Analyzing and Tuning SPECmpi Performance on Lenovo ThinkSystem Servers

Explains the difference between parallel and distributed programming paradigm

Introduces SPECmpi, the industry standard Message Passing Interface (MPI) benchmark

Describes SPECmpiM characteristics through system runtime behavior

Provides tuning guidance to optimize the SPECmpiM performance on ThinkSystem servers

Jimmy Cheng

Abstract

Message Passing Interface (MPI) is a message passing library standard widely used in parallel computing to adapt single-threaded workloads for parallelism. The SPEC MPI 2007 benchmark suite is designed to provide an objective representation of the suitability of servers for parallel computing workloads.

This paper describes how to configure Lenovo® ThinkSystem™ servers to obtain the best performance for parallel computing workloads and other applications that have similar characteristics to the SPECmpiM benchmark.

This paper is intended for data center administrators, ThinkSystem end users and technical sales representatives who want to understand how to tune systems for MPI workloads. The paper assumes readers are familiar with Linux and have experience with programming languages such as C/C++.

Contents

Introduction

Parallel and distributed programming models such as Message Passing Interface (MPI) and OpenMP have been developed to solve the problems with the dramatic increase in size and complexity of applications in the modern data center.

By placing OpenMP directives in the existing code, the parallel region-like loop can be easily accelerated within a multi-processor node. For example, the OpenMP directive **`#pragma omp parallel for`** was added to the Triad function of the Stream benchmark source code to speed up the **`for`** loop as shown in Figure 1 below.

```
void tuned_STREAM_Triad(double scalar)
{
   int j;
#pragma omp parallel for
   for (j=0; j<N; j++)
      a[j] = b[j]+scalar*c[j];
}
```

Figure 1 Source code of Stream benchmark

Unlike the OpenMP paradigm, an application launched by MPI library can be deployed within and across nodes in the cluster of data center. As a result, MPI has become one of the dominant programming models in the modern data center. For instance, the Intel MPI library, `mpirun`, uses shared memory for intra-node communication and a tag matching interface (TMI)-capable network fabric for inter-node communication.

The sample usage of mpirun is as follows:

```
# mpirun -n <# of processes> -host hostfile -genv I_MPI_FABRICS shm:tmi ./App
```

In this paper, we introduce MPI and the MPI benchmark SPECmpi. We then describe the testing we performed in our lab using the Lenovo ThinkSystem SR650 server: the effect of MPI workloads have on CPU and memory and how we tuned these subsystems to obtain the best performance for MPI workloads.

As a result of this tuning experience, Lenovo was able to set a new single node 2-socket SPECmpi world record on the ThinkSystem SR650 server with the SPECmpiM_base2007 metric from the MPI M2007 suite of SPEC MPI 2007 Benchmark. Read about our SPECmpi benchmark results on this page:

`https://lenovopress.com/servers/benchmarks/specmpi`

Message Passing Interface

Message Passing Interface (MPI), a message-passing library specification, provides developers a clearly defines set of MPI routines to enable them to implement their own MPI library efficiently. MPI was originally designed for distributed memory architectures, however, it is now widely used in modern high performance computing (HPC) environments to take advantage of multicore architectures and parallel applications.

A MPI library call invokes a number of processes for a parallel application, each of which is assigned a unique rank number which used to communicate between the process to best utilize the resources in the cluster to complete the work.

Figure 2 illustrates a sample of MPI execution in a cluster environment. The MPI library invokes N processes with rank numbers 0, 1, 2, ... N for a parallel application. Each process has its own hardware resource from nodes in a cluster as illustrate in by the blue lines, communicating with other processes to complete the work as illustrated by the red lines.

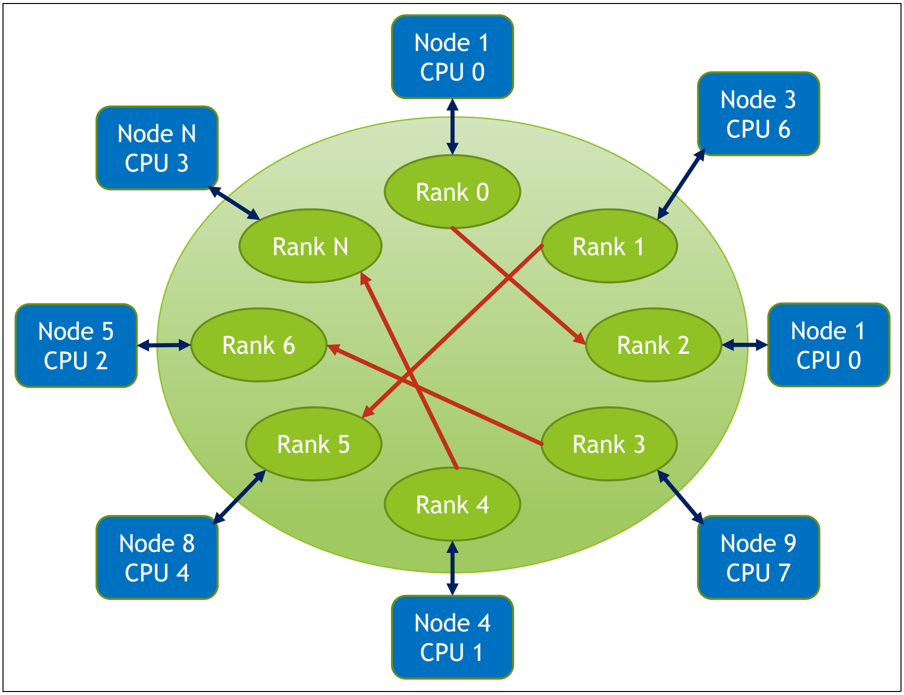


Figure 2   A example of MPI execution in a cluster environment

SPECmpi 2007

With the increase in MPI-parallel applications, Standard Performance Evaluation Corporation (SPEC) developed a standard benchmark named SPEC MPI 2007 specifically for MPI to measure MPI applications performance between different clusters. Instead of using synthetic benchmark or parallelized versions of sequential benchmark, SPEC developed a set of native MPI-parallel compute-intensive applications for SPECmpi.

The benchmark provides medium and large suite for different scale of cluster. The medium suite is suitable for smaller-scale clusters from 4 to 128 ranks (cores) whereas the large suite is designed for larger-scale cluster up to 2048 ranks.

Table 1 lists each sub-benchmark, whether the sub-benchmark is for the Medium or Large suite, the programing language used to implement the test, and the application area each sub-benchmark applies to.

Table 1   Details of SPECmpi 2007 Benchmarks

Benchmark	**Medium Suite**	**Large Suite**	**Programming Language**	**Application Area**
104.milc	Yes		C	Physics: Quantum chromodynamics (QCD)
107.leslie3d	Yes		Fortran	Computational fluid dynamics (CFD)
113.GemsFDTD	Yes		Fortran	Computational electromagnetics (CEM)

Benchmark	Medium Suite	Large Suite	Programming Language	Application Area
115.fds4	Yes		C/Fortran	Computational fluid dynamics (CFD)
121.pop2	Yes	Yes	C	Ocean modeling
122.tachyon	Yes	Yes	C	Graphics: parallel ray tracing
125.RAxML		Yes	C	DNA matching
126.lammps	Yes	Yes	C++	Molecular dynamics simulation
127.wrf2	Yes		C/Fortran	Weather prediction
128.GAPgeofem	Yes	Yes	C/Fortran	Heat transfer using Finite Element Methods (FEM)
129.tera_tf	Yes	Yes	Fortran	3D eulerian hydrodynamics
130.socorro	Yes		C/Fortran	Molecular dynamics using Density-Functional Theory (DFT)
132.zeusmp2	Yes	Yes	C/Fortran	Physics: Computational fluid dynamics (CFD)
137.lu	Yes	Yes	Fortran	Computational fluid dynamics (CFD)
142.dmilc		Yes	C	Physics: Quantum chromodynamics (QCD)
143.dleslie		Yes	Fortran	Computational fluid dynamics (CFD)
145.lGemsFDTD		Yes	Fortran	Computational electromagnetics (CEM)
147.l2wrf2		Yes	C/Fortran	Weather prediction

The SPECmpi benchmark uses “base” and “peak” metrics to evaluate the performance of a server:

- Base metric is minimum requirement for valid submission for both medium and large suite, each sub-benchmark need to be build and run within same environment for base metric.
- Peak metric, on the other hand, is optional for submission, optimizing each individual sub-benchmark with different set of environment settings such as compiler flags.

Each sub-benchmark score is actual elapsed time ratio of the reference system (Celestica A2210) and the system under test (SUT). The base/peak metric are calculate as a Geometric Mean of all individual ratios in the medium or large suite.

Table 2 shows the example of SPECmpiM base score calculation. The first row of the table shows that sub-benchmark 104.milc runs 1565 seconds on the reference system and 75.5 seconds on the SUT, so the ratio is 20.7. The SPECmpiM_Base result 34 is calculate by Geometric Mean of all sub-benchmark’s ratio at the last column of the table.

Table 2 SPECmpi base score calculation

Benchmarks	Run Time	Reference Time	Ratio
104.milc	75.5	1565	20.7
107.leslie3d	185	5220	28.3
113.GemsFDTD	166	6308	38.0
115.fds4	79.2	1951	24.6
121.pop2	134	4128	30.8

Benchmarks	**Run Time**	**Reference Time**	**Ratio**
122.tachyon	85.7	2797	32.7
126.lammps	111	2915	26.3
127.wrf2	147	7796	53.0
128.GAPgeofem	49.7	2065	41.6
129.tera_tf	112	2768	24.7
130.socorro	62.8	3817	60.8
132.zeusmp2	85.8	3103	36.2
137.lu	79.6	3676	46.2
SPECmpiM_Base			**34**

ThinkSystem SR650

We used the Lenovo ThinkSystem SR650 for our testing in the lab.

ThinkSystem SR650 is a two-socket server based on the Intel Xeon Scalable family processors. It offers superior system performance with the Intel Xeon Processor Scalable Family with up to 28-core processors, up to 38.5 MB of last level cache, up to 2666 MHz memory speeds, and two Ultra Path Interconnect (UPI) links between processors that operate at up to 10.4 GT/s.

Figure 3 Lenovo ThinkSystem SR650

The server offers configuration flexibility, powerful computing power, and upgrade capability for MPI applications in the data center.

For information about the SR650, see the Lenovo Press product guide:

https://lenovopress.com/lp0644-lenovo-thinksystem-sr650-server

The configuration we used is summarized in Table 3 on page 7.

Table 3 Test configuration

Feature	1DPC	2DPC	2DPC w/ SNC
Hardware components			
CPU	2 x Intel Xeon Platinum 8180 Processors (28 cores, 2.50 GHz)		
Memory	384 GB 12x 32 GB RDIMMs 2666MHz	768 GB 24x 32 GB RDIMMs 2666MHz	768 GB 24x 32 GB RDIMMs 2666MHz
Disk	800 GB 12Gbps SAS 2.5" SSD		
OS	Red Hat Enterprise Linux 7.3		
UEFI settings			
Operating Mode	Max Performance	Max Performance	Max Performance
SNC	Disabled	Disabled	Enabled

Analysis of CPU performance

CPU utilization and operating frequency are crucial performance metrics to investigate workloads. As shown in Figure 4, SPECmpiM is a compute intensive workload. Both CPU0 and CPU1 keeps running near 100%, CPU utilization falls down only when each sub-benchmark switches.

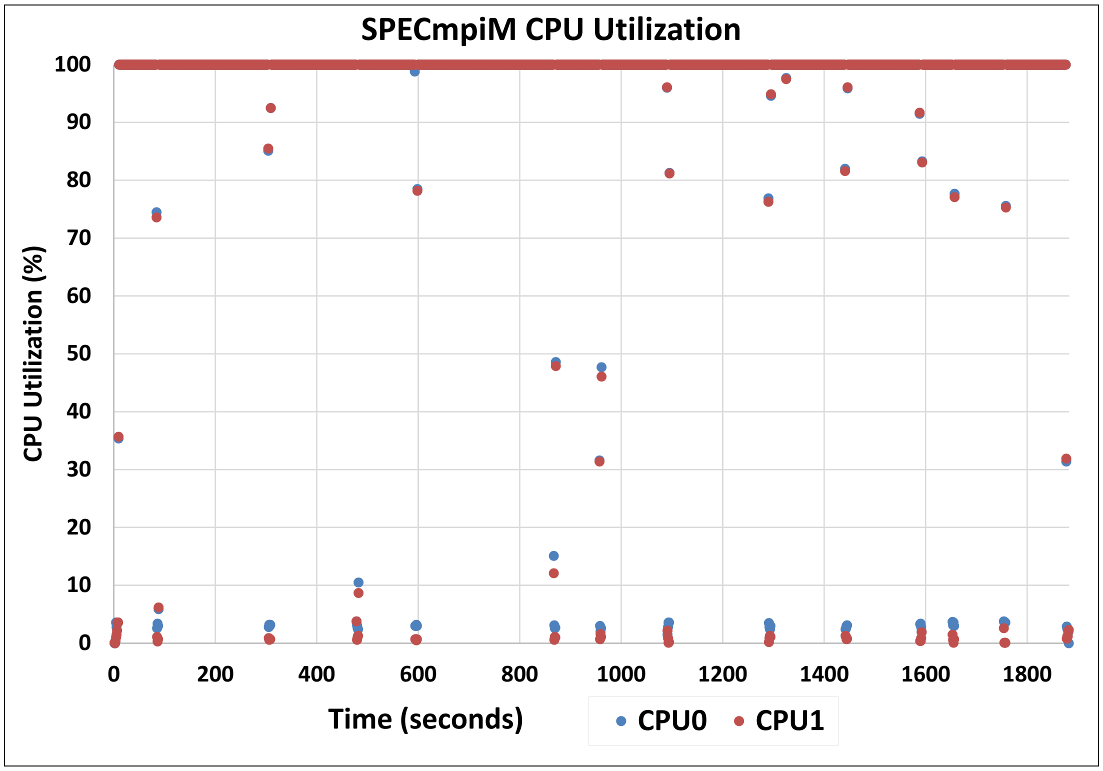


Figure 4 CPU utilization when running SPECmpiM

In addition to CPU utilization, we looked further into Instruction Per Cycle (IPC) to identify whether each processor is busy on computation or waiting (including idle or stalled) on memory I/O. In general, the CPU is more likely to be working on real jobs if IPC > 1.0, and more like to be stalled waiting on memory I/O if IPC < 0.

In Figure 5, we sampled IPC every second while running SPECmpiM. In the chart, the X-axis shows IPC, grouped by a step of 0.1 in the range from 0.11 to 2.29. The Y-axis, left side, shows the number of times the IPC value was in each IPC grouping (bar graph) and the Y-axis right side shows the cumulative IPC as a percentage of each group (line graph).

As you can see from the chart (indicated with an arrow), we note that an IPC less than 1.0 occurs 65% of time, and the average IPC is 0.9 which indicate processor spend more time on waiting rather than computation during SPECmpiM.

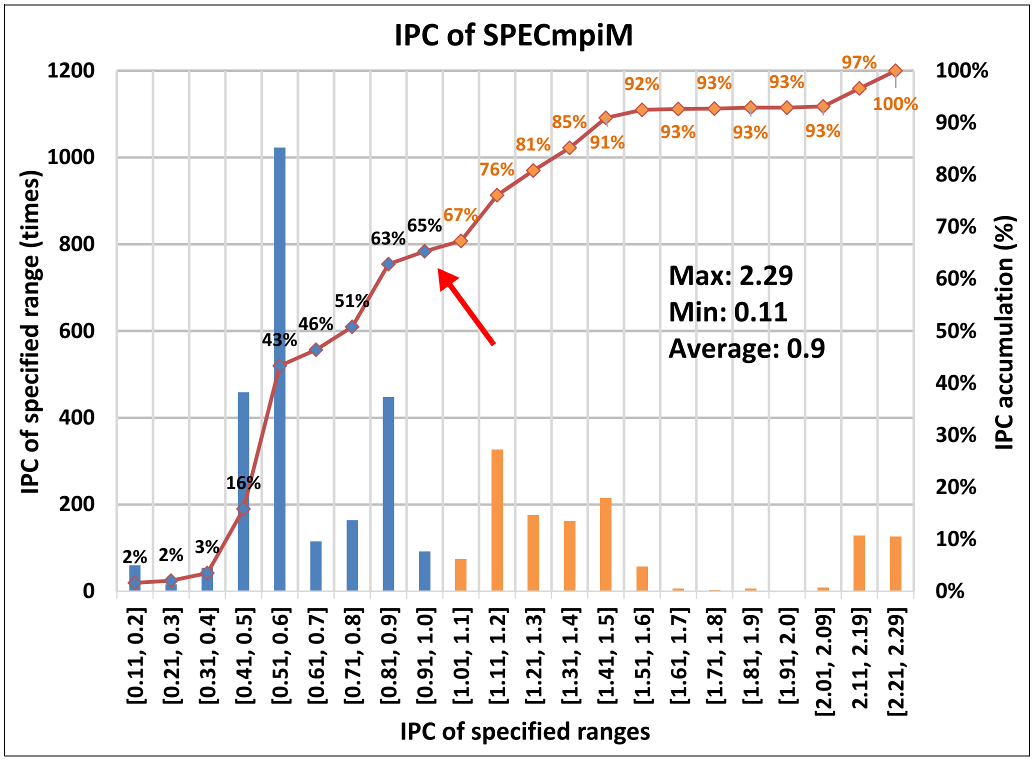


Figure 5 IPC of SPECmpiM

CPU operating frequency is another important metric for performance. Higher frequency brings better performance when running same instruction on same CPU. Figure 6 on page 9 illustrates CPU operating frequency when running the SPECmpiM benchmark suite. Note that, the slight frequency difference between CPU0 and CPU1 is comes from the operating system overhead.

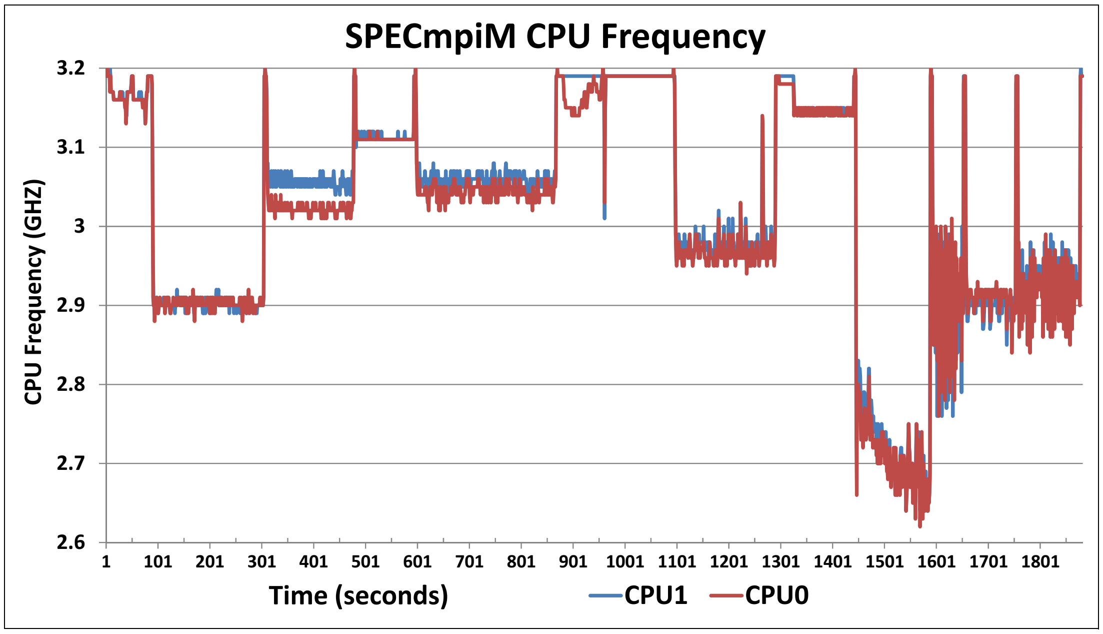


Figure 6 CPU frequency of SPECmpiM

Processor run by default on their base frequency, however for processors with Intel Turbo Boost Technology 2.0 support (such as the Intel Xeon 8180 processor we used in our tests), the processor can run at a higher frequency for a short period of time to obtain better performance.

The actual operating frequency is highly depending on two main factors:

- Type of workload
- Number of active cores

Different workload can be compiled into integer or floating point instructions depending on the source code. AVX, AVX2 and AVX512 instructions are used to speed up floating point by using SIMD (single instruction, multiple data) technique. However, AVX2 and AVX512 instructions consume more processor power than non-AVX instructions. This means that if you use AVX2 or AVX512 instructions instead of non-AVX, and processor frequency remain the same, the Thermal Design Power (TDP) rating of the processor may be exceeded.

As a result, operating turbo frequencies will also be based on the type of instructions used as well as the number of cores the instructions run on.

Table 4 on page 10 shows the base and turbo frequency for Intel Xeon 8180 processor based on the number of active cores. For other processor models, see Figure 2 on page 13 of the *Intel Xeon Processor Scalable Family Specification Update* document, available from:

`https://www.intel.com/content/dam/www/public/us/en/documents/specification-updates/xeon-scalable-spec-update.pdf`

Table 4 The base/turbo frequency of Intel 8180 Processor

Number of active cores	Base frequency			Turbo frequency		
	Non AVX	**AVX 2**	**AVX 512**	**Non AVX**	**AVX 2**	**AVX 512**
1-2 cores	2.5 GHz	2.1 GHz	1.7 GHz	3.8 GHz	3.6 GHz	3.5 GHz
3-4 cores				3.6 GHz	3.4 GHz	3.3 GHz
5-8 cores				3.5 GHz	3.3 GHz	3.2 GHz
9-12 cores				3.5 GHz	3.3 GHz	3.1 GHz
13-16 cores				3.5 GHz	3.3 GHz	2.8 GHz
17-20 cores				3.5 GHz	3.1 GHz	2.6 GHz
21-24 cores				3.3 GHz	2.9 GHz	2.4 GHz
25-28 cores				3.2 GHz	2.8 GHz	2.3 GHz

Analysis of memory performance

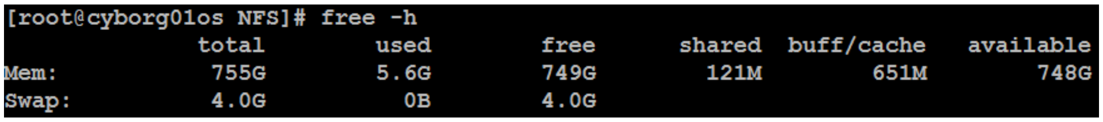
Since the 1980s, processor speed has increased significantly faster than memory speed.[1] As a result, the processor-memory performance gap has become a primary obstacle of system performance especially for memory-bound workloads.

In this section, we will focus on memory characteristic analysis of SPECmpiM through memory footprint, memory bandwidth and memory access pattern.

Memory footprint

In general, a program needs to be load from a drive to physical memory when it's running by CPU. Virtual memory can be used if physical memory capacity is not sufficient for the application, and virtual memory is usually emulated by the drive to extend memory capacity. However, the drive are much slower compared to physical memory (especially technologies such as SATA HDDs with spinning platters) which significantly impacts the performance of the server.

To ensure if SPECmpi performance impact by the virtual memory, we use Linux command **`free`** to diagnosis the memory footprint and SWAP memory usage. The example usage of free command is shown in Figure 7.



```
[root@cyborg01os NFS]# free -h
              total        used        free      shared  buff/cache   available
Mem:           755G        5.6G        749G        121M        651M        748G
Swap:          4.0G          0B        4.0G
```

Figure 7 free -h command

The **`-h`** parameter is used for human-readable output which convert the raw data with proper unit. The `used` column indicates the amount of system memory and SWAP that has been allocated respectively.

[1] Patterson D., Anderson T. et al.: A Case for Intelligent RAM: IRAM. IEEE Micro (1997)

In our lab, we use the free command to sample system memory and SWAP usage every 3 seconds during SPECmpiM running. The results are shown in Figure 8.

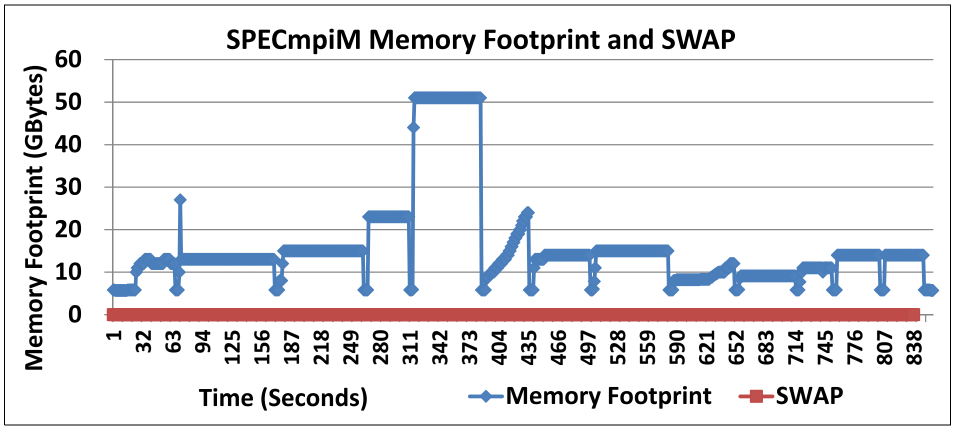


Figure 8 Memory footprint and SWAP of SPECmpi

The system memory footprint is around 5 GB when the server is idle, up to 51 GB when running SPECmpiM. Of importance is to note that SPECmpiM doesn't use SWAP during its operation.

Memory bandwidth

The theoretical memory bandwidth can be calculated by the following formula:

```
Memory frequency × (data width · 8) × number of channels × number of sockets × 90%
```

Our lab test was based on the ThinkSystem SR650 with the following hardware configuration:

- Two Intel Xeon 8180 Scalable Family processors
- Memory frequency 2.666GHz
- 3 memory channels on each of 2 memory controller
- 64-bit memory data width

Using the formula, the highest theoretical memory bandwidth is calculated as follows:

```
2.666 GHz × 64 bit ÷ 8 × 6 channels × 2 sockets × 90%
= 230.3 GBytes/second
```

Figure 9 on page 12 shows the memory read, write and system (aggregation of read and write bandwidth) bandwidth during SPECmpiM running. The SPECmpiM is a memory bandwidth-intensive workload, utilizing more than 50% of the theoretical memory bandwidth most of time during the run. In addition, it required more memory-read bandwidth than memory-write bandwidth by about 3:1.

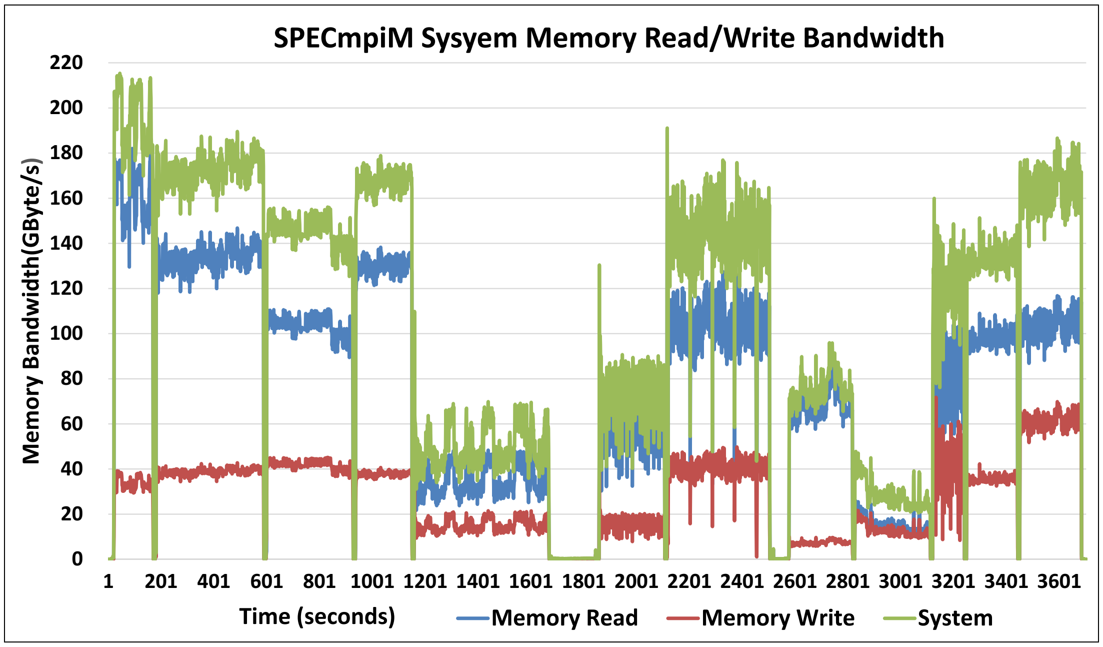


Figure 9 System memory bandwidth of SPECmpiM

Memory access pattern

With Non-Uniform Memory Access (NUMA), memory is divided into multiple local domains managed by each processor and memory controller, rather than one single memory pool. This was done to increase local data access speed and to reduce the memory bus traffic.

The downside to NUMA is that there is a performance penalty to access remote memory -- memory controlled by another processor -- since the request needs one or more hops to other memory controller. An application workload therefore needs to aware the NUMA architecture of a server to avoid this performance penalty and others such as the well-known database “swap insanity” problem.[2]

As shown in Figure 10 on page 13, each Intel Xeon Scalable Family processor contains two sub-NUMA clusters (SNCs). Accessing cores on same SNC domain memory is expected to have lower memory latency than accessing cores across different SNC domains. For example, an application that runs on cores of CPU 1’s SNC Domain 1 will get lower latency when it accesses memory in the same domain than the memory in CPU1’s SNC Domain 2.

[2] See “The MySQL “swap insanity” problem and the effects of the NUMA architecture” by Jeremy Cole, `https://blog.jcole.us/2010/09/28/mysql-swap-insanity-and-the-numa-architecture`

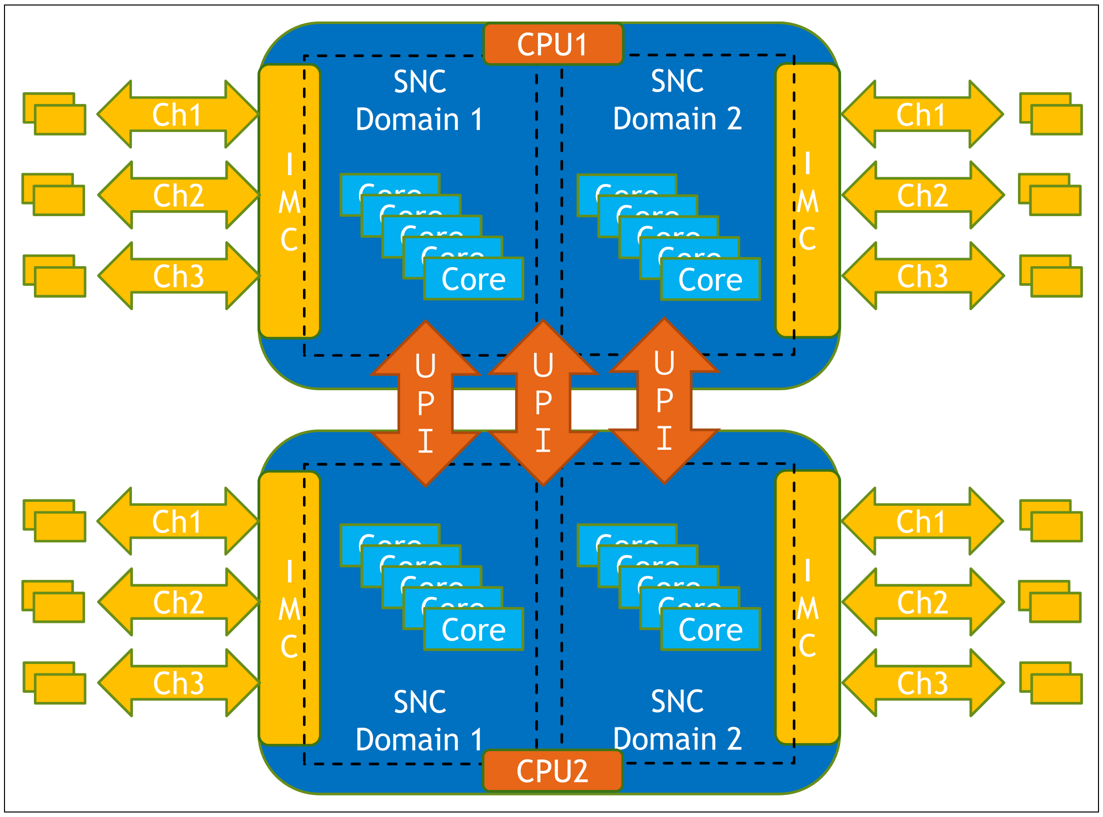


Figure 10 SNC architecture on Intel Scalable Processor

Figure 11 shows the memory access pattern of an SPECmpiM workload. The blue line shows that nearly all memory references go to local socket memory rather than remote socket memory.

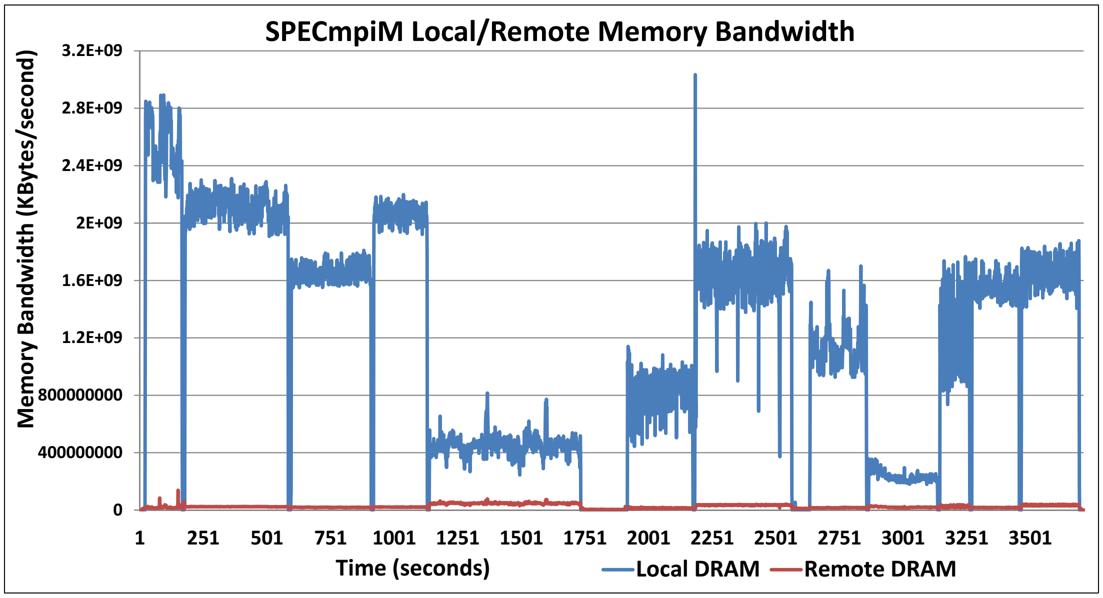


Figure 11 Memory access pattern of SPECmpiM

Performance tuning

As confirmed by our testing in the lab, SPECmpiM requires high CPU frequency and high memory bandwidth, especially for local memory access. As a result, for parallel computing and other SPECmpiM-like workloads, we recommend you configure the ThinkSystem SR650 as follows:

- Intel Xeon Platinum 8180 Processor
- Two DIMMs per channel with 2666 MHz
- Maximum Performance mode (in UEFI)
- SNC Enable (in UEFI)

The ThinkSystem SR650 supports a large range of Intel processor options, however for MPI workload we suggest the use of the Intel Xeon Platinum 8180 Processor. It has highest core count, which gives MPI library maximum capability to parallel the MPI workloads.

The SR650 also supports two DIMMs per channel (DPC), which will result in 3% higher memory bandwidth compared to only having one DIMM per channel, which improves SPECmpiM performance by 2.9%. This is shown in Figure 12.

For more information, see the Lenovo Press paper, *Intel Xeon Scalable Family Balanced Memory Configurations*, https://lenovopress.com/lp0742

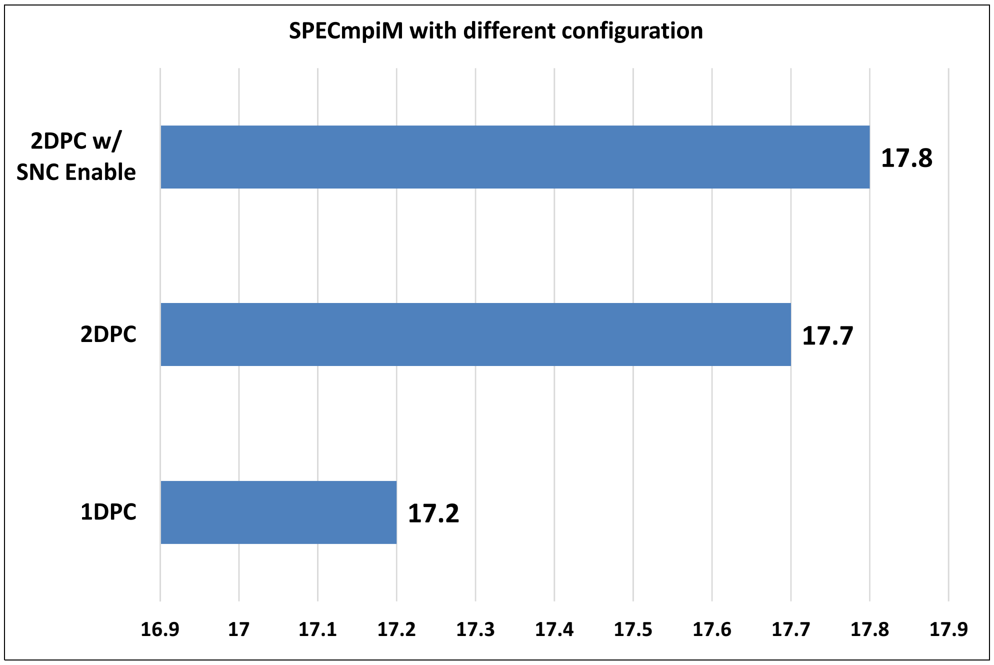
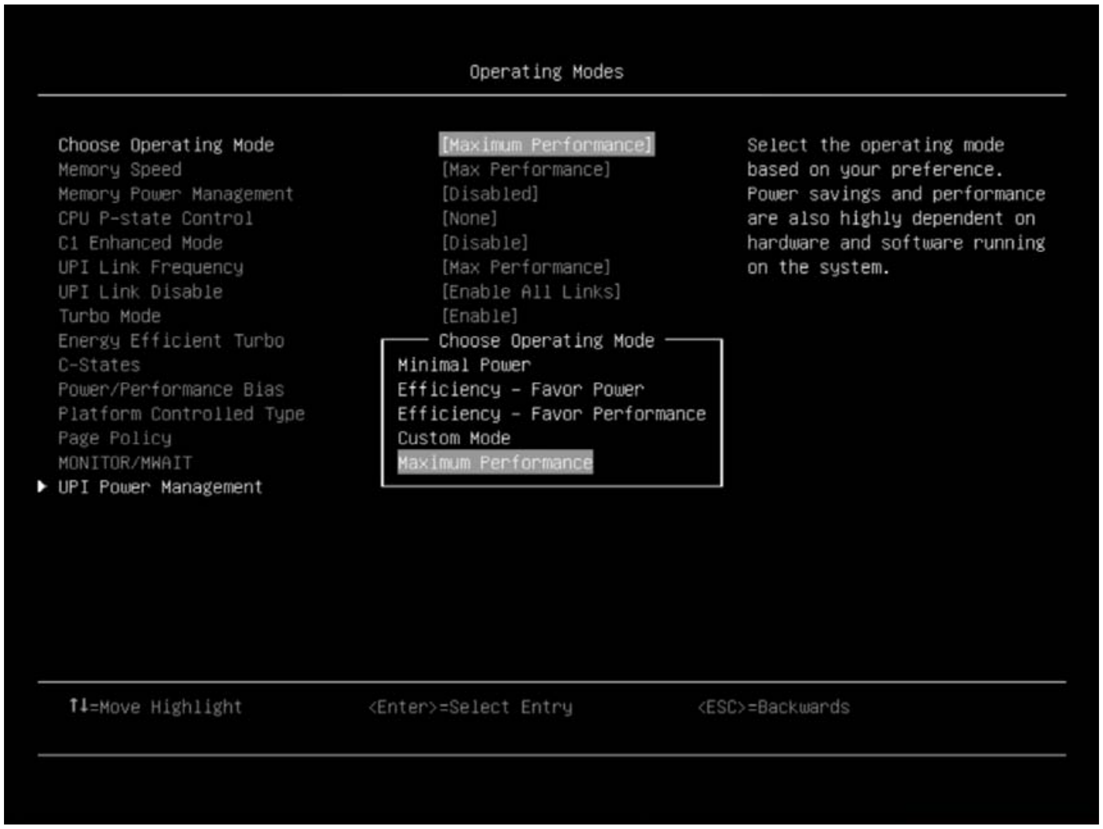


Figure 12 SPECmpiM result with different configuration

In addition to memory hardware configuration, Lenovo provide preset Operating Modes in the UEFI menu to provide different user scenarios. The operating mode **Maximum performance** is recommend for general MPI workload. The selection is made in UEFI as shown in Figure 13 on page 15.



```
                                  Operating Modes

  Choose Operating Mode          [Maximum Performance]     Select the operating mode
  Memory Speed                   [Max Performance]         based on your preference.
  Memory Power Management        [Disabled]                Power savings and performance
  CPU P-state Control            [None]                    are also highly dependent on
  C1 Enhanced Mode               [Disable]                 hardware and software running
  UPI Link Frequency             [Max Performance]         on the system.
  UPI Link Disable               [Enable All Links]
  Turbo Mode                     [Enable]
  Energy Efficient Turbo      ┌── Choose Operating Mode ──┐
  C-States                    │ Minimal Power               │
  Power/Performance Bias      │ Efficiency - Favor Power    │
  Platform Controlled Type    │ Efficiency - Favor Performance
  Page Policy                 │ Custom Mode                 │
  MONITOR/MWAIT               │ Maximum Performance         │
► UPI Power Management        └─────────────────────────────┘

  ↑↓=Move Highlight          <Enter>=Select Entry          <ESC>=Backwards
```

Figure 13 Operating Mode in UEFI menu

The Operating Mode choices are as follows:

- Minimal Power

  Minimal Power mode strives to minimize the absolute power consumption of the system while it is operating. The tradeoff is that performance may be reduced in this mode depending on the application that is running.

- Efficiency - Favor Power

  Efficiency - Favor Power mode maximizes the performance/watt efficiency with a bias towards power savings. It provides the best features for reducing power and increasing performance in applications where maximum bus speeds are not critical.

- Efficiency - Favor Performance

  Efficiency - Favor Performance mode optimizes the performance/watt efficiency with a bias towards performance.

- Maximum Performance

  Maximum Performance mode will maximize the absolute performance of the system without regard for power. In this mode, power consumption is not taken into consideration. Attributes like fan speed and heat output of the system may increase in addition to power consumption. Efficiency of the system may go down in this mode, but the absolute performance may increase depending on the workload that is running.

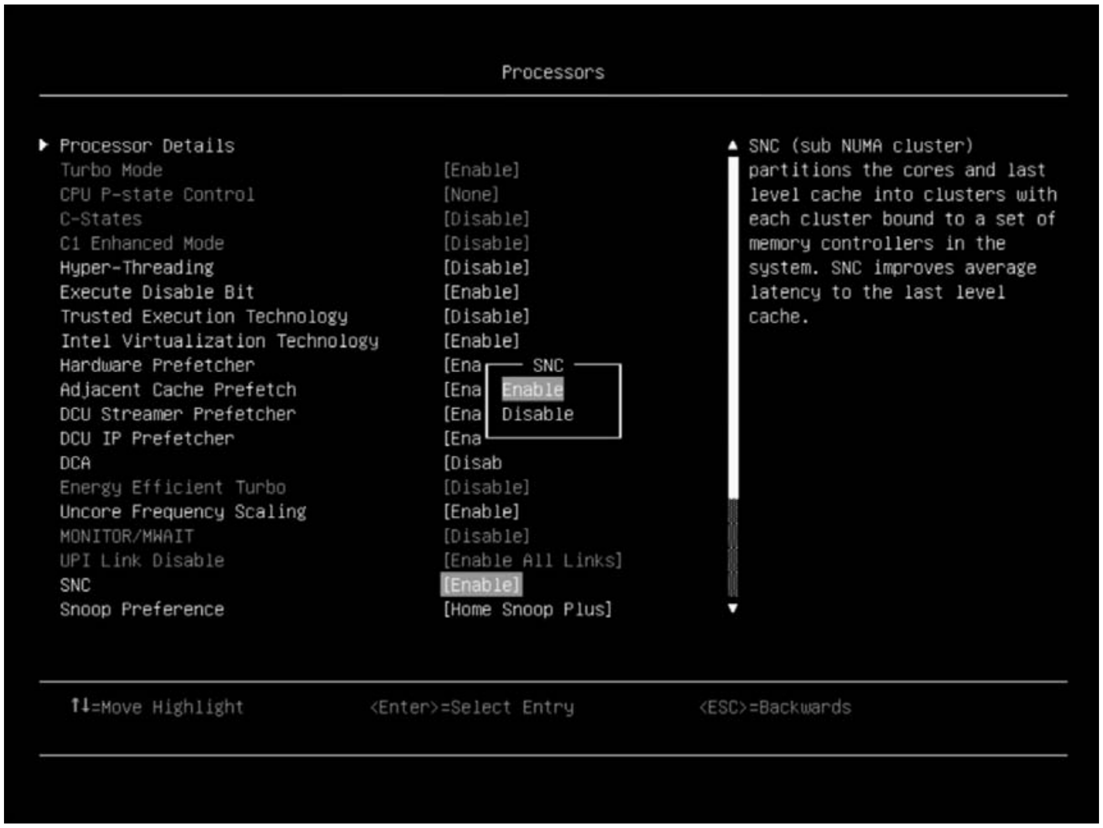
- Custom Mode

  Custom Mode allows the user to individually modify any of the low-level settings that are preset and unchangeable in any of the other 4 preset modes.

Intel Xeon Scalable Family processors introduced the sub-NUMA Cluster (SNC) to optimize the local memory bandwidth and access latency. Similar to Cluster-On-Die (COD) used in previous generation processors, SNC creates two localization domains in a processor. Each SNC domain has its own mapping address with one half of last level cache (LLC) as illustrated in Figure 10 on page 13.

The Enable SNC option in UEFI helps highly NUMA-optimized workloads like SPECmpiM to improve overall performance. Figure 12 on page 14 shows that enabling SNC improves SPECmpiM from 17.7 to 17.8.

Figure 14 shows the SNC setting page in the UEFI.



```
                                   Processors

▸ Processor Details                                       ▲ SNC (sub NUMA cluster)
  Turbo Mode                       [Enable]                 partitions the cores and last
  CPU P-state Control              [None]                   level cache into clusters with
  C-States                         [Disable]                each cluster bound to a set of
  C1 Enhanced Mode                 [Disable]                memory controllers in the
  Hyper-Threading                  [Disable]                system. SNC improves average
  Execute Disable Bit              [Enable]                 latency to the last level
  Trusted Execution Technology     [Disable]                cache.
  Intel Virtualization Technology  [Enable]
  Hardware Prefetcher              [Ena┌── SNC ──┐
  Adjacent Cache Prefetch          [Ena│ Enable  │
  DCU Streamer Prefetcher          [Ena│ Disable │
  DCU IP Prefetcher                [Ena└─────────┘
  DCA                              [Disab
  Energy Efficient Turbo           [Disable]
  Uncore Frequency Scaling         [Enable]
  MONITOR/MWAIT                    [Disable]
  UPI Link Disable                 [Enable All Links]
  SNC                              [Enable]
  Snoop Preference                 [Home Snoop Plus]        ▼

  ↑↓=Move Highlight          <Enter>=Select Entry          <ESC>=Backwards
```

Figure 14   Enabling SNC in UEFI

Reference information

For more information, consult these resources:

- STREAM Benchmark Reference Information

  http://www.cs.virginia.edu/stream/ref.html

Author

Jimmy Cheng is a performance engineer in the Lenovo Data Center Group Laboratory in Taipei Taiwan. Jimmy joined Lenovo in December 2016. Prior to this, he worked on IBM POWER system assurance and validation, ATCA system integration, automation development as well as network performance. Jimmy holds a Master’s Degree in Electronic and Computer Engineering from National Taiwan University of Science and Technology in Taiwan, and a Bachelor’s Degree in Computer Science and Engineering from Yuan-Ze University, Taiwan.

Thanks to the following people for their contributions to this project:

- David Watts, Lenovo Press

Notices

This document was created or updated on March 13, 2018.

Send us your comments via the **Rate & Provide Feedback** form found at
http://lenovopress.com/lp0752

Trademarks