Intel® Ethernet Fabric

Performance Tuning Guide

Rev. 1.1

July 2021

Doc. No.: 632488, Rev.: 1.1

Revision History

Date	Revision	Description
July 2021	1.1	Product 11.1.0.0 release - Changes to this document include: • MPI collective tuning algorithms added to MPI Collective Tunings on page 30 • Added guidance to lower NIC Tx/Rx queues for improved application performance over TCP. IRQ affinity and irqbalance on page 19. • Updated description of the `roce_ena` parameter in irdma module settings on page 22. • Added section on CUDA and GPUDirect* on page 32 • Added discussion on using PSM3_PRINT_STATS to detect fabric drops in PSM3 Environment Variables on page 29 • Added details on Arista 7170 PFC tuning in Priority Flow Control on page 15
February 2021	1.0	Initial release.
December 2020	0.7	Initial (Beta) release.

Contents

Tables

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/design/products-and-solutions/networking-and-io/high-performance-networking/technical-library.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, aka libfabric) provider which implements an optimized user-level communications protocol. The audience for this document includes cluster administrators and those running or implementing Message-Passing Interface (MPI) programs.

continued...

Task	Document Title	Description
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 (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 8 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. 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® Ethernet Fabric Suite products is available 24 hours a day, 365 days a year. Please contact Intel Customer Support or visit https://www.intel.com/content/www/us/en/support/contact-support.html for additional detail.

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 on page 6 of this document. You may also consult the *Intel® Ethernet 800 Series Linux Performance Tuning Guide* for E810 Ethernet 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
GCC	GNU Compiler Collection
GUPS	Giga-Updates per Second
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
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
TCP	Transmission Control Protocol
THP	Transparent Huge Pages
UD	Unreliable Datagram
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.
- 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. In this chapter, we specify 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
Snoop Holdoff Count	9[4]

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.

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 RHEL* 8.x 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. 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.

3.1.1 Using the Intel® P-State Driver

The Intel® P-State Driver is the default driver for RHEL*8.x, so 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. We recommend 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`

It is possible to enforce a slower clock frequency for benchmarking or validation purposes with the Intel P-State frequency driver. To do this, first disable Turbo mode, then set `min_perf_pct` and `max_perf_pct` such that `[min/max]_perf_pct = ceiling(target_clock/base_clock*100)`. For example, if we want to enforce a clock frequency of 1.8 GHz on a processor with a 2.1 GHz base frequency, we would set `[min/max]_perf_pct = ceiling(1.8/2.1*100)= 86`.

- `echo 1 > /sys/devices/system/cpu/intel_pstate/no_turbo`
- `echo 86 > /sys/devices/system/cpu/intel_pstate/min_perf_pct`
- `echo 86 > /sys/devices/system/cpu/intel_pstate/max_perf_pct`

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, please 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 on page 11).
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® Gold 6148 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 one of the more challenging aspects 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 8 nodes or more, depend heavily on PFC for the best possible performance and lowest run to run variation.

Detailed discussion of PFC is beyond the scope of this document, but we aim to provide examples of how Intel has enabled PFC on test clusters. In general the Ethernet NICs are configured to use firmware Data Center Bridging (DCB), and willing mode. Then the switches are configured for DCB (priority settings, traffic classes, bandwidth allocations, etc.) on the switch ports. 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. Below 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 CVL NICs

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

Disable Link-level Flow Control (LFC)

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

Verify that LFC is disabled

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

Configure the NIC for firmware DCB mode (as opposed to software DCB mode)

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

Verify that firmware DCB is enabled

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

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 for `<interface>`:

```
:> cat /etc/NetworkManager/dispatcher.d/20-ethtool
#!/bin/bash
if [ "$1" = "<interface>" ] && [ "$2" = "up" ]; then
  /sbin/ethtool --set-priv-flags <interface> fw-lldp-agent on
  /sbin/ethtool -A <interface> rx off tx off
fi
```

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

Configuring Switches for PFC

Please consult your switch vendor documentation for all details regarding enabling PFC. On one Arista DCS-7170-32CD-F (Tofino silicon) switch with software version 4.22.1FX-CLI , the following steps were necessary at a minimum:

```
localhost>enable
localhost#config terminal
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
```

then 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". CVL will automatically use priority 0 when running in RoCEv2 mode. Other than setting up the NICs as described in the section above, no additional flags are necessary to target priority 0 when using Open MPI as packaged with Intel® EFS or the Intel® MPI Library

Validating PFC with ethtool counters

The switch output above shows 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 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 there are both tx and rx pause frames being xon (requested on) and xoff (requested off). This is essential. 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
```

and also zero RDMA discards reported by irdma:

```
:> grep . /sys/class/infiniband/<devname>/hw_counters/*Discards
/sys/class/infiniband/<devname>/hw_counters/ip4InDiscards:0
/sys/class/infiniband/<devname>/hw_counters/ip6InDiscards:0
```

Validating PFC with FastFabric tools

In the Intel Ethernet FastFabric Tools TUI, you can validate the functionality of the PFC configuration. This tool performs many-to-one incast traffic tests and detects packet loss. In the FastFabric TUI, select 2 (Host Verification/Admin), 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 headroom buffer is not large enough as described in the next section).

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 in flight at the time the Tx pause is sent. The headroom absorbs all in flight 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, lower level registers must be adjusted. Unfortunately, the EOS does not provide this capability so tuning the switch must be done manually. First log into the switch and enter enable mode:

```
localhost#enable
```

then enter the following command which queries pipe 1 for the existing headroom limits:

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

note that 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). Unfortunately, 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 - 232x80=18,560 bytes for each headroom buffer. In practice we have seen increasing this up to 0xe80 (3712*80=296,960 bytes) is large enough to prevent drops in a cluster size of 32 nodes. In order to set this parameter, the following must be run on pipe 1 and 3, for the PPG ids that are non-zero above:

```
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
```

You may then validate the setting took effect by re-running the first command above:

```
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]
...
```

3.3 IRQ affinity and irqbalance

The purpose of irqbalance is to distribute hardware interrupts across processors on a multiprocessor 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. However, if you want to adjust the IRQ affinity, Intel® EFS follows 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
```

or to ensure it 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. 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 8 has resulted in improved application performance, for example, 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. This is a similar methodology to making PFC settings persistent as shown in a previous section.

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 Slurm* Settings

The following settings have been adjusted on very large clusters (thousands of nodes) and shown to improve Slurm's tolerance to unexpected communication delays and other potentially disruptive scenarios such as temporary network outages. Values are in seconds:

- `Slurmctldtimeout=360` (default is 120)
- `SlurmdTimeout=360` (default is 300)
- `BatchStartTimeout=20` (default is 10)
- `CompleteWait=15` (default is 0)
- `MessageTimeout=60` (default is 10)
- `MinJobAge=600` (default is 300)
- `TCPTimeout=15` (default is 2)

Smaller clusters may operate fine with lower values closer to, or at, the default.

To adjust the settings above, perform the following:

1. Modify the values in `/etc/slurm.conf`.
2. Restart the `slurmd` and `slurmctld` processes for the changes to take effect.

   Refer to the slurm documentation for more information including definitions for each variable (https://slurm.schedmd.com/slurm.conf.html).

3.6 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 install process, this module parameter should be set automatically. To check the value,

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

if the value instead 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 by other contents in the `irdma.conf` file and the above will just append to them. 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 4KB, for example:

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

this requires the corresponding network interface is using an MTU of at least roughly 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 which 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 happens, you can add this 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.

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. In this chapter we discuss 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.

To run Open MPI with the PSM3 provider:

1. 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` or in your run script:

   ```
   :> source /usr/mpi/gcc/openmpi-x.y.z-ofi/bin/mpivars.sh
   ```

   This will set the PATH and LD_LIBRARY_PATH and MANPATH variables for this MPI version.

2. Specify the location of the installed PSM3 provider. If Intel® Ethernet Host Software was installed properly, this should be

   ```
   :> export FI_PROVIDER_PATH=/usr/lib64/libfabric
   ```

3. Use the options in your `mpirun` command to specify the use of libfabric (ofi) with the PSM3 provider.

   For example:

   ```
   :> mpirun -mca mtl ofi -x FI_PROVIDER_PATH=$FI_PROVIDER_PATH -x
   FI_PROVIDER=psm3 …
   ```

4. Verify that the PSM3 library was found and is being used by MPI.

   The following command will display PSM3 information to stdout:

   ```
   :> mpirun ... -x PSM3_IDENTIFY=1 ...
   ```

   NOTE

   If you do not see PSM3 output with this variable set, then PSM3 was not found nor is being used.

5. Due to the failover nature of Open MPI and libfabric, it is possible if something is not exactly correct with the PSM3 software environment, a lower performing stack could be used that does not include PSM3. This can happen silently. The below options should reduce the likeliness of a silent failover and cause an error if PSM3 is not working correctly.

   a. Construct FI_PROVIDER to specifically exclude any provider other than identically psm3:

   ```
   :> export FI_PROVIDER="${FI_PROVIDER:-^$(fi_info | sed -n 's/^provider: //p'
   | sort -u | grep -v 'psm3$' | tr '\n' ',' | sed 's/,$//')}"
   ```

   If this worked it should construct a string for `FI_PROVIDER` such as:

   ```
   :> echo $FI_PROVIDER
   ^psm2,usnic,verbs,ofi_rxm,ofi_rxd,shm,UDP,tcp,sockets,ofi_perf_hook,ofi_noop_h
   ook,ofi_mrail,psm3;ofi_rxd
   ```

   Note the last `psm3;ofi_rxd` is the layered provider, different than `psm3`.

6. **b. Disable the tcp btl.** This will prevent Open MPI from using it in case the ofi mtl fails completely:

   ```
   :> mpirun ... -mca btl ^tcp ...
   ```

   Alternatively, Open MPI could be configured with `--enable-mca-no-build=btl-tcp`.

4.1 MPI Benchmark Fundamentals

Two common benchmark applications 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 Open MPI as packaged with Intel® EFS to measure latency, bandwidth, and message rate. In these examples we will use the IMB-MPI1 benchmark. For more information, refer to the Intel® MPI Benchmarks User Guide.

NOTE

Examples are shown here for Open MPI, however, for many applications, Intel® MPI Library may offer better performance and stability.

To begin, load Open MPI into the environment:

```
:> source /usr/mpi/gcc/openmpi-4.1.1-ofi/bin/mpivars.sh
```

NOTE

You must have password-less ssh enabled between all nodes where you want to run benchmarks.

MPI Latency

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

```
:> mpirun -np 2 --map-by ppr:1:node -host node1,node2 -x FI_PROVIDER=psm3 ./IMB-
MPI1 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 on page 30). 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. We use the `Uniband` and `Biband` benchmarks for MPI bandwidth measurements 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:

```
:> mpirun -np 2 --map-by ppr:1:node -host node1,node2 -x FI_PROVIDER=psm3 ./IMB-
MPI1 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 example above. 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:

```
:> mpirun -np 8 --map-by ppr:4:node -host node1:4,nodes2:4 -x FI_PROVIDER=psm3 ./
IMB-MPI1 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.

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.

If we have nodes with 32 physical cores per node,

```
:> mpirun -np 64 --map-by ppr:32:node -host node1:32,node2:32 -x
FI_PROVIDER=psm3 ./IMB-MPI1 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.

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:

```
:> mpirun -np $((128*32)) --map-by ppr:32:node -hostfile 128hosts -x
FI_PROVIDER=psm3 ./IMB-MPI1 Allreduce
-npmin $((128*32))
...
#---------------------------------------------------------------- #
Benchmarking Allreduce # #processes = 4096
```

```
#------------------------------------------------------------
#bytes #repetitions t_min[usec] t_max[usec] t_avg[usec]
...
```

Typically, `t_avg` latency gives a good idea of the performance of the system. Sometimes, large deviations between `t_min` and `t_max` can indicate sub-optimal 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 wish 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` will vary from system to system. Then, 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, especially 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. It is possible that adjusting these variables for other workloads not shown below may also help improve performance.

Table 3. PSM3 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=1,2, or 3` increases bandwidth from the default of `PSM3_RDMA=0`
MPI Uni and Bi-directional bandwidth, `PSM3_RDMA=1` mode, 128KB+	`PSM3_RV_QP_PER_CONN=4` (default). Uses multiple queue pairs per MPI process.
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 (e.g. `mpirun ... -genv PSM3_RDMA=1 ...`). In order to set these environment variables with Open MPI, you must pass them as mpirun command arguments (e.g. `mpirun ... -x PSM3_RDMA=1 ...`)

PSM3_RDMA modes

PSM3 supports multiple RoCE data movement modes which are configurable with the environment variable `PSM3_RDMA`. For more detail, 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 very minimal or 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 very 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 that PSM3 has to recover from. Check the PFC configuration, and also try to pace PSM3 by reducing `PSM3_FLOW_CREDITS`, to alleviate the drops and possibly improve performance.

4.4 MPI Collective 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 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, 8 nodes, 52 processes per node)
MPI Alltoall	Intel® MPI Library 2019 Update 9	`-genv I_MPI_ADJUST_ALLTOALL=4` for 16 nodes, 52ppn, 512B-1KB, 8KB-512KB	Other algorithms may help other node counts, ppn, and message sizes
MPI_Allreduce	Intel® MPI Library 2021 Update 2	`-genv I_MPI_ADJUST_ALLREDUCE="4:0-1048575;2:1048576-104857600"` for 16-32 nodes, 52ppn	use in conjunction with PSM3_RDMA=1. Algorithm 4 (Topology aware Reduce and Bcast) is used for message sizes below 1MB, and algorithm 2 (Rabenseifner's) is used for message sizes >=1MB.

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://software.intel.com/content/www/us/en/develop/documentation/mpi-developer-reference-windows/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`.

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 E810 NIC:

```
:> 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)
```

The first column is the slot number. Then, 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 output above you can see the E810-C NIC is connected to NUMA node 0. This corresponds to the first 26 CPU cores, as seen with:

```
:> 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 above, you must make sure 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, we would have to tell 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 we pinned to any of cores 0-25 (the default). With the Intel® MPI Library, we can 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 CUDA and GPUDirect*

As with 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) - lowering to 65536 significantly improves large message pingpong latency (osu_latency) but has a negative effect on streaming bandwidth (osu_bw/osu_bibw).
- PSM3_CUDA_THRESH_RNDV (default 8000) - larger values decrease latency in 8KB-32KB message size range, but may have a negative impact on bandwidth

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