socket 0, setting (for a CPU with 12 cores):

```
-genv I_MPI_PROCESSOR_LIST 0-11
```

The result will be the best possible osu_latency or IMB-MPI1 Pingpong latency by running the first MPI rank on each node on CPU0 (or it could be set to other core number on the first socket). This is necessary because, when running one rank per node with version 5.1, Intel® MPI Library does not pin the rank to a particular core. This one processes per node (PPN) behavior enables better hybrid OpenMP/MPI performance when using many threads, and only one MPI rank per node. This setting will also perform well when running the multi-PPN cases of the OMB or IMB benchmarks.

5.3 Tuning for LS-DYNA* Performance

You can get significant performance improvements for the Livermore Software Technology Corporation's (LSTC) LS-DYNA application by setting the eager buffer size to 4194304 or 8388608.

To set the eager_buffer_size:

1. Edit `/etc/modprobe.d/hfi1.conf` (see Setting HFI1 Driver Parameters on page 20)
2. Add or change the following:

```
eager_buffer_size=4194304
```

or

```
eager_buffer_size=8388608
```

In our testing up to 32 nodes, this change improved performance with three standard test models: Neon, Car2Car, and 3Cars.

Other applications that may benefit from setting eager_buffer_size to 4194304 or 8388608 are ANSYS Mechanical, weather code WRF, and ODB-10M.

Note: These settings, especially 8388608, may hurt multi-process IPoIB throughput. If that is an important workload on these nodes, use smaller values or do not set this parameter.

5.4 Tuning for MPI Performance on Nodes with Intel® Xeon Phi™ Product Family x200

Note: This section is not relevant to Intel® Xeon Phi™ Coprocessor x100 Product Family.

To significantly improve MPI performance on Intel® Xeon Phi™ Product Family x200, certain driver parameter and PSM2 environment variable settings are needed.

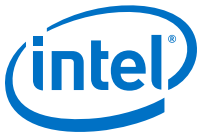
To set the number of user contexts per node:

1. Edit `/etc/modprobe.d/hfi1.conf` (see "Setting HFI1 Driver Parameters on page 20")
2. Add the following hfi1 driver parameter settings to the "options hfi1" line.

```
num_user_contexts=X
```

where *X* can be one of the following:

- *X*=16 – This value provides the best performance when running a few MPI ranks per node (16 or fewer), such as when using a Hybrid programming model with many more threads than MPI ranks.
- *X*="Number of physical cores per node" (Default setting) – This value is used if you do not plan on running more MPI ranks than cores.



- *X*=2x "Number of physical cores per node" – This value is used if you plan on running an MPI rank-count up to 2x the number of cores on a node (assuming Hyper-Threading is turned on).

3. To improve large-message MPI bandwidth: If you are running OPA SW 10.3 or newer, you may add to the "`options hfi1`" line in hfi1.conf:

```
max_mtu=10240
```

To set the MPI rank on one or more processors using taskset:

When running single-process-per-node benchmarks, you may obtain the best performance using processor 7 (as enumerated by the OS) and the lowest performance using processor 6, with performance between these levels occurring on the remaining OS-enumerated processors, but this may vary from node to node. This assumes a properly running irqbalance service (as described in irqbalance on page 14) on nodes with Intel® Xeon Phi™ Processors.

1. Run the `mpirun` command using taskset.

   The following example uses "taskset" to place the MPI rank on processor 7 on both nodes of this two-node test:

```
mpirun  -mca pml cm -mca mtl psm2 -H host1,host2 taskset -c 7
/usr/mpi/gcc/openmpi-1.10.2-hfi/tests/osu_benchmarks-3.1.1/osu_bw
```

To set the maximum number of user contexts for 288 MPI ranks:

Intel® Xeon Phi™ Product Family x200 nodes have up to 72 cores, and with Hyper-Threading turned on, up to 288 CPUs enumerated by the OS. It is unlikely that running with 288 MPI ranks per node will perform well (due to MPI memory consumption), but it might be worth a try.

1. Edit `/etc/modprobe.d/hfi1.conf` (see "Setting HFI1 Driver Parameters on page 20")
2. Modify `num_user_contexts=144` (max value is 160)

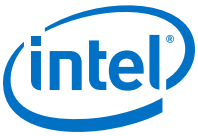
   With this value set, running 288 MPI ranks is possible with 2:1 context sharing.

An alternative to avoid the need for context sharing is to install two Intel® Omni-Path adapters per node. This will provide enough hardware contexts to support the 288 MPI ranks.

5.5 Tuning for Improved 1 KB to 8 KB Message Bandwidth at High Processes per Node

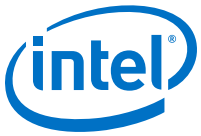
You can get significant performance improvements, from 1 KB to 8 KB message sizes in benchmarks like OSU Micro Benchmark's osu_gatherv, Intel MPI Benchmark's biband (bidirectional streaming bandwidth), and HPC Challenge's MPI Random Access (GUPS) benchmark, by setting the `PSM2_BOUNCE_SZ` environment variable to `8192`. The improvements typically occur when running with a number of MPI processes per node (PPN) at or near the core-count for the node, as is typically the case. This setting may also help applications where the gather collective or MPI SendRecv in these message sizes is important.

To set the tuning for improved message bandwidth, perform the following:



Set `PSM2_BOUNCE_SZ=8192` so that it gets propagated by your `mpirun` command.

For example with Intel® MPI's mpirun, you add `-genv PSM2_BOUNCE_SZ 8192` to the command line.



6.0 System Settings for Verbs Performance

The following settings can improve performance for verbs performance and stability, and, likely, performance for storage systems that use these protocols.

6.1 Parallel File System Concurrency Improvement

Adaptive Cache Memcpy is an option that will improve throughput in high concurrency situations - for example, many outstanding requests in flight, or many threads or processes receiving storage traffic concurrently.

To turn on the Adaptive Cache Memcpy heuristic:

1. Edit `/etc/modprobe.d/hfi1.conf file` (see Setting HFI1 Driver Parameters on page 20).
2. Add this setting:

```
options hfi1 sge_copy_mode=2
```

3. Restart the hfi1 driver to activate the setting as described in Setting HFI1 Driver Parameters on page 20.

Note: Working together with this `sge_copy_mode=2` setting is a `wss_threshold` (working set size threshold percentage) parameter with '80' as the default. To use that default, it is not necessary to set it in the `options hfi1` line above. You may want to experiment to see if the value of `wss_threshold=70` performs better on your file system benchmarks.

To improve parallel file system scalability using kernel receive queues:

Another `/etc/modprobe.d/hfi1.conf` file setting that can improve parallel file (storage) system scalability is krcvqs (kernel receive queues).

On a compute node that is a storage system client node, you may want to try values larger than the default of `krcvqs=2`.

1. Edit `/etc/modprobe.d/hfi1.conf file` (see Setting HFI1 Driver Parameters on page 20).
2. Modify the `krcvqs` value as shown in the example below:

```
options hfi1 sge_copy_mode=2 krcvqs=4
```

The above setting assumes one active VL. If, for example, there were three active VLs (00, 01, 02) and the parallel file system traffic were on VL 01, then a good krcvqs setting might be `krcvqs=2,4,2`, to avoid using too many resources on the non-storage VLs.

Notes:

- The larger the `krcvqs` value, the higher chance that it will have some adverse effect on the MPI performance for that compute node. But values in the range of 3 to 5 could be evaluated.
- For a storage server node (for example an OSS for Lustre or a NSD Server for IBM Spectrum Scale (GPFS)), where there is no MPI workload, values larger than five can be tested to see which performs best. Also, you need to be aware of how many CPU cores are on these server nodes. A setting of `krcvqs=8`, for example, would mean that eight CPU cores on the server would have the servicing of kernel receive interrupts as part of their workload.

6.2 Lustre Parameter Tuning for Intel Omni-Path

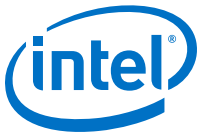
For Lustre* Version 2.8 or newer or Intel® Enterprise Edition for Lustre* software Version 3.0 or newer, you do not need to do anything. The Lustre module load process will automatically load these necessary tunings.

For earlier versions of Lustre, additional settings to the Lustre configuration file will enable the best performance with the Intel® Omni-Path interconnect.

To modify the Lustre configuration file:

1. Edit the `/etc/modprobe.d/lustre.conf` file.
2. Add the following parameter settings:

```
options lnet networks="o2ib0(ib0)"
options ko2iblnd peer_credits=128 peer_credits_hiw=64 credits=1024
concurrent_sends=256 ntx=2048 map_on_demand=32 fmr_pool_size=2048
fmr_flush_trigger=512 fmr_cache=1
```



7.0 Verbs Benchmarks

This section describes how to use the perftest suite of benchmarks to best measure RDMA/verbs performance on Intel® Omni-Path Architecture.

7.1 Perftest

Perftest is an open source benchmark from OpenFabrics Enterprise Distribution (OFED) for verbs performance. It is a set of microbenchmarks written over user-level verbs to measure latency, and uni- and bi-directional bandwidth.

7.2 RDMA Performance

The best perftest to measure verbs Remote Direct Memory Access (RDMA) bandwidth performance is the `ib_write_bw test`, with the default connection type `Reliable Connection` (RC). Please refer to the available options by running `ib_write_bw -h`. The following discussion highlights optimizations for the `ib_write_bw` test , but is applicable to other tests as well.

MTU Size can have a significant impact on bandwidth performance. For larger message sizes, such as 1 MB, the greater the MTU size, the higher the resulting band width. However, for small message sizes, such as 8B, the larger MTU sizes are not optimal. Therefore, the selection of the optimal MTU size is dependent on a particular mix of application message sizes.

InfiniBand* supports MTU sizes of 256B, 512B, 1024B, 2048B, and 4096B only. OPA on the other hand can support MTU sizes from 2048B (2K) up to 8192B (8KB) for verbs traffic. Intel recommends you use the 8KB MTU default for RDMA requests of 8KB or more.

To select the 8KB MTU size in the `ib_write_bw` test:

1. The `-R` switch should be specified to connect Queue Pairs (QPs) with `rdma_cm`.
2. Use the address for the server node's ib0 port to specify the IPoIB interface.

Before using the `rdma_cm` path for QPs, the ib_ipoib driver must be loaded. The sequence of driver installations and execution of the `ib_write_bw` test is:

```
1.    sudo modprobe hfi1
2.    sudo modprobe ib_ipoib
3.    sudo ifup ib0
4.    ib_write_bw -F -R -s 1048576                          // on server node
5.    ib_write_bw -F -R -s 1048576 <server's IPoIB address> // on client node
```

The `-F` switch is used to prevent the test from failing when the cpufreq_ondemand module is used. Refer to CPU Frequency Scaling Drivers on page 14 for more information.

8.0 IPoFabric Performance

The traditional term for sending IP traffic over the InfiniBand* fabric is IP over IB or *IPoIB*. The Intel® Omni-Path fabric does not implement InfiniBand*. Instead, the Intel implementation for OPA is known as IP over Fabric or *IPoFabric*. From the software point of view, IPoFabric behaves the same way as IPoIB, and in fact uses an ib_ipoib driver to send IP traffic over the ib0 and/or ib1 ports. Therefore, we will primarily refer to this traffic as IPoFabric, but will contunue to use the terms ib_ipoib and refer to the ib0/ib1 ports, and measure performance with traditional IP-oriented benchmarks such as qperf and iperf.

8.1 IPoFabric Connected Mode Configuration

For IPoFabric bandwidth benchmarks, a prerequisite for the best-possible performance is having 8KB MTU enabled on the fabric.

To enable the 8K MTU automatically:

1. Edit the `/etc/sysconfig/network-scripts/ifcfg-ib0` file.
2. Modify the following settings:

```
ONBOOT=yes
NM_CONTROLLED=no
MTU=65520
CONNECTED_MODE=yes
```

3. If you have two OPA ports in your system, perform Steps 1 and 2 for the `ifcfg-ib1` file.

8.2 IPoFabric Datagram (UD) Mode Configuration

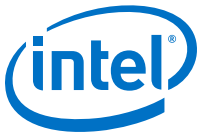
Although Connected Mode will provide the best performance in general, situations may arise in which you want to set Datagram mode, also known as Unreliable Datagram (UD) mode.

UD mode only uses one Queue Pair (QP) per node, so it will have a lower memory footprint than Connected Mode (which has one QP for each destination node for which IPoFabric communications are desired).

To set UD mode:

Set IPoIB to Datagram mode as shown in the example below.

```
modprobe ib_ipoib
ifup ib0
echo datagram > /sys/class/net/ib0/mode
```



When you view the IP MTU size for ib0, you will see that it gets set to 2044 bytes:

```
# cat /sys/class/net/ib0/mtu
2044
```

You can double the size of this UD mode MTU on RHEL* distributions to nearly 4K bytes to improve throughput.

To increase the size of MTU in UD mode:

1. Change the FM's configuration file to allow larger MTU for multicast.

   ```
   vi /etc/sysconfig/opafm.xml
   ```

   Change this:

   ```
         <MulticastGroup>
           <Create>1</Create>
           <MTU>2048</MTU>
           <Rate>25g</Rate>
   ```

   to this (MTU and speed updated):

   ```
         <MulticastGroup>
           <Create>1</Create>
           <MTU>4096</MTU>
           <Rate>25g</Rate>
   ```

2. Allow the 'ifup-ib' script in RHEL to accept the larger MTU size.

   ```
   vi /etc/sysconfig/network-scripts/ifup-ib
   ```

   Change this:

   ```
       else
           echo datagram > /sys/class/net/${DEVICE}/mode
           # cap the MTU where we should based upon mode
           [ -z "$MTU" ] && MTU=2044
           [ "$MTU" -gt 2044 ] && MTU=2044
       fi
   ```

   to this (comment out the two lines using #):

   ```
       else
           echo datagram > /sys/class/net/${DEVICE}/mode
           # cap the MTU where we should based upon mode
   #       [ -z "$MTU" ] && MTU=2044
   #       [ "$MTU" -gt 2044 ] && MTU=2044
       fi
   ```

3. Stop ib_ipoib on all hosts, restart the FM, and then restart ib_ipoib as shown below:

   ```
   modprobe -r ib_ipoib
   systemctl stop opafm
   systemctl start opafm
   opainfo | grep PortState
      PortState:     Active
   modprobe ib_ipoib
   ```

```
ifup ib0
ifconfig ib0
ib0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 4092
        inet 10.228.216.150  netmask 255.255.255.0  broadcast 10.255.255.255
        inet6 fe80::211:7501:165:b0ec  prefixlen 64  scopeid 0x20<link>
…
```

8.3 TCP Parameter Tuning for IPoFabric Performance

The default TCP parameters supplied by Red Hat and SUSE operating systems perform reasonably well with the Intel® Omni-Path Fabric's IPoFabric, so *typically no tuning is required*.

If you have nodes where memory usage is not a concern, or that communicate with only a few other nodes, and where there is bi-directional (simultaneous read/write) traffic, the following tunings could improve total IPoFabric bi-directional throughput up to 10%.

To improve IPoFabric traffic performance, apply the following settings:

- For Intel® Xeon® Processors and Intel® Xeon Phi™ Processors, set the following:

```
sudo sysctl -w net.ipv4.tcp_rmem="16384 349520 16777216"
sudo sysctl -w net.ipv4.tcp_wmem="16384 349520 16777216"
sudo sysctl -w net.core.rmem_max=16777216
sudo sysctl -w net.core.wmem_max=16777216
```

- For Intel® Xeon® Processors nodes only (not Intel® Xeon Phi™ Processors), set the following:

```
sudo sysctl -w net.core.somaxconn=2048
sudo sysctl -w net.ipv4.tcp_mtu_probing=1
sudo sysctl -w net.core.netdev_max_backlog=250000
```

Note: The above commands will set these parameters and affect performance only until the next node reboot.

To make the above changes persistent, perform the following:

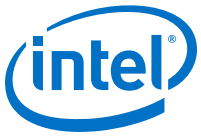
1. Edit the `/etc/sysctl.conf` file to add the appropriate settings listed above, in a format such as: `net.ipv4.tcp_rmem="16384 349520 16777216"`.
2. Run `sysctl -p` in a system start-up script, such as `/etc/rc.local`, to apply the tunings.

8.4 qperf

qperf is a benchmark that is included in IFS or OFED. It is designed to be run on a pair of nodes. You arbitrarily designate one node to be the server and the other to be the client.

To run a qperf test:

1. Run a ping test to ensure that ib_ipoib is running.



2. If you are concerned about performance, run a quick and useful qperf pretest on both the server and the client node.

```
qperf <server ipoib addr> -m 1M –ca 7  tcp_bw
```

In the above command line:

Option	Description
`<server ipoib addr>`	Specifies the IP address of the ib0 port of the server node.
`-ca 7`	Can be considered a tuning that will pin both the server and client-side qperf process to core 7, which is typically on the first socket. You do not want to specify core 0 or other lower number cores where OS or driver interrupts will interfere with the benchmark.
`-m`	Specifies a message size. In the example above, 1M specifies a 1 megabyte message size.

3. Type `<Ctrl+C>` on the server side to stop the test.

8.5 iperf

iperf is a tool for active measurements of the maximum achievable bandwidth on networks that carry IP traffic. It supports tuning of various parameters related to timing, protocols, and buffers. For each test, iperf reports the bandwidth, loss, and other parameters.

To improve Intel® Omni-Path Architecture's IPoIB throughput, run the following iperf command lines:

1. For server, specify:

```
iperf3 -s -1 -f G -A 6
```

2. For client, specify:

```
iperf3 -c <server ipoib addr> -f G -t 12 -O 2 --len 1M -A 6
```

In the above command lines:

Option	Description
`-A 6`	Sets CPU affinity to core 6.
`-t 12 -O 2`	Runs the test for 12 seconds, but omits the first two seconds when calculating the bandwidth result at the bottom of the output. This typically improves performance and makes performance results more stable.
`-f G`	Indicates the output bandwidth should be in GB (gigabyte) units.
`--len`	Indicates the message size in bytes, in this case 1M = 1 Megabyte
`<server ipoib addr>`	Specifies the IPoIB address of the server.

Note: The latest version of iperf is available for download at: http://software.es.net/iperf/.

9.0 Driver IRQ Affinity Assignments

MPI programs over PSM2 use a Send DMA mechanism (SDMA) for the sending of large messages. The hardware provides 16 SDMA engines and the host driver distributes the MPI processes over these SDMA engines to spread the work. The hfi1 host driver sets up interrupt handlers to process the SDMA completions and to keep the SDMA hardware queue filled up with any pending SDMA work. The driver makes affinity selections for these interrupt handlers at driver initialization time. This section describes how these choices are made and how they can be influenced. Note that some of the details can vary from software release to software release. The details below are valid for the V10.0 through the V10.3 software releases.

The number of SDMA engines (default 16) that are used by the host driver can be modified using the num_sdma module parameter:

```
parm:           num_sdma:Set max number SDMA engines to use (uint)
```

The number of VLs that are used is a fabric manager parameter (see opafm.xml). opaportinfo can be used to determine the number of enabled VLs. The following output shows that one data VL is in use (VL0) as all other data VLs (not VL15) have the MTU configured to 0 bytes. This is a fairly typical configuration:

```
   MTU  Supported: (0x7) 8192 bytes
   MTU  Active By VL:
   00:8192  01:    0 02:    0 03:    0 04:    0 05:    0 06:    0 07:    0
   08:    0 09:    0 10:    0 11:    0 12:    0 13:    0 14:    0 15: 2048
   16:    0 17:    0 18:    0 19:    0 20:    0 21:    0 22:    0 23:    0
   24:    0 25:    0 26:    0 27:    0 28:    0 29:    0 30:    0 31:    0
```

The driver partitions the configured number of SDMA engines over the number of VLs. For example, 16 SDMA engines distributed over 1 VL means that all SDMA engines can be used by that VL.

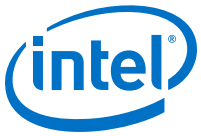
Two methods can be used to identify the SDMA engine to CPU number affinity assignment:

1. Use a four-line script to print SDMA engine identifier and the CPU number assigned for it.

   This method is more reliable, but needs a longer command, and the output is a bit terse.

2. Use the command "`dmesg | grep hfi1_0 | grep IRQ`".

   This is a shorter, easier-to-remember command, and produces more readable output, but it is not guaranteed to work in all OSes, and your system administrator may have limited the verbosity level in /var/log/messages so that this command doesn't provide the indicated output.



Method 1

```
# for i in `grep "hfi1_.* sdma" /proc/interrupts | awk '{print $NF}'` ; \
do printf "%s " $i ; irq=`awk "(\\\$NF == \"$i\") \
{print substr(\\\$1,1,length(\\\$1)-1)}" < /proc/interrupts` ; \
cat /proc/irq/$irq/smp_affinity_list ; done
```

Results:

```
sdma0 3
sdma1 4
sdma2 5
sdma3 6
sdma4 7
sdma5 8
sdma6 9
sdma7 10
sdma8 11
sdma9 12
sdma10 13
sdma11 0
sdma12 3
sdma13 4
sdma14 5
sdma15 6
```

Essentially this script finds all of the interrupt numbers in `/proc/interrupts` that are used by hfi1 SDMA engines, and then looks up their affinity settings in `/proc/irq/$irq/smp_affinity_list`.

Method 2

An alternate method to identify the affinity assignment for SDMA engines (that may not work on all systems):

```
# dmesg | grep hfi1 | grep IRQ
```

Results:

```
[ 165.513884] hfi1 0000:05:00.0: hfi1_0: IRQ vector: 74, type GENERAL -> cpu: 0
[ 165.521955] hfi1 0000:05:00.0: hfi1_0: IRQ vector: 75, type SDMA engine 0 -> cpu: 3
[ 165.530510] hfi1 0000:05:00.0: hfi1_0: IRQ vector: 76, type SDMA engine 1 -> cpu: 4
[ 165.539066] hfi1 0000:05:00.0: hfi1_0: IRQ vector: 77, type SDMA engine 2 -> cpu: 5
[ 165.547627] hfi1 0000:05:00.0: hfi1_0: IRQ vector: 78, type SDMA engine 3 -> cpu: 6
[ 165.556183] hfi1 0000:05:00.0: hfi1_0: IRQ vector: 79, type SDMA engine 4 -> cpu: 7
[ 165.564743] hfi1 0000:05:00.0: hfi1_0: IRQ vector: 80, type SDMA engine 5 -> cpu: 8
[ 165.573316] hfi1 0000:05:00.0: hfi1_0: IRQ vector: 81, type SDMA engine 6 -> cpu: 9
[ 165.581887] hfi1 0000:05:00.0: hfi1_0: IRQ vector: 82, type SDMA engine 7 -> cpu: 10
[ 165.590541] hfi1 0000:05:00.0: hfi1_0: IRQ vector: 83, type SDMA engine 8 -> cpu: 11
[ 165.599196] hfi1 0000:05:00.0: hfi1_0: IRQ vector: 84, type SDMA engine 9 -> cpu: 12
[ 165.607849] hfi1 0000:05:00.0: hfi1_0: IRQ vector: 85, type SDMA engine 10 -> cpu: 13
[ 165.616601] hfi1 0000:05:00.0: hfi1_0: IRQ vector: 86, type SDMA engine 11 -> cpu: 3
[ 165.625256] hfi1 0000:05:00.0: hfi1_0: IRQ vector: 87, type SDMA engine 12 -> cpu: 4
[ 165.633928] hfi1 0000:05:00.0: hfi1_0: IRQ vector: 88, type SDMA engine 13 -> cpu: 5
[ 165.642582] hfi1 0000:05:00.0: hfi1_0: IRQ vector: 89, type SDMA engine 14 -> cpu: 6
[ 165.651235] hfi1 0000:05:00.0: hfi1_0: IRQ vector: 90, type SDMA engine 15 -> cpu: 7
[ 165.659981] hfi1 0000:05:00.0: hfi1_0: IRQ vector: 91, type RCVCTXT ctxt 0 -> cpu: 0
[ 165.668726] hfi1 0000:05:00.0: hfi1_0: IRQ vector: 92, type RCVCTXT ctxt 1 -> cpu: 1
[ 165.677476] hfi1 0000:05:00.0: hfi1_0: IRQ vector: 93, type RCVCTXT ctxt 2 -> cpu: 2
```

In this case, the SDMA engines were assigned by the host driver to CPU cores starting at core 3 and wrapping around from CPU core 13 to CPU core 0. Note that CPU cores 1 and 2 were skipped in the assignment. The reason is that these were assigned to the

"kernel receive queues" (that is, receive contexts) that are used for processing incoming Verbs packets, as is apparent from the last 2 rows of the Method 2 output. The number of kernel receive queues defaults to 2 but can be modified using the krcvqs module parameter:

```
parm:           krcvqs:Array of the number of non-control kernel receive queues
by VL (array of uint)
```

Note that if the Method 1 script output indicates a CPU range similar to the output below, then it is likely that the irqbalance service is not running correctly and needs to be restarted:

```
sdma0 0-13
sdma1 0-13
sdma2 0-13
sdma3 0-13
sdma4 0-13
sdma5 0-13
sdma6 0-13
sdma7 0-13
sdma8 0-13
sdma9 0-13
sdma10 0-13
sdma11 0-13
sdma12 0-13
sdma13 0-13
sdma14 0-13
sdma15 0-13
```

Mapping from MPI Processes to SDMA Engines

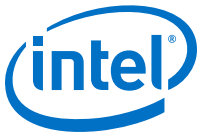
Typically, MPI jobs are assigned to a specific service level (SL) and this is mapped to a service channel (SC) and virtual lane (VL) by mappings set up by the fabric manager (configured by the `opafm.xml` configuration file). Each MPI process is associated with a VL, a context number and potentially a sub-context number (if context sharing is used). The VL selects the set of SDMA engines that can be used. In many cases, only one VL is configured (as shown in the output above) allowing an MPI job to use all 16 SDMA engines. The context and sub-context number (if required) are used to distribute the MPI processes over these SDMA engines. This is essentially a round-robin mapping. However, the context numbers that are assigned to MPI jobs typically do not start at 0, so there is often an offset for the first SDMA engine that is used.

Intel® Omni-Path Fabric Suite V10.3 offers a way to control a mapping of CPU core A to SDMA engine X. Thus, any MPI process running on that Core A will get assigned to SDMA engine X. To accomplish this, the driver will expose new sysfs entries per SDMA engine. For each SDMA engine, the driver will create a new sysfs directory called `sdma<N>`, where *N* is the SDMA engine id.

```
/sys/devices/pci…/infiniband/hfi1_<N>/sdma<N>/
```

"`pci…`" refers to several directory levels (a method for listing those directory names is provided below). Each directory will expose two new files (attributes):

- `cpu_list`: A read/write attribute that allows you to set up the process to SDMA engine assignments based on which CPU a process is running.
- `vl`: A read-only attribute that allows you to find out the existing SDMA engine to Virtual Lane mappings.



To print out the full path names to these files, use either of the following commands:

- `find /sys/devices/ -name vl | grep hfi1`
- `find /sys/devices/ -name cpu_list | grep hfi1`

Or, if you only want to print the files for the first HFI, use `hfi1_0`:

```
$ find /sys/devices/ -name cpu_list | grep hfi1_0
/sys/devices/pci0000:00/0000:00:02.0/0000:01:00.0/infiniband/hfi1_0/sdma0/cpu_list
/sys/devices/pci0000:00/0000:00:02.0/0000:01:00.0/infiniband/hfi1_0/sdma1/cpu_list
/sys/devices/pci0000:00/0000:00:02.0/0000:01:00.0/infiniband/hfi1_0/sdma2/cpu_list
/sys/devices/pci0000:00/0000:00:02.0/0000:01:00.0/infiniband/hfi1_0/sdma3/cpu_list
/sys/devices/pci0000:00/0000:00:02.0/0000:01:00.0/infiniband/hfi1_0/sdma4/cpu_list
/sys/devices/pci0000:00/0000:00:02.0/0000:01:00.0/infiniband/hfi1_0/sdma5/cpu_list
/sys/devices/pci0000:00/0000:00:02.0/0000:01:00.0/infiniband/hfi1_0/sdma6/cpu_list
/sys/devices/pci0000:00/0000:00:02.0/0000:01:00.0/infiniband/hfi1_0/sdma7/cpu_list
/sys/devices/pci0000:00/0000:00:02.0/0000:01:00.0/infiniband/hfi1_0/sdma8/cpu_list
/sys/devices/pci0000:00/0000:00:02.0/0000:01:00.0/infiniband/hfi1_0/sdma9/cpu_list
/sys/devices/pci0000:00/0000:00:02.0/0000:01:00.0/infiniband/hfi1_0/sdma10/cpu_list
/sys/devices/pci0000:00/0000:00:02.0/0000:01:00.0/infiniband/hfi1_0/sdma11/cpu_list
/sys/devices/pci0000:00/0000:00:02.0/0000:01:00.0/infiniband/hfi1_0/sdma12/cpu_list
/sys/devices/pci0000:00/0000:00:02.0/0000:01:00.0/infiniband/hfi1_0/sdma13/cpu_list
/sys/devices/pci0000:00/0000:00:02.0/0000:01:00.0/infiniband/hfi1_0/sdma14/cpu_list
/sys/devices/pci0000:00/0000:00:02.0/0000:01:00.0/infiniband/hfi1_0/sdma15/cpu_list
```

Note: Since the typical installation only uses one VL, number 0, and there are 16 SDMA engines for that VL, we will not deal with the VL number from this point on.

To print the CPUs that the SDMA engines and Receive contexts (# defined by krcvqs parameter), use this command:

```
$ dmesg | grep hfi1_0 | grep IRQ
[71396.873706] hfi1 0000:01:00.0: hfi1_0: IRQ vector: 43, type GENERAL  -> cpu: 0
[71396.873736] hfi1 0000:01:00.0: hfi1_0: IRQ vector: 44, type SDMA engine 0 -> cpu: 3
[71396.873763] hfi1 0000:01:00.0: hfi1_0: IRQ vector: 45, type SDMA engine 1 -> cpu: 4
[71396.873789] hfi1 0000:01:00.0: hfi1_0: IRQ vector: 46, type SDMA engine 2 -> cpu: 5
[71396.873816] hfi1 0000:01:00.0: hfi1_0: IRQ vector: 47, type SDMA engine 3 -> cpu: 6
[71396.873843] hfi1 0000:01:00.0: hfi1_0: IRQ vector: 49, type SDMA engine 4 -> cpu: 7
[71396.873878] hfi1 0000:01:00.0: hfi1_0: IRQ vector: 50, type SDMA engine 5 -> cpu: 8
[71396.873905] hfi1 0000:01:00.0: hfi1_0: IRQ vector: 51, type SDMA engine 6 -> cpu: 9
[71396.873937] hfi1 0000:01:00.0: hfi1_0: IRQ vector: 52, type SDMA engine 7 -> cpu: 10
[71396.873963] hfi1 0000:01:00.0: hfi1_0: IRQ vector: 53, type SDMA engine 8 -> cpu: 11
[71396.873989] hfi1 0000:01:00.0: hfi1_0: IRQ vector: 54, type SDMA engine 9 -> cpu: 12
[71396.874015] hfi1 0000:01:00.0: hfi1_0: IRQ vector: 55, type SDMA engine 10 -> cpu: 13
[71396.874040] hfi1 0000:01:00.0: hfi1_0: IRQ vector: 56, type SDMA engine 11 -> cpu: 14
[71396.874066] hfi1 0000:01:00.0: hfi1_0: IRQ vector: 57, type SDMA engine 12 -> cpu: 15
[71396.874098] hfi1 0000:01:00.0: hfi1_0: IRQ vector: 58, type SDMA engine 13 -> cpu: 16
[71396.874125] hfi1 0000:01:00.0: hfi1_0: IRQ vector: 59, type SDMA engine 14 -> cpu: 17
[71396.874151] hfi1 0000:01:00.0: hfi1_0: IRQ vector: 60, type SDMA engine 15 -> cpu: 18
[71396.874403] hfi1 0000:01:00.0: hfi1_0: IRQ vector: 61, type RCVCTXT ctxt 0 -> cpu: 0
[71396.874743] hfi1 0000:01:00.0: hfi1_0: IRQ vector: 62, type RCVCTXT ctxt 1 -> cpu: 1
[71396.875100] hfi1 0000:01:00.0: hfi1_0: IRQ vector: 63, type RCVCTXT ctxt 2 -> cpu: 2
```

Note: If this `dmesg` command does not work, you can use the "Method 1 script" above to get the SDMA engine to cpu core assignment.

As an example, you can assign a single core or a range of cores to the cpu_list file as follows:

```
# echo “0-3” > /sys/devices/pci0000:00/0000:00:02.0/0000:01:00.0/infiniband/hfi1_0/sdma0/
cpu_list
```

Note: You need to use the "**`find /sys/devices/ -name cpu_list | grep hfi1_0`**" command shown above to find the correct pathname to use with this echo command for sdma engine 0 on hfi1_0. The above directories "`pci0000:00/0000:00:02.0/0000:01:00.0`", most likely, won't exist on your system.

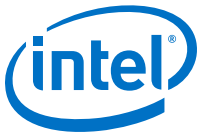
To illustrate an application of the feature to Intel® Xeon Phi™ Product Family x200 CPUs, let's consider the situation described in Tuning for MPI Performance on Nodes with Intel® Xeon Phi™ Product Family x200 on page 24: "When running single-process-per-node benchmarks, you may obtain the best performance using processor 7 (as enumerated by the OS) and the lowest performance using processor 6, with performance between these levels occurring on the remaining OS-enumerated processors." An example of these performance levels of the OSU Micro Benchmark, osu_bw, at a 4 MB message size (run on nodes with Intel® Xeon Phi™ Product Family x200 at 1.5 GHz, 72 cores, with `max_mtu=10240` set, Turbo mode off), are as follows:

Pin-to-Core Number	osu_bw in MB/s
0	10205
1	10185
2	10431
3	10391
4	10300
5	10260
6	**8038**
7	**12381**
8	10318
9	10315
10	10468
11	10426
12	10282
…	

Suppose your objective was to increase the number of "best-cores" from 1 to 4 consecutive cores, and to eliminate the slower performance of Core 6.

First, we need to understand that the best-performing core 7 above has that higher performance because it happened to be assigned to the SDMA engine assigned to core 6. From above, we can see that sdma3 is assigned to this core: "`SDMA engine 3 → cpu: 6`". On Intel® Xeon Phi™ Product Family x200 CPUs, consecutive even-odd pairs like cores 0-1 or 6-7 are each in one 'tile' which shares the same L2 cache. Thus there is very fast communications by cache-sharing between cores 6 and 7.

Note: Core 7 is not always the best-performing core on Intel® Xeon Phi™ Processor.



Using the two lists above showing the file associated with the SDMA engines and the CPUs each SDMA engine is assigned to, as root, the following commands can be executed:

```
echo 6 > /sys/devices/pci0000:00/0000:00:02.0/0000:01:00.0/infiniband/hfi1_0/sdma4/cpu_list
echo 7 > /sys/devices/pci0000:00/0000:00:02.0/0000:01:00.0/infiniband/hfi1_0/sdma3/cpu_list
echo 8 > /sys/devices/pci0000:00/0000:00:02.0/0000:01:00.0/infiniband/hfi1_0/sdma6/cpu_list
echo 9 > /sys/devices/pci0000:00/0000:00:02.0/0000:01:00.0/infiniband/hfi1_0/sdma5/cpu_list
```

As an example of how these commands were determined, for the first command above, we are trying to assign an MPI process that will be pinned to core 6 to and SDMA engine assigned to core 7 (the other member of the tile—even-odd pair that includes core 6); and the 'dmesg' list above shows: "`type SDMA engine 4 → cpu: 7`". Thus we pick the file with 'sdma4' in the path. Similar reasoning was used to construct the other three commands. The resulting performance is below.

Pin-to-Core Number	osu_bw in MB/s
0	10222
1	10185
2	10436
3	10438
4	10326
5	10279
6	**12381**
7	**12379**
8	**12380**
9	**12378**
10	10435
11	10446
12	10294
…	

For both this and the previous table, the … indicates that for cores 13 - 71 (on this 72-core system) the performance level is all in the 10300–10500 MB/s range.

Note that there is now no core with the lower 8000 MB/s performance level, and there are four cores with near line rate performance of ~12.4 GB/s.