Technical Note

P320h/P420m SSD Performance Optimization and Testing

Introduction

This technical note describes how to optimize and test the performance of Micron's P320h and P420m solid state drives (SSDs).

This technical note should be used with Micron's *Best Practices for SSD Performance Measurement* technical marketing brief for specific instructions on testing both form factors (half-height, half-length [HHHL] and 2.5-inch) of the P320h and P420m SSDs.

Sections in this document include:

SSD vs. HDD Performance Testing

Unlike hard disk drives (HDDs) that use magnetic media to store user data, allowing data to be written and erased repeatedly to the same logical block addresses (LBAs), SSDs use NAND Flash media. NAND Flash media is a nonvolatile memory type that has an inherent limit on the number of write and erase cycles that can be performed on it during its lifetime.

As an SSD is written to and erased, the NAND media can eventually wear out over time and become unavailable for writes. This wearout behavior occurs more quickly if the same NAND addresses are regularly written to or erased. To prevent this, SSDs implement advanced wear-leveling algorithms in their NAND controllers that dynamically spread writes across different NAND addresses in the memory array to help maintain the endurance of the drive. Although these internal hardware algorithms can help maintain endurance, they also influence how an SSD performs. Therefore, an SSD, unlike an HDD, displays sensitivity to workload type, data pattern, and the state of the drive prior to a performance test.

The purpose of this technical note is to provide guidelines for optimizing the performance of the SSD and testing it to achieve repeatable, reliable results. This technical note also discusses specific system and drive performance adjustments that are unique to the P320h/P420m to enable you to achieve the best performance available on the drives.

Performance Testing Goals

Accuracy, consistency, and repeatability are the most important goals of any performance test. A brief overview of these goals is provided below; for more details, see Micron's *Best Practices for SSD Performance Measurement* technical marketing brief.

Accuracy

Results of a performance test should correctly represent the actual performance behavior of an SSD under a sustained workload, under specific test conditions. Inaccurate results can provide misleading and incorrect performance data and can generate invalid expectations of SSD performance when used in an application.

Consistency

During a performance test, system variables such as the operating system, BIOS, and hardware parameters (including processor type and number of cores), in addition to the SSD itself, should be consistent. Any variation in these parameters can affect both the accuracy and reliability of the performance results.

Reliability

Performance results must be repeatable. Performance data should achieve the same or similar results within a defined margin of error when the same workload is repeated on the same SSD using the same system. Performance data that varies from run to run provides an inaccurate and invalid measurement of what the actual performance of the SSD is with the specified workload.

Performance States

The state of an SSD prior to a performance test is important and affects the consistency and reliability of the output.

Figure 1 and Figure 2 illustrate actual performance of a P320h with a defined workload during different performance states.

Figure 1 shows the three performance states an SSD enters:

- Fresh-out-of-box (FOB)
- Transition
- Steady state

Figure 1: Performance States (4KB Random Write Performance)

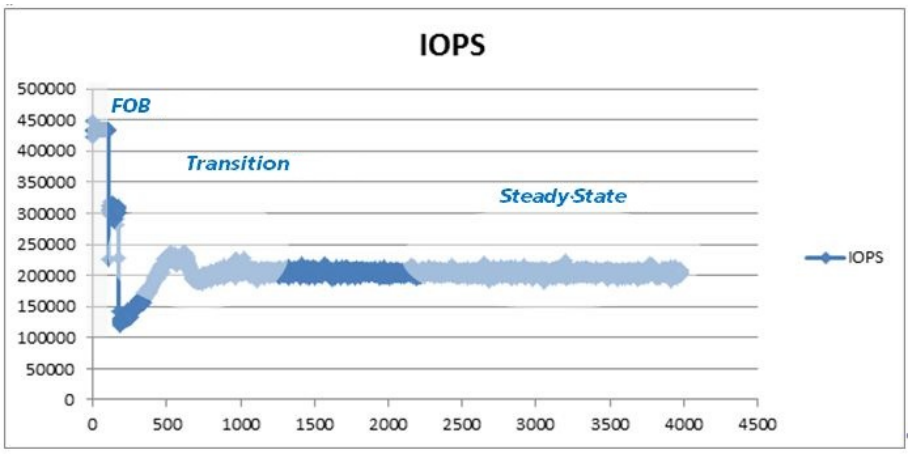
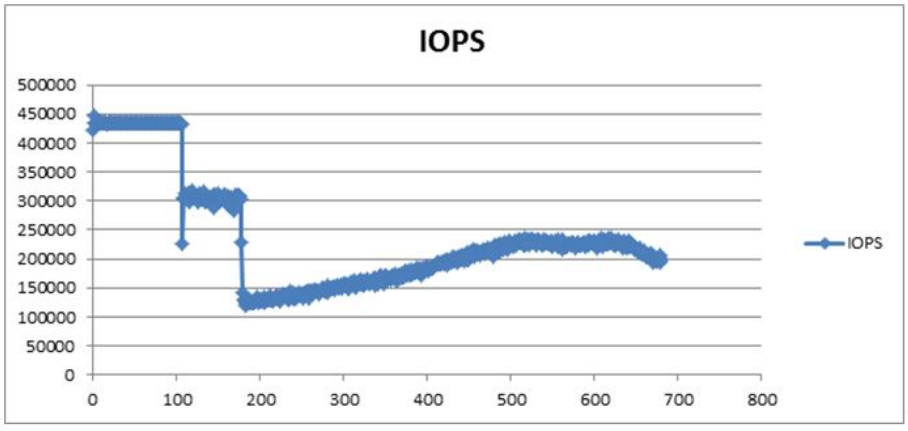


Figure 2: FOB and Transition State Detail (4KB Random Write Performance)



FOB State

The FOB state is defined as the performance state when I/O is run on the drive immediately after the drive has been initially shipped from the factory or secure erased (see Preparing the Drive for Testing (page 8) for descriptions of these terms). Typically, it is in this state where higher input/output operations per second (IOPS), or bandwidth numbers, are expected because the SSD's NAND management operations have not yet started.

In the FOB state, data received from the host is directly written to the SSD with a minimal amount of internal Flash block move operations. The SSD remains in this state for a finite amount of time before the performance eventually drops and the SSD moves to the transition state. The amount of time the SSD remains in the FOB state varies due to several factors, including capacity, workload type, transfer size (mixture of read/write requests), and NAND media type (single-level cell [SLC] or multilevel cell [MLC]).

Transition State

After an SSD is written to for a period of time, its performance in the FOB state eventually decreases and then fluctuates. (This behavior is illustrated in Figure 2 (page 3).) This fluctuation, or transition state, is due to the onset of NAND management operations occurring in the background, where incoming writes are periodically treated as read/modify/write operations in the NAND Flash media. From the host side, the SSD's latency response to incoming writes increases due to these background operations, resulting in performance drops. The amount of time the SSD spends in this state depends on factors similar to those described in the FOB state.

Steady State/Worst-Case Performance

Eventually, the performance fluctuations seen during the transition state should settle into a defined margin of variation, usually ±10% from a baseline value for an extended period of time. When this occurs, the SSD is in the steady state. In the steady state, the SSD achieves a balance between background NAND management operations that store data and the task of processing incoming write data from the host. Because the worst-case performance can be expected in this state, it can provide an accurate and repeatable measurement of the drive's performance.

Some SSD vendors quote FOB performance numbers because the performance is much higher in this state than other states. Micron does not quote performance numbers for our SSDs in this state because the SSD's worst case, steady state performance will be lower and steady state performance is a more realistic measure of the drive's true performance over time.

Micron conforms to the Storage Networking Industry Association (SNIA) Performance Testing Specification V1.0 for the test methodology that determines when the steady state has been reached, and Micron uses the steady state for measuring and reporting performance data for our SSDs.

Testing Requirements

This section describes the requirements for preparing a P320h/P420m for a performance test, including hardware, software, and temperature/airflow considerations.

Micron Driver

Download and install the latest Micron driver, which is included in the support pack available from www.micron.com. Detailed hardware and software requirements for the P320h/P420m are documented in the installation guides included with the support pack.

Hardware and Software

Verify that the system on which the performance test is being run meets the hardware and software requirements and recommendations (as noted) listed in Table 1.

Note: The P320h/P420m requires a 64-bit Windows or Linux operating system installed on a 64-bit processor platform. Refer to the product data sheets for the latest list of operating systems supported by the P320h/P420m. (Although you can use 64-bit operating systems that are not on the list.)

If you plan to use a non-supported Linux operating system (with the exception of CentOS, which is compliant with RHEL Kernel), you will need to build the Micron driver from source code and install it in your system. This can be done by extracting and making the driver from the files included in the *_src.rpm file in the Linux Driver subdirectory of the support pack. Before building the driver, be aware of the Linux kernel version requirement (2.6.25+) and make sure your kernel revision is equivalent or higher. Otherwise, you may encounter build or dependency errors when compliling the driver source code.

Performance data for the P320h/P420m available in the product data sheets reflect a maximum value; your performance could be equal to or less than the stated values, depending on your system's hardware capabilities. In general, higher performance has been observed on higher-frequency CPU platforms with faster DDR clock speeds.

Table 1: PCIe SSD Hardware and Software Requirements

Category	Description	Requirements
System	Type, BIOS	• Enterprise server-class system with Intel® Tylersburg, Sandy Bridge, or later chipset • Performance optimized legacy or UEFI BIOS
Driver	Micron Windows, Linux	• Latest Micron SSD driver (driver must be installed first before the SSD can be used in the target system) • Latest version available in the current Windows or Linux support pack
Firmware	Micron, Unified Image	• Latest SSD firmware included in the support pack (recommended, but not required)
Micron tools	Micron RealSSD™ Manager (RSSDM)	• Latest RSSDM version available in the support pack
Testing tools	IO stress	• Windows Iometer • Linux: FIO or Vdbench™ (FIO is recommended)

Table 1: PCIe SSD Hardware and Software Requirements (Continued)

Category	Description	Requirements
Operating system	Windows, Linux	• Windows Server 2012 R2 (x86-64), Hyper-V (x86-64) • Windows Server 2012 (x86-64), Hyper-V (x86-64) • Windows Server 2008 R2 SP1 (x86-64), Hyper-V (x86-64) • Windows 8, 8.1 (x86-64 and x86) • Windows 7 (x86-64 and x86) • Red Hat Enterprise Linux (RHEL) 5.5-5.10, 6.0-6.5 (x86-64) • SUSE Linux Enterprise (SLES) 11 SP1, SP2, SP3 (x86-64) • VMware 5.0, 5.1 (x86-64) • VMware 5.5 (inbox driver) • Citrix XenServer 6.1 • Ubuntu 12.04-12.04.3 LTS Server (64-bit)
Processor	CPU type, cores, cache size	• 64-bit processor • Single, dual, or higher number of processors (single processor is preferred) • Intel® Xeon® (Nehalem-EP) or later generation processor with four or more cores/processors • 12MB local cache (recommended) • Processors with clock speeds greater than 3 GHz (recommended for best performance) • Up to 8 CPU cores (logical + physical) with hyperthreading (recommended)
Memory	System RAM	• 6GB DRAM minimum (recommended)
Chipset	Type	• Intel® 5520 or later
PCIe slots	Type, speed, link width	• PCIe Gen2 x8 slot recommended (PCIe Gen1 slot can be used; however, drive performance will be impacted)
Power requirements	Slot power consumption and external supply	• Slots must be at least 25W PCIe-compatible • External power connection not required

Temperature/Airflow

To achieve optimum performance when testing the P320h/P420m, it is important to maintain its temperature below the maximum data sheet ratings and provide a minimum airflow within the limits specified in the data sheet. If these limits are exceeded, the drive can protect itself from damage using a feature called thermal throttling; however, performance will be reduced until the temperature falls below the throttling threshold temperature.

Thermal throttling protects the drive by progressively reducing (throttling) host I/O traffic received by the drive until the temperature falls below a preset threshold. After the temperature goes below this threshold, the drive's I/O performance is restored to original levels. The drive firmware monitors temperature by reading an on-die sensor within the NAND controller in the drive hardware. If the temperature continues to increase above the threshold, all I/O traffic will eventually be throttled and the drive will enter thermal shut down. After the drive enters thermal shut down, it will continue to be detected by the operating system, however, all I/O traffic to the drive will be disabled. When thermal shutdown occurs, the drive can be restored by powering down the system, removing it from the system and allowing it to cool, and then reinstalling it and powering on the system.

If thermal throttling begins during a performance test, much lower than expected or decreased performance will be seen. This behavior can generate an invalid performance measurement.

Thermal throttling and shutdown can be prevented by increasing the airflow around the drive through external fans and proper placement in a well-ventilated slot on the system motherboard.

To ensure data sheet requirements are met, an anemometer should be used to measure the airflow and temperature on both sides of the drive after it has been installed in the target system. The measurement must be taken in the system's normal operating environment with the system's cover on and after it has warmed up to its operating temperature. This process can take approximately 20 minutes.

Preparing the Drive for Testing

The goal of any SSD performance test is to prepare and test the SSD so that consistent and repeatable steady state results are achieved. To accomplish this, Micron recommends the following steps:

- Perform a secure erase
- Precondition the SSD

Perform a Secure Erase

Before running a performance test, erase all existing data on the P320h/P420m using the SECURE ERASE command to place the SSD in the FOB state. This can be accomplished using Micron's RealSSD Manager (included in the support pack).

Important: The SECURE ERASE command removes all data and files from the SSD. Make sure to back up the data on alternate media before performing a SECURE ERASE operation.

Precondition the SSD

After the drive is erased, fill it completely with data so that it reaches the preconditioned state. In this state, the SSD's LBA table has all of its entries filled with LBA and NAND addresses, enabling the SSD to perform reads quicker because addresses can be looked up from the LBA table instead of creating a new entry for each read access.

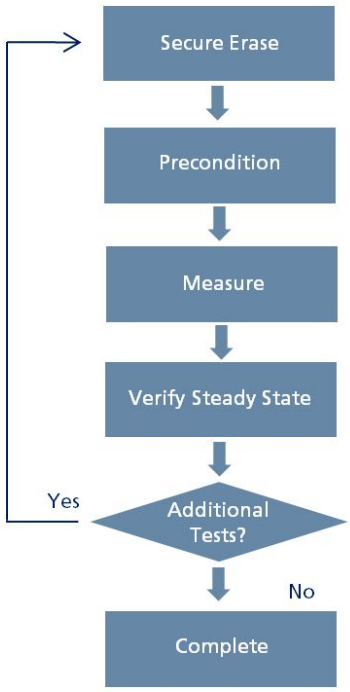
Important: Reads on an erased SSD may result in lower performance (approximately 50% decrease). This low performance is expected because internal LBA table entries are being populated, which increases READ latency.

Recommended Performance Testing Workflow

To achieve the performance testing goals outlined in this document, it is necessary to define and adhere to a test flow and methodology that takes into account the differences in performance of SSDs versus HDDs.

Figure 3 outlines Micron's recommended workflow for testing the performance of the P320h/P420m. This workflow implements a subset of tests defined in the SNIA Performance Testing Specification.

Figure 3: Performance Test Workflow



The secure erase, precondition, and steady state steps are described in the previous sections. The measure step refers to the recording of performance results at each predefined time interval, for a sequence of time intervals, until the steady state is reached. Measuring too soon at either the FOB or transition state can result in inaccurate or misleading data.

Note: This workflow applies to 100% write workloads and any mixed read/write workloads. If more than one test is required, repeat this process for each test. For 100% read tests of any workload size or type (random or sequential I/O), the secure erase and precondition steps need to be performed only once before the first read; they are not required on subsequent 100% read workload runs of different sizes or types.

Testing Performance in Windows

Micron's *Best Practices for SSD Performance Measurement* technical marketing brief defines a test flow in Windows using Iometer that can be used to test any SSD, including the P320h/P420m.

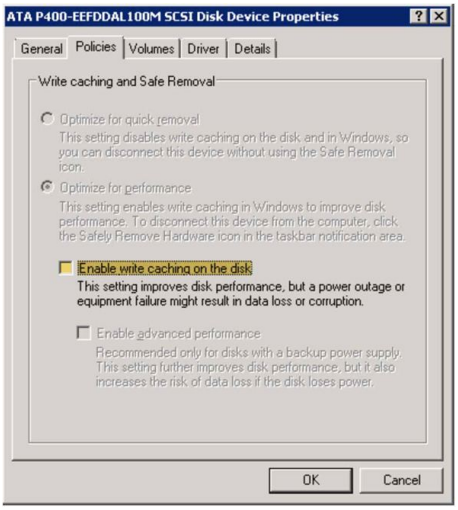
Follow the test flow documented in the technical marketing brief, noting the following exceptions for the P320h/P420m outlined in this section.

Write Cache Setting

The P320h/P420m has two write cache settings: one that can be set in Windows and one that is internal to the SSD.

Enabling or disabling the write cache setting in Windows (Figure 4) has no effect on the P320h/P420m; therefore, it is not necessary to change this setting.

Figure 4: Write Cache Setting



The write cache setting internal to the P320h/P420m can be enabled or disabled on the P320h only using RSSDM. However, be aware in the case of unexpected power loss, data loss may occur in the write cache if this setting is enabled.

The internal drive setting on the P420m is enabled by default and cannot be changed. The P420m supports power holdup protection, which protects data during an unexpected power loss.

Using Iometer

Iometer is the recommended tool to validate P320h/P420m performance in Windows operating systems. It is recommended to run Iometer on a raw drive; performance data with a drive in this format has been shown through internal testing to be better than a drive formatted for NTFS. (A raw drive is one that has only the master boot record (MBR) written to it by the Windows Disk Manager. While Iometer can be run with an NTFS-formatted drive, it takes some time for it to prepare drives prior to running a test.)

Iometer version 2006 is recommended. Equivalent performance tests with the same workload and test flow have shown some performance differences between version 2006 and 2008, with improved performance results reported on version 2006.

Iometer Queue Depth

The P320h/P420m has been designed to operate at peak performance at a queue depth of 256. (Other SSDs typically require a queue depth of 32.)

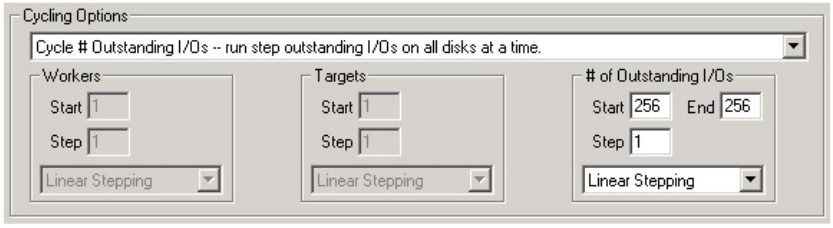
Set the P320h/P420m to a queue depth of 256 before running a preconditioning or I/O workload. The queue depth in Iometer can be set one of two ways:

- Setting Max Queue Depth (Auto Spawn Workers)
- Setting Queue Depth/Worker (Manually Spawning Workers)

Setting Max Queue Depth (Auto Spawn Workers)

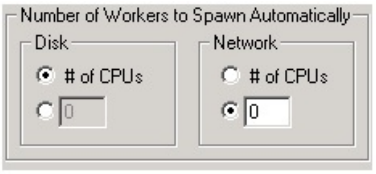
To automatically spawn workers, set the total queue depth to 256 workers on the Iometer Test Setup tab, following the sample in Figure 5.

Figure 5: Iometer Maximum Queue Depth Setting



Note: Performance may be impacted if your system contains fewer workers than the number of CPU cores in the system. Therefore, set the **Number of Workers to Spawn Automatically** section to match the number of CPU cores available in your system by selecting the **# of CPUs** option in the Disk subsection, as shown in Figure 6.

Figure 6: Iometer Worker Spawn Setting



Setting Queue Depth/Worker (Manually Spawning Workers)

To manually spawn four or eight workers and configure the queue depth per worker such that Equation 1 (EQ.1) is met, follow the steps below.

Number of Workers * Queue Depth/Worker = 256 (EQ.1)

1. For each worker, set the targets (under **Disk Targets**) for the disk under test.

   Note: The disk number under test can be found by using Windows Disk Manager and reviewing which of the disk numbers is the P320h/P420m under test.

2. For each worker, select the correct access specification (or workload type, size) from the list of available workloads or create your own.

   Note: If creating your own workload, make sure the I/Os are aligned on 4KB boundaries and not sector boundaries. (Figure 8 (page 13) shows a sample 4KB random write access specification.)

3. For each worker, select the number of outstanding I/Os (or queue depth) setting such that EQ.1 is satisfied.

   For example, for four workers, the queue depth setting per worker should be 64. For eight workers it should be 32, and so on.

4. Change the **Cycling Options** section in the Test Setup tab to **Normal; run all selected targets for all workers**.

Figure 7: Iometer Queue Depth/Worker Setting, Spawned Workers

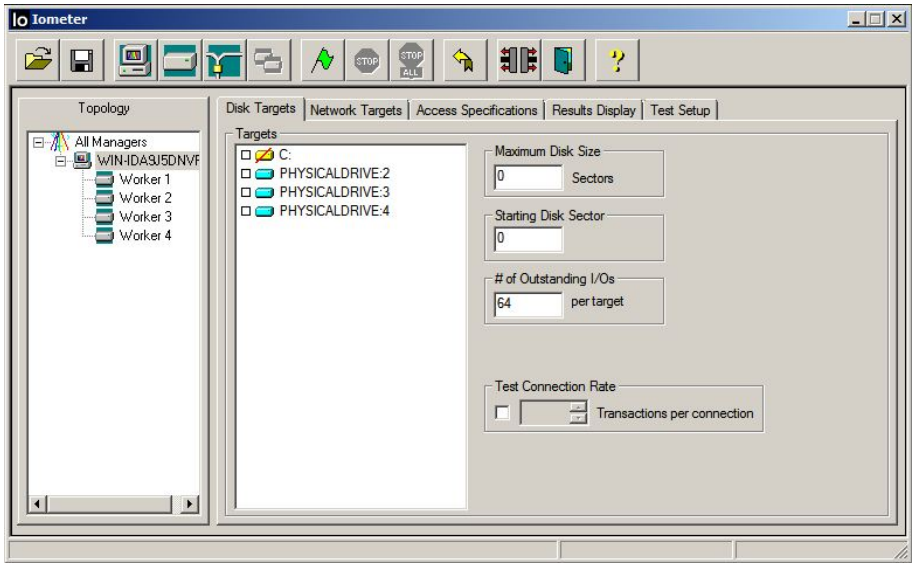
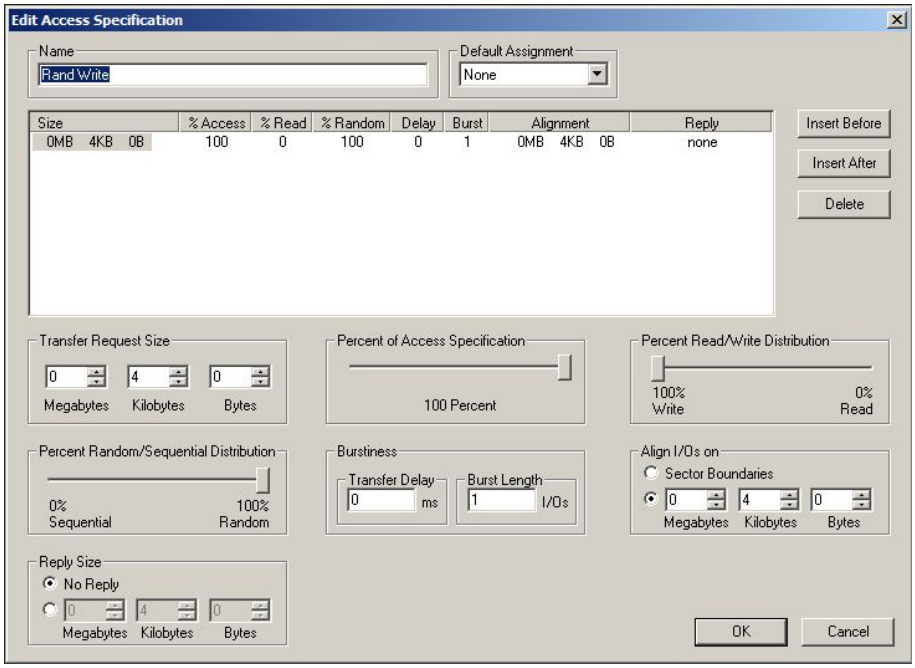


Figure 8: Iometer 4KB Random Write Access Specification



Other I/O Testing Tools in Windows

Several other testing tools are available in Windows to benchmark P320h/P420m performance; however, if you use one of these tools, be aware of the following:

- Make sure the tool has an option to adjust the queue depth setting to 256 prior to use; otherwise, performance results may not be optimized for the P320h/P420m.
- Make sure the tool is capable of performing write transfers that are aligned to 4KB boundaries. Like all Micron SSDs, the write performance of a P320h/P420m is adversely affected if unaligned writes are used.
- If the tool is capable of reporting latency data, verify the latency results are reported correctly. Very large latencies (greater than 10 seconds) have been observed to be erroneously reported on some tool versions. (The P320h/P420m is unlikely to generate latencies of this magnitude without the system reporting errors.)

Testing Performance in Linux

This section includes samples of pseudo code for Linux systems that can be used to implement the testing workflow defined in Recommended Performance Testing Workflow (page 9) for one P320h/P420m installed in a system.

Note: There may be other variations to the script code that could yield equivalent results.

If there are multiple SSDs in the system, run the SECURE ERASE command, precondition each drive, and then run the workload test on each drive individually, in order. You will likely need to modify the scripts described below to support this.

Secure Erasing the Drive

Run the SECURE ERASE command on each drive separately using RSSDM with the commands shown in the figure below.

Figure 9: Secure Erase Command

```
DRIVE_NUM="`rssdm -L | grep Drive\ Id | sed s/Drive.*:\ //g'"

yes | rssdm -X -p ffff -n $DRIVE_NUM
```

Note: The commands above will perform a secure erase on one drive in a system. If there are multiple drives in a system, enter a value to replace **$DRIVE_NUM** in the **-n** option for RSSDM.

The drive number can be extracted by running **rssdm –L** and filtering the first row output. The **yes** command will automatically complete the Y/N response that the RSSDM generates when running the SECURE ERASE command.

For these commands to execute, the bash shell path may need to be modified on the system to include the path to the RSSDM application appended to the system's $PATH variable.

Preconditioning the Drive

Run the **dd** command once on the drive, as shown in the figure below.

Figure 10: Preconditioning Command

```
dd if=/dev/zero of=$FILE_NAME oflag=direct bs=1M
```

The $FILE_NAME variable is the Linux device path of the P320h/P420m (typically **/dev/rssd<x>**); in this case, where x = a, b, and so on.

Depending on the capacity of the drives and the system, this command can take five minutes or longer to complete. Make sure to use direct IO (oflag=direct); otherwise, the command can take much longer.

If there are multiple P320h/P420m SSDs in a system, a different value will need to be used for the $FILE_NAME variable.

Note: If using a tool other than the dd command (for example, FIO), it is recommended to double fill the drive's capacity.

Running Workload Performance Test

Run a sample read/write test flow on a P320h/P420m using the suggested psuedo code shown below.

Figure 11: Psuedo Code for Sample Linux Performance Test Script

```
Thread_Count=8 # Could be lower as long as: Thread Count x IO_Depth = 256.
IO_Depth=32    # Suggest either 32,64 with the appropriate Thread Count adjustment.
Duration=1600  # Roughly the amount of time shown above.
               # Your value may be different depending on your system.
Run_Dir= ./SH/      # Optional, directory where scripts are run.
Log_Dir= ./Results/ # Directory where performance results are stored.
Test_Pattern = (Random, Sequential)

for Pattern_Type in Test_Pattern
do
 Block_Size = (512, 1024, 2048, 4096, 8192..)
 for size in Block_Size

  do
#   rwmix_values = (0,10,30,50,70,90,100) # Optional if you want mixed workloads.
#   Uncomment to use for mix in rwmix_values

#   do
    $Run_Dir/Secure_Erase_Drive # These are the commands or subroutines listed in SE-
CURE ERASE command above
    $ Run_Dir/Pre_Condition_Drive # These are the commands or subroutines listed in
preconditioning command above
    If Pattern_Type == Random # This will loop the Random tests for another block
size and workload

    then
    echo " Random IO Tests..."
    $Run_Dir/Random_Read $Thread_Count $IO_Depth $Pattern_Type $Block_Size $Duration
    $Run_Dir/Random_Write $Thread_Count $IO_Depth $Pattern_Type $Block_Size $Duration

    else # This will loop the Sequential test for another block size and workload
    echo " Sequential IO Tests..."
    $Run_Dir/Random_Read $Thread_Count $IO_Depth $Pattern_Type $Block_Size $Duration
    $Run_Dir/Random_Write $Thread_Count $IO_Depth $Pattern_Type $Block_Size $Duration

    Endif # Read/Write tests will repeat for new block sizes

#   Done # End optional IO mix loop

  Done # End inner do block size loop

Done # End outer do access type loop
```

The script consists of two do loops:

- Inner do loop—cycles the 100% read/write tests with different transfer sizes
- Outer do loop—repeats the transfer size tests for different I/O types, whether sequential or random

Note: An optional third do loop is available (commented out in the sample pseudo code above) to iterate the test for mixed read/write workloads.

Within the inner loop, a secure erase and preconditioning pass is performed before each test. The actual I/O test is referenced as a read/write subroutine in the pseudo code, whose function is to execute an I/O stress tool with the block size, thread count, io_depth, and duration parameters passed from the top-level script. These subroutines can be functions included in the main script or referenced in a calling script.

If only 100% read tests are being performed, the script flow described above would be different, in that the secure erase and precondition passes would not be required after the first read workload is executed. The script above would have to be modified to support this flow. These modifications are outside the scope of this document.

Using FIO

FIO for Linux is recommended as the I/O stress tool for the random and sequential read/write subroutines shown in the psuedo code in Running Workload Performance Test (page 15). Based on internal benchmarking tests Micron has performed to date, using FIO for Linux appears to show better performance for the same workload and hardware platform than Vdbench. If a different tool such as Vdbench is used, these scripts would need to be modified appropriately to support the command options of that tool.

Below is a sample command line for FIO for a 4KB random write test, which could be used as part of a subroutine in the psuedo code above.

Figure 12: Sample FIO Command Line

```
fio --filename=dev/rssda --bs=4K --runtime=$Duration -ioengine=libaio \
--iodepth=$IO_Depth --numjobs$Thread_Count --name=io_test --group_reporting \
--direct=1 --end_fsync=0 --invalidate=1 --norandommap --ramp_time=3000
--randrepeat=0 --refill_buffers --thread --gtod_reduce=1 --rw=randrw -rwmixwrite=100
--cpus_allowed=0-7 -nonrandommap 2> &1 | tee \
-i /$Log_dir/4K_Random_Write_$Thread_Count_$IO_Depth
```

The **>redirect** and **tee** commands allow FIO to output results to standard output and to save them to a log file, indicated by the log directory and defined by script variable $Log_dir.

Some parameters on the FIO command should be fixed and not passed through script parameters. For example, set the ioengine type to **linux asynchronous IO (libaio)** so that I/O is sent in asynchronous queued mode. This allows I/Os to be sent in parallel across all threads with a combined queue depth of 256. This is the optimal method for sending I/O to the P320h/P420m for best performance.

Also note:

- **-cpu_allowed** limits the number of CPU cores the I/O can be threaded across to 8 maximum. This is the recommended total (logical+physical) CPU core depth for testing with the drive.

  Note: The CPU number for each core in your system may be different. Use the **Numactl --hardware** command to list the CPU core associated with each processor node. Use the output from **dmesg** to ensure correct assignment of CPU cores to nodes.

- **-direct=1** indicates I/O will not be buffered by the system and will be sent directly to the drive. (Buffering I/O could increase host I/O latency and reduce performance.)
- **-group_reporting** summarizes the cumulative performance results across all I/O threads instead of itemizing results for each thread separately. This reduces the FIO output and improves readability.

Performance Tuning

Occasionally, even when following Micron's recommended test flow, performance may be lower than your application requirements. If this occurs, check the drive and system settings described in this section.

Drive Settings

Specific adjustments can be made to the following drive settings to improve performance:

- Power limiting
- Interrupt coalescing

Power Limiting

When enabled, the power limiting setting reduces IOPS processed by the drive to indirectly reduce power so the drive does not exceed the 25W PCIe slot power limit. Exceeding the slot's power limit may cause a temporary power supply brownout on some systems, which can cause the drive to become unresponsive until the system is power cycled. If power limiting is enabled on the drive, both random and sequential I/O are lowered by approximately 40%.

When power limiting is disabled, the drive is unlocked to achieve its highest performance potential. Power spikes that exceed the 25W PCIe slot limit may occur for up to 400µs, but the long-term RMS average power of the drive will still meet the 25W requirement.

The power limiting setting on the P320h/P420m HHHL SSD is disabled by default if the drive has been updated with the latest firmware in this or later support packs. If you are concerned about power draw, you can enable the power limiting setting on the P320h/P420m HHHL SSD.

Note: The power limiting setting on the P320h/P420m 2.5-inch SSD is disabled by default and cannot be changed.

To enable power limiting on a P320h/P420m HHHL SSD:

1. Install the latest support pack available from the Micron web site, which includes RealSSD Manager.
2. Verify the power limiting setting by running the following command at the command line:

   rssdm -L -d

3. To enable power limiting, enter the following command:

   rssdm -M -p 1 -n <Drive_id>

   <Drive_id> is the number on the first row when running rssdm -L.

   A message appears instructing you to power cycle the system for the changes to take effect.

4. Power cycle the system again to allow the new power limiting setting to take effect.
5. Verify that the power limiting setting has been changed by running the following command:

   rssdm -L -d

Interrupt Coalescing

The P320h/P420m has a unique option that can improve performance by coalescing the command response of several outstanding NCQ WRITE commands into a single interrupt instead of separate acknowledge interrupts for each WRITE command. In the latter case, commands that are not coalesced can increase CPU utilization, particularly on a specific core that is configured by the operating system to handle the interrupt response. This can slow down the system and reduce performance.

When commands are coalesced, there are fewer interrupts for the CPU to process; however, the result is increased command latency.

Interrupt coalescing is enabled by default on the P320h/P420m; there are three options to configure the performance of the P320h/P420m with this setting, as shown in Table 2.

Figure 13 and Figure 14 show some performance data comparing the different interrupt coalescing options.

Table 2: Interrupt Coalescing Settings

Interrupt Coalescing Value	Description
0xD200F	Balanced setting that trades high IOPS for lower latency and is suitable for the widest range of queue depths (default).
0xD801F	Optimized for IOPS; useful if you have a high queue depth (greater than 128) workload. More NCQ writes will be posted and acknowledged with a single interrupt; highest latency.
0xD020F	Optimized for throughput; suitable for lower queue depth (less than 32) workloads; lowest latency.

Figure 13: P320h Performance with Interrupt Coalescing Settings

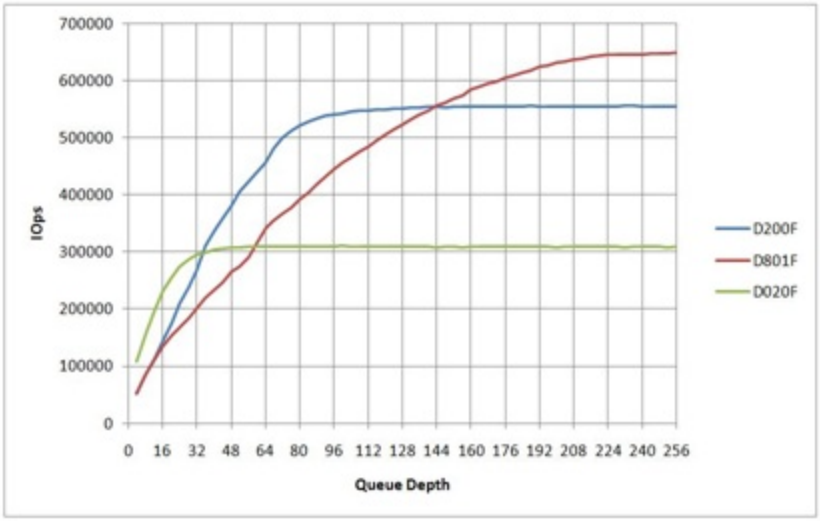
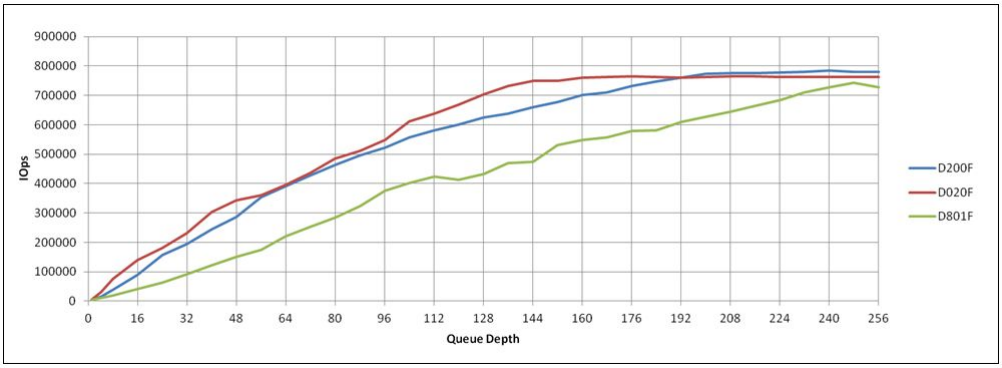


Figure 14: P420m Performance with Interrupt Coalescing Settings



To configure interrupt coalescing settings to one of the values in Table 2, use the RSSDM command line interface (CLI):

1. In the command prompt, run:

   Rssdm-M-i<intrp_coalesce_value> -n<drive_id>

   Where <intrp_coalesce_value> is one of the three values listed above in Table 2. Do not use any other value; unexpected results may occur.

2. After the command completes, a message displays indicating that a power cycle is required.
3. Power cycle the system.
4. When the system reboots again, type the following in the command prompt: **rssdm –L –d**
5. Verify the new interrupt coalescing setting has taken effect.

Note: The interrupt coalescing setting can also be changed in the RSSDM GUI. A power cycle is still required.

System Settings

Specific adjustments can be made in the system BIOS to improve drive performance, if the system BIOS supports these adjustments.

For example, Micron's drive testing has shown that with dual-processor configurations, one CPU should be used for I/O and the other for interrupt processing for best performance. For a single CPU system, adjusting core allocation such that the core used for interrupt processing is different than the core used for I/Os can optimize performance.

Micron's *How System Settings Impact PCIe SSD Performance* white paper documents these performance differences and provides data and recommendations to optimize drive performance by adjusting CPU and core affinity on single and multiple CPU systems running the Linux operating system.

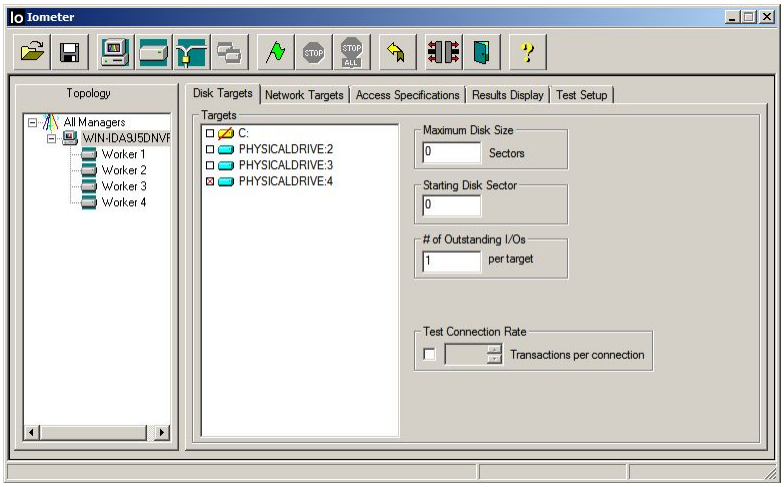
For Windows, the following section explains how to adjust CPU core affinity.

Setting CPU Affinity in Windows with Iometer

1. Review Micron's *How System Settings Impact PCIe SSD Performance* white paper and your system configuration to determine which slot and CPU will process both I/Os and interrupts.
2. If your system contains more than one drive, open a separate Iometer instance for each drive.

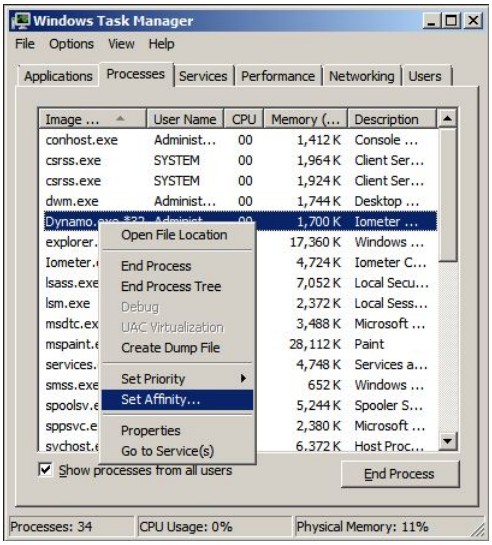
   Figure 15 shows the Iometer main window.

Figure 15: Iometer Manager



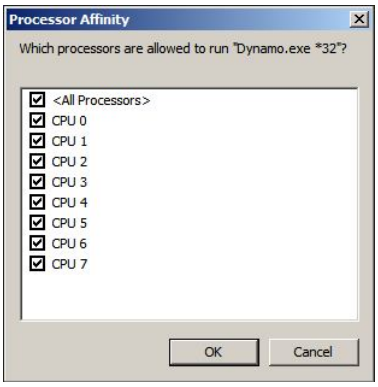
3. Open the Windows Task Manager and select **Dynamo.exe** in the Process tab.
4. Right-click on Dynamo.exe and select **Set Affinity**, as shown in Figure 16.

Figure 16: Setting Affinity in Windows Task Manager



5. Assign the Dynamo process to specific CPU cores, unchecking the cores that do not apply, as shown in Figure 17.

Figure 17: Setting Processor Affinity



6. Repeat Step 5 for the next Dynamo process, making sure that core assignments do not overlap.

   Note: These steps do not have to be repeated if the same system has more than one drive.

Conclusion

This technical note provides a performance testing methodology and recommendations tailored to the P320h/P420m to help you achieve optimal performance results in your system.

This document is intended to be a work in progress with additional updates planned whenever new information regarding optimizing performance for the P320h/P420m becomes available. Check the Micron website (www.micron.com) for updates.

References

- SNIA V1.0 Performance Testing Specification
- PCIe CEM Specification V2.1
- Micron's Best Practices for SSD Performance Measurement Technical Marketing Brief
- Micron's How System Settings Impact PCIe SSD Performance White Paper
- P320h/P420m HHHL and 2.5-inch data sheets
- P320h/P420m HHHL and 2.5-inch installation guides
- RealSSD Manager (RSSDM) User Guide

Revision History

Rev. B – 04/14

- Added information for the P420m

Rev. A – 10/12

- Initial release

8000 S. Federal Way, P.O. Box 6, Boise, ID 83707-0006, Tel: 208-368-3900
www.micron.com/productsupport Customer Comment Line: 800-932-4992