Intel® Omni-Path Fabric Performance Tuning

User Guide

Rev. 10.0

October 2017

Order No.: H93143-10.0

Revision History

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

Date	Revision	Description
October 2017	10.0	Updates to this document include: • Changed Preface: The *Intel® Omni-Path Fabric Suite FastFabric Command Line Interface Reference Guide* has been merged into the *Intel® Omni-Path Fabric Suite FastFabric User Guide*. See the Intel® Omni-Path Documentation Library for details. • Global product name change from Intel® Xeon Phi™ Product Family x200 to Intel® Xeon Phi™ x200 Product Family. • Updated Performance Tuning Checklist. • Updated Intel® Xeon® Scalable Processors. • Updated irqbalance. • Updated Using the Intel P-State Driver. • Removed obsolete section, Avoid acpi_pad Consuming CPU Resources. • Updated Transparent Huge Pages. • Updated Listing the Driver Parameters. • Updated Current Values of Module Parameters. • Updated Intel® MPI Library Settings. • Updated Tuning for High-Performance Linpack Performance: Added new subsections: — Expected Levels of Performance — Selection of HPL Binary and MPI — MPI Flags and Proper Job Submission Parameters/Syntax — HPL.dat Input File — Recommended Procedure for Achieving Best HPL Score • Updated MPI Applications Performance Tuning. • Updated Driver Parameter Settings for Intel® Xeon Phi™ x200 Product Family • Updated Lustre. • Updated IPoFabric Connected Mode Configuration. • Updated IPoFabric Datagram Mode Configuration. • Updated RPS and GSO Tuning for IPoFabric Performance. • Added new sections: RPS Tuning for Intel® Xeon® Processor Nodes and RPS and GSO Tuning for Intel® Xeon Phi™ x200 Product Family Nodes.
August 2017	9.0	Updates to this document include: • Added Terminology. • Updated Performance Tuning Checklist. • Updated Intel® Xeon® Processor E5 v3 and v4 Families. • Added Intel® Xeon® Scalable Processors • Updated Intel® Xeon Phi™ x200 Product Family. • Updated Using the Intel P-State Driver. • Updated Listing the Driver Parameters. • Updated Current Values of Module Parameters. • Updated Setting HFI1 Driver Parameters. • Updated Intel® MPI Library Settings. • Added Intel® MPI 2017: New Support of OFI as a Fabric.

continued...

Date	Revision	Description
		• Updated Tuning for High-Performance Linpack Performance. • Updated MPI Applications Performance Tuning. • Updated Driver Parameter Settings for Intel® Xeon Phi™ x200 Product Family • Updated GPUDirect* RDMA Tuning for MPI Benchmarks and Applications. • Updated Accelerated RDMA. • Updated Parallel File System Concurrency Improvement. • Added IBM Spectrum Scale (aka GPFS) Tuning for Intel® Omni-Path. • Updated RDMA Performance. • Added RPS and GSO Tuning for IPoFabric Performance. • Updated Driver IRQ Affinity Assignments.
May 2017	8.0	Updates to this document include: • Updated Performance Tuning Checklist. • Updated Tuning for High-Performance Linpack Performance. • Added new sections: MPI Collective Scaling Guidelines for Large Clusters and Driver Parameter Settings for Intel® Xeon Phi™ x200 Product Family
April 2017	7.0	Updates to this document include: • Updated Performance Tuning Checklist. • Updated Intel® Xeon® Processor E5 v3 and v4 Families. • Updated Intel® Xeon Phi™ x200 Product Family. • Updated irqbalance. • Added new section Do Not Enable intel_iommu. • Added new section Dealing with Memory Fragmentation. • Updated Listing the Driver Parameters. • Updated Current Values of Module Parameters. • Added new section Switching to the Intel P-State Driver to Run Certain FastFabric Tools. • Updated Intel® MPI Library Settings. • Added new section Tuning for High-Performance Linpack Performance. • Changed section name from Tuning for LS-DYNA* Performance to MPI Applications Performance Tuning. • Updated MPI Applications Performance Tuning. • Added new section Tuning for Improved Performance on QCD Applications. • Added new section GPUDirect* RDMA Tuning for MPI Benchmarks and Applications. • Added new section Accelerated RDMA. • Updated Parallel File System Concurrency Improvement. • Updated IPoFabric Connected Mode Configuration. • Updated IPoFabric Datagram Mode Configuration.
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™ x200 Product Family. • Updated Tuning for LS-DYNA* Performance. • Added new section Tuning for MPI Performance on Nodes with Intel® Xeon Phi™ x200 Product Family. • Added new section Tuning for Improved 1 KB to 8 KB Message Bandwidth at High Processes per Node.

continued...

Date	Revision	Description
		• 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.

Intel® Omni-Path Documentation Library

Intel® Omni-Path publications are available at the following URLs:

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

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

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

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

Use the tasks listed in this table to find the corresponding Intel® Omni-Path document.

Task	Document Title	Description
Key: Shading indicates the URL to use for accessing the particular document.		
• Intel® Omni-Path Switches Installation, User, and Reference Guides: http://www.intel.com/omnipath/SwitchPublications		
• Intel® Omni-Path Software Installation, User, and Reference Guides (includes HFI documents): http://www.intel.com/omnipath/FabricSoftwarePublications (no shading)		
• Drivers and Software (including Release Notes): http://www.intel.com/omnipath/Downloads		

continued...

Task	Document Title	Description
Using the Intel® OPA documentation set	*Intel® Omni-Path Fabric Quick Start Guide*	A roadmap to Intel's comprehensive library of publications describing all aspects of the product family. It outlines the most basic steps for getting your Intel® Omni-Path Architecture (Intel® OPA) cluster installed and operational.
Setting up an Intel® OPA cluster	*Intel® Omni-Path Fabric Setup Guide* (Old title: *Intel® Omni-Path Fabric Staging Guide*)	Provides a high level overview of the steps required to stage a customer-based installation of the Intel® Omni-Path Fabric. Procedures and key reference documents, such as Intel® Omni-Path user guides and installation guides are provided to clarify the process. Additional commands and BKMs are defined to facilitate the installation process and troubleshooting.
Installing hardware	*Intel® Omni-Path Fabric Switches Hardware Installation Guide*	Describes the hardware installation and initial configuration tasks for the Intel® Omni-Path Switches 100 Series. This includes: Intel® Omni-Path Edge Switches 100 Series, 24 and 48-port configurable Edge switches, and Intel® Omni-Path Director Class Switches 100 Series.
	Intel® Omni-Path Host Fabric Interface Installation Guide	Contains instructions for installing the HFI in an Intel® OPA cluster. A cluster is defined as a collection of nodes, each attached to a fabric through the Intel interconnect. The Intel® HFI utilizes Intel® Omni-Path switches and cabling.
Installing host software Installing HFI firmware Installing switch firmware (externally-managed switches)	*Intel® Omni-Path Fabric Software Installation Guide*	Describes using a Text-based User Interface (TUI) to guide you through the installation process. You have the option of using command line interface (CLI) commands to perform the installation or install using the Linux* distribution software.
Managing a switch using Chassis Viewer GUI Installing switch firmware (managed switches)	*Intel® Omni-Path Fabric Switches GUI User Guide*	Describes the Intel® Omni-Path Fabric Chassis Viewer graphical user interface (GUI). It provides task-oriented procedures for configuring and managing the Intel® Omni-Path Switch family. Help: GUI online help.
Managing a switch using the CLI Installing switch firmware (managed switches)	*Intel® Omni-Path Fabric Switches Command Line Interface Reference Guide*	Describes the command line interface (CLI) task information for the Intel® Omni-Path Switch family. Help: -help for each CLI.
Managing a fabric using FastFabric	*Intel® Omni-Path Fabric Suite FastFabric User Guide* (Merged with: *Intel® Omni-Path Fabric Suite FastFabric Command Line Interface Reference Guide*)	Provides instructions for using the set of fabric management tools designed to simplify and optimize common fabric management tasks. The management tools consist of TUI menus and command line interface (CLI) commands. Help: -help and man pages for each CLI. Also, all host CLI commands can be accessed as console help in the Fabric Manager GUI.
Managing a fabric using Fabric Manager	*Intel® Omni-Path Fabric Suite Fabric Manager User Guide*	The Fabric Manager uses a well defined management protocol to communicate with management agents in every Intel® Omni-Path Host Fabric Interface (HFI) and switch. Through these interfaces the Fabric Manager is able to discover, configure, and monitor the fabric.
	Intel® Omni-Path Fabric Suite Fabric Manager GUI User Guide	Provides an intuitive, scalable dashboard and set of analysis tools for graphically monitoring fabric status and configuration. It is a user-friendly alternative to traditional command-line tools for day-to-day monitoring of fabric health. Help: Fabric Manager GUI Online Help.

continued...

Task	Document Title	Description
Configuring and administering Intel® HFI and IPoIB driver Running MPI applications on Intel® OPA	*Intel® Omni-Path Fabric Host Software User Guide*	Describes how to set up and administer the Host Fabric Interface (HFI) after the software has been installed. The audience for this document includes both cluster administrators and Message-Passing Interface (MPI) application programmers, who have different but overlapping interests in the details of the technology.
Writing and running middleware that uses Intel® OPA	*Intel® Performance Scaled Messaging 2 (PSM2) Programmer's Guide*	Provides a reference for programmers working with the Intel® PSM2 Application Programming Interface (API). The Performance Scaled Messaging 2 API (PSM2 API) is a low-level user-level communications interface.
Optimizing system performance	*Intel® Omni-Path Fabric Performance Tuning User Guide*	Describes BIOS settings and parameters that have been shown to ensure best 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.
Designing an IP or storage router on Intel® OPA	*Intel® Omni-Path IP and Storage Router Design Guide*	Describes how to install, configure, and administer an IPoIB router solution (Linux* IP or LNet) for inter-operating between Intel® Omni-Path and a legacy InfiniBand* fabric.
Building a Lustre* Server using Intel® OPA	*Building Lustre* Servers with Intel® Omni-Path Architecture Application Note*	Describes the steps to build and test a Lustre* system (MGS, MDT, MDS, OSS, OST, client) from the HPDD master branch on a x86_64, RHEL*/CentOS* 7.1 machine.
Building Containers for Intel® OPA fabrics	*Building Containers for Intel® Omni-Path Fabrics using Docker* and Singularity* Application Note*	Provides basic information for building and running Docker* and Singularity* containers on Linux*-based computer platforms that incorporate Intel® Omni-Path networking technology.
Writing management applications that interface with Intel® OPA	*Intel® Omni-Path Management API Programmer's Guide*	Contains a reference for programmers working with the Intel® Omni-Path Architecture Management (Intel OPAMGT) Application Programming Interface (API). The Intel OPAMGT API is a C-API permitting in-band and out-of-band queries of the FM's Subnet Administrator and Performance Administrator.
Learning about new release features, open issues, and resolved issues for a particular release	*Intel® Omni-Path Fabric Software Release Notes*	
	Intel® Omni-Path Fabric Manager GUI Release Notes	
	Intel® Omni-Path Fabric Switches Release Notes (includes managed and externally-managed switches)	

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 12 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 http://www.intel.com/omnipath/support for additional detail.

1.0 Introduction

The Intel® Omni-Path Architecture (Intel® OPA) is designed for excellent out-of-the-box performance. However, you may be able to 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.

For details about the other documents for the Intel® Omni-Path product line, refer to Intel® Omni-Path Documentation Library on page 9 of this document.

1.1 Terminology

The table below lists the abbreviations and acronyms used in this document.

Table 1. Terminology

Term	Description
ACPI	Advanced Configuration and Power Interface
BIOS	Basic Input/Output System
CPU	Central Processing Unit
FM	Intel® Omni-Path Fabric Suite Fabric Manager
GCC	GNU Compiler Collection
GUPS	Giga-Updates per Second
HFI	Host Fabric Interface
HPC	High-Performance Computing
HPL	High-Performance Linpack
HT	Intel® Hyper Threading
IFS	Intel® Omni-Path Fabric Suite
IMB	Intel® MPI Benchmarks
IO	Input/Output
IP	Internet Protocol
IPoFabric	Internet Protocol over Fabric
IPoIB	Internet Protocol over InfiniBand*
IRQ	Interrupt Request
MPI	Mesage Passing Interface
MTU	Maximum Transmission Unit

continued...

Term	Description
NUMA	Non-Uniform Memory Access
OFED	OpenFabrics Enterprise Distribution*
OFI	OpenFabrics Interface
OMB	OSU Micro Benchmarks
OPA	Intel® Omni-Path Architecture
OS	Operating System
OSU	Ohio State University
PPN	Processes per Node
PSM2	Performance Scaled Messaging 2
QCD	Quantum Chromodynamics
QP	Queue Pair
RDMA	Remote Direct Memory Access
RPS	Receive Packet Steering
SDMA	Send Direct Memory Access
SMP	Symmetric Multiprocessing
TBB	Intel® Threading Building Blocks
TCP	Transmission Control Protocol
THP	Transparent Huge Pages
TID	Thread ID
TMI	Tag Matching Interface
UD	Unreliable Datagram
VL	Virtual Lane
VM	Virtual Machine
VT	Intel® Virtualization Technology

1.2 Performance Tuning Checklist

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

Table 2. Performance Tuning Checklist

Tunings	Section
Set recommended BIOS settings.	• Intel® Xeon® Processor E5 v3 and v4 Families • Intel® Xeon® Scalable Processors • Intel® Xeon Phi™ x200 Product Family
Distribute hardware interrupts across processors via irqbalance (Linux*).	irqbalance
Run CPU at its maximum non-Turbo frequency (Intel P-State Driver).	Using the Intel P-State Driver

continued...

Tunings	Section
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 frequency 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
Remove `intel_iommu` from the grub file.	Do Not Enable intel_iommu
Set THP to "always".	Transparent Huge Pages
Alleviate memory fragmentation.	Dealing with Memory Fragmentation
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
Reduce the MPI start-up time in large node-count runs when using Intel® MPI 5.x.	Intel® MPI Library Settings
Test with Intel® MPI 2017 Supporting OFI	Intel® MPI 2017: New Support of OFI as a Fabric
Pin the MPI rank to a socket.	Intel® MPI Benchmarks or OSU Micro Benchmarks
Tune for HPL Performance.	Tuning for High-Performance Linpack Performance
Load pre-compiled HPL Binary found in Intel® Parallel Studio XE.	Selection of HPL Binary and MPI
Ensure you are using the PSM2 software layer.	MPI Flags and Proper Job Submission Parameters/Syntax
Tune to achieve the best HPL score.	Recommended Procedure for Achieving Best HPL Score
Set the `rcvhdrcnt` parameter for LS-DYNA*.	MPI Applications Performance Tuning
Reduce OS and hardware noise to improve MPI collectives scalability.	MPI Collective Scaling Guidelines for Large Clusters
Set the user contexts per node (Intel® Xeon Phi™ x200 Product Family).	Driver Parameter Settings for Intel® Xeon Phi™ x200 Product Family
Set the maximum number of user context for 288 MPI ranks (Intel® Xeon Phi™ x200 Product Family).	Driver Parameter Settings for Intel® Xeon Phi™ x200 Product Family
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
Speed up performance of some QCD applications.	Tuning for Improved Performance on QCD Applications
Obtain maximum bandwidth and best latency between CUDA-enabled GPU devices.	GPUDirect* RDMA Tuning for MPI Benchmarks and Applications
Set tuning based on common GPUDirect* use cases.	GPUDirect* RDMA Tuning for MPI Benchmarks and Applications
Enable Accelerated RDMA.	Accelerated RDMA
Use Accelerated RDMA.	Accelerated RDMA
Turn on the Adaptive Cache Memcpy heuristic.	Parallel File System Concurrency Improvement

continued...

Tunings	Section
Improve parallel file system scalability using kernel receive queues (krcvqs).	Parallel File System Concurrency Improvement
Use Lustre* Version 2.8 or newer or Intel® Enterprise Edition for Lustre* software Version 3.0 or newer.	Lustre
Using Accelerated RDMA with IBM Spectrum Scale parallel file system.	IBM Spectrum Scale (aka GPFS) Tuning for Intel® Omni-Path
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 Mode Configuration
Increase the size of MTU in Datagram (UD) mode.	IPoFabric Datagram Mode Configuration
Improve IPoFabric performance for a large number of clients (Intel® Xeon® Processors).	RPS Tuning for Intel® Xeon® Processor Nodes
Improve throughput when using multiple IPoFabric (IPoIB) streams of traffic per node (Intel® Xeon Phi™ x200 Product Family).	RPS and GSO Tuning for Intel® Xeon Phi™ x200 Product Family Nodes
Improve IPoFabric traffic performance through TCP Parameters.	TCP Parameter Tuning for IPoFabric Performance
Identify affinity assignments for SDMA engines.	Driver IRQ Affinity Assignments
Map MPI processes to SDMA engines.	Mapping from MPI Processes to SDMA Engines

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 may 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 3. 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	4096B[4]
Snoop Holdoff Count	9[5]

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 27.
5. Also known as `Snooped Response Wait Time for Posted Prefetch` in some BIOSes.

2.2 Intel® Xeon® Scalable Processors

For the Intel® Xeon® Scalable Processors CPUs, Intel recommends the following:

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

Table 4. Recommended BIOS Settings for Intel® Xeon® Scalable Processors

BIOS Setting	Value
CPU Power and Performance Policy	Performance or Balanced Performance[a]
Intel® Hyper-Threading Technology	No recommendation (Test with your applications to see if a benefit occurs.)
Sub_NUMA Cluster	Disabled[b]
Snoop Holdoff Count	9[c]
Uncore Frequency Scaling	Enabled
WFR Uncore GV Rate Reduction	Disabled (where available)[d]

Notes: a. To get the most consistent Turbo mode performance for demanding workloads, set this parameter to "Performance". Either "Performance" or "Balanced Performance" will result in good fabric performance.
b. Also known as `Cluster-on-Die` in some BIOSes.
c. Also known as `Snooped Response Wait Time for Posted Prefetch` in some BIOSes.
d. May be named differently in some BIOSes.

2. Use the default settings, including Intel® Turbo Boost Technology= Enabled.

   Settings for Intel® Xeon® Processor E5 V3 and V4 Family CPUs are listed in Table 3 on page 17. Intel recommends the same values for these settings be used on Intel® Xeon® Scalable Processors CPUs, where the setting is available.

3. Enable Turbo speeds as specified in CPU Frequency Scaling Drivers on page 20.

2.3 Intel® Xeon Phi™ x200 Product Family

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

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

Table 5. Recommended BIOS Settings for Intel® Xeon Phi™ x200 Product Family

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[c]

Notes: a. Application-dependent, but "Quadrant" mode has shown some benefit to peak MPI message rate.

b. While virtualization is not supported with Intel® Xeon Phi™ x200 Product Family, Intel® Xeon Phi™ Processors operate in X2APIC Mode, which requires that the Intel® VT for Directed I/O (VT-d) remain enabled. If Intel® VT for Directed I/O (VT-d) is exposed in your BIOS, we recommend that it is set to "enabled". When verifying host settings via `verifyhosts.sh` script or `opaverifyhosts`, please do not select/include the vtd test.

c. Also known as `Snooped Response Wait Time for Posted Prefetch` in some BIOSes.

2. Use the default settings, including Intel® Turbo Boost Technology= Enabled.
3. Enable Turbo speeds as specified in CPU Frequency Scaling Drivers on page 20.

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. Install the `irqbalance` package, if not already installed.

```
# yum install irqbalance
```

2. Add the following line to the `/etc/sysconfig/irqbalance` file:

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

3. *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.x is the Intel P-State (`intel_pstate`). An alternative driver called the Advanced Configuration and Power Interface (ACPI) CPUfreq (`acpi_cpufreq`) is also available. Both have their advantages and disadvantages, but only one can be active at a time. In this section we 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.x, 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
elif [ -e /boot/grub2/grub.cfg ]; then
GRUB_CFG=/boot/grub2/grub.cfg
fi
grub2-mkconfig -o $GRUB_CFG
```

3. Reboot.

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.

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 2.3 GHz and `Performance` governor, when using the acpi-cpufreq driver.

```
sudo cpupower -c all frequency-set --min 2.3GHz --max 2.3GHz \
-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 in the BIOS (as recommended in BIOS Settings on page 17).
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
```

3.2.3 Switching to the Intel P-State Driver to Run Certain FastFabric Tools

Some Intel-provided tools require the use of the Intel P-State driver rather than the acpi_cpufreq driver. For example, the `hostverify.sh` tool fails with RHEL* 6.7 due to the Intel P-State driver not being the default cpufreq driver.

If you are using the acpi_cpufreq driver, perform one of the following methods to switch to Intel P-State driver in order to use the target tool.

Temporary Switch to Intel P-State Driver

To temporarily switch to the Intel P-state driver, perform the following steps:

1. Make sure cpupowerutils package is installed.

   ```
   # yum install cpupowerutils
   ```

2. Check if any other cpufreq kernel driver is active.

   ```
   # cpupower frequency-info -d
   ```

3. Unload another cpufreq kernel driver (if any).

   ```
   # rmmod acpi_cpufreq
   ```

4. Load intel_pstate driver.

   ```
   # modprobe intel_pstate
   ```

5. Set cpufreq governor to "performance".

   ```
   # cpupower -c all frequency-set -g performance
   ```

6. After using `hostverify.sh` or other tools that needed the Intel P-state set, you may reboot to return to the acpi_cpufreq driver.

Load Intel P-State Driver at Boot Time

To load the Intel P-state driver at boot time, perform the following steps:

1. Create a script file `/etc/sysconfig/modules/intel_pstate.modules` and add below text to it.

   ```
   #!/bin/sh
   /sbin/modprobe intel_pstate >/dev/null 2>&1
   ```

2. Add executable permissions for above file.

```
# chmod +x /etc/sysconfig/modules/intel_pstate.modules
```

3. Reboot the system for the changes to take effect.
4. Verify that the Intel P-state driver is loaded.
5. Install the cpupowerutils package, if not already installed.

```
# yum install cpupowerutils
```

6. Set cpufreq governor to 'performance'.

```
# cpupower -c all frequency-set -g performance
```

To re-enable the acpi_cpufreq driver, perform the following:

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.

3.3 Do Not Enable intel_iommu

Setting `intel_iommu=on` in the kernel command line can hurt verbs and MPI performance. Sometimes enabling `intel_iommu` is necessary if you are running applications from a virtual machine (VM). But typically for High-Performance Computing (HPC), you do not use a VM, therefore, `intel_iommu` is not set.

To remove `intel_iommu` from the grub file:

1. Navigate to `/etc/default/grub`.
2. Remove `intel_iommu=on` from the kernel command lines.

3.4 Transparent Huge Pages

Transparent Huge Pages is set to “always”. It is enabled in RHEL* 7.2 and later by default. Note that 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 the default "always" is not set on each node, you can set it by using the following command:

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

3.5 Dealing with Memory Fragmentation

When a Linux system has been running for a while, memory fragmentation, which depends heavily on the nature of the applications that are running on it, can increase. The more processes that request the kernel to allocate and free physical memory, the quicker the physical memory becomes fragmented. If that happens, performance on applications can suffer significantly. Over time, the performance of benchmarks and applications can decrease because of this issue.

Cluster/system administrators and users can take steps to address the memory fragmentation issue as described below. Note that users will not be able to apply their settings until the system administrators have applied theirs first.

System Administrator Settings

The following settings are performed by system administrators.

1. Enable THP to "always" as per Transparent Huge Pages.
2. As an alternative to THP, reserve huge pages with the sysfs entries, `nr_hugepages` or `nr_overcommit_hugepages`.
3. To better ensure that the system will allocate 2M pages to the job, set the cluster's job submission system to drop the caches and compact memory before each user job with these commands:

   ```
   echo 3 >/proc/sys/vm/drop_caches
   echo 1 >/proc/sys/vm/compact_memory
   ```

4. Setting a smaller number of User Contexts, using driver parameter `num_user_contexts`, can allocate a larger number of TID descriptors per context, which can deal better with the effects of memory fragmentation. See the Tuning for MPI Performance on Nodes with Intel® Xeon Phi™ x200 Product Family on page 37, Step 2 for guidelines on this hfi1 driver parameter.

User Settings

The following settings are performed by users.

1. Assuming that the system administrator has enabled THP (described in #1 above), the user can align larger MPI buffers on 2M boundaries and pad the total size to a multiple of 2M.

   You can use `posix_memalign` or Intel's `_mm_malloc` to cause the OS to try to allocate 2 MB pages.

2. Assuming that the system administrator has enable the alternative to THP (described in #2 above), the user can explicitly allocate huge pages using mmap, Intel® Threading Building Blocks (TBB) malloc with `TBB_MALLOC_USE_HUGE_PAGES=1`, or libhugetlbfs.

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/3.10.0-514.el7.x86_64/extra/ifs-kernel-updates/
hfi1.ko
version:        0.9-294
description:    Intel Omni-Path Architecture driver
license:        Dual BSD/GPL
rhelversion:    7.3
srcversion:     E5A668BCD78F5DCE1719E58
alias:          pci:v00008086d000024F1sv*sd*bc*sc*i*
alias:          pci:v00008086d000024F0sv*sd*bc*sc*i*
depends:        rdmavt,ib_core,i2c-core,i2c-algo-bit
vermagic:       3.10.0-514.el7.x86_64 SMP mod_unload modversions
signer:         Intel Corporation: Intel(R) Omni-Path HFI UEFI
sig_key:        71:9E:BA:A1:25:17:2B:A6:9A:D0:1B:85:0B:74:58:F8:5C:89:BB:07
sig_hashalgo:   sha256
parm:           lkey_table_size:LKEY table size in bits (2^n, 1 <= n <= 23) (uint)
parm:           no_user_mr_percpu:Avoid percpu refcount for user MRs (default 0)
(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:           tid_rdma_seg_min_size:Min TID RDMA segment size, default: 64KiB
(uint)
parm:           tid_rdma_seg_max_size:Max TID RDMA segment size, default: 256KiB
(uint)
parm:           tid_rdma_max_read_segs:Max TID RDMA READ segments per QP request,
default: 6 (uint)
parm:           tid_rdma_max_write_segs:Max TID RDMA WRITE segments per QP
request, default: 2 (uint)
parm:           tid_rdma_proto_enable:[En|Dis]able TID RDMA READ and/or WRITE
protocols. (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:           pcie_ctle:PCIe static CTLE mode, bit 0 - discrete on/off, bit 1 -
integrated on/off (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: 8MB (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:           port_reorder:Device port reorder: 1 - order HFIs on the same ASIC
in increasing port order, or 0 - order exactly as the kernel enumerates (default)
(bool)
parm:           max_mtu:Set max MTU bytes, default is 10240 (uint)
parm:           cu:Credit return units (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)
```

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 (these are the default values):

```
aspm 0
cache_size 256 cap_mask 0x4c09a00cb9a cu 1
desct_intr 64
eager_buffer_size 8388608
hdrq_entsize 32
kdeth_qp 128 krcvqs link_crc_mask 3
lkey_table_size 16
loopback 0
max_ahs 65535
max_cqes 3145727
max_cqs 131071
max_mcast_grps 16384
max_mcast_qp_attached 16
max_mtu 10240
max_pds 65535
max_qps 32768
max_qp_wrs 16383
max_sges 96
max_srqs 1024
max_srq_sges 128
max_srq_wrs 131071
no_user_mr_percpu 0
num_sdma 0
num_user_contexts 44
num_vls 8
pcie_caps 0
pcie_ctle 3
pcie_force 0
pcie_pset 2
pcie_retry 5
pcie_target 3
piothreshold 256 port_reorder N 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 SLES 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.

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 28 to see if your changes to `hfi1.conf` took effect.

The `krcvqs=3` setting only affects the first virtual lane (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

MPI libraries are a key type of middleware for building 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-x.y.z-hfi  mvapich2-x.y-hfi
gcc:
openmpi-x.y.z-hfi  openmpi-x.y.z  mvapich2-x.y-hfi  mvapich2-x.y
```

Note: *x.y.z* refers to the latest version of openmpi; *x.y* refers to the latest mvapich2.

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 Performance Scaled Messaging 2 (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-x.y.z-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-x.y-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`

  or the equivalent "`-genv I_MPI_FABRICS_LIST tmi -genv I_MPI_TMI_PROVIDER psm2 -genv I_MPI_FALLBACK 0`"

  Uses PSM2/tmi even for 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, `shm:tmi` performs the best overall; but this is application-dependent. Also, "`-genv I_MPI_TMI_PROVIDER psm2`" is useful with either the second or third bullets above to make sure you are using PSM2.

On clusters with Intel® Xeon Phi™ x200 Product Family CPUs, The setting "`-genv I_MPI_FABRICS tmi`," or the equivalent `tmi:tmi`, greatly reduces MPI latency (such as measured by the IMB-MPI1 PingPong benchmark) and improves the small message performance of MPI collectives such as Barrier and Allreduce.

To reduce the MPI start-up time in large node-count runs the following `mpirun` options should be added to your command line:

```
export  I_MPI_HYDRA_PMI_CONNECT=alltoall
export  I_MPI_PLATFORM_CHECK=0
export  I_MPI_HYDRA_BRANCH_COUNT=4
```

Notes:

- If you are using Intel® MPI 2017 or a later version, `I_MPI_HYDRA_PMI_CONNECT=alltoall` need not be specified.
- If you are using a version of Intel® MPI library 2017 Update 3 or newer, then the environment variable `I_MPI_JOB_FAST_STARTUP=on` has been shown to greatly improve MPI initialization/start-up time for very large node counts (hundreds or more). The feature is available for `I_MPI_FABRICS=shm:tmi` or `I_MPI_FABRICS=shm:ofi` modes only.
- If you are seeing stalled runs using the Intel® MPI Benchmarks—Remote Memory Access (IMB-RMA) benchmark, add `-genv I_MPI_FAIR_READ_SPIN_COUNT=1` as a workaround.

5.2 Intel® MPI 2017: New Support of OFI as a Fabric

With Intel® MPI 2017, Update 3 or newer, the default for `I_MPI_FABRICS` changes from `shm:tmi` to `shm:ofi`. `I_MPI_FABRICS=shm:ofi` implies the OpenFabrics Interface (OFI) library will be used for inter-node communications, if it is installed on all the nodes taking part in the communications. With an OPA fabric installed on these nodes, PSM2 will be the default provider used by OFI. With the Intel® Omni-Path Software 10.5 release, the OFI library will be included in the Basic and IFS packages. This change of default could provide an unexpected change in performance from the previous Intel® OP Software and Intel® MPI releases. The change to OFI over PSM2 could provide better or worse performance than the previous default, the TMI library over PSM2 that is provided with the Intel® MPI package.

If you follow the recommendations in the previous section of this guide, then Tag Matching Interface (TMI) is specified and OFI will not be used.

If you want to test with OFI, use settings such as the following:

- `-genv I_MPI_FABRICS shm:ofi,`
- `-genv I_MPI_FABRICS ofi`
- `-OFI`

If you are using a version of Intel® Omni-Path Software that is older than 10.5, then OFI may not be installed on your nodes. If you want to experiment with OFI, you may download the library and associated software from the "Downloads for Intel® Omni-Path Host Fabric Interface Products" site, or obtain the latest from https://ofiwg.github.io/libfabric/.

5.3 Intel® MPI Benchmarks or OSU Micro Benchmarks

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.

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 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, 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 also performs well when running the multi-PPN cases of the OMB or IMB benchmarks.

5.4 Tuning for High-Performance Linpack Performance

High-Performance Linpack (HPL) is a software package that solves a uniformly random, dense system of linear equations in double precision (64 bits) on distributed-memory computers, reporting time and floating-point execution rate. The HPL implementation of the HPC Linpack Benchmark is portable and freely available (see http://www.netlib.org/benchmark/hpl/).

Intel® Parallel Studio XE contains pre-compiled versions of HPL (referred to as the Intel® Distribution for LINPACK Benchmark) that are optimized using Intel® Math Kernel Library tools for best performance. The pre-compiled binaries are also processor- and platform-aware such that they automatically choose the most optimal core pinning and other environment variables at runtime.

5.4.1 Expected Levels of Performance

The peak flops for a node is given by the simple formula:

`Rpeak = Ncores * GHz(base) * (operations/cycle)`

Intel® Xeon Phi™ Processors deliver 32 operations/cycle and Intel® Xeon® Processor v3 and v4 families deliver 16, double-precision floating-point operations/cycle per CPU core. The Intel® Xeon® Scalable Processors delivers 32 double-precision floating-point operations/cycle per CPU core.

Base GHz for the Intel® Xeon Phi™ Processor 7210 is 1.3 GHz. So, the peak GFlops for this processor is `(68 cores)*(1.3 GHz)*(32 double-precision flops/cycle)` or ~ 2662.4 GFlops. The expected level of actual performance is 1933 GFlops (refer to Intel® Xeon Phi™ Processor Applications Showcase). This corresponds to a single node efficiency (`Rmax/Rpeak = 1933/2662.4)` of 72.6%.

Important: Before doing any multi-node HPL runs, be sure that individual node performance is evaluated and is in the expected range for the processor. The multi-node HPL calculation is throttled by the lowest performing node.

5.4.2 Selection of HPL Binary and MPI

Intel recommends that you use the pre-compiled version of HPL contained within Intel® Parallel Studio XE, known as the Intel® Distribution for LINPACK Benchmark. Currently, these are the versions located in the `compilers_and_libraries_2017.4.196 (2017.4.196)/linux/mkl/benchmarks/mp_linpack/` directory tree.

Be sure to use the matching Intel® MPI Library. For the recommended version above, use Intel® MPI Library for Linux* OS, Version 2017 Update 3 Build 20170405 (id: 17193).

To load the library into your environment:

```
source /opt/intel/parallel_studio_xe_2017.4.056/bin/psxevars.sh
```

5.4.3 MPI Flags and Proper Job Submission Parameters/Syntax

No specific MPI parameters are required to achieve the best HPL performance on Intel® OPA. However, the following flags are recommended to ensure that you are using the PSM2 software layer, which is higher performing than the verbs software layer.

Refer to Intel® MPI Library Settings on page 31 for the recommended TMI settings to ensure PSM2 is being used.

5.4.4 HPL.dat Input File

Detailed tuning of HPL.dat file is not generally required to achieve the bulk of the expected performance and are out of scope for this document.

Note: An online tool can be used to generate an example HPL.dat file.

The main parameters in HPL.dat file are:

- The "problem size" *N* dictates the solution of an *NxN* matrix, which determines the total amount of memory used.
  - To estimate the total amount of memory for a given *N*, use this formula:

    $GB = (N*N*8)/2^{30}$

    where

    `GB` is the total amount of memory required in GibiBytes

    `8` is due to the size of an 8 byte word,

    2^{30} is the number of bytes in a Gibibytes.
  - Conversely, to determine *N* for a desired memory usage, use this formula:

    $N= SQRT((GB* 2^{30})/8)$
  - Typically, we use 60-90% of the system memory (`%M = 0.6-0.9`) for the most optimal HPL score. Then, for a multi-node (*C* nodes) where NODEMEM is the total memory per node in Gibibytes, the cluster-level calculation is:

    $N = ROUND(SQRT((C * \%M * NODEMEM * 2^{30})/8))$

  where `ROUND` is a function that rounds to a nearby integer.
- The "block size" *NB*: An optimal block size that typically will return the best performance and no adjustment is necessary. Optimal block sizes are defined for each processor below:
  - Intel® Xeon Phi™ Processor: 336
  - Intel® Xeon® Processor E5 v3 Family and v4 Family: 192
  - Intel® Xeon® Scalable Processors: 384
- "*Ps* and *Qs*": *P* and *Q*. *P* and *Q* govern the dimension of the problem matrix.

  Selecting *P* and *Q* is somewhat of a "fine-tuning" but always make sure `P ~ Q` and `P * Q = number of MPI ranks.`

  The best P and Q are determined by the balance of computation and network performance. If higher computational performance is required, a larger P is required. If the network becomes the bottleneck, then a larger Q is required to reduce vertical communication in the matrix. For example, if a node has add-in coprocessor cards, the best performance may be seen if `P > Q`.

  Very large scale calculations on Intel® Xeon Phi™ Processors have shown that the best scaling occurs where `Q ~ 4*P`. For instance, for a 1024-node calculation using one MPI rank per node, set `P=16` and `Q=64`.

5.4.5 Recommended Procedure for Achieving Best HPL Score

Typically, the Intel-provided binary provides good performance by automatically choosing optimal MPI/Open MP core affinity. However, you still must execute the binary and example scripts requesting the appropriate number of MPI ranks and MPI ranks per node. The binary automatically selects the number of Open MP threads and placement based on the exact processor detected, and environment variables that are set by the run scripts.

- For Intel® Xeon Phi™ Processors, the recommendation is to use one MPI rank per node.

- On dual socket Intel® Xeon® Processor systems, the recommendation is to use one MPI rank per NUMA node.

During an HPL run, if you run the Linux* `top` command on the node, you should see one HPL process using all of the core resources. For example, on a 64-core Intel® Xeon Phi™ Processor system, you would see one HPL process using 6400% of CPU during peak execution. On a dual socket, 16-core Intel® Xeon® Processor system, you would see two HPL processes each consuming 1600% of CPU resources. This implies that the Open MP threads are active within each MPI task.

To achieve the best HPL score, perform the following:

1. Load the Intel tools into your path as discussed in Selection of HPL Binary and MPI on page 34.
2. Perform a single node HPL run on each node you plan to use in the multi-node run.

   If you are able to choose from a group of nodes greater than the target multi-node count and eliminate the lowest performing nodes, you may achieve a greater multi-node score. For example, if you want to perform a 128-node HPL run and have 144 nodes available to test, remove the lowest 16 performing nodes for your final multi-node run.

   Run a single node HPL calculation using a small problem size (such as 10000) as a first functional test. Note that you can pass arguments after the binary so you can adjust parameters on the fly, but HPL.dat still needs to be present in the directory. Make sure you save the log files for all runs for future reference if required.

   The score in GFlops is reported in the right-most column of the program output. You should check this to make sure the performance is as expected for the particular CPU you are using. A general rule of thumb is ±5% as an acceptable performance range.

   Note the run time increases appreciably with increasing *N*, especially for multi-node studies, so it is important to weigh the benefit of increasing *N* for both score and runtime allowance.

   Ideally, this study should be performed for all nodes individually; and, low-performing nodes should be identified and corrected.
3. Perform a multi-node run with all of the selected hosts, using the included run scripts with the Intel® Distribution for LINPACK Benchmark. Helpful details can be found at https://software.intel.com/en-us/mkl-linux-developer-guide-overview-of-the-intel-distribution-for-linpack-benchmark.

   For very large scale multi-node runs (up to thousands), the efficiency may be up to 5–10% lower than the lowest single node efficiency used in the run. For smaller cluster runs, the fabric impact on efficiency should be less.

5.5 MPI Applications Performance Tuning

Certain non-default settings for either HFI1 driver parameters or PSM2 or Intel® MPI environment variables may improve HPC applications performance. For Intel® Omni-Path Fabric Suite software release 10.5 or newer, the following tunings have been tested on the indicated processor types with positive results.

Table 6. Intel® Omni-Path Fabric Suite Software Release 10.5 or Newer

Application	Processor	Tuning Parameters[1]
Ansys Mechanical 18.0, Case=bga	Intel® Xeon® Processor E5 v4 Family	`I_MPI_ADJUST_GATHERV=3`[2]
LSTC's LS-DYNA*, test models: Neon, Car2Car, and 3Cars	Intel® Xeon® Processor E5 v4 Family	`rcvhdrcnt=8192`
OpenFOAM	Intel® Xeon® Scalable Processors	`I_MPI_FABRICS=ofi`
115.fds4[3], Fire Dynamics Simulator 4	Intel® Xeon® Scalable Processors	`rcvhdrcnt=4096`

Notes: 1. To set the `rcvhdrcnt` driver parameter, refer to Setting HFI1 Driver Parameters on page 29.
2. Parameters that begin with `I_MPI_` are Intel® MPI environment variables.
3. This application is part of the SPEC MPI2007 Medium suite. This tuning has no negative effects on the other 12 codes in the suite, so can be used for a base run.

For older Intel® Omni-Path Software releases, please refer to the *Intel® Omni-Path Fabric Performance Tuning User Guide* for software release 10.4.2 and earlier.

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

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

5.6.1 MPI Collective Scaling Guidelines for Large Clusters

Recent large-scale performance tuning on a large Intel® Xeon Phi™ x200 Product Family cluster resulted in key learnings for better performance. If your cluster is larger than 1000 nodes and if applications are run at a scale of over 1000 nodes, you can significantly improve scaling of certain MPI collectives, such as Allreduce and Barrier, by adding some Linux boot command line settings and a BIOS setting.

To reduce OS and hardware noise to improve MPI collectives scalability, perform the following steps as root:

1. Disable Hyper-Threading in the BIOS.
2. Add two settings to the boot command line, as follows:
   a. Edit `/etc/default/grub` by adding "`idle=halt nohz_full=1-`*XX*" to `GRUB_CMDLINE_LINUX`, where *XX* is one less than the number of CPU cores on the node. For example, on a node with a 68-core CPU:

```
GRUB_CMDLINE_LINUX=vconsole.keymap=us console=tty0
vconsole.font=latarcyrheb-sun16 crashkernel=256M
console=ttyS0,115200 idle=halt nohz_full=1-67
```

3. Apply the changes using:

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

4. Reboot.

The default idle (`idle=mwait`) means that the OS wakes up periodically on every idle CPU (all the hyper threads) to issue a scheduling check. The main effect of `idle=halt` with HT off is to keep turbo transitions from happening, which is source of hardware noise. `nohz_full` results in a tick-less kernel on that list of CPU cores/threads.

If you see improved applications scalability with these settings, you may want to evaluate whether this improvement offsets any potential loss of applications performance due to turning HT off.

5.6.2 Driver Parameter Settings for Intel® Xeon Phi™ x200 Product Family

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

1. With Intel® MPI, to significantly improve small message latency and scalability of collectives, set "`-genv I_MPI_FABRICS tmi`" as part of your mpirun/mpiexec command.
2. To set the number of user contexts per node:
   a. Edit `/etc/modprobe.d/hfi1.conf` (see Setting HFI1 Driver Parameters on page 29)
   b. 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).

The MPI bandwidth obtained when running a benchmark like osu_bw may vary from core to core. If the test was run separately when pinned to each Intel® Xeon Phi™ Processor core, then there would likely be one best-performing core, one slower-performing core, and the rest of the cores yielding average performance. For details on the reason for this and how to tune to get more "best-performing" cores, see Mapping from MPI Processes to SDMA Engines on page 54.

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

Intel® Xeon Phi™ x200 Product Family 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 29)

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 provides enough hardware contexts to support the 288 MPI ranks.

5.7 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 such as 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 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.

5.8 Tuning for Improved Performance on QCD Applications

QCD applications are important at many HPC centers. QCD describes how protons and neutrons interact and what goes on at high-energy accelerators. The following are some tuning tips to speed the performance of some of these applications:

- Set the PSM2 environment variable `PSM2_RTS_CTS_INTERLEAVE=1`. This setting can significantly help some QCD tests that use large messages (16 MB or larger).
- Refer to Dealing with Memory Fragmentation on page 25 for additional tips that can aid QCD performance.

5.9 GPUDirect* RDMA Tuning for MPI Benchmarks and Applications

Note: *GPUDirect Remote Direct Memory Access (RDMA) was formerly known as GPUDirect v3.*

From the NVIDIA* CUDA* Toolkit Documentation:

> *"GPUDirect RDMA is a technology which enables a direct path for data exchange between the GPU and third-party peer devices using standard features of PCI Express. Examples of third-party devices include network interfaces, … "*

The Intel® OP HFI Adapter is an example of a network interface device that supports GPUDirect RDMA. This section discusses how to best use this capability on Intel® Omni-Path-enabled compute nodes.

GPUDirect RDMA is available on two families of NVIDIA GPUs: Tesla* and Quadro*.

Prerequisites

The following prerequisites are needed to obtain maximum bandwidth and best latency between CUDA-enabled GPU devices:

1. Connect the GPU and Intel® Omni-Path adapter card to the same CPU (socket) via PCIe buses. As stated in the CUDA documentation, the two devices must share the same upstream PCI Express root complex.
2. Use a CUDA-aware MPI, such as the Open MPI provided as part of the Intel® OPA Software 10.4 (or newer) release, installed by default at `/usr/mpi/gcc/openmpi-X.Y.Z-cuda-hfi`.
3. Use a CUDA-enabled application or benchmark, such as OMB 5.3.2 configured and built with `--enable-cuda` and other settings documented in the OMB README file; and, run using `-d cuda D D` at runtime.
4. Set PSM2 environment variables `PSM2_CUDA=1` and `PSM2_GPUDIRECT=1`. For example, Open MPI mpirun command line options `-x PSM2_CUDA=1 -x PSM2_GPUDIRECT=1` would propagate these variable settings to the compute nodes running the job.

Use Cases

Usage tips and examples depend on the number of Intel® Omni-Path adapters used per node. Four cases include: 1) single-HFI systems; 2) multi-HFI systems where each MPI rank uses one HFI; 3) multi-HFI systems where each MPI rank uses multiple HFIs (to increase single-rank bandwidth); and, 4) single-HFI systems where the HFI and GPU are connected to different CPU sockets (that is, Prerequisite 1 above was not possible).

1. Single HFI per node

   This is the most common situation. The HFI adapter and GPU are connected to the same socket. First assume that the compute nodes have the HFI adapter connected to socket 0. An example mpirun command line that follows these recommendations is:

   ```
   /usr/mpi/gcc/openmpi-1.10.4-cuda-hfi/bin/mpirun -mca pml cm \
    -mca mtl psm2 -H node01,node02 -x PSM2_CUDA=1 \
    -x PSM2_GPUDIRECT=1 ./osu_bw -d cuda D D
   ```

   Note: Typically, the full pathname to mpirun is not required if the bin directory is in your path. However, in the command line above, we wanted to emphasize that the CUDA-enabled MPI, that is `openmpi-1.10.4-cuda-hfi`, should be used for GPU workloads.

2. Dual HFI per node where each MPI rank running on the node uses one HFI.

   If the GPU device is connected to the second CPU socket (socket 1) and a second HFI on the system also is connected to socket 1 (HFI_UNIT=1), then a good-performing mpirun command line would be:

   ```
   mpirun -mca pml cm -mca mtl psm2 -H node01,node02 -x HFI_UNIT=1 \
    -x PSM2_CUDA=1 -x PSM2_GPUDIRECT=1 ./osu_bw -d cuda D D
   ```

The command line above leaves the placement of the osu_bw process to the OS scheduler. If the compute nodes had two 14-core CPUs, and you wanted more control over which core and socket on which the benchmark process ran, the following command with `taskset` could be employed:

```
mpirun -mca pml cm -mca mtl psm2 -H node01,node02 -x PSM2_CUDA=1 \
 -x PSM2_GPUDIRECT=1 taskset -c 20 ./osu_bw -d cuda D D
```

Alternatives to `taskset -c 20` include: `taskset -c 14-27` or `numactl -N 1` to place the task on any core of socket 1 (also known as NUMA node 1) and potentially allow movement between cores on NUMA node 1. Note that we did not specify which HFI in the command above.

If there are more than one HFI is on same socket, PSM2 spreads the ranks uniformly across the HFIs within the socket (default behavior).

You can use Round-Robin algorithm to consider all HFIs in the system (and not restrict to just the socket) by setting the environment variable `HFI_SELECTION_ALG="Round Robin All"` (note that this is not recommended for GPU workloads).

3. Dual HFI per node where each MPI rank uses both HFIs, when possible.

   - For each MPI rank NOT running GPU workloads, you may set the environment variable, `PSM2_MULTIRAIL=1` to stripe large messages across both HFIs.
   - For the MPI ranks running GPU workloads, you have the following two cases:

     a. Both HFIs are connected to the same socket as the GPU.

        In this case, you can set `PSM2_MULTIRAIL=2` to stripe large messages across both HFIs. If both HFIs were on socket 1 (of 0,1), then the following would be a good mpirun command:

        ```
        mpirun -mca pml cm -mca mtl psm2 -H node01,node02 -x PSM2_MULTIRAIL=2 \
        -x PSM2_CUDA=1 -x PSM2_GPUDIRECT=1 numactl -N 1 ./osu_bw -d cuda D D
        ```

     b. If the two HFIs are connected to different sockets, you should **not** use PSM2_MULTIRAIL. For the GPU workload, you should specify to use only the single HFI on the same socket as the GPU. For example, if both the GPU and hfi1_1 were on NUMA node 1 (of 0,1), you could use the following mpirun command to run a bandwidth test:

        ```
        mpirun -mca pml cm -mca mtl psm2 -H node01,node02 -x HFI_UNIT=1 \
         -x PSM2_CUDA=1 -x PSM2_GPUDIRECT=1 numactl -N 1 ./osu_bw -d cuda D D
        ```

4. HFI adapter and GPU connected to different sockets.

   The HFI adapter and GPU are PCIe-connected to different CPUs/sockets. A loss of performance occurs in this situation (as compared to having both connected to the same socket), but most of it can be gained back by setting the environment variable `PSM2_GPU_DIRECT_RECV_THRESH` to `4096` as per the example below:

```
/usr/mpi/gcc/openmpi-1.10.4-cuda-hfi/bin/mpirun -mca pml cm \
-mca mtl psm2 -H node01,node02 -x PSM2_CUDA=1 -x PSM2_GPUDIRECT=1 \
-x PSM2_GPU_DIRECT_RECV_THRESH=4096 ./osu_bw -d cuda D D
```

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 Accelerated RDMA

Accelerated RDMA is a Verbs protocol extension to improve the performance of RDMA write and RDMA read operations on Intel® Omni-Path hardware. This extension improves the efficiency of large message transfers to provide performance benefits for storage protocols and other Verbs-based protocols. The performance benefits include increased achievable bandwidth with reduced CPU utilization. This feature accelerates the OpenFabrics Alliance* (OFA) Verbs API with no changes required to API consumers. Since the acceleration technique is performed by the host driver, the application running over the OFA Verbs API does not need to make any code change.

Accelerated RDMA is off by default. The following conditions must be true to enable and engage Accelerated RDMA.

To enable Accelerated RDMA, perform the following steps:

1. Edit `/etc/modprobe.d/hfi1.conf` file (see Setting HFI1 Driver Parameters on page 29).
2. Add the `cap_mask=0x4c09a01cbba` setting to the 'options hfi1' line in the file.
3. Restart the hfi1 driver to activate the setting as described in Setting HFI1 Driver Parameters on page 29.

To use Accelerated RDMA, ensure that your application or middleware (such as file system software) meets the following conditions:

1. Use a payload size of at least 256 KB.
2. Set the payload size as a multiple of 4 KB.
3. Align the data buffers to 4 KB page boundaries.

6.2 Parallel File System Concurrency Improvement

The Accelerated RDMA setting should be a preferred setting for high concurrency file system traffic compared to the Adaptive Cache Memcpy heuristic parameter setting.

If the Accelerated RDMA setting does not work well with your workload, as an alternative you can turn on the Adaptive Cache Memcpy, by setting the parameter `sge_copy_mode=2` similar to other driver parameters as discussed in Setting HFI1 Driver Parameters on page 29.

Another driver parameter 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 29).
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.

On a storage system server node, typically there is no competing MPI traffic or floating point compute activity, so more of the cores can be used for kernel receive queues. In this case, using settings such as `krcvqs=4` up to `krcvqs=8` may provide the best throughput and IO operations per second for the file system.

6.3 Lustre

Lustre* Version 2.8 or newer or Intel® Enterprise Edition for Lustre* software Version 3.0 or newer is recommended. The Lustre module load process will automatically load the necessary tunings for optimal Intel® Omni-Path performance.

Lustre* Version 2.10 includes a performance improvement relevant to Intel® Omni-Path and other fabrics, particularly for 1-client to 1-server tests.

6.4 IBM Spectrum Scale (aka GPFS) Tuning for Intel® Omni-Path

As mentioned previously, the Accelerated RDMA setting should be a preferred setting for high concurrency file system traffic, such as a cluster with an IBM Spectrum Scale parallel file system (previously known as a General Parallel File System, GPFS). The accelerated RDMA setting must made on all the nodes involved with the file system large message/block traffic. For example, the `cap_mask=0x4c09a01cbba` setting must appear in the `/etc/modprobe.d/hfi1.conf` file on both the NSD server and the client nodes.

The conditions to engage Accelerated RDMA are specified in Accelerated RDMA on page 42. For the first two conditions—payload size is ≥ 256 KB and payload size is a multiple of 4 KB—use the following Spectrum Scale parameter settings (tunings):

```
NSD Server:
mmchconfig verbsRdmaMaxSendBytes=1024k, scatterBufferSize=1024k,
verbsRdmaMaxSendSge=40, maxMBpS=32000
```

```
Client/compute node:
mmchconfig verbsRdmaMaxSendBytes=1024k, scatterBufferSize=1024k,
verbsRdmaMaxSendSge=40, pagepool=4096M, prefetchPct=50, maxMBpS=16000
```

For the third condition—data buffer is 4K page aligned—this is always in effect for Spectrum Scale RDMA traffic.

If any of the nodes involved with the Spectrum Scale file system traffic has two Intel® Omni-Path adapters, and you want both to support the file system traffic, then for each such node, use the `mmchconfig` command to set the verbsPorts parameter as:

```
verbsPorts="hfi1_0 hfi1_1"
```

The following Intel® Xeon® Processor E5 v3 and v4 families BIOS tunings, found in Table 3 on page 17, were found to be particularly important for a Spectrum Scale file system workload:

```
IOU Non-posted Prefetch= Disabled
NUMA Optimized= Enable (aka Memory.SocketInterleave=NUMA)
Snoop Holdoff Count=9
```

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 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 RDMA bandwidth performance is the `ib_write_bw test`, with the default connection type `Reliable Connection`. 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 Perftest benchmarks 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 ib_ipoib`
2. `sudo ifup ib0`
3. `ib_write_bw -F -R -s 1048576 // on server node`
4. `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 20 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.

Note: If your interface names are non-standard (not using conventional `ib*`), these settings may not get applied properly from the `ifcfg-ib*` files. For example, if your interface name is `opa_ib0`, you will need to modify the settings in `ifcfg-opa_ib0` to enforce the above settings. Please confirm `ib*` interface names on your system to apply these settings.

8.2 IPoFabric Datagram 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/opa-fm/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>100g</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 Intel® Omni-Path Fabric Suite Fabric Manager, 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 RPS and GSO Tuning for IPoFabric Performance

Receive Packet Steering (RPS) tuning may improve performance when a larger number of client nodes access services on one or more servers using IPoFabric transport. In terms of logic, RPS is a software implementation of receive-side scaling. Being in the software, it is necessarily called later in the datapath. RPS selects the CPU to perform protocol processing above the interrupt handler.

RPS requires a kernel compiled with the `CONFIG_RPS kconfig` symbol (on by default for SMP). Even when compiled, RPS remains disabled until explicitly configured. The list of CPUs to which RPS may forward traffic can be configured for each receive queue using a sysfs file entry:

```
/sys/class/net/<dev>/queues/rx-<n>/rps_cpus
```

Generic Segmentation Offload (GSO) uses the TCP or UDP protocol to send large packets. GSO performs segmentation/fragmentation operations, bypassing the NIC hardware. This is achieved by delaying segmentation until as late as possible, for example, when the packet is processed by the device driver. This became default behavior in RHEL 7.4.

8.3.1 RPS Tuning for Intel® Xeon® Processor Nodes

For a service being accessed by clients over the ib0 device, the following tuning has been shown to work on dual processor systems of the Intel® Xeon® Processor E5 v4 Family, and Intel® Xeon® Scalable Processors generations:

```
echo ffffff00 > /sys/class/net/ib0/queues/rx-0/rps_cpus
```

Configurations of one server and up to five or six clients have been tested with an iperf stream between the server and each of the clients. Significant bandwidth improvements have been observed when five or more clients are accessing one server. In these tests, hfi1 driver parameter `krcvqs=3` was set on the 44-core Intel® Xeon® Processor E5 v4 Family, and `krcvqs=4` was set on the 48-core Intel® Xeon® Scalable Processors. (See Setting HFI1 Driver Parameters on page 29 for details on setting this or other driver parameters.)

To make the above tuning persist after reboots, the next section describes a method using the `/etc/rc.local` file.

8.3.2 RPS and GSO Tuning for Intel® Xeon Phi™ x200 Product Family Nodes

For Intel® Xeon Phi™ x200 Product Family processor nodes, the following tuning is recommended to improve throughput when using multiple IPoFabric (IPoIB) streams of traffic per node.

Perform the following steps using root privileges:

1. Ensure the OPA hfi1 driver and ipoib driver are configured to autostart.
2. Add the following two lines to the file `/etc/rc.local`:

   ```
   ethtool -K <IPoIB-port-name> gso off
   echo "00,00000000,ffff0000" > /sys/class/net/<IPoIB-port-name>/queues/rx-0/
   rps_cpus
   ```

   where `<IPoIB-port-name>` is often `ib0`, but could be `ib1`, `opa`, `opa_ib0`, etc. So, for one common case, the commands are:

   ```
   ethtool -K ib0 gso off
   echo "00,00000000,ffff0000" > /sys/class/net/ib0/queues/rx-0/rps_cpus
   ```

3. Make sure `/etc/rc.local` script file is executable by issuing:

   ```
   # chmod +x /etc/rc.local
   ```

4. Reboot to activate the changes.

8.4 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.

8.5 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.

8.6 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 gigabyte (GB) 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 (or programs or middleware using the verbs API) use a SDMA mechanism 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) 10240 bytes
   MTU  Active By VL:
   00:10240 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 one 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
```

9.1 Mapping from MPI Processes to SDMA Engines

Typically, MPI jobs are assigned to a specific service level; and, this is mapped to a service channel and virtual lane 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 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 exposes new sysfs entries per SDMA engine. For each SDMA engine, the driver creates 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, will not exist on your system.

To illustrate an application of the features and principles described above, we run the OSU Micro Benchmark, osu_bw, at a 4 MB message size (run on nodes with Intel® Xeon Phi™ x200 Product Family at 1.5 GHz, 72 cores, with `max_mtu=10240` set, Turbo mode off), and see the results 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™ x200 Product Family 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 the 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.