AMD

NVMe SSD Performance Evaluation Guide for Windows Server® 2016 and Red Hat Enterprise Linux® 7.4

Publication # **56367**	Revision: **0.70**
Issue Date: **August 2018**	

Advanced Micro Devices

Contents

List of Tables

List of Figures

Revision History

Date	Revision	Description
August 2018	0.70	Initial public release.

Evaluating NVMe SSD Performance using Windows Server® 2016

Prerequisites

Evaluating NVMe performance on Windows Server® 2016 requires a few modifications before benchmarks can be run. Below is a list of required prerequisites to ensure that benchmark results are accurate and repeatable. For more information on system configuration consult Appendix A.

SSD Preconditioning

Before running any benchmarks, it is crucial that you prepare the drive by "preconditioning" the drive. Preconditioning a drive is recommended to achieve sustained performance on a "fresh" drive. To better understand why SSD preconditioning is important, you can visit the following site:

http://www.snia.org/sites/default/education/tutorials/2011/fall/SolidState/EstherSpanjer_The_Why_How_SSD_Performance_Benchmarking.pdf.

The preconditioning process utilizes three steps to ensure that benchmarking results are accurate and repeatable. It is recommended to run the following workloads with twice the advertised capacity of the SSD to guarantee that all available memory is filled with data including the factory provisioned area.

- Secure erase the SSD
- Fill SSD with 128k sequential data twice
- Fill the drive with 4k random data

If you're running a sequential workload to estimate the read or write throughput, you may skip the last step, although it is not recommended.

Disable Write-Caching

To measure the true performance of the NVMe SSDs being used, it is recommended that you disable write-caching in your Windows installation. This setting can be found by:

1. Go to "Device Manager"
2. Under "Disk Drives", right click on the device name and click on "Policies"
3. Under "Removal Policy", choose "Quick Removal"
4. Click "OK to save the setting.

Power Options

For the highest performance possible, it is important to ensure that the power plan is set to achieve the highest performance. This setting can be verified by navigating to:

Control Panel → Hardware → Power Options → Set to "High Performance"

Performance Options

Similarly, it is recommended that any additional visual effects be turned off. This setting is accessible by navigating to:

Control Panel → System and Security → Advanced System Settings → Performance → Visual Effects and select "Adjust for best performance"

Using Diskspd for Performance Testing

Diskspd is a command line tool for storage benchmarking on Microsoft Windows that generates a variety of requests against computer files, partitions, or storage devices.

Here is an example for diskspd usage and description of options used in our testing:

```
> diskspd.exe -Suw -b4K -t16 -ag1,16,17,18,19,20,21,22,23,24,25,
26,27,28,29,30,31 -o32 -w100 -W600 -d300 #1
```

Table 1. Diskspd Options Used in Performance Testing

Option	Description
-Suw	Controls caching behavior. u: disable software caching w: enable writethrough (no hardware write caching)
-b4K	Size of block
-t16	Number of threads per target
-ag1,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31	Advanced CPU affinity – Assigns threads round-robin to the CPUs provided. g1 specifies processor group 1, and 16-31 are the core numbers within that
-o32	Number of outstanding I/O requests per target per thread
-w100	Percentage of write requests
-W60	Warm up time
-d300	Test duration
#1	Target (physical drive number)

To determine the physical drive number of the desired target SSD in Windows:

1. Open a command prompt window
2. Run “diskpart”. Running this command switches to diskpart’s command line interface
3. Type “list disk” to see the disk numbers and information

To determine the assigned NUMA node for each physical device in Windows:

1. Go to Control Panel
2. Open Device Manager
3. Under “Disk Drives”, right click on the device name and click on “Properties”.
4. Go to the “Details” tab and click on the “Property” field.
5. Scroll down the list and click on “NUMA”. The value provided is the NUMA node the device is assigned to.

Use the following table to interpret the color coding used in the commands below:

Color	Description
Red	Process group number
Green	8 CPUs on the same NUMA node the NVMe SSD is located on
Blue	8 CPUs located on an adjacent NUMA node to the NUMA node NVMe SSD is located on, in the same process group

Table 2. Samsung PM1725a 1.6TB (NUMA node 3, Physical Drive #1)

Operation	Command Line
Sequential Read	diskspd.exe -Suw -b128K -si128K -t16 -ag0,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31 -o32 -w0 -W600 -d300 #1
Sequential Write	diskspd.exe -Suw -b128K -si128K -t16 -ag0,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31 -o32 -w100 -W600 -d300 #1
Random Read	diskspd.exe -Suw -b4K -t16 -ag0,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31 -o32 -w0 -W600 -d300 #1
Random Write	diskspd.exe -Suw -b4K -t16 -ag0,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31 -o32 -w100 -W600 -d300 #1

Table 3. Micron 9200 3.2TB (NUMA node 6, Physical Drive #2)

Operation	Command Line
Sequential Read	diskspd.exe -Suw -b128K -si128K -t16 -ag1,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31 -o32 -w0 -W600 -d300 #2
Sequential Write	diskspd.exe -Suw -b128K -si128K -t16 -ag1,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31 -o32 -w100 -W600 -d300 #2
Random Read	diskspd.exe -Suw -b4K -t16 -ag1,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31 -o32 -w0 -W600 -d300 #2
Random Write	diskspd.exe -Suw -b4K -t16 -ag1,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31 -o32 -w100 -W600 -d300 #2

The affinity is set based on the locality of the cores to the NUMA node the physical drive is located on. Always run your tests long enough to counter the effect of any caching (use -W for warmup time).

Using fio for Performance Testing

fio is a tool that can spawn threads or processes doing a particular type of I/O operation as specified by the user. One way this tool can be leverage is to use a jobfile that invokes fio as follows under the command line:

```
> fio <jobfile>
```

The intended parameters for the job could be included in the jobfile. Below is an example of a jobfile used for random write operation:

```
[global]
ioengine=windowsaio
direct=1
iodepth=32
group_reporting=1
numjobs=16
ramp_time=600
runtime=300
[4k-ramdwr]
bs=4k
rw=randwrite
filename=\\.\PhysicalDrive1
```

Table 4. fio Options Used in Windows®

Option	Description
ioengine=windowsaio	Defines how the job issues I/O. Use Windows® native asynchronous I/O
direct=1	Use non-buffered I/O
iodepth=32	Number of I/O units to keep in flight against the file
group_reporting=1	Display per-group reports instead of per-job when **numjobs** is specified
Numjobs=16	Number of clones (processes/threads performing the same workload) of this job
ramp_time=600	fio will run the specified workload for this amount of time (in seconds) before logging any performance numbers. Useful for letting performance settle before logging results, thus minimizing the runtime required for stable results
runtime=300	Terminate processing after the specified number of seconds
bs=4k	Block size for I/O units
rw=randwrite	Type of I/O pattern
filename=\\.\PhysicalDrive1	This is how you define raw devices in Windows

Evaluating NVMe Performance using Red Hat Enterprise Linux® 7.4

Prerequisites

Evaluating NVMe performance on Red Hat Enterprise Linux® 7.4 requires a few modifications before benchmarks can be run. Below is a list of required prerequisites to ensure that benchmark results are accurate and repeatable. For more information on system configuration consult Appendix A.

SSD Preconditioning

Before running any benchmarks, it is crucial that you prepare the drive by "preconditioning" the drive. Preconditioning a drive is recommended to achieve sustained performance on a "fresh" drive. To better understand why SSD preconditioning is important, you can visit the following site:

http://www.snia.org/sites/default/education/tutorials/2011/fall/SolidState/EstherSpanjer_The_Why_How_SSD_Performance_Benchmarking.pdf.

The preconditioning process utilizes three steps to ensure that benchmarking results are accurate and repeatable. It is recommended to run that the following workloads be run with twice the advertised capacity of the SSD to guarantee that all available memory is filled with data including the factory provisioned area.

- Secure erase the SSD
- Fill SSD with sequential data twice
- Fill the drive with 4k random data twice

If you're running a sequential workload to estimate the read or write throughput, you may skip the last step, although it is not recommended.

Using fio for Performance Testing

fio is a tool that can spawn threads or processes doing a particular type of I/O operation as specified by the user. One way this tool can be leverage is to use a jobfile that invokes fio as follows under the command line:

```
> fio <jobfile>
```

The intended parameters for the job could be included in the jobfile. Below is an example of a jobfile used for random write operation:

```
[global]
ioengine=libaio
direct=1
iodepth=32
group_reporting=1
numjobs=16
ramp_time=600
runtime=300
[4k-ramdwr]
bs=4k
rw=randwrite
filename=/dev/nvme[0]n1 #0 is the device identifier here
```

Table 5. fio Options Used in Linux®

Option	Description
ioengine=libaio	Defines how the job issues I/O.
direct=1	Use non-buffered I/O
iodepth=32	Number of I/O units to keep in flight against the file
group_reporting=1	Display per-group reports instead of per-job when **numjobs** is specified
numjobs=16	Number of clones (processes/threads performing the same workload) of this job
ramp_time=600	fio will run the specified workload for this amount of time before logging any performance numbers. Useful for letting performance settle before logging results, thus minimizing the runtime required for stable results
runtime=300	Terminate processing after the specified number of seconds
bs=4k	Block size for I/O units
rw=randwrite	Type of I/O pattern
filename=/dev/nvme[device identifier] n1	This is how Linux® defines raw devices

Please note that to achieve higher IOPS performance in Linux, the numjobs parameter as well as the iodepth can be increased. You can also specify the NUMA nodes to be used by parameter “numa_cpu_nodes”. Always start with the CPUs that are located on the same NUMA node as the NVMe SSD device.

Performance Results

Sequential Throughput Performance

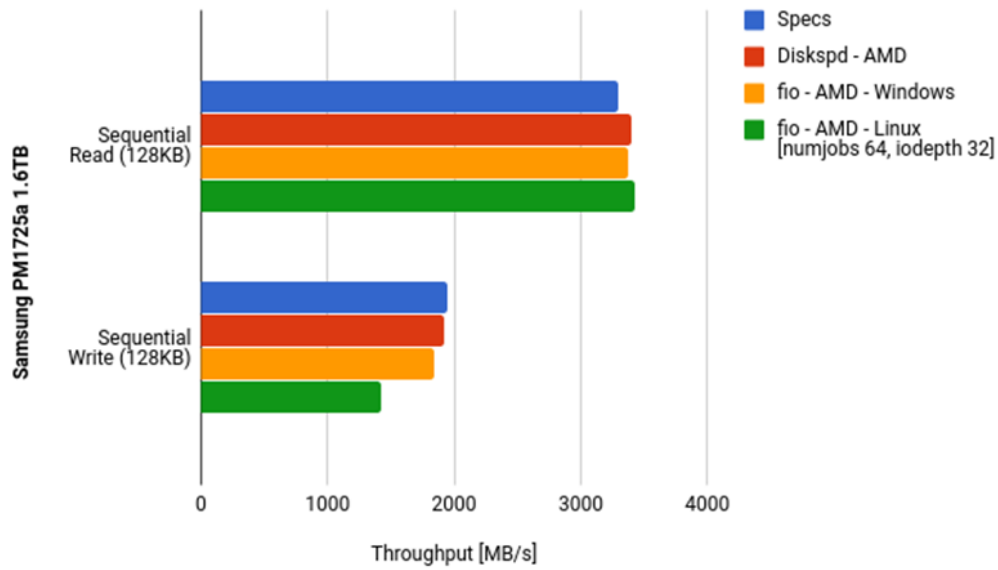

Figure 1. Sequential Throughput Performance on Samsung PM1725a 1.6TB NVMe SSD

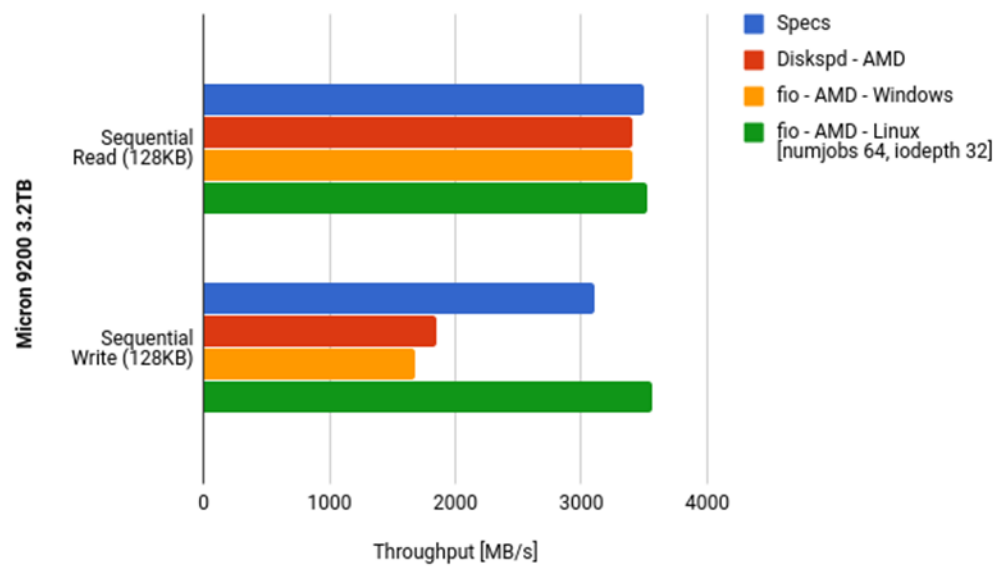

Figure 2. Sequential Throughput Performance on Micron 9200 3.2TB NVMe SSD

Random IOPS Performance

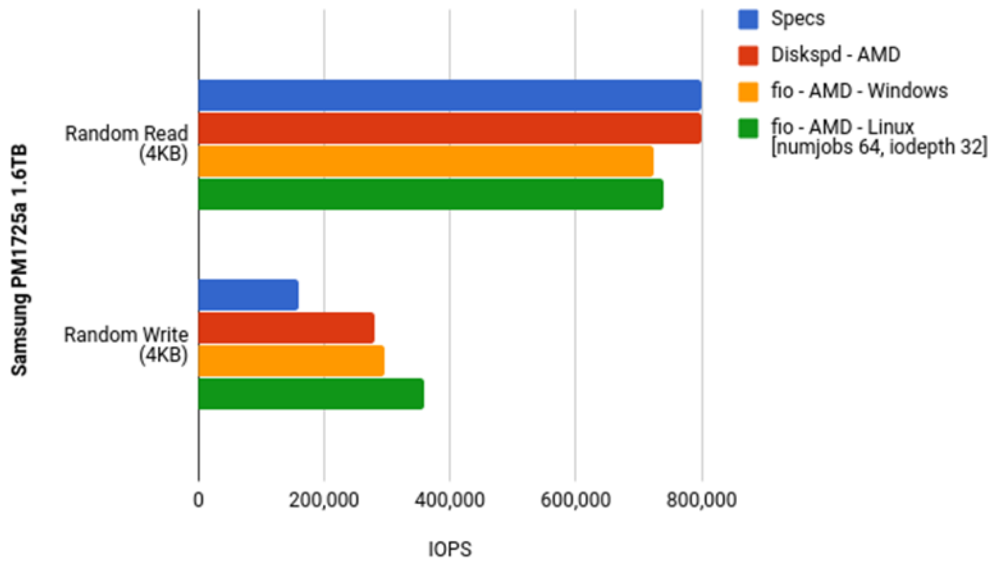

Figure 3. Random IOPS Performance on Samsung PM1725a 1.6TB NVMe SSD

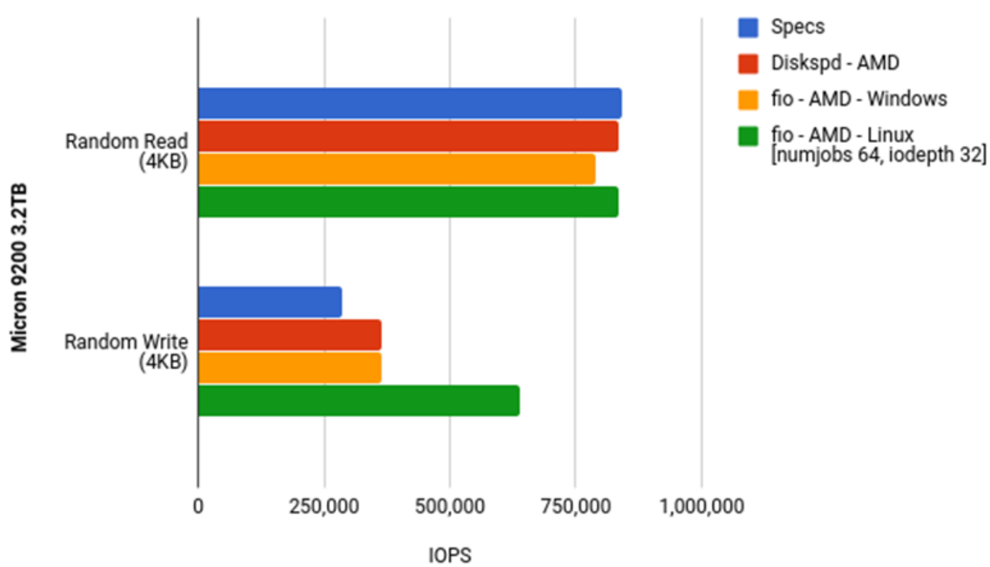

Figure 4. Random IOPS Performance on Micron 9200 3.2TB NVMe SSD

Appendix A Test System Configuration

Table 6. Configuration of the Systems Under Test (SUTs)

Platform	AMD	AMD
Processor	2x AMD EPYC™ 7601	2x AMD EPYC™ 7601
RAM	16x 16GB DRAM @ 2666MHz	16x 16GB DRAM @ 2666MHz
OS	Windows Server® 2016 DataCenter 64-bit version 10.0.14393	Red Hat Enterprise Linux® Server release 7.4 (Maipo)
NVMe SSD #1	Samsung PM1725a 1.6TB (NUMA node 3)	Samsung PM1725a 1.6TB (NUMA node 3)
NVMe SSD #2	Micron 9200 MAX 3.2TB (NUMA node 6)	Micron 9200 MAX 3.2TB (NUMA node 2)
Diskspd version	2.0.20a	N/A
fio version	3.6	3.1

Table 7. Required BIOS Settings for Testing

BIOS Settings	Desired Value
Simultaneous Multithreading (SMT)	Disabled