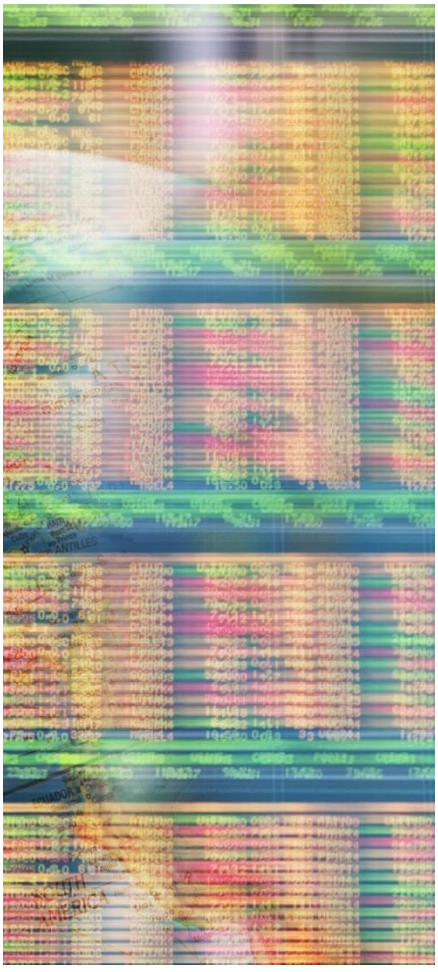
Optimization Guide

High Performance Computing
Intel® Xeon® Scalable Processor

HPC Cluster Tuning on 3rd Generation Intel® Xeon® Scalable Processors



Revision Record

Date	Rev.	Description
05/26/21	1.1	Fixed small typo and added conclusion and feedback sections
04/06/2021	1.0	Initial public release.

1. Introduction

Optimizing performance of servers used for the high-performance computing (HPC) applications may require different configuration options than for servers used with other enterprise applications. For HPC clusters, the goal is to reduce workload runtimes for applications using MPI libraries and high-performance fabrics This guide will cover many system and software configuration options that have been demonstrated to improve application performance in internal controlled tests.

All configuration settings are intended for multi-node HPC clusters running 2-socket 3rd Generation Intel® Xeon processor-based servers. No other hardware has been evaluated for this guide.

The objective of this guide is to provide an environment optimized for typical multi-user production clusters. Configuration settings should be beneficial to a broad list of multi-node applications using MPI libraries. However, HPC applications may be affected differently by settings using in this guide; therefor performance improvement for any single application cannot be guaranteed.

1.1. 3rd Generation Intel® Xeon Scalable Processors

3rd Generation Intel® Xeon® Scalable processors (former codename “Ice Lake”) deliver industry-leading, workload-optimized platforms with built-in AI acceleration, providing a seamless performance foundation to help speed data’s transformative impact, from the multi-cloud to the intelligent edge and back. Here are some of the features in these new processors:

- Enhanced performance
- Enhanced Intel® Deep Learning Boost with VNNI
- More Intel® Ultra Path Interconnect (UPI) links
- Increased DDR4 memory speed and capacity (2 integrated memory controllers; 4 channels per controller)
- Intel® Advanced Vector Extensions (Intel® AVX)
- Intel® Security Essentials supporting Intel® Security Libraries for Data Center (Intel® SecL-DC)
- Intel® Speed Select Technology (Intel® SST)
- Support for Intel® Optane™ Persistent Memory 200 series

2. Hardware Configuration

2.1. DIMM Slot Configuration

Populate all memory channels with the fastest DIMM speed supported by the platform.

Intel 3rd Generation Xeon Scalable processors supports 8 memory channels per processor. Every memory channel should be occupied by at least one DIMM. Use identical dual-rank, registered DIMMS for all memory slots. Dual-rank DIMMs will perform better than single rank DIMMs. DIMM speed should be the fasted speed supported by the platform.

At the same memory speed, 2 DIMMs per channel may perform slightly better than 1 DIMM per channel, if memory speed is not reduced by using more than 1 DIMM.

Do not use Intel® Optane™ memory for HPC benchmarks.

2.2. Memory Size

Most HPC applications and benchmarks will benefit from larger physical memory size. Specific requirements will be determined by the application. As an example, each cluster node running the GROMACS or LAMMPS molecular dynamics codes should have a minimum 96 GB of RAM installed. Following memory population guidelines and using one 16-GB DDR4 DIMM in each memory channel would provide a total 256GB of RAM for a 2-socket system.

2.3. Memory Errors

A repeating, corrected memory error will reduce performance. If memory correction is enabled, check **dmsg** or the system event log to confirm there are no corrected memory events and replace any DIMMs that show repeated memory errors.

2.4. BIOS Settings

Enable Sub-NUMA Clusters, enable One-way IMC Interleave, and set power profiles to “Performance”.

CPU Power and Performance Policies and Fan Profiles should always be set to “Performance”.

Enable Turbo Mode. It is unlikely to improve HPC application results due to high CPU utilization, but it will not reduce performance. You may disable it if performance metrics for a benchmark run must be consistent with previous runs.

The recommended setting for hyper-threading (SMT or Symmetric Multi-Threading) is enabled; however, performance benefits will vary for each application. For some applications, a small decrease in performance may be observed. It is recommended to evaluate hyper-threading performance for the applications that you will use. For the STREAM benchmark specifically, the recommended setting for hyper-threading is Disabled.

2.4.1. Recommended settings.

Use optimized configuration settings and recommended values. Default values are noted by an asterisk (*).

Configuration Item	Recommended Value
Hyper-Threading (SMT)	Enabled* (see text)
Core Prefetchers	Enabled*
Turbo Boost Technology	Enabled*
Intel® SpeedStep® (P-States)	Disabled
SNC (Sub-NUMA Clusters)	Enabled
IMC Interleave	One-way
UPI Prefetch	Enabled*
XPT Prefetch	Enabled*
Total Memory Encryption (TME)	Disabled
Memory controller page policy	Static closed
Autonomous Core C-State	Disabled*
CPU C6 Report	Disabled*
Enhanced Halt State (C1E)	Disabled*
Package C State	C0/C1 State*
Relax Ordering	Disabled*
Intel VT for Directed I/O (Intel VT-D)	Disabled*
CPU Power Policy	Performance
Local/Remote Threshold	Auto*
LLC Prefetch	Disabled*

Configuration Item	Recommended Value
LLC Dead Line Alloc.	Enabled*
Directory AToS	Disabled*
Direct-to-UPI (D2U)	Enabled
DBP-for-F	Enabled

2.4.2. Description of Settings

Sub-NUMA Cluster (SNC)

SNC is a feature that provides similar localization benefits as Cluster-On-Die (COD), a feature found in previous processor families, without some of COD's downsides. SNC breaks up the last level cache (LLC) into disjoint clusters based on address range, with each cluster bound to a subset of the memory controllers in the system. SNC improves average latency to the LLC and is a replacement for the COD feature found in previous processor families.

For all HPC applications, both SNC and XPT/UPI prefetch should be enabled. This will set two clusters per socket and utilize LLC capacity more efficiently and reduces latency due to core/IMC proximity.

Integrated Memory Controller (IMC) Interleaving

This controls the interleaving between the Integrated Memory Controllers (IMCs). If SNC is enabled, IMC Interleaving is set to one-way, and there will be no interleaving.

XPT (eXtended Prediction Table) Prefetch

Extended prediction table (XPT) Prefetch is a new capability that is designed to reduce local memory access latency. XPT Prefetch is an "LLC miss predictor" in each core that will issue a speculative DRAM read request in parallel to an LLC lookup, but only when XPT predicts a "miss" from the LLC lookup.

Ultra-Path Interconnect (UPI) Prefetch

UPI Prefetch is another new capability that is designed to reduce remote memory access latency. The UPI controller issues a UPI Prefetch, also in parallel to an LLC lookup, to the memory controller when a remote read arrives to the home socket.

Direct-to-UPI (D2U or D2K)

D2U is a latency-saving feature for remote read transactions. With D2U enabled, the IMC will send the data directly to the UPI instead of going through the Caching and Home Agent (CHA), reducing latency. Keep enabled, although workloads that are highly NUMA-optimized or that use high levels of memory bandwidth are less likely to be affected by disabling D2U.

DBP-for-F

DBP-for-F is a new feature that can benefit multi-threaded workloads, but workloads that are single-threaded could experience lower performance.

3. Linux Optimizations

3.1. Virtual Machines

Running applications inside of a virtual machine will reduce performance, although the reduction may be small. Performance impact is dependent on the hypervisor and the configuration used. Do not execute HPC benchmarks in a virtual machine.

3.2. Network Configuration

Many HPC applications are fine-grained, requiring more frequent inter-process communication with smaller payloads. As a result, performance is dependent more on communication latency than on bandwidth. High-performance fabrics are used to minimize message latency.

For maximum performance, use one fabric host controller per CPU. In a server, each PCIe expansion slot is associated with one of the CPUs. Install the fabric host controllers so that each CPU is associated with its own fabric host controller. You will need to consult the system board technical specification to determine PCIe lane assignment.

Fabric performance may be further enhanced using multiple links (dual-rail or multi-rail). For many, bandwidth needs are limited and using multi-rail does not benefit performance.

3.3. Disk Configuration

Disk configuration has little to no impact on benchmark performance, including HPL, HPCG, or STREAM.

Storage requirements for other HPC applications will vary.

3.4. CPU Configuration

Before running benchmarks, all CPUs should be set to performance mode. For example, to use the **cpupower** utility, run

```
cpupower -c all frequency-set --governor performance
```

3.5. Services

Disable all unnecessary services and cron jobs.

When running HPC benchmarks, do not run them as a job using a batch scheduler or resource manager, and those services should be disabled until the benchmark is complete. Make certain that no other users are logged into any systems that will be used during the benchmark.

4. Application Settings

4.1. Development Environment

Build and execute applications using the latest Intel® oneAPI HPC Toolkit.

Intel® oneAPI Toolkits enable the development with a unified toolset, allowing developers to deliver applications and solutions across CPU, GPU, and FPGA architectures. The Intel® oneAPI HPC Toolkit delivers what's needed to build, analyze, optimize, and scale HPC applications with the latest techniques in vectorization, multithreading, multi-node parallelization, and memory optimization. The HPC toolkit is an add-on to the Intel® oneAPI Base Toolkit, which is required.

The oneAPI Toolkits are available for installation using a local installer, or through online APT and YUM repositories. To

install the latest toolkit, go to
https://software.intel.com/content/www/us/en/develop/tools/oneapi/hpc-toolkit/download.html

4.1.1. Intel® oneAPI Math Kernel Library (oneMKL)

The oneMKL provides enhanced math routines and libraries, such as BLAS, LAPACK, sparse solvers, fast Fourier transforms (FFT), random number generator functions (RNG), summary statistics, data fitting, and vector math. Use of oneMKL is recommended for optimal performance of HPC applications and benchmarks.

The library is included in the Intel® oneAPI Base Toolkit, but oneMKL support for Intel® MPI library or Intel® Fortran Compilers requires the Intel® oneAPI HPC Toolkit.

4.1.2. Intel MPI Library

Use Intel MPI Library minimum version 2021.2.0. MPI applications should be compiled with this version using compilers and libraries from Intel oneAPI toolkit 2021.2.0 or later.

4.2. User Environment

4.2.1. Intel® Hyper-Threading Technology

For OpenMP based applications that do not benefit from using simultaneous multi-threaded cores, make sure that the number of OpenMP threads do not exceed the available number of physical cores on the system. OpenMP threads are controlled by setting the `OMP_NUM_THREADS` environment variable.

Also set the appropriate thread to core affinity, based on how Hyper-Threading is enabled on the server.

If Hyper-Threading is enabled:

```
export KMP_AFFINITY=granularity=fine,compact,1,0
```

If Hyper-Threading is disabled:

```
export KMP_AFFINITY=compact
```

4.2.2. Multi-Rail Omni-Path Fabric

If your system is configured with Omni-Path fabric and multiple links, enable multi-rail communication. Set the variable PSM2_MULTIRAIL equal to the number of cable links on each host controller. By default, it is set to 1.

4.3. Benchmark Optimization

Optimized settings for HPC benchmarks may differ from applications.

4.3.1. HP LINPACK

Optimum tuning of the HP LINPACK benchmark uses custom configuration that may impact performance of other benchmarks and applications. For that reason, it is not included here. For information on tuning the HP LINPACK benchmark, contact your Intel representative.

4.3.2. HPCG

Use the Intel® Optimized HPCG benchmark included with the oneAPI Math Kernel Library.

The High Performance Conjugate Gradients (HPCG) Benchmark project (http://hpcg-benchmark.org) is designed to complement the HP LINPACK (HPL) benchmark by providing metrics that more closely match a different and broad set of important applications. It is designed to measure the performance of

- Sparse matrix-vector multiplication.
- Vector updates.
- Global dot products.
- Local symmetric Gauss-Seidel smoother.
- Sparse triangular solver

The Intel® Optimized HPCG benchmark provides an implementation of the HPCG benchmark optimized for Intel® Xeon® processors with support for the latest processor technologies, including Intel® Advanced Vector Extensions (Intel® AVX), Intel® Advanced Vector Extensions 2 (Intel® AVX2), Intel® Advanced Vector Extensions 512 (Intel® AVX-512).

The benchmark can be found in the "`/benchmarks/hpcg`" subdirectory under the oneAPI MKL installation. To prepare the benchmark, follow the instructions at https://software.intel.com/content/www/us/en/develop/documentation/onemkl-linux-developer-guide/top/intel-math-kernel-library-benchmarks/intel-optimized-high-performance-conjugate-gradient-benchmark/getting-started-with-intel-optimized-hpcg.html

Use the Scalable processor optimized binary **xhpcg_skx**. The Intel Optimized HPCG package also includes the source code necessary to build these versions of the benchmark for other MPI implementations. Also note that:

- Small problem sizes will produce very good results. However, the problem size must be large enough so that it will not fit into cache; otherwise it is considered an invalid run. It should occupy a minimum 25% of physical memory. Optimum local dimension grid size will need to determine through practical evaluation.
- Longer runtimes beyond the required 3600s runtime do not appear to impact performance.
- Best results are obtained when using from 1 to 1.25 MPI process per total core count and 12 to 16 OpenMP threads per MPI process. The optimal configuration will need to be determined through experimentation. Skip SMT cores when assigning threads.

4.3.3. STREAM

The STREAM benchmark is a simple, synthetic benchmark designed to measure sustainable memory bandwidth (in MB/s) and a corresponding computation rate for four simple vector kernels (Copy, Scale, Add and Triad). It is also part of the HPCC benchmark suite. Its source code is freely available from http://www.cs.virginia.edu/stream/. It measures:

- Sustainable memory bandwidth
- Corresponding computation rate for a simple vector kernel

The general rule for STREAM is that each array must be at least four times (4×) the sum of all last-level caches used in the run. STREAM may be run in its standard form, or it may be optimized. When optimized, results must be identified as such (see the STREAM FAQ at http://www.cs.virginia.edu/stream/ref.html).

For instructions on how to obtain the best performance of the standard STREAM benchmark on Intel processors, see https://software.intel.com/content/www/us/en/develop/articles/optimizing-memory-bandwidth-on-stream-triad.html

5. References

5.1. numactl

The **numactl** tool can be used to view the configuration and status of the NUMA node of the current server. For example, CPU core count, memory size of each node, and the distance between different nodes. The process can be bound to the specified CPU core through this tool, and the specified CPU core will run the corresponding process.

Figure: numactl sample output

```
[root@localhost ~]# numactl -H
available: 4 nodes (0-3)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
node 0 size: 64794 MB
node 0 free: 55404 MB
node 1 cpus: 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
node 1 size: 65404 MB
node 1 free: 58642 MB
node 2 cpus: 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
node 2 size: 65404 MB
node 2 free: 61181 MB
node 3 cpus: 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
node 3 size: 65402 MB
node 3 free: 55592 MB
node distances:
node   0   1   2   3
  0:  10  15  20  20
  1:  15  10  20  20
  2:  20  20  10  15
  3:  20  20  15  10
```

5.1.1. numastat

You can view the of the status of current NUMA nodes using **numastat**. This includes local and remote memory access by CPU cores.

Figure: numastat sample output

```
[root@localhost test]# numastat
                           node0           node1           node2           node3
numa_hit                68641597        59333134        50159172        55076006
numa_miss                     1         1005288          188567            2132
numa_foreign            1171587            2133               0           22268
interleave_hit             1476            1463            1477            1410
local_node              68640860        59318730        50157010        55073766
other_node                   738         1019692          190729            4372
```

5.2. PCM

The Processor Count Monitor (PCM) can be used to monitor performance indicators of the Intel CPU core. PCM is often used to monitor the bandwidth of persistent memory. The tool can be downloaded from https://github.com/opcm/pcm.

Figure: PCM example of monitoring persistent memory bandwidth

```
|---------------------------------------||---------------------------------------||---------------------------------------||---------------------------------------|
|--            Socket  0              --||--             Socket  1             --||--             Socket  2             --||--             Socket  3             --|
|---------------------------------------||---------------------------------------||---------------------------------------||---------------------------------------|
|--     Memory Channel Monitoring      --||--     Memory Channel Monitoring      --||--     Memory Channel Monitoring      --||--     Memory Channel Monitoring      --|
|---------------------------------------||---------------------------------------||---------------------------------------||---------------------------------------|
|-- Mem Ch  0: Reads (MB/s):     1.85 --||-- Mem Ch  0: Reads (MB/s):     1.75 --||-- Mem Ch  0: Reads (MB/s):     1.77 --||-- Mem Ch  0: Reads (MB/s):     1.93 --|
|--            Writes(MB/s):     1.78 --||--            Writes(MB/s):     1.72 --||--            Writes(MB/s):     1.73 --||--            Writes(MB/s):     1.76 --|
|--       PMM Reads(MB/s)   :     0.00 --||--       PMM Reads(MB/s)   :     0.00 --||--       PMM Reads(MB/s)   :     0.00 --||--       PMM Reads(MB/s)   :     0.00 --|
|--       PMM Writes(MB/s)  :     0.00 --||--       PMM Writes(MB/s)  :     0.00 --||--       PMM Writes(MB/s)  :     0.00 --||--       PMM Writes(MB/s)  :     0.00 --|
|-- Mem Ch  1: Reads (MB/s):     1.81 --||-- Mem Ch  1: Reads (MB/s):     1.75 --||-- Mem Ch  1: Reads (MB/s):     1.77 --||-- Mem Ch  1: Reads (MB/s):     1.94 --|
|--            Writes(MB/s):     1.75 --||--            Writes(MB/s):     1.72 --||--            Writes(MB/s):     1.73 --||--            Writes(MB/s):     1.77 --|
|--       PMM Reads(MB/s)   :     0.00 --||--       PMM Reads(MB/s)   :     0.00 --||--       PMM Reads(MB/s)   :     0.00 --||--       PMM Reads(MB/s)   :     0.00 --|
|--       PMM Writes(MB/s)  :     0.00 --||--       PMM Writes(MB/s)  :     0.00 --||--       PMM Writes(MB/s)  :     0.00 --||--       PMM Writes(MB/s)  :     0.00 --|
|-- Mem Ch  2: Reads (MB/s):     1.83 --||-- Mem Ch  2: Reads (MB/s):     1.75 --||-- Mem Ch  2: Reads (MB/s):     1.77 --||-- Mem Ch  2: Reads (MB/s):     1.90 --|
|--            Writes(MB/s):     1.77 --||--            Writes(MB/s):     1.73 --||--            Writes(MB/s):     1.73 --||--            Writes(MB/s):     1.76 --|
|--       PMM Reads(MB/s)   :     0.00 --||--       PMM Reads(MB/s)   :     0.00 --||--       PMM Reads(MB/s)   :     0.00 --||--       PMM Reads(MB/s)   :     0.00 --|
|--       PMM Writes(MB/s)  :     0.00 --||--       PMM Writes(MB/s)  :     0.00 --||--       PMM Writes(MB/s)  :     0.00 --||--       PMM Writes(MB/s)  :     0.00 --|
|-- Mem Ch  3: Reads (MB/s):     1.87 --||-- Mem Ch  3: Reads (MB/s):     1.78 --||-- Mem Ch  3: Reads (MB/s):     1.82 --||-- Mem Ch  3: Reads (MB/s):     1.97 --|
|--            Writes(MB/s):     1.76 --||--            Writes(MB/s):     1.74 --||--            Writes(MB/s):     1.74 --||--            Writes(MB/s):     1.78 --|
|--       PMM Reads(MB/s)   :     0.00 --||--       PMM Reads(MB/s)   :     0.00 --||--       PMM Reads(MB/s)   :     0.00 --||--       PMM Reads(MB/s)   :     0.00 --|
|--       PMM Writes(MB/s)  :     0.00 --||--       PMM Writes(MB/s)  :     0.00 --||--       PMM Writes(MB/s)  :     0.00 --||--       PMM Writes(MB/s)  :     0.00 --|
|-- Mem Ch  4: Reads (MB/s):     1.88 --||-- Mem Ch  4: Reads (MB/s):     1.79 --||-- Mem Ch  4: Reads (MB/s):     1.79 --||-- Mem Ch  4: Reads (MB/s):     1.94 --|
|--            Writes(MB/s):     1.78 --||--            Writes(MB/s):     1.74 --||--            Writes(MB/s):     1.73 --||--            Writes(MB/s):     1.77 --|
|--       PMM Reads(MB/s)   :     0.00 --||--       PMM Reads(MB/s)   :     0.00 --||--       PMM Reads(MB/s)   :     0.00 --||--       PMM Reads(MB/s)   :     0.00 --|
|--       PMM Writes(MB/s)  :     0.00 --||--       PMM Writes(MB/s)  :     0.00 --||--       PMM Writes(MB/s)  :     0.00 --||--       PMM Writes(MB/s)  :     0.00 --|
|-- Mem Ch  5: Reads (MB/s):     1.85 --||-- Mem Ch  5: Reads (MB/s):     1.79 --||-- Mem Ch  5: Reads (MB/s):     1.79 --||-- Mem Ch  5: Reads (MB/s):     1.92 --|
|--            Writes(MB/s):     1.74 --||--            Writes(MB/s):     1.74 --||--            Writes(MB/s):     1.73 --||--            Writes(MB/s):     1.76 --|
|--       PMM Reads(MB/s)   :     0.00 --||--       PMM Reads(MB/s)   :     0.00 --||--       PMM Reads(MB/s)   :     0.00 --||--       PMM Reads(MB/s)   :     0.00 --|
|--       PMM Writes(MB/s)  :     0.00 --||--       PMM Writes(MB/s)  :     0.00 --||--       PMM Writes(MB/s)  :     0.00 --||--       PMM Writes(MB/s)  :     0.00 --|
|-- NODE 0 Mem Read (MB/s) :    11.08 --||-- NODE 1 Mem Read (MB/s) :    10.61 --||-- NODE 2 Mem Read (MB/s) :    10.70 --||-- NODE 3 Mem Read (MB/s) :    11.59 --|
|-- NODE 0 Mem Write(MB/s) :    10.58 --||-- NODE 1 Mem Write(MB/s) :    10.39 --||-- NODE 2 Mem Write(MB/s) :    10.40 --||-- NODE 3 Mem Write(MB/s) :    10.60 --|
|-- NODE 0 PMM Read (MB/s):     0.00 --||-- NODE 1 PMM Read (MB/s):     0.00 --||-- NODE 2 PMM Read (MB/s):     0.00 --||-- NODE 3 PMM Read (MB/s):     0.00 --|
|-- NODE 0 PMM Write(MB/s):     0.00 --||-- NODE 1 PMM Write(MB/s):     0.00 --||-- NODE 2 PMM Write(MB/s):     0.00 --||-- NODE 3 PMM Write(MB/s):     0.00 --|
|-- NODE 0.0 NM read hit rate :  0.09 --||-- NODE 1.0 NM read hit rate :  0.05 --||-- NODE 2.0 NM read hit rate :  0.06 --||-- NODE 3.0 NM read hit rate :  0.13 --|
|-- NODE 0.1 NM read hit rate :  0.11 --||-- NODE 1.1 NM read hit rate :  0.07 --||-- NODE 2.1 NM read hit rate :  0.08 --||-- NODE 3.1 NM read hit rate :  0.14 --|
|-- NODE 0.2 NM read hit rate :  0.00 --||-- NODE 1.2 NM read hit rate :  0.00 --||-- NODE 2.2 NM read hit rate :  0.00 --||-- NODE 3.2 NM read hit rate :  0.00 --|
|-- NODE 0.3 NM read hit rate :  0.00 --||-- NODE 1.3 NM read hit rate :  0.00 --||-- NODE 2.3 NM read hit rate :  0.00 --||-- NODE 3.3 NM read hit rate :  0.00 --|
|-- NODE 0 Memory (MB/s):       21.66 --||-- NODE 1 Memory (MB/s):       21.00 --||-- NODE 2 Memory (MB/s):       21.10 --||-- NODE 3 Memory (MB/s):       22.19 --|
|---------------------------------------||---------------------------------------||---------------------------------------||---------------------------------------|
|---------------------------------------||
|--           System DRAM Read Throughput(MB/s):         43.99               --|
|--          System DRAM Write Throughput(MB/s):         41.96               --|
|--            System PMM Read Throughput(MB/s):          0.00               --|
|--           System PMM Write Throughput(MB/s):          0.00               --|
|--                System Read Throughput(MB/s):         43.99               --|
|--               System Write Throughput(MB/s):         41.96               --|
|--              System Memory Throughput(MB/s):         85.95               --|
|---------------------------------------||
```

6. Conclusion

We understand every application is unique. We shared many of our experiences with HPC clusters hoping that some of our learnings could be applied to your specific application. HPC clusters have been well tested on Intel platforms. With 3rd Generation Intel® Xeon® Scalable processor, Intel takes it even further by optimizing the platform as a whole -- CPU, memory, storage, and networking working together for the best user experience.

7. Feedback

We value your feedback. If you have comments (positive or negative) on this guide or are seeking something that is not part of this guide, please reach out to us here:

https://community.intel.com/t5/Software-Tuning-Performance/bd-p/software-tuning-perf-optimization