

Intel® Ethernet Fabric

Performance Tuning Guide

Rev. 1.9

July 2024

Doc. No.: 632488, Rev.: 1.9



You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All product plans and roadmaps are subject to change without notice.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Your costs and results may vary.

Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Copyright © –2023, Intel Corporation. All rights reserved.

Revision History

Date	Revision	Description
June 2024	1.9	Product 11.7.0.0 release - Changes to this document include: • Removed MPI uni and bi-directional bandwidth tuning from PSM3 Environment Variables
February 2024	1.8	Product 11.6.0.0 release - Changes to this document include: • Added Active-State Power Management to BIOS and Platform Settings
September 2023	1.7	Product 11.5.1.0 release - Changes to this document include: • Updated Using the Intel P-State Driver • Added Note to oneAPI Level Zero and GPUDirect • Added missing QoS setting to Switch Configurations for PFC • Assorted grammatical, formatting and style improvements through the document.
May 2023	1.6	Product 11.5.0.0 release - Changes to this document include: • Updated Switch Configurations for PFC to use Traffic Class 0 instead of 2 • Added Performance Tuning for Intel GPU • Updated MPI Performance and MPI Benchmark Fundamentals to point to *Intel® Ethernet Fabric Suite FastFabric User Guide* on building and running benchmarks. As well as updating example commands to use `run_*` scripts. • Updated GPU Direct Requirements to include section on Intel® GPU Direct.
January 2023	1.5	Product 11.4.1.0 release - Changes to this document include: • Updated MPI Collective and Intel® oneCCL Tunings
November 2022	1.4	Product 11.4.0.0 release - Changes to this document include: • Updated file paths for Ubuntu system • Updated NIC Configuration for PFC to use software DCB
June 2022	1.3	Product 11.3.0.0 release - Changes to this document include: • Increase PSM3_RV_MR_CACHE_SIZE for very large MPI messages with PSM3_RDMA mode 1. PSM3 Environment Variables • Improving MPI Alltoall performance. MPI Collective and Intel® oneCCL Tunings • Improving PSM3 TCP performance with PSM3_TCP_SKIPPOLL_COUNT. TCP Performance • Added Intel®oneCCL Multi-NIC guidance: MPI Collective and Intel® oneCCL Tunings • Increase PSM3_CUDA_THRESH_RNDV to a very large value (always use eager) when using CUDA with TCP. CUDA and GPUDirect • PSM3 as of IEFS 11.3 release may provide higher GPU Direct bandwidth on systems with PCIe switches. CUDA and GPUDirect
March 2022	1.2	Product 11.2.0.0 release - Changes to this document include: • Added guidance to disable PCIe Access Control Services. GPU Direct Requirements • Removed guidance on how to reduce CPU clock frequency with intel_pstate, since it is not commonly used

continued...

Date	Revision	Description
		• Enhanced chapter on Priority Flow Control (PFC) configuration on NICs and example switches Priority Flow Control Configuration and Tuning • Added section on Intel®oneCCL MPI Collective and Intel® oneCCL Tunings • Added discussion on NVIDIA Multi-Process Service (MPS) CUDA and GPUDirect and NCCL NVIDIA Collectives Communication Library (NCCL) • Added section on TCP performance tuning with PSM3 TCP Performance
July 2021	1.1	Product 11.1.0.0 release - Changes to this document include: • MPI collective tuning algorithms added to MPI Collective and Intel® oneCCL Tunings • Added guidance to lower NIC Tx/Rx queues for improved application performance over TCP. IRQ Affinity and irqbalance. • Updated description of the `roce_ena` parameter in irdma Module Settings. • Added section on CUDA and GPUDirect • Added discussion on using PSM3_PRINT_STATS to detect fabric drops in PSM3 Environment Variables • Added details on Arista 7170 PFC tuning in Priority Flow Control
February 2021	1.0	Initial release.

Contents

Revision History....3

Preface....7

Intended Audience....7
Intel® Ethernet Fabric Suite Documentation Library....7
How to Search the Intel® Ethernet Fabric Suite Documentation Set....8
Documentation Conventions....8
Best Practices....9
License Agreements....9
Technical Support....9

1.0 Introduction....10

1.1 Terminology....10
1.2 Performance Tuning Quick Start Guide....11

2.0 BIOS and Platform Settings....12

2.1 BIOS Recommendations....12
2.2 GPU Direct Requirements....13

3.0 Linux Settings....14

3.1 CPU Frequency Scaling Drivers....14
3.1.1 Using the Intel P-State Driver....14
3.1.2 Using the ACPI CPUfreq Driver and cpupower Governor....15
3.2 Priority Flow Control....17
3.3 IRQ Affinity and irqbalance....17
3.4 Memory Fragmentation....18
3.5 irdma Module Settings....18
3.6 Intel® Ethernet Driver (ice) Settings....20
3.7 TuneD Tuning Service....20

4.0 MPI Performance....22

4.1 MPI Benchmark Fundamentals....22
4.2 Intel® MPI Library Settings....25
4.3 PSM3 Environment Variables....26
4.4 MPI Collective and Intel® oneCCL Tunings....28
4.5 MPI Affinity....31
4.6 Dual/Multi-Rail....32
4.7 TCP Performance....32

5.0 Performance Tuning for Intel GPU....33

5.1 oneAPI Level Zero and GPUDirect....33

6.0 Performance Tuning for NVIDIA GPU....34

6.1 CUDA and GPUDirect....34
6.2 NVIDIA Collectives Communication Library (NCCL)....35

7.0 Priority Flow Control Configuration and Tuning....36

7.1 NIC Configuration for PFC....36
7.2 Switch Configurations for PFC....38
7.3 Verification for PFC....42

Tables

1 Terminology.....10
2 Recommended BIOS Settings (Intel® Xeon® Scalable Processors).....12
3 PSM3 RoCEv2 (Verbs) Performance Tunings..... 26

Preface

This manual is part of the documentation set for the Intel® Ethernet Fabric Suite Fabric (Intel® EFS Fabric), which is an end-to-end solution consisting of Network Interface Cards (NICs), fabric management, and diagnostic tools.

The Intel® EFS Fabric delivers the next generation, High-Performance Computing (HPC) network solution that is designed to cost-effectively meet the growth, density, and reliability requirements of HPC and AI training clusters.

Intended Audience

The intended audience for the Intel® Ethernet Fabric Suite (Intel® EFS) document set is network administrators and other qualified personnel.

Intel® Ethernet Fabric Suite Documentation Library

Intel® Ethernet Fabric Suite publications are available at the following URL:

https://www.intel.com/content/www/us/en/support/articles/000088090/ethernet-products/intel-ethernet-software.html

Use the tasks listed in this table to find the corresponding Intel® Ethernet Fabric Suite document.

Task	Document Title	Description
Installing host software Installing NIC firmware	*Intel® Ethernet Fabric Suite 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 fabric using FastFabric	*Intel® Ethernet Fabric Suite FastFabric User 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 Text-based User Interface (TUI) menus and command line interface (CLI) commands.
Running MPI applications on Intel® EFS Running middleware that uses Intel® EFS	*Intel® Ethernet Fabric Suite Host Software User Guide*	Describes how to set up and administer the Network Interface Card (NIC) after the software has been installed and provides a reference for users working with Intel PSM3. Performance Scaled Messaging 3 (PSM3) is an Open Fabrics Interface (OFI, also called libfabric) provider which implements an optimized user-level communications protocol. The audience for

continued...

Task	Document Title	Description
		this document includes cluster administrators and those running or implementing Message-Passing Interface (MPI) programs.
Optimizing system performance	*Intel® Ethernet Fabric Performance Tuning Guide*	Describes BIOS settings and parameters that have been shown to ensure best performance, or make performance more consistent, on Intel® Ethernet Fabric Suite Software. If you are interested in benchmarking the performance of your system, these tips may help you obtain better performance.
Learning about new release features, open issues, and resolved issues for a particular release	*Intel® Ethernet Fabric Suite Software Release Notes*	

How to Search the Intel® Ethernet Fabric Suite Documentation Set

Many PDF readers, such as Adobe Reader and Foxit Reader, allow you to search across multiple PDFs in a folder.

Follow these steps:

1. Download and unzip all the Intel® Ethernet Fabric Suite PDFs into a single folder.
2. Open your PDF reader and use **CTRL-SHIFT-F** to open the Advanced Search window.
3. Select **All PDF documents in...**
4. Select **Browse for Location** in the dropdown menu and navigate to the folder containing the PDFs.
5. Enter the string you are looking for and click **Search**.

Use advanced features to further refine your search criteria. Refer to your PDF reader Help for details.

Documentation Conventions

The following conventions are standard for Intel® Ethernet Fabric Suite 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 to a figure, table, or section in this guide. Links to websites are also shown in blue. For example:

  See License Agreements 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® Ethernet Fabric Suite 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® Ethernet Host Software or Intel® Ethernet Fabric Suite is being used on the hosts.
- Tasks that are generally applicable to all environments are not marked.

Best Practices

- Intel recommends that users update to the latest versions of Intel® Ethernet Fabric Suite software to obtain the most recent functional and security updates.
- To improve security, the administrator should log out users and disable multi-user logins prior to performing provisioning and similar tasks.

License Agreements

This software is provided under one or more license agreements. 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

Creating a technical support ticket for Intel® Ethernet Fabric Suite products is available 24 hours a day, 365 days a year. Contact Intel® Customer Support or visit https://www.intel.com/content/www/us/en/support.html for additional details.

1.0 Introduction

The Intel® Ethernet Fabric Suite (Intel® EFS) 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 settings and parameters that have been shown to improve MPI/HPC performance on Intel® Ethernet Fabric Suite. 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® EFS product line, refer to Intel® Ethernet Fabric Suite Documentation Library of this document. You may also consult the *Intel® Ethernet 800 Series Linux Performance Tuning Guide* for Intel® Ethernet E810 Network Adapter-specific tunings.

This version of the tuning guide is focused only on the optimization of MPI/HPC applications. Future versions will contain guidance for optimization with parallel file systems for high performance storage.

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
CUDA	Compute Unified Device Architecture
GCC	GNU Compiler Collection
HPC	High-Performance Computing
HPL	High-Performance Linpack
HT	Intel® Hyper Threading
EFS	Intel® Ethernet Fabric Suite
IMB	Intel® MPI Benchmarks
IO	Input/Output
IP	Internet Protocol
IRQ	Interrupt Request
MPI	Message Passing Interface
MTU	Maximum Transmission Unit

continued...

Term	Description
NCCL	NVIDIA Collective Communication Library
NUMA	Non-Uniform Memory Access
OFED	OpenFabrics Enterprise Distribution
OFI	OpenFabrics Interface
OMB	OSU Micro Benchmarks
OS	Operating System
OSU	Ohio State University
PPN	Processes per Node
PSM3	Performance Scaled Messaging 3
QP	Queue Pair
RDMA	Remote Direct Memory Access
RoCE	RDMA over Converged Ethernet
SDMA	Send Direct Memory Access
SMP	Symmetric Multiprocessing
TBB	Intel® Threading Building Blocks
TC	Traffic Class
TCP	Transmission Control Protocol
THP	Transparent Huge Pages
VM	Virtual Machine
VT	Intel® Virtualization Technology

1.2 Performance Tuning Quick Start Guide

The list below is intended to outline the most important tunings for Intel® EFS performance, sorted by most important at the top, to least important moving towards the bottom of the list. This is only a rough guide and individual clusters may require other tunings, discussed in other sections of this guide.

- Set BIOS settings. (See BIOS and Platform Settings.)
- Enable Intel® Turbo Boost Technology, if possible.

  Enable "Performance Governor" with either ACPI or Intel® P-State frequency driver:

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

- Make sure the MPI is using libfabric with the PSM3 provider. (See Intel® MPI Library Settings.)

  Use the latest available version of the Intel® MPI Library for optimized application performance.
- Confirm that priority flow control (PFC) is configured correctly and there is no packet loss in the fabric impacting performance. (See Priority Flow Control.)
- Consider adjusting PSM3 environment variables to further tune performance.

2.0 BIOS and Platform Settings

Setting the system BIOS is an important step in configuring a cluster to provide the best mix of application performance and power efficiency. This section lists settings that can maximize 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 BIOS Recommendations

This section provides an example of the recommended BIOS settings.

Table 2. Recommended BIOS Settings (Intel® Xeon® Scalable Processors)

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
Package C-State	C6(Retention) state
C1E	Enabled
Processor C6	Enabled
Intel® Hyper-Threading Technology	No recommendation
IOU Non-posted Prefetch	Disabled (where available)[2]
NUMA Optimized	Enable[3]
Sub_NUMA Cluster	Disabled

continued...

BIOS Setting	Value
Snoop Holdoff Count	9[4]
Active-State Power Management	Disabled

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® Ethernet Fabric performance.
2. May not be visible in the BIOS settings. A setting of *enabled* may cause limits in peak bandwidth.
3. Also known as `Memory.SocketInterleave=NUMA` in some BIOSes.
4. Also known as `Snooped Response Wait Time for Posted Prefetch` in some BIOSes.

2.2 GPU Direct Requirements

This section provides an example fo the recommenced BIOS settings for use with GPU Direct.

Intel® GPU Direct

For GPU Direct to function properly, Intel recommends disabling PCIe Access Control Services (ACS), also known as IO virtualization, VT-d, or IOMMU. If left enabled, unpredictable behavior such as application failures may be experienced.

Additionally, setting `Memory Mapped I/O Size` to `1024G` and `MMIO High Base` to `56T` can improve performance, if available.

Refer to https://dgpu-docs.intel.com/driver/installation.html for more information on configuring the host system.

NVIDIA GPU Direct

For GPU Direct to function properly, NVIDIA recommends disabling PCIe Access Control Services (ACS), also known as IO virtualization, VT-d, or IOMMU. If left enabled, unpredictable behavior such as application failures may be experienced. Refer to the NVIDIA documentation, PCIe Access Control Services (ACS), for how to disable these services.

3.0 Linux Settings

Intel recommends the following settings to enable consistent performance measurements on the Linux distributions supported with Intel® Ethernet Fabric Suite.

3.1 CPU Frequency Scaling Drivers

Methods for power saving on CPUs can adversely impact performance. 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.

The default scaling driver in our supported OSes is the Intel® P-State (`intel_pstate`) driver. 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. This section describes how to use each driver for consistent, best-effort performance measurements. Setting a 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/supercomputer, 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.

3.1.1 Using the Intel P-State Driver

The default scaling driver, for our supported OSes, is the Intel P-State (intel_pstate) driver. No additional setup is required. A detailed description of the design and features available with Intel P-State drivers is available here: https://www.kernel.org/doc/html/v4.18/admin-guide/pm/intel_pstate.html. Detailed explanation of these features is beyond the scope of this document. In general, no customization beyond the default is required for the best fabric performance, other than ensuring that the turbo frequencies are enabled and the performance governor is enabled.

The following settings are sysfs entries that can be controlled by the system administrator in real time, and a reboot is not required in order to take effect. However, due to the nature of Intel P-State, it is not always straight-forward to monitor the core frequencies and confirm your settings are in effect. For example, a command such as `grep MHz /proc/cpuinfo` will return a wide range of clock frequencies at any given time, unlike ACPI, which would return a consistent value in a format like "2X00000" or "2X01000" if Turbo mode is enabled. Intel recommends confirming and monitoring the clock frequencies using a kernel tool such as `turbostat`.

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**
- `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 you are running and other frequency information, use the command:

  `cpupower frequency-info`

If you have previously disabled the P-state driver, you must re-enable it *before* applying the tunings listed above. 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
```

   NOTE

   The code example above is for Red Hat. Other OSes may require a different method for modifying grub boot parameters.

3. Reboot.

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

3.1.2 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
elif [ -e /boot/grub2/grub.cfg ]; then
GRUB_CFG=/boot/grub2/grub.cfg
fi
grub2-mkconfig -o $GRUB_CFG
```

   NOTE

   The code example above is for Red Hat. Other OSes may require a different method for modifying grub boot parameters.

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 during benchmarking, you may want to pin the CPU clock frequency to a specific value and 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 server chassis temperature will most likely rise if the above scheme is used.

To get the maximum advantage from Intel® Turbo Boost Technology:

1. Ensure that Turbo mode is set to Enabled in the BIOS (as recommended in BIOS and Platform Settings).
2. Set the frequencies appending "01" to the clock rate. This will enable Intel® Turbo Boost Technology.

For example, if running on an Intel® Xeon® Scalable 8360Y Processor (nominal 2.4 GHz clock rate), then the corresponding command option would be:

```
sudo cpupower -c all frequency-set --min 2.401GHz --max 2.401GHz \
-g Performance
```

3.2 Priority Flow Control

Enabling priority flow control (PFC) and confirming it is in use is an important aspect of performance tuning. For small node counts and point-to-point microbenchmarks, it is not essential to have PFC enabled for acceptable performance. However, collective communications or using high core counts per node, or HPC applications running on roughly eight nodes or more, depend heavily on PFC for the best possible performance and lowest run to run variation.

Refer to Priority Flow Control Configuration and Tuning that covers various examples for how to configure Ethernet NICs and switches for PFC.

3.3 IRQ Affinity and irqbalance

The purpose of irqbalance is to distribute hardware interrupts across cores on a multi-core system in order to increase performance. Intel has not identified a crucial role in manually setting IRQ affinities for the ice driver in order to obtain good performance with MPI applications using PSM3/RoCEv2. However, if you want to adjust the IRQ affinity, follow the guidance provided by Intel with the *ice Linux Base Driver for the Intel(R) Ethernet Controller 800 Series* driver package.

To stop irqbalance, execute:

```
systemctl stop irqbalance
```

To ensure irqbalance is disabled on boot:

```
systemctl disable irqbalance
```

The next step is to run the `set_irq_affinity` script, as outlined in the ice driver readme file.

Some MPI applications running over TCP may benefit from lowering the number of Tx/Rx queues. See Intel® Ethernet Driver (ice) Settings for details.

3.4 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**.
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
```

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 enabled 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.

3.5 irdma Module Settings

The irdma (Intel RDMA) module is used to communicate using the RoCE protocol over compatible Intel NICs. The module parameter `roce_ena=1` must be set in order to globally set all ports on the NIC to run in RoCE mode. If the cluster was installed using the Intel® Ethernet Fabric Suite FastFabric installation process, this module parameter should be set automatically. To check the value:

```
:> cat /sys/module/irdma/parameters/roce_ena
1
```

If the value is `0`, you must change it to 1. Perform the following to make the change persistent on reboots:

```
:> echo 'options irdma roce_ena=1' >> /etc/modprobe.d/irdma.conf
:> dracut -f
:> reboot
```

Setting all ports globally with `roce_ena=1` is sufficient for most HPC use cases. In the case where you want to only set a specific port to use RoCE, use the parameter `roce_port_cfg` as described in *README_irdma.txt* that is contained within the irdma software release.

Note that there may already be other contents in the `irdma.conf` file. You will append the file with the previous setting. Also, you may want to check the contents of the file to be sure it fits the needs of your system.

4K MTU

For the highest possible bandwidth, ensure the MTU for the device is 4 KB, for example:

```
:> ibv_devinfo -v -d <devname> | grep active_mtu
                        active_mtu:             4096 (5)
```

This requires that the corresponding network interface is using an MTU of at least 100 bytes larger (which includes IP and RoCE headers, maybe VLAN headers), but is typically set to 9K (jumbo). For example, in `/etc/sysconfig/network-scripts/ifcfg-<interface>`, insert the line:

```
MTU=9000
```

Increasing irdma Queue Pair (QP) Limit

There may be some applications that fail with the following message (or similar):

```
libirdma-irdma_vmapped_qp: failed to create QP, status 75
libirdma-irdma_ucreate_qp: failed to map QP
node1.314829Ran out of memory (err=4)
node1.314851Process connect/disconnect error: 4, opcode 206
```

The default number of QPs for irdma is 4096. If this occurs, you can add the additional module parameter to irdma.conf:

```
:> echo 'options irdma limits_sel=5' >> /etc/modprobe.d/irdma.conf
```

Again, `dracut -f` and `reboot` is required to make the changes persistent. This new value allows for 65,532 QPs.

3.6 Intel® Ethernet Driver (ice) Settings

Some MPI applications running over TCP may benefit from lowering the number of Tx/Rx queues. The default number of queues enabled for each Ethernet port by the driver at initialization is equal to the total number of cores, including hyper threads. In platforms with high core count CPUs, this configuration can cause resource contention. In practice, Intel has found that reducing the number of Tx/Rx queues down to eight has resulted in improved performance for applications such as the Weather Research and Forecasting Model (WRF).

```
ethtool -L <interface> combined 8
```

where *<interface>* is the name of the Ethernet NIC in use. In order to make these settings persistent, you may wish to use the NetworkManager utility. This script would make the settings described above persistent on boot:

```
:> cat /etc/NetworkManager/dispatcher.d/20-ethtool
#!/bin/bash
if [ "$1" = "<interface>" ] && [ "$2" = "up" ]; then
  /sbin/ethtool -L <interface> combined 8
fi
```

Make sure 755 permissions are assigned to the file for it to be properly executed at boot time.

NOTE

Intel recommends that you unload and reload irdma after the system boots, or perform this change to the ice driver before loading irdma.

To confirm the settings took effect:

```
[root@node ~]# ethtool -l <interface>
...
Current hardware settings:
RX:             0
TX:             0
Other:          1
Combined:       8
```

3.7 TuneD Tuning Service

The TuneD tuning service has the potential to improve network latency and/or bandwidth depending on the profile selected, especially when running applications or benchmarks over TCP. It should be used carefully because certain profiles consume significantly more power and demand more cooling. A complete description of the TuneD service can be found here: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/monitoring_and_managing_system_status_and_performance/getting-started-with-tuned_monitoring-and-managing-system-status-and-performance.

Based on internal testing on sixteen 3rd Generation Intel® Xeon® Scalable Processors nodes connected with a 200 GB Ethernet network, the best point-to-point latency is achieved with the *latency-performance, hpc-compute*, and *network-latency* profiles.

These profiles deliver the lowest and most consistent point-to-point latency. The best application performance is achieved with the *throughput-performance* profile (which is the default). Conversely, although the *latency-performance* profile gave the lowest and most consistent point-to-point latency, it impacted some applications negatively by approximately 5%.

Although the TuneD tuning service has the potential to improve latency, its impact on application performance, power consumption, and the thermal state of the servers should be understood before deploying a more aggressive profile in production.

4.0 MPI Performance

MPI libraries are a key type of middleware for building HPC applications. The Intel® Ethernet Fabric Suite Software package includes a build of Open MPI, though many times it is recommended and preferred to use the latest Intel® MPI Library. This chapter shows how to use Open MPI and the Intel® MPI Library, followed by general performance tuning recommendations.

If the Open MPI package was installed, the exact version can be seen under `/usr/mpi/gcc` such as:

```
[/usr/mpi/gcc]$ ls -r *
openmpi-x.y.z-ofi
```

NOTE

x.y.z refers to the latest version of `openmpi`.

The `gcc` directory means that the GNU Compiler Collection (GCC) was used to build the MPI library.

For best performance, run MPIs using libfabric with the Performance Scaled Messaging 3 (PSM3) provider included with Intel® Ethernet Host Software.

Intel® Ethernet Fabric Suite FastFabric contains a collection of several common MPI applications and benchmarks. Refer to the *Intel® Ethernet Fabric Suite FastFabric User Guide* for building and running the included sample applications and benchmarks. Also included in the FastFabric suite are the *<MPI>*`.params` files. Each file includes required and optional arguments, as well as brief descriptions, for working with the related MPI and PSM3.

NOTE

<MPI> refers to the specific MPI (that is, `intelmpi.params` or `openmpi.params`).

4.1 MPI Benchmark Fundamentals

Two common benchmark applications, also included in FastFabric, are used to measure MPI performance: OSU Micro-Benchmarks (OMB) (https://mvapich.cse.ohio-state.edu/benchmarks/) and Intel® MPI Benchmarks (IMB) (https://github.com/intel/mpi-benchmarks). In general, the goal of these benchmarks is to measure point-to-point performance (latency, bandwidth, and message rate) between two nodes. Additionally, MPI collectives performance can be measured using a large group of nodes.

For simplicity, this section demonstrates how to run the Intel® MPI Benchmarks using FastFabric as packaged with Intel® EFS to measure latency, bandwidth, and message rate. These examples use the IMB-MPI1 benchmark. For more information, refer to the Intel® MPI Benchmarks User Guide.

NOTE

Examples are shown using the FastFabric `./run_*` scripts included in the Intel® Ethernet Fabric Suite.

To begin, follow instruction in the *Intel® Ethernet Fabric Suite FastFabric User Guide* to build and run sample MPI benchmarks.

NOTE

You must have password-less ssh enabled between all nodes where you want to run benchmarks and `mpi_hosts` file must be updated to include each node.

MPI Latency

MPI latency is measured between two nodes using one core (MPI rank) per node.

```
:> ./run_imb 2 Pingpong
...
#---------------------------------------------------
# Benchmarking PingPong
# #processes = 2
#---------------------------------------------------
#bytes #repetitions t[usec] Mbytes/sec
...
```

The resulting output in the third column (`t[usec]`) is latency as a function of message size. Typically, 8-byte latency is used for performance analysis. The analogous benchmark with OMB is called `osu_latency`. Sometimes, the MPI rank needs to be pinned to a certain CPU socket in order to achieve the best latency (see MPI Affinity). Note that the bandwidth returned is from a single buffer (mpi_send/recv) and does not fully stress the throughput capability of the network.

MPI Bandwidth

MPI bandwidth is measured between two nodes using one or more ranks per node. As you use more ranks per node, the aggregate bandwidth increases for lower message sizes. The `Uniband` and `Biband` benchmarks for MPI bandwidth measurements are used because they perform many simultaneous, non-blocking sends and are able to *stream* messages continuously and saturate the network.

The following is an example of one rank per node for uni-directional bandwidth:

```
:> ./run_imb 2 Uniband
...
#---------------------------------------------------
# Benchmarking Uniband
# #processes = 2
#---------------------------------------------------
#bytes #repetitions Mbytes/sec Msg/sec
...
```

The third column (`Mbytes/sec`) reports MPI bandwidth. The fourth column (`Msg/sec`) is message rate (discussed in the next section).

To run a bidirectional bandwidth test, replace `Uniband` with `Biband` in the previous example. The analogous benchmarks in OMB are `osu_bw` and `osu_bibw`.

The following example shows how to run both Uniband and Biband simultaneously, using four MPI ranks per node:

```
:> PROCS_PER_NODE=4 ./run_imb 8 Uniband Biband -npmin 8
```

NOTE

The `-npmin 8` flag is required to ensure that exactly four communicating pairs are running, the first four ranks on node1 and the second four ranks on node2.

NOTE

PROCS_PER_NODE var can also be added to the params file.

The analogous benchmark in OMB is `osu_mbw_mr`. There is no equivalent bidirectional benchmark.

MPI Message Rate

Message rate is also measured with Uniband and Biband benchmarks, but using as many ranks per node as there are cores on the node. The message rate is the total number of MPI messages (typically eight bytes) sent between the two nodes. This is a derived quantity that can be calculated from the bandwidth output.

The following is an example on nodes with 32 physical cores per node:

```
:> PROCS_PER_NODE=32 ./run_imb 64 Uniband -npmin 64
...
#---------------------------------------------------
# Benchmarking Uniband
# #processes = 64
#---------------------------------------------------
#bytes #repetitions Mbytes/sec Msg/sec
...
```

The fourth column (`Msg/sec`) is the message rate and is typically quoted for eight bytes. The same method can be used to measure bidirectional message rate with the Biband benchmark.

NOTE

If you are using OpenMPI within a SLURM job, depending on the environment you may need to properly specify the environment variable `SLURM_TASKS_PER_NODE`. For example, for a 2-node, 32-process per node job, it may be necessary to `export SLURM_TASKS_PER_NODE="32(x2)"`. This may also be handled by directives to `srun` or `sbatch`.

MPI Collectives

Performance can be measured for a variety of collectives such as Allreduce. These benchmarks can be run between many nodes. For example, a 128-node, 32-rank per node Allreduce can be run with the following command:

```
:> PROCS_PER_NODE=32 ./run_imb $((128*32)) Allreduce -npmin $((128*32))
...
#---------------------------------------------------------------- #
Benchmarking Allreduce # #processes = 4096
#----------------------------------------------------------------
#bytes #repetitions t_min[usec] t_max[usec] t_avg[usec]
...
```

Typically, `t_avg` latency gives you a good idea of the performance of the system. Sometimes, large deviations between `t_min` and `t_max` can indicate suboptimal performance and perhaps system jitter effects.

4.2 Intel® MPI Library Settings

NOTE

The information in this section assumes the use of Intel® MPI Library as recommended in the *Intel® Ethernet Fabric Suite Software Release Notes*.

For best performance, Intel recommends that you use the PSM3 libfabric (OFI) provider—a high-performance interface to the Intel® Ethernet Fabric. Note that as of Intel® MPI 2019, only OFI fabric is supported.

First, load the Intel® MPI library (and other Intel tools) into your environment:

```
:> source /opt/intel/oneapi/setvars.sh
```

If this was successful, you may check the version of MPI loaded in the environment:

```
:>  mpirun -version
Intel(R) MPI Library for Linux OS, Version 2021.3 Build 20210601 (id: 6f90181f1)
Copyright 2003-2021, Intel Corporation.
```

You may instead want to source the exact MPI library instead of the one bundled with oneAPI. The exact paths and location of the corresponding `mpivars.sh` or `env/vars.sh` vary from system to system. Set these two environment variables:

- `export I_MPI_FABRICS=shm:ofi` (preferred)

  or

  `export I_MPI_FABRICS=ofi` to not use Intel® MPI's shared-memory communications
- `export FI_PROVIDER=psm3`

For more details on available options, refer to the *Intel® MPI Library Developer Reference for Linux OS* found at https://software.intel.com/en-us/mpi-developer-reference-linux, specifically the section titled "*Environment Variables for Fabrics Control*". To ensure that the Intel® MPI fabric or provider is what you expect (especially that PSM3 is the provider for OFI), use `-genv I_MPI_DEBUG=5` option to view the debug output. You should see output such as:

```
[0] MPI startup(): libfabric version: 1.12.1-impi
[0] MPI startup(): libfabric provider: psm3
```

4.3 PSM3 Environment Variables

Certain non-default settings for PSM3 environment variables may improve HPC applications or microbenchmark performance. The following tunings have been tested on Intel® Xeon® Scalable Processors with positive results. See the *Intel® Ethernet Fabric Suite Host Software User Guide* for additional details of the PSM3 environment variables.

NOTE

It is possible that adjusting these variables for other workloads not shown in the following table may also help improve performance

NOTE

Do not enable every setting and expect to improve performance for an arbitrary application. In most cases, the optimal performance is achieved with the default settings.

It is possible that adjusting these variables for other workloads not listed may also help improve performance.

Table 3. PSM3 RoCEv2 (Verbs) Performance Tunings

Application/Benchmark/Metric	Tuning Parameters
HPCC PTRANS	`PSM3_FLOW_CREDITS=16` significantly increases PTRANS GB/s performance (measured at 16 nodes, 52 cores per node).
HPCC MPIFFT	`PSM3_FLOW_CREDITS=16` significantly increases MPIFFT GFlops performance (measured at 16 nodes, 52 cores per node). `PSM3_RDMA=0` (currently the default) performs better than RDMA mode 1.
QCD, Alltoall collectives, and other bandwidth dependent applications	`PSM3_ERRCHK_TIMEOUT=20:640:2`, default is `160:640:2` (min:max:factor) in milliseconds. Decreasing the first number increases the rate of PSM3 retries in the case of packet drops.
122.tachyon (Graphics, parallel ray tracing - Spec MPI 2007, medium suite)	`PSM3_RCVTHREAD_FREQ=600:1000:1`, increasing frequency of receive thread polling (default is `PSM3_RCVTHREAD_FREQ=10:100:1`).
MPI uni and bi-directional bandwidth, `PSM3_RDMA=1` mode, greater than 128 KB	`PSM3_RV_QP_PER_CONN=4` (default). Uses multiple queue pairs per MPI process.

continued...

Application/Benchmark/Metric	Tuning Parameters
	`PSM3_RV_MR_CACHE_SIZE=1024` increases the maximum amount of CPU memory to be pinned per process by the rendezvous module's CPU MR cache. In units of megabytes. Prevents major bandwidth drops for very large message sizes such as greater than 128 MB.
MPI uni and bi-directional bandwidth, `PSM3_RDMA=1` mode, 8192-32768 bytes	`PSM3_MQ_RNDV_NIC_THRESH=8000` switches to rendezvous protocol at smaller message sizes (default is `PSM3_MQ_RNDV_NIC_THRESH=64000`).
MPI uni-directional bandwidth	`PSM3_QP_PER_NIC=2 (or 4)` increases uni-directional streaming bandwidth (measured with IMB-MPI1 Uniband or osu_mbw_mr). No impact on bi-directional bandwidth.
MPI latency	`PSM3_RDMA=3` decreases single core latency relative to other modes.

In order to set these environment variables with the Intel® MPI Library, you can export them in your environment (e.g. `export PSM3_RDMA=1`), or you can pass them as mpirun command arguments (for example, `mpirun ... -genv PSM3_RDMA=1 ...`). In order to set these environment variables with Open MPI, you must pass them as mpirun command arguments (for example, `mpirun ... -x PSM3_RDMA=1 ...`)

You can confirm the environment variables took effect, or determine what the existing settings are, by setting `PSM3_VERBOSE_ENV=2:`. By including the colon at the end will output only for rank number 0 and prevent a lot of repeated output.

PSM3_RDMA Modes

PSM3 supports multiple RoCE data movement modes that are configurable with the environment variable `PSM3_RDMA`. For more details, refer to the *Intel® Ethernet Fabric Suite Host Software User Guide*. In general , `PSM3_RDMA=1` mode provides the best single thread bandwidth performance, and `PSM3_RDMA=3` provides the lowest latency. Most applications perform the best with either mode 0 or 1.

Packet Loss/Drops

If PFC is configured and tuned properly, there should be minimal to no packet losses or drops. You can use PSM3 profiling (`PSM3_PRINT_STATS`) to determine if PSM3 is experiencing drops and re-transmitting messages, which is bad for performance. If you see non-zero entries for `err_chk_send` or `err_chk_recv`, this means that the network is experiencing losses from which PSM3 has to recover. Check the PFC configuration. Also try to pace PSM3 by reducing `PSM3_FLOW_CREDITS` to alleviate the drops and possibly improve performance.

PSM3_PRINT_STATS

The profiling tool, `PSM3_PRINT_STATS`, can be used to debug low performance problems. Set to -1 to output statistics after the end of the run, or to a positive integer value (in seconds) to print statistics at every interval in seconds. Note that a file is generated per PSM3 process.

For delicate comparisons between runs, you may want to enable `PSM3_PRINT_STATSMASK=0xffffff` in order to print all values, including zero values, so side-by-side comparisons of output are formatted similarly.

4.4 MPI Collective and Intel® oneCCL Tunings

Intel recommends using the latest Intel® MPI Library when possible for optimized MPI collectives performance. The following table is a collection of additional tuning recommendations. It is possible that the tunings apply to other versions of Intel® MPI Library, but the version where it was discovered is listed for completeness.

Application/ Collective	MPI	Tuning	Notes
NAS Parallel Benchmarks, FT kernel	Intel® MPI Library 2019 Update 9	`-genv I_MPI_ADJUST_ALLTOALL=3`	Significantly improves NAS Parallel Benchmarks performance for FT kernel (class C, eight nodes, 52 processes per node).
MPI Alltoall	Intel® MPI Library 2019 Update 9	`-genv I_MPI_ADJUST_ALLTOALL=4` for 16 nodes, 52 ppn, 512 Bytes-1 KB, 8 KB-512 KB	Other algorithms may help other node counts, ppn, and message sizes.
MPI Alltoall	Intel® MPI Library 2021 Update 5	`-genv I_MPI_ADJUST_ALLTOALL=2` for 32 nodes, 1 ppn	Use in conjunction with . `-genv PSM3_RDMA=1`.
MPI_Allreduce	Intel® MPI Library 2021 Update 2	`-genv I_MPI_ADJUST_ALLREDUCE="4:0-1048575;2:1048576-104857600"` for 16-32 nodes, 52 ppn	Use in conjunction with `PSM3_RDMA=1`. Algorithm 4 (Topology aware Reduce and Bcast) is used for message sizes below 1 MB, and algorithm 2 (Rabenseifner's) is used for message sizes >=1 MB.

Note that Intel® MPI Library is specifically tuned for PSM3 running with RDMA mode 0. When running in a mode other than 0 (such as `PSM3_RDMA=1`), it is possible that other MPI collective algorithms may provide improved performance. See https://www.intel.com/content/www/us/en/develop/documentation/mpi-developer-reference-linux/top/environment-variable-reference/i-mpi-adjust-family-environment-variables.html for a summary of the available variables.

The autotuner feature of Intel® MPI Library can be used to re-tune an application and replace the existing default tunings. See https://software.intel.com/content/www/us/en/develop/documentation/mpi-developer-reference-windows/top/environment-variable-reference/tuning-environment-variables/autotuning.html for details. This may be beneficial after making a change such as switching from `PSM3_RDMA=0` (default) to `PSM3_RDMA=1`.

Intel® oneCCL

Intel® oneAPI is a set of APIs and tools that provide a multi-vendor cohesive environment for developing and executing high performance applications on CPUs as well as various accelerated processing elements such as GPUs. Within oneAPI, the Intel® oneAPI Collectives Communications Library (oneCCL) is a scalable and high-performance communication library for Deep Learning (DL) and Machine Learning (ML) workloads.

oneCCL is not included in the Intel® Ethernet Fabric Suite software, but is available separately. Go to https://www.intel.com/content/www/us/en/developer/tools/oneapi/oneccl.html for more information on oneCCL and oneAPI.

oneCCL provides efficient collective communications that are heavily used in DL and ML. It also allows the user to easily trade off compute for communication performance to drive scalability of communication patterns. The following table outlines helpful tunings for applications using oneCCL.

Worker Threads

oneCCL provides environment variables to control worker thread configuration. With these variables, you can trade off the CPU resources used for computing and communication. For example, you can run Deep Learning Recommendation Model (DLRM) with hyperthreading enabled, one MPI process per socket, and dedicate the first four hyperthreaded CPU cores for communication, and all of the physical cores for computing. To do this, you set the environment variables as shown in the following table.

Environment Variable	Tuning Impact
I_MPI_PIN_DOMAIN	Define domains (non-overlapping subnet of logic processes) on a node, and the rules for how MPI processes are bound to the domains. MPI process threads can freely migrate from one logic processor to another within its corresponding domain. For example, for nodes with two CPU sockets, you can define two domains as shown below, and then have one MPI process per socket with all the threads on separate physical CPU cores: `I_MPI_PIN_DOMAIN=[0xffffffffffffff,0xffffffffffffff00000000000000,]` See https://www.intel.com/content/www/us/en/develop/documentation/mpi-developer-reference-linux/top/environment-variable-reference/process-pinning/interoperability-with-openmp-api.html for more information.
OMP_NUM_THREADS	Set the maximum number of OpenMP threads that can be used for a parallel region. For example, to allow an OpenMP application to use all cpu cores, you can set `OMP_NUM_THREADS` to the number of physical cores per cpu socket. See https://www.openmp.org/spec-html/5.0/openmpse50.html for more information.
CCL_WORKER_COUNT	Specify the number of oneCCL worker threads. For example, to use four CPU cores for communication per process, you can set the following to have four CCL workers. `CCL_WORKER_COUNT=4` See https://oneapi-src.github.io/oneCCL/env-variables.html#ccl-worker-count for more information.
CCL_WORKER_AFFINITY	Specify CPU affinity for oneCCL worker threads. For example, on a system with 112 physical CPU cores, you can use the following setting to have the first four hyperthreaded CPU cores dedicated for communication. `CCL_WORKER_AFFINITY='112,113,114,115,168,169,170,171'` See https://oneapi-src.github.io/oneCCL/env-variables.html#ccl-worker-affinity for more information.

Multi-NIC

oneCCL supports multi-NIC environments and is controlled primarily by three environment variables: `CCL_MNIC`, `CCL_MNIC_NAME`, and `CCL_MNIC_COUNT`.

- Setting `CCL_MNIC` to *local* can improve data communication performance because it will select NICs local to the NUMA node where the worker thread is running.
- Specifying `CCL_MNIC_COUNT` to use multiple NICs for communication can significantly improve performance as well.

Refer to the oneCCL documentation for more information: https://oneapi-src.github.io/oneCCL/env-variables.html#multi-nic.

NOTE

This methodology is used in place of the traditional `PSM3_MULTIRAIL` method of striping data across multiple NICs.

You should carefully study the difference in performance between the two methods (using oneCCL MNIC capability or PSM3 multi-rail capability) before choosing one over the other. In many cases, especially for AI workloads with larger messages, allowing oneCCL to do the multi-NIC load balancing can provide performance advantages due to a higher PSM3 MR cache hit rates since each NIC will repeatedly see the same slice of each IO buffer. In contrast, when `PSM3_MULTIRAIL` is used, the IO buffers will be load balanced across the specified NICs and each NICs MR cache may need to be larger so that any NIC may see any slice of a given IO buffer.

Collective Algorithms

For some oneCCL operations, specifying a collective algorithm may provide better performance. For example, you can control the *Allreduce* algorithm used by setting `CCL_ALLREDUCE`. See https://oneapi-src.github.io/oneCCL/env-variables.html#collective-algorithms-selection for details. In general, oneCCL attempts to use sensible defaults. In testing at 32 nodes with one process per node and eight worker threads, the *ring_rma* algorithm was found to perform the best for message sizes greater than 8 MB. For intermediate message sizes, the *Rabenseifner* algorithm was found to perform the best. The optimal algorithm will vary based on node count, message sizes, and other factors.

PSM3 Tunings

For AI training and other applications with large message Allreduce collectives, the performance may be improved by enabling `PSM3_RDMA=1`. At a scale of 32 nodes with one process per node and eight worker threads, performance may be further improved by setting `PSM3_MQ_RNDV_NIC_WINDOW=524288` (the default is 131072, or 128 KB).

4.5 MPI Affinity

The choice of the NIC with respect to the location of the MPI process has a measurable impact on performance. For example, latency-sensitive applications that use a NIC on a remote NUMA node will incur a performance cost related to memory/cache locality of the MPI process and an additional delay related to inter-NUMA interconnect traffic.

To determine which NUMA node a NIC is connected to, for example, an Intel® Ethernet 800 Series PCIe Adapter:

```
:> lspci | grep 810
18:00.0 Ethernet controller: Intel Corporation Ethernet Controller E810-C for
QSFP (rev 02)
18:00.1 Ethernet controller: Intel Corporation Ethernet Controller E810-C for
QSFP (rev 02)
```

NOTE

The first column is the slot number.

To find the NUMA node the slot is connected to:

```
:> lspci -v -s 18:00.0 | grep NUMA
        Flags: bus master, fast devsel, latency 0, IRQ 39, NUMA node 0
```

From the previous output, you can see the adapter is connected to NUMA node 0. This corresponds to the first 26 CPU cores as seen in:

```
:> lscpu | grep NUMA
NUMA node(s):         2
NUMA node0 CPU(s):    0-25,52-77
NUMA node1 CPU(s):    26-51,78-103
```

To minimize the cross-NUMA latency penalties described previously, you must make sure that the MPI process is pinned to any of the CPU cores 0-25. Typically, this is the default behavior of an MPI library. In the case that the NIC is connected to NUMA node 1, instruct the MPI library to pin the rank to any of CPU cores 26-51. For example, with Open MPI:

```
:> mpirun ... taskset -c 26 ./osu_latency
```

Here the utility affinitizes the MPI process to the first core on socket 1, and a lower latency will result than if pinned to any of cores 0-25 (the default). With the Intel® MPI Library, use the built-in environment variable `-genv I_MPI_PIN_PROCESSOR_LIST=26`. This environment variable can take a list or range of cores for multi-PPN tests. For more details, see the Intel® MPI Library documentation.

4.6 Dual/Multi-Rail

On systems with more than one NIC per node, PSM3 can use a feature known as multi-rail in order to use multiple NICs and increase the available bandwidth to the node. For bandwidth-hungry applications, multi-rail configurations may offer improved performance. See the *Intel® Ethernet Fabric Suite Host Software User Guide* for more details on configuring and using multi-rail. Note that on systems with more than one active NIC, each PSM3 process will use the NUMA-local NIC for communication. You do not have to explicitly set `PSM3_MULTIRAIL` in order to use all NICs, as long as there is at least one PSM3 process on the same NUMA node.

4.7 TCP Performance

While most HPC applications use RDMA-capable networks and the RoCE implementation of PSM3, it is also possible to run PSM3 using standard TCP. This section outlines helpful tunings for running PSM3 with TCP. Consult the *Intel® Ethernet Fabric Suite Host Software User Guide* for details on how to run PSM3 over TCP.

In the 11.2 software release (and later), you must specify at a minimum `PSM3_HAL=sockets`. By default, PSM3 will select the fastest NIC in the system. However, you may want to specify exactly which NIC to use with `PSM3_NIC=<interface>`, where *interface* is the name of the net device (not RDMA device).

PSM3 Environment Variable	Tuning Impact
PSM3_TCP_SKIPPOLL_COUNT	Default is "20:10". Some applications benefit from disabling this feature (set to 0:0) such as tachyon, socorro, and dmilc from the SPEC MPI 2007 application suite.
PSM3_MTU	Default is 65536. Higher values may increase bandwidth for single-rank performance, but lower values (such as PSM3_MTU=16384) have been shown to improve performance for some applications (for example, LAMMPS, rhodopsin protein benchmark).
FI_PSM3_LAZY_CONN	Default is 0 (off). Set to 1 to increase performance of some applications. This only establishes connections between endpoints when first used for communications, instead of establishing connections between all endpoints at job start.

5.0 Performance Tuning for Intel GPU

The focus of this chapter is on network performance tuning when running with Intel GPUs.

5.1 oneAPI Level Zero and GPUDirect

As with the non-GPU enabled PSM3, the oneAPI Level Zero-enabled PSM3 is also tuned to deliver optimized out-of-the-box performance for most workloads. There are certain PSM3 thresholds that are user-configurable and may deliver improved performance depending on the workload. See the *Intel® Ethernet Fabric Suite Host Software User Guide* for detailed explanation on each environment variable.

For example:

- PSM3_RV_GPU_CACHE_SIZE and PSM3_RV_MR_CACHE_SIZE: Increasing to 4096 may improve MR cache hit rates and application performance.

When running Intel GPU applications, it may be necessary to increase the maximum allowed open files through the Linux `ulimit -n limit` command or by editing `/etc/security/limits.conf` and increasing the `nofile` limit. Depending on the applications and hardware configuration, values as large as 524288 may be required.

When running an Intel GPU application, the following additional environment variable settings are recommended:

```
EnableImplicitScaling=0
NEOReadDebugKeys=1
```

When running an Intel GPU application with Intel MPI, the following additional environment variable settings are recommended:

```
I_MPI_OFFLOAD=1
I_MPI_OFFLOAD_PIPELINE=0
I_MPI_OFFLOAD_RDMA=1
I_MPI_OFFLOAD_RDMA_THRESHOLD=0
```

NOTE

In order to take advantage of the performance improvement from Direct GPU Access, you will need to set the parameter `prelim_override_p2p_dist` to 1 for the `i915` Graphics Driver. See *Intel® Ethernet Fabric Suite Software Installation Guide* for more information.

6.0 Performance Tuning for NVIDIA GPU

The focus of this chapter is on network performance tuning when running with NVIDIA GPUs.

6.1 CUDA and GPUDirect

As with the non-CUDA enabled PSM3, the CUDA-enabled PSM3 is also tuned to deliver optimized out-of-the-box performance for most workloads. There are certain PSM3 thresholds that are user-configurable and may deliver improved performance depending on the workload. See the *Intel® Ethernet Fabric Suite Host Software User Guide* for detailed explanation on each environment variable.

For example:

- PSM3_MQ_RNDV_NIC_WINDOW (default 2097152): Decreasing to 65536 significantly improves large message ping-pong latency (osu_latency) but has a negative effect on streaming bandwidth (osu_bw/osu_bibw).
- PSM3_CUDA_THRESH_RNDV (default 8000): Defines the message size, in bytes, when the protocol switches from eager to rendezvous. Messages below the threshold use eager, and at or above the threshold use rendezvous.
  - For the verbs HAL (PSM3_HAL=verbs), larger values decrease latency in 8 KB-32 KB message size range, but may have a negative impact on bandwidth
  - For the sockets HAL (PSM3_HAL=sockets), increased bandwidth is seen in the 8 KB-256 KB message size range when increasing this to a very large value such as 2147483648 (or just above whatever the largest message size in the application is). This effectively disables rendezvous and uses eager for all the messages sizes.
- PSM3_GPUDIRECT_RDMA_SEND_LIMIT: As of the IEFS 11.3 release, this limit is set to UINT_MAX such that PSM3 always uses GPU Direct for RDMA sends. On systems with PCIe switches, this gives the best performance. On systems without PCIe switches, you may see a slight benefit to bandwidth by reducing this environment variable down to the previous default, or approximately `PSM3_GPUDIRECT_RDMA_SEND_LIMIT=30000` (bytes).

These observations are made when running the CUDA-enabled osu_latency, osu_bw, and osu_bibw within the OSU Microbenchmarks suite.

NVIDIA Multi-Process Service (MPS)

For the majority of use cases, MPI applications using GPUs assign one MPI rank per GPU. All of the communication for the GPU is handled through the single MPI rank. In the case where it is desired to use multiple MPI ranks per GPU, significantly lower performance may be seen. In this case, it may be beneficial to deploy NVIDIA Multi-Process Service (MPS). See https://docs.nvidia.com/deploy/mps/index.html for more details.

6.2 NVIDIA Collectives Communication Library (NCCL)

The *Intel® Ethernet Fabric Suite Host Software User Guide* describes how to set up and run PSM3 with NCCL. This section highlight some of the NCCL and PSM3 environment variables that impact performance when running with NCCL.

When running PSM3 and NCCL, performance benefits have been seen when reducing the size of the buffer used by NCCL by setting `NCCL_BUFFSIZE=262144`. The default buffer size is 4194304 (4 MB). See https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/env.html#environment-variables for more details on the available NCCL environment variables.

7.0 Priority Flow Control Configuration and Tuning

Priority Flow Control (PFC) allows Ethernet to be configured as a lossless network. Lossless behavior is a prerequisite of RoCEv2 and essential to getting good performance with RoCEv2 for both PSM3 and storage uses.

This chapter will provide guidelines and examples on how to configure PFC on various Ethernet NICs and switches. Consult your NIC and switch vendor documentation for all details regarding enabling PFC. Note that these examples are aimed at providing lossless network traffic for one traffic class.

7.1 NIC Configuration for PFC

NOTE

A detailed discussion of PFC is beyond the scope of this document.

In general, the Ethernet NICs are configured to use software Data Center Bridging (DCB) in willing mode. Then, the switches are configured for DCB (priority settings, traffic classes, bandwidth allocations, headroom, and so on) on the switch ports. A detailed explanation of these implementations can be found in the Intel® Ethernet 800 Series Linux Flow Control Configuration Guide for RDMA Use Cases. Note that when non-willing mode is used to explicitly configure PFC (such as for a back-to-back test configuration without a switch), additional steps must be taken to ensure the recipe is persistent across node reboots.

The *Intel® Ethernet Fabric Suite Software Installation Guide* discusses how to use the FastFabric installation procedure to automate setting up the NICs for PFC in willing mode. Following is a step-by-step recipe that is known to work if you wish to customize or understand more fully what settings are being implemented. In the examples, replace <*interface*> with the name of the RDMA device's corresponding network interface name.

Enable Willing Mode on Intel Ethernet NICs

When the NICs are configured in Willing mode and connected to a switch with DCB configured, the NICs will automatically apply the same DCB configuration. This example shows how to enable software willing mode on a CVL NIC. Consult your switch manual for DCB configuration steps, or the next section for an example.

1. Disable Link-level Flow Control (LFC).

```
:>ethtool -A <interface> rx off tx off
```

2. Verify that LFC is disabled.

```
:> ethtool -a <interface>
Pause parameters for <interface>:
Autonegotiate:  on
RX:             off
TX:             off
RX negotiated:  off
TX negotiated:  off
```

3. Disable Firmware DCB.

```
:>ethtool --set-priv-flags <interface> fw-lldp-agent off
```

4. Verify that Firmware DCB is disabled.

```
:>ethtool --show-priv-flags <interface> | grep fw-lldp-agent
fw-lldp-agent                : off
```

5. Install OpenLLDP if not already installed.
   - For RHEL system:

     ```
     yum install lldpad
     ```

   - For SLES system:

     ```
     zypper install open-lldp
     ```

   - For Ubuntu system:

     ```
     apt-get install lldpad
     ```

6. Start the Open LLDP daemon.

```
lldpad -d
```

7. Disable CEE transmission.

```
lldptool -Ti <interface> -V CEE-DCBX enableTx=no
```

8. Reset the DCBX mode to be *auto*.

```
lldptool -Ti <interface> -V IEEE-DCBX mode=reset
```

9. Configure willing configuration.

```
lldptool -Ti <interface> -V ETS-CFG enableTx=yes willing=yes
```

10. Configure willing recommendation.

```
lldptool -Ti <interface> -V ETS-REC enableTx=yes
```

11. Configure willing PFC.

```
lldptool -Ti <interface> -V PFC willing=yes enableTx=yes
```

12. Terminate the first instance of lldpad.

```
lldpad -k
```

13. Remove lldpad state records from shared memory.

```
lldpad -s
```

14. Restart service lldpad.

```
systemctl restart lldpad
```

15. Verify CEE mode is disabled.

```
lldptool -ti <interface> -V CEE-DCBX -c
enableTx=no
```

16. Verify DCBX mode is *auto*.

```
lldptool -ti <interface> -V IEEE-DCBX -c
mode=auto
```

7.2 Switch Configurations for PFC

This section provides examples for how to configure and tune various Ethernet switches for the best PFC performance.

Configuring PFC on Arista 7060 Switches

The following recipe maps all traffic to priority 0 on the Arista DCS-7060CX-32S-R switch running EOS 4.24.1.1F, which has been shown to reliably enable PFC.

1. Open the switch configuration terminal.

```
myswitch>enable
myswitch#config terminal
```

2. Configure the QoS policy.

```
myswitch(config)#sh run sec test
myswitch(config)#ip access-list test
myswitch(config-acl-test)#counters per-entry
myswitch(config-acl-test)#10 permit ip any any
myswitch(config-acl-test)#sh run sec class
myswitch(config-acl-test)#class-map type qos match-any test
myswitch(config-cmap-qos-test)#match ip access-group test
myswitch(config-cmap-qos-test)#policy-map type quality-of-service test
myswitch(config-pmap-quality-of-service-test)#class test
myswitch(config-pmap-c-quality-of-service-test-test)#set cos 0
myswitch(config-pmap-c-quality-of-service-test-test)#set traffic-class 0
myswitch(config-pmap-c-quality-of-service-test-test)#class class-default
```

```
myswitch(config-pmap-c-quality-of-service-test-class-default)#exit
myswitch(config-pmap-quality-of-service-test)#exit
myswitch(config)#
```

3. Apply the policy on the input interface.

```
myswitch(config)#interface ethernet 1/1-32/1
myswitch(config-if-Et1/1,2/1,3/1,...,31/1,32/1)#service-policy type qos input
test
myswitch(config-if-Et1/1,2/1,3/1,...,31/1,32/1)#priority-flow-control
priority 0 no-drop
myswitch(config-if-Et1/1,2/1,3/1,...,31/1,32/1)#qos trust cos
myswitch(config-if-Et1/1,2/1,3/1,...,31/1,32/1)#exit
```

4. (Optional) Copy the running configuration to the startup configuration to persist a reboot.

```
myswitch(config)# copy running-config startup-config
```

Set Up PFC on Arista DCS-7170-32CD-F

On an Arista DCS-7170-32CD-F switch with Arista EOS software version 4.22.1FX-CLI, the following steps will set up PFC (see the complete Arista EOS documentation at https://www.arista.com/en/um-eos for more guidance).

1. Open the switch configuration terminal.

```
myswitch>enable
myswitch#config terminal
```

2. Configure PFC.

```
localhost(config)# interface ethernet 1/1-32/1
localhost(config-if-Et1/1,2/1,3/1,...,31/1,32/1)#priority-flow-control on
localhost(config-if-Et1/1,2/1,3/1,...,31/1,32/1)#dcbx mode ieee
localhost(config-if-Et1/1,2/1,3/1,...,31/1,32/1)#priority-flow-control
priority 0 no-drop
localhost(config-if-Et1/1,2/1,3/1,...,31/1,32/1)#flowcontrol send off
localhost(config-if-Et1/1,2/1,3/1,...,31/1,32/1)#flowcontrol receive off
localhost(config-if-Et1/1,2/1,3/1,...,31/1,32/1)#qos trust cos
```

3. Confirm that PFC is actually enabled on priority 0.

```
localhost(config-if-Et1/1,2/1,3/1,...,31/1,32/1)#show priority-flow-control
The hardware supports PFC on priorities 0 1 2 3 4 5 6
PFC receive processing is enabled on priorities 0 1 2 3 4 5 6 7
Global PFC : Enabled

E: PFC Enabled, D: PFC Disabled, A: PFC Active, W: PFC Watchdog Enabled
Port       Status  Priorities  Action      Timeout  Recovery
Polling         Note
                                                      Interval/Mode   Config/Oper
-------------------------------------------------------------------------------
---------
Et1/1      E A -   0           -           -        - / -           - / -
Et2/1      E A -   0           -           -        - / -           - / -
Et3/1      E A -   0           -           -        - / -           - / -
...
...
Port                RxPfc          TxPfc
Et1/1          2843498825        79913457
Et2/1          2203656611        73527802
```

```
Et3/1           2459768514        70122076
Et4/1           2508639684        74898208
...
```

NOTE

Only priority 0 is listed as "Enabled" and "Active". The Intel® Ethernet 800 Series PCIe Adapter will, by default, use priority 0 when running in RoCEv2 mode. Other than setting up the NICs as described in the previous section, no additional flags are necessary to target priority 0 when using Open MPI as packaged with Intel® EFS or the Intel® MPI Library.

Further PFC Tuning of the Arista 7170 Switch

Under highly loaded scenarios, the default headroom buffer sizes on the Arista 7170 switch with EOS are not large enough, and packet loss occurs even if PFC appears to be functioning correctly. When a receiving node sends a Tx pause packet, it takes time for that packet to traverse the network and reach its destination. Headroom buffers on the switch exist to absorb all packets that are already transmitting at the time the Tx pause is sent. The headroom tries to absorb all transmitting packets before the node receiving the pause packet stops sending. If the headroom is exceeded, then drops still occur and poor performance may result.

In order to tune the PFC headroom, the lower level registers must be adjusted. This can be accomplished using the following steps:

1. Enter the following command, which queries pipe 1 for the existing headroom limits:

```
localhost#enable
localhost#platform barefoot access rr dev_0 device_select tm_top tm_wac_top
wac_pipe[1] csr_mem_wac_ppg_hdr_lmt
```

NOTE

You can replace `wac_pipe[1]` with `wac_pipe[3]` to view the same limits for pipe 3.

The output will look similar to:

```
0 [0041a000] : 00000000 : hdr_lmt[0]
0 [0041a004] : 000000e8 : hdr_lmt[1]
0 [0041a008] : 00000000 : hdr_lmt[2]
0 [0041a00c] : 000000e8 : hdr_lmt[3]
0 [0041a010] : 00000000 : hdr_lmt[4]
0 [0041a014] : 000000e8 : hdr_lmt[5]
0 [0041a018] : 00000000 : hdr_lmt[6]
0 [0041a01c] : 000000e8 : hdr_lmt[7]
0 [0041a020] : 00000000 : hdr_lmt[8]
0 [0041a024] : 000000e8 : hdr_lmt[9]
0 [0041a028] : 00000000 : hdr_lmt[10]
0 [0041a02c] : 000000e8 : hdr_lmt[11]
0 [0041a030] : 00000000 : hdr_lmt[12]
0 [0041a034] : 000000e8 : hdr_lmt[13]
0 [0041a038] : 00000000 : hdr_lmt[14]
0 [0041a03c] : 000000e8 : hdr_lmt[15]
0 [0041a040] : 00000000 : hdr_lmt[16]
0 [0041a044] : 000000e8 : hdr_lmt[17]
0 [0041a048] : 00000000 : hdr_lmt[18]
```

```
0 [0041a04c] : 000000e8 : hdr_lmt[19]
0 [0041a050] : 00000000 : hdr_lmt[20]
0 [0041a054] : 000000e8 : hdr_lmt[21]
0 [0041a058] : 00000000 : hdr_lmt[22]
0 [0041a05c] : 000000e8 : hdr_lmt[23]
0 [0041a060] : 00000000 : hdr_lmt[24]
0 [0041a064] : 000000e8 : hdr_lmt[25]
0 [0041a068] : 00000000 : hdr_lmt[26]
0 [0041a06c] : 000000e8 : hdr_lmt[27]
0 [0041a070] : 00000000 : hdr_lmt[28]
0 [0041a074] : 000000e8 : hdr_lmt[29]
0 [0041a078] : 00000000 : hdr_lmt[30]
0 [0041a07c] : 000000e8 : hdr_lmt[31]
```

In this example, the non-zero entries exist in the non-default PPG ids 1, 3, 5, 7,...31 (odd numbered). The exact non-default PPGs may vary from switch reboot to switch reboot. The value in hex is 0xe8 cells (232) and each cell is 80 bytes. Therefore, 232x80=18,560 bytes for each headroom buffer. In practice, increasing this value up to 0xe80 (3712x80=296,960 bytes) is large enough to prevent drops in a cluster size of 32 nodes.

2. To set this parameter for the non-zero PPG IDs in the previous output, the following must be run on pipe 1 and 3:

```
platform barefoot access wr dev_0 device_select tm_top tm_wac_top wac_pipe[1]
csr_mem_wac_ppg_hdr_lmt  hdr_lmt[1] 00000e80

platform barefoot access wr dev_0 device_select tm_top tm_wac_top wac_pipe[3]
csr_mem_wac_ppg_hdr_lmt  hdr_lmt[1] 00000e80

platform barefoot access wr dev_0 device_select tm_top tm_wac_top wac_pipe[1]
csr_mem_wac_ppg_hdr_lmt  hdr_lmt[3] 00000e80

platform barefoot access wr dev_0 device_select tm_top tm_wac_top wac_pipe[3]
csr_mem_wac_ppg_hdr_lmt  hdr_lmt[3] 00000e80

platform barefoot access wr dev_0 device_select tm_top tm_wac_top wac_pipe[1]
csr_mem_wac_ppg_hdr_lmt  hdr_lmt[5] 00000e80

platform barefoot access wr dev_0 device_select tm_top tm_wac_top wac_pipe[3]
csr_mem_wac_ppg_hdr_lmt  hdr_lmt[5] 00000e80

...

platform barefoot access wr dev_0 device_select tm_top tm_wac_top wac_pipe[1]
csr_mem_wac_ppg_hdr_lmt  hdr_lmt[31] 00000e80

platform barefoot access wr dev_0 device_select tm_top tm_wac_top wac_pipe[3]
csr_mem_wac_ppg_hdr_lmt  hdr_lmt[31] 00000e80
```

3. Validate the setting took effect by re-running the command in step 1.

```
localhost#platform barefoot access rr dev_0 device_select tm_top tm_wac_top
wac_pipe[1] csr_mem_wac_ppg_hdr_lmt
0 [0040a000] : 00000000 : hdr_lmt[0]
0 [0040a004] : 00000e80 : hdr_lmt[1]
0 [0040a008] : 00000000 : hdr_lmt[2]
0 [0040a00c] : 00000e80 : hdr_lmt[3]
0 [0040a010] : 00000000 : hdr_lmt[4]
0 [0040a014] : 00000e80 : hdr_lmt[5]
0 [0040a018] : 00000000 : hdr_lmt[6]
...
```

Configuring PFC on SONiC OS

At this time, non-willing mode must be used with SONiC, so PFC must be configured explicitly on both the NIC and the switch.

To enable PFC explicitly on the NIC on TC0:

```
systemctl start lldpad
ethtool -A <interface> rx off tx off

ethtool --set-priv-flags ${interface} fw-lldp-agent off

lldptool -Ti ${interface} -V ETS-CFG willing=no
up2tc=0:0,1:0,2:0,3:0,4:0,5:0,6:0,7:0 \\
tsa=0:ets,1:strict,2:strict,3:strict,4:strict,5:strict,6:strict,7:strict
    \\
tcbw=100,0,0,0,0,0,0,0

lldptool -Ti ${iface} -V PFC willing=no enabled=0
```

Configuring PFC in the SONiC config_db.json file is beyond the scope of this document. However, the following buffer sizes (in bytes) have been shown to work well for 16-32 nodes connected to a single 64-port Edgecore Mavericks switch with SONiC version SONiC.202012.26143-dirty-20210729.160956:

```
    "BUFFER_PROFILE": {
        "ingress_lossless_profile": {
            "dynamic_th": "7",
            "pool": "[BUFFER_POOL|ingress_lossless_pool]",
            "size": "4096",
            "xoff": "100000",
            "xon": "18430"
        },
```

For more details on configuring SONiC, see the documentation found here: https://github.com/Azure/SONiC/wiki/Configuration

7.3 Verification for PFC

It is important to verify PFC is set up properly and takes effect. This section provides instructions for validating PFC.

Validating PFC with FastFabric Tools

In the Intel® Ethernet Fabric Suite FastFabric TUI, you can validate the functionality of the PFC configuration. See *Intel® Ethernet Fabric Suite FastFabric User Guide* for more details. This tool performs many-to-one incast traffic tests and detects packet loss. In the FastFabric TUI, select 2 (Host Verification/Admin), then 6 (Verify PFC via empirical test). This tool is more thorough than just verifying Tx/Rx pause counters, because an improperly *tuned* PFC can still be lossy (for example, if the headroom buffer is not large enough as described earlier in this chapter).

Validating PFC with ethtool Counters

The switch outputs in the previous example show that the switch is receiving and sending PFC pause frames. These should increase over time when an application is running and heavily loading the switch. In order to confirm that pause frames are also being sent and received by the hosts, you can use a `watch` command while the application is running:

```
:> watch -d -n1 "ethtool -S <interface> | grep priority_0"
Every 1.0s: ethtool -S <interface> | grep priority_0

     tx_priority_0_xon.nic: 8281348
     tx_priority_0_xoff.nic: 2835583234
     rx_priority_0_xon.nic: 39950042
     rx_priority_0_xoff.nic: 39963415
```

Notice that there are both Tx and Rx pause frames for both XON (requested on) and XOFF (requested off). You should see non-zero counters for all four. If you only see Tx or Rx, PFC is not fully enabled on the system. In addition to seeing these pause frames increasing, you should see zero LAN packet drops:

```
:> ethtool -S <interface> | grep drop
     rx_dropped: 0
     tx_dropped_link_down.nic: 0
     rx_dropped.nic: 0
```

You should also see zero RDMA discards reported by irdma:

```
:> grep . /sys/class/infiniband/<devname>/ports/1/hw_counters/*Discards

/sys/class/infiniband/<devname>/ports/1/hw_counters/ip4InDiscards:0
/sys/class/infiniband/<devname>/ports/1/hw_counters/ip6InDiscards:0
```