

# ezFIO

## Version 1.0

### User Guide



ezFIO Test Progress

Testing Drive: \\.\PHYSICALDRIVE1, Virtual Disk(UNKNOWN), 4GB

Current Test: Sustained Multi-Threaded Random Read Tests by Block Size, BS=131072

Current Test Runtime: 00:01:53

Total Test Runtime: 00:41:09

Access Pattern	Write %	Block Size	Queue Depth	IOPS	Bandwidth (MB/s)	Latency (us)
Sequential Preconditioning						
Seq Pass 1	100%	128K	256	DONE	DONE	DONE
Seq Pass 2	100%	128K	256	DONE	DONE	DONE
Sustained Multi-Threaded Sequential Read Tests by Block Size						
Preparation						
Seq	0	512b	64	103,806	50.7	593.6
Seq	0	1K	64	101,267	98.9	611.8
Seq	0	2K	64	86,476	168.9	719.3
Seq	0	4K	64	77,850	304.1	806.6
Seq	0	8K	64	53,457	417.6	1187.5
Seq	0	16K	64	30,997	484.3	2060.0
Seq	0	32K	64	15,903	497.0	4021.3
Seq	0	64K	64	8,200	512.5	7799.1
Seq	0	128K	64	4,120	515.0	15520.1
Sustained Multi-Threaded Random Read Tests by Block Size						
Preparation						
Rand	0	512b	256	91,415	44.6	2791.7

Open Graphs Spreadsheet

**Document Number:** 61600-00235-102

**Document Version:** 1.2

**Software Version:** 1.0

**Revision Date:** February 19, 2016

1<sup>ST</sup> Edition (Document Version 1.2)

February 19, 2016

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: HGST, INC., PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer or express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of this publication. HGST may make improvements or changes in any products or programs described in this publication at any time.

It is possible that this publication may contain reference to, or information about, HGST products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that HGST intends to announce such HGST products, programming, or services in your country.



HGST does not maintain any technical information about this product. ezFIO is distributed under the GNU Public License V2.0 or later.

HGST may have patents or pending patent applications covering the subject matter in this document. The furnishing of this document does not give you any license to these patents.

© 2016 HGST, Inc., All rights reserved.

## Conventions

The following icon and text conventions are used throughout this document to identify additional information of which the reader should be aware.

Conventions		Description
CAUTION		This icon denotes the use of extreme caution and the user must exercise good judgment according to previous experience before advancing to the next procedure. The icon also indicates the existence of a hazard that could result in equipment or property damage, or equipment failure if the instructions are not observed.
NOTE		This icon identifies information that relates to the clarification of instructions and highlights points of interest.
<b>Bold.</b>	<b>Text</b>	Used to indicate <b>important technical notes</b> .
<b><i>Bold Italic</i></b>	<b><i>Text</i></b>	Used to indicate <b><i>critical instructions</i></b> .
<i>Light Blue Italic</i>	<i>Text</i>	Used to indicate a <i>hyperlink</i> or “ <i>jump</i> ” to a related <i>section</i> or <i>subsection</i> .

## Revision History

Revision	Date	Page(s)	Description
-101	02/05/2016	All	Initial release.
-102	02/19/2016	Title	Struck warning statement on Title page.
		8	Corrected two minor typographical errors.
		19	Figure 9; corrected figure text.
			Figure 10; corrected figure text.
		20	Section 5.9 struck. Obsolete specification.

## GNU Public License

ezFIO is distributed under the GNU Public License V2.0 or later. Full terms are included in the "LICENSE" file included with the distribution. Below is the abridged notice:

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either Version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

## FIO Moral License

While FIO is not included in the ezFIO distribution, it obviously relies on FIO to perform the actual testing. Below is FIO's "moral-license" file, as requested by Jens Axboe, the author:

As specified by the COPYING file, fio is free software published under Version 2 of the GPL License. That covers the copying part of the license. When using fio, you are encouraged to uphold the following moral obligations:

- If you publish results that are done using fio, it should be clearly stated that fio was used. The specific version should also be listed.
- If you develop features or bug fixes for fio, they should be sent upstream for inclusion into the main repository. This is not specific to fio; it is a general rule for any open source project. It is just the right thing to do. Plus it means that you do not have to maintain the feature or change internally. In the long run, this is saving you a lot of time.

I would consider the above to fall under "common courtesy", but since people tend to have differing opinions of that, it does not hurt to spell out my expectations clearly.

## Contributions

ezFIO is available on GitHub hosting, under the URL <https://github.com/earlephilhower/ezfio>, and may be forked or contributed to using the GitHub standard pull request branching operations.

# Table of Contents

<b>1. Introduction .....</b>	<b>8</b>
1.1 Overview .....	8
1.2 Why Sustained Mode? .....	8
1.3 Why use FIO? .....	8
1.4 Making Test Results Easy To Use .....	8
1.5 System Requirements .....	9
1.5.1 Windows 64-Bit .....	9
1.5.2 Linux .....	9
1.5.3 Installation .....	9
<b>2. Using ezFIO under Windows .....</b>	<b>10</b>
2.1 Overview .....	10
2.2 Using the ezFIO GUI .....	10
2.3 Using the ezFIO CLI .....	12
<b>3. Using ezFIO under Linux .....</b>	<b>14</b>
3.1 Overview .....	14
3.2 Running the Script .....	14
<b>4. Test Sequence .....</b>	<b>16</b>
4.1 Overview .....	16
4.2 Sequential .....	16
4.3 Random .....	16
<b>5. Interpreting the Test Results .....</b>	<b>17</b>
5.1 Overview .....	17
5.2 System and DUT Summary .....	17
5.3 Sustained Mixed Read-Write Test .....	17
5.4 Queue Depth .....	18
5.5 4K Random Read Performance with Varying Queue Depths .....	18
5.6 4K Pure Random Writes .....	19
5.7 Random Read Performance / Sequential Reads / Identical Block Sizes .....	19
5.8 Full Write Workloads with Varying Block Sizes .....	19
5.9 Raw FIO Output .....	20

## List of Figures

Figure 1: ezFIO Drive Selection Window .....	10
Figure 2: Verifying the ezFIO Test .....	11
Figure 3: The ezFIO Test Progress Window .....	11
Figure 4: System and DUT Summary .....	17
Figure 5: Sustained Mixed Read-Write Test .....	17
Figure 6: IOPS vs. Queue Depth .....	18
Figure 7: 4K Random Read, Queue Depth and Latency .....	18
Figure 8: 4K Pure Random Writes .....	19
Figure 9: Sequential vs. Random Read Performance by Block Size .....	19
Figure 10: Sequential vs. Random Write Performance by Block Size .....	19

# 1. Introduction

## 1.1 Overview

ezFIO is a script-based tool optimized for demonstrating the sustained performance over a series of varying workloads for NVMe storage devices. It utilizes the cross-platform open-source tool FIO (FIO can be found at <http://www.github.org/>) in an easy to use GUI (Windows) or CLI (Windows and Linux).

## 1.2 Why Sustained Mode?

The performance of most NVMe device can vary due to the underlying technology and the storage methodology used to write data to the media. For example, NAND flash devices require the implementation of *garbage collection* to manage data operations on flash chips that do not allow for the updating of existing data. Also, *preconditioning* or *seasoning* is a method often employed to ensure that the devices under test are fully utilized and will require garbage collection to operate. This simulates, as closely as possible, the long term performance of these devices better than simple, immediate testing.



*The preconditioning operations can require substantial time to complete while the write performance, size, and current state of the device are tested. Preconditioning stages can require some 15 minutes on smaller, faster devices, but up to several hours on larger, multi-TB flash devices.*

## 1.3 Why use FIO?

ezFIO is designed to track sustained performance by performing a series of scripted tests that include random and sequential preconditioning. It also operates at the lowest level possible on the raw device to ensure that file system overhead is removed and that the true device characteristics can be obtained.

FIO was chosen because of its power and cross-platform capabilities; it can run on everything from embedded ARM Linux systems to the latest Microsoft server and desktop operating systems. Because FIO is so powerful it is also often difficult to configure and run properly. ezFIO implements a predefined sequence of operations and manages the selection of the appropriate command line options.

Also, because many users are unfamiliar with the concept of sustained performance, they do not properly precondition the NVMe drives prior to running FIO to obtain realistic, long-term performance rates, resulting in misleading outcomes that do not reflect real-world performance.

## 1.4 Making Test Results Easy To Use

ezFIO will generate a simple OpenDocument Spreadsheet (ODS) after the test run that can be imported into Microsoft Excel, OpenOffice Calc, or any other spreadsheet program. The spreadsheet will include the raw test output and a series of graphs that show the performance characteristics of the drive over different I/O types, queue depths, block sizes, and read/write ratios. These graphs present the performance data in an intuitive format.

All tests are run using the FIO tool, and all executed test command lines and results are preserved in intermediate files for later examination if needed. As ezFIO is implemented as scripting languages, it is easy for users to examine and modify. All actual IO testing is performed by FIO, so there is no performance penalty often associated with scripting environments.



## 1.5 System Requirements

### 1.5.1 Windows 64-Bit

ezFIO for Windows is written in PowerShell and requires the following:

- Windows Server 2008 R2, Windows Server 2012, or Windows Server 2012 R2.
- Administrator privileges are required to allow raw device access by FIO.
- Microsoft .Net Framework 4.5 (<https://www.microsoft.com/en-us/download/details.aspx?id=42643>)
- FIO for Windows, Version 2.2 or later, is required. The port of FIO to Windows is available at <http://www.bluestop.org/fio/>.

The following computer settings should be enabled to ensure the highest performance:

- Screen Saver (desk.cpl,screensaver,@screensaver) – Disable
- Power Management (powercfg.cpl) – High Performance



If Windows should report, "ezFio.ps1 cannot be loaded because running scripts is disabled on this system...", run the following command line to enable execution of local PowerShell scripts:

```
Set-ExecutionPolicy -scope CurrentUser RemoteSigned
```

### 1.5.2 Linux

ezFIO for Linux is written in Python 2.7 and requires the following:

- Linux Distribution that includes a Python interpreter.
- Root privileges are required to allow raw device access by FIO.
- FIO for Linux with asynchronous (libaio) support (often available from distribution repositories such as Debian, Ubuntu or EPEL, or compiled from sources available at <https://github.com/axboe/fio>).



When building FIO it is important to ensure that `libaio-devel` is installed prior to running `make` to include the asynchronous IO required for testing.

### 1.5.3 Installation

Since ezFIO is a script, the user only needs to unpack the ZIP (Windows) or tarball (Linux) archive file.

- For the Windows ezFIO implementation, the ZIP archive contains `ezfio.ps1` and `ezfio.bat` (a wrapper to start `ezfio.ps1` with the appropriate administrator privileges).
- For the Linux ezFIO implementation, the tarball archive contains `ezfio.py`.
- An OpenOffice ODS template, `original.ods`, is the test results spreadsheet template.

## 2. Using ezFIO under Windows

### 2.1 Overview

ezFIO for Windows consists of GUI (Graphical User Interface) and CLI (Command Line Interface) components. The user should first ensure that the device under test does not have any drive letters assigned to it. The Disk Management tool can be launched by running “diskmgmt.msc” from the Start menu (Start -> Run) and removing any mounted volumes from the drive that is to be tested.



Make sure that FIO for Windows is installed before running ezFIO, otherwise the script will prompt the user to install FIO.

### 2.2 Using the ezFIO GUI

1. Double-click the **ezfio.bat** script. A UAC window may appear requesting Administrator privileges. Once the privileges are granted, the ezFIO Drive Selection window will appear.

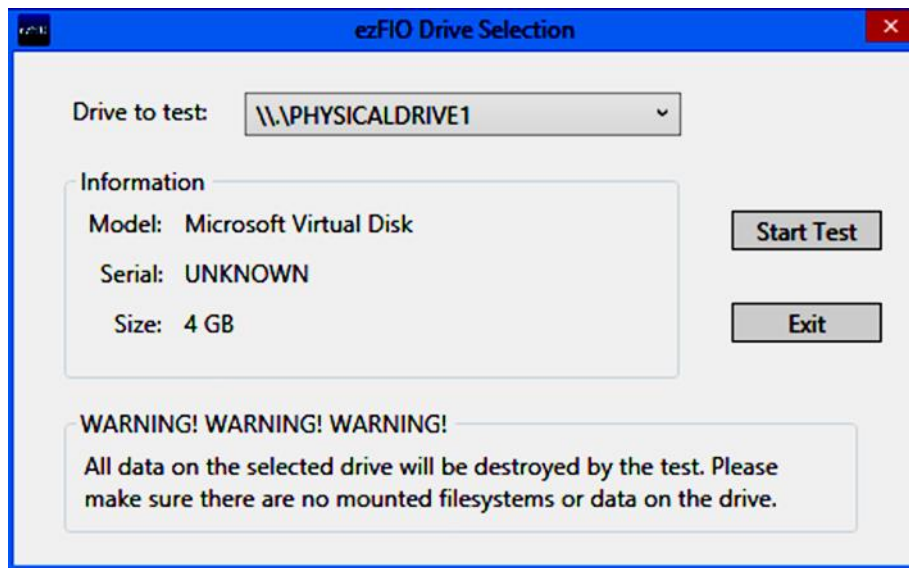


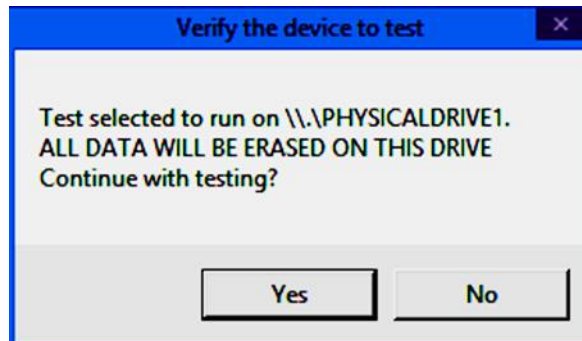
Figure 1: ezFIO Drive Selection Window

2. Select the NVMe device to test from the **Drive to test** drop-down list. The **Information** area below the listing will show the model name, serial number, and raw size of the selected drive.



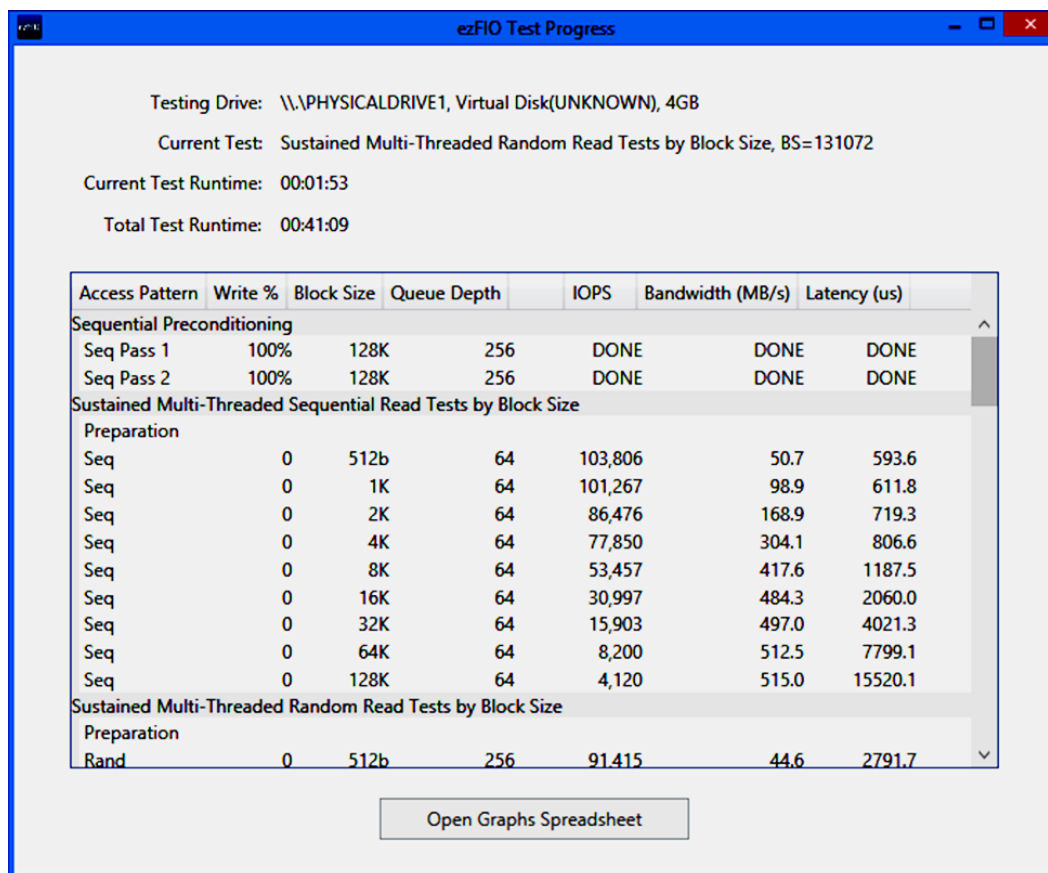
*Make sure that the correct device is selected for testing before clicking the **Start Test** button.*

- Click the **Start Test** button. The user is prompted to confirm or to abort the operation. Click **Yes** to continue or **No** to abort the test.



**Figure 2: Verifying the ezFIO Test**

- If the user selects **Yes**, the testing window will appear and the ezFIO test suite will begin running in the background as shown below.



**Figure 3: The ezFIO Test Progress Window**

The Testing Drive, Current Test, Current Test Runtime and Total Test Runtime will indicate the drive under test, the test that is currently in progress, the run time of the test, and the test duration respectively. Because this is a sustained-performance test, there will be multiple preconditioning passes that may require 30 minutes to several hours, depending on the NVMe device size and write performance.



*Do not perform any other tasks on the server while the tests are running to ensure repeatable measurements. Any additional CPU or I/O loads may produce incorrect test results.*

The lower part of the window lists the tests to be performed and the sequence order of the tests. As the tests complete the results will be listed in the window. While it is possible to copy and paste directly from this view, it is recommended to only use the final spreadsheet results to ensure that all the data was collected.

A notification will appear once the tests are complete and the **Open Graphs Spreadsheet** option will be enabled. Click the **Open Graphs Spreadsheet** button to load the ODS file into a spreadsheet application. The spreadsheet file will be created in the directory in which ezFIO was installed. The file will have a timestamp name assigned to it for easy identification.

### 2.3 Using the ezFIO CLI

The user may also choose to run an unattended test using the ezFIO CLI. Start a PowerShell window using administrator privileges, navigate to the directory containing `ezfio.ps1`, and run:

```
.\ezfio.ps1 -drive <#> [-yes]
```

Where <#> is the physical drive number of the device under test. To list the physical device numbers, run `ezFIO.ps1` with the `-h` (-help) option:

```
PS C:\Users\Administrator\Desktop\ezfio> .\ezfio.ps1 -h
ezfio V0.9, an in-depth IO tester for NVME devices
WARNING: All data on any tested device will be destroyed!
```

Usage:

```
.\ezfio.ps1 -drive <PhysicalDiskNumber> [-util <1..100>]
EX: .\ezfio.ps1 -drive 2 -util 100
```

PhysDrive is the ID number of the \\PhysicalDrive to test

Usage is the percent of total size to test (100%=default)

Physical disks:

```
0. DELL PERC H710 SCSI Disk Device, Serial: 004b952204d9, Size: 500GB
1. STAASSDModel SCSI Disk Device, Serial: S0018C5, Size: 800GB
2. NVMEDevel SCSI Disk Device, Serial: V00023, Size: 4849GB
3. NVMEModel2 SCSI Disk Device, Serial: R01020, Size: 4849GB
```



If the `-yes` option is not specified, the script will give the user a final chance to abort the test before continuing; otherwise, type `yes` and press Enter to start the test.

-----  
WARNING! WARNING! WARNING! WARNING! WARNING! WARNING! WARNING!

THIS TEST WILL DESTROY ANY DATA AND FILESYSTEMS ON \\.\PhysicalDrive6

Please type the word "yes" and hit return to continue, or anything else to abort: <yes>

The test will then commence, with status output going to the console. On completion, the test will show the path to the generated ODS file containing all test results.

...

---Sustained Perf Stability Test - 4KB Random 30% Write for 20 minutes---

Sustained Perf Stability Test - 4KB Random 30% Write for 20 minutes 755.10...

---Sustained 4KB Random Write Tests by Number of Threads---

Sustained 4KB Random Write Tests by Number of Threads, Threads=1 26.19...

Sustained 4KB Random Write Tests by Number of Threads, Threads=2 41.54...

Sustained 4KB Random Write Tests by Number of Threads, Threads=4 68.34...

Sustained 4KB Random Write Tests by Number of Threads, Threads=8 98.84...

Sustained 4KB Random Write Tests by Number of Threads, Threads=16 138.84...

Sustained 4KB Random Write Tests by Number of Threads, Threads=32 188.57...

Sustained 4KB Random Write Tests by Number of Threads, Threads=64 220.39...

Sustained 4KB Random Write Tests by Number of Threads, Threads=128 229.67...

Sustained 4KB Random Write Tests by Number of Threads, Threads=256 219.79...

---Sustained Multi-Threaded Random Write Tests by Block Size---

Sustained Multi-Threaded Random Write Tests by Block Size, BS=512 10.18...

Sustained Multi-Threaded Random Write Tests by Block Size, BS=1024 18.89...

Sustained Multi-Threaded Random Write Tests by Block Size, BS=2048 31.57...

Sustained Multi-Threaded Random Write Tests by Block Size, BS=4096 218.45...

Sustained Multi-Threaded Random Write Tests by Block Size, BS=8192 240.91...

Sustained Multi-Threaded Random Write Tests by Block Size, BS=16384 216.23...

Sustained Multi-Threaded Random Write Tests by Block Size, BS=32768 251.16...

Sustained Multi-Threaded Random Write Tests by Block Size, BS=65536 254.44...

Sustained Multi-Threaded Random Write Tests by Block Size, BS=131072 381.38...

COMPLETED! Output file:

C:\Users\Administrator\Desktop\ezfio\_results\_2222GB\_12cores\_2500MHz\_PhysicalDrive6\_TM32\_2016-01-29\_13-08-21.ods

## 3. Using ezFIO under Linux

### 3.1 Overview

ezFIO is a Python 2.7 script that requires an unmounted `/dev/node` on which to operate. The user should make sure that no mounted filesystems or user data is present on the device under test, as ezFIO will overwrite all the data.

### 3.2 Running the Script

Make sure that FIO is installed and accessible via the PATH system variable. There are Linux distributions that have precompiled versions found in the online repositories, or the user may compile the latest from scratch. ezFIO can use the `sdparm` utility (included in most online depositories) to extract the NVMe model names and serial numbers for tested devices. If `sdparm` is not installed, then these will need to be manually entered after the test.

Run ezFIO as a root user, either via `sudo ./ezfio.py` or with root privileges. The user can run the script without any parameters or use the `-h` (`--help`) option to print the usage help:

```
[root@tm33 ezfio]# ./ezfio.py -h
usage: ezfio.py [-h] --drive PHYSDRIVE [--utilization UTILIZATION] [--yes]
```

A tool to easily run FIO to benchmark sustained performance of NVME and other types of SSD.

optional arguments:

<code>-h, --help</code>	show this help message and exit
<code>--drive PHYSDRIVE, -d PHYSDRIVE</code>	Device to test (ex: <code>/dev/nvme0n1</code> )
<code>--utilization UTILIZATION, -u UTILIZATION</code>	Amount of drive to test (in percent), 1...100
<code>--yes</code>	Skip the final warning prompt (for scripted tests)

Requirements:

- \* Root access (log in as root, or `sudo {prog}`)
- \* No filesystems or data on target device
- \* FIO IO tester (available <https://github.com/axboe/fio>)
- \* `sdparm` to identify the NVME device and serial number

WARNING: All data on the target device will be DESTROYED by this test.

The user can supply the `-d` parameter with the device node under test (for NVMe devices, this is often `/dev/nvme0n1` for the first namespace of the first device). The script will prompt the user for verification before beginning the test.



If the `-yes` option is not specified, the script will give the user a final chance to abort the test before continuing; otherwise, type `yes` and press Enter to start the test.

```
-----
WARNING! WARNING! WARNING! WARNING! WARNING! WARNING! WARNING!
THIS TEST WILL DESTROY ANY DATA AND FILESYSTEMS ON /dev/nvme0n1
Type the word "yes" and hit enter to continue, or anything else to abort:
yes
```

ezFIO will begin the test at this point. When run in an interactive text console, the script will display a blinking test runtime as the tests progress; the test results will print once completed. When run with output piped to a file, the script will only report the test and results, one per line:

```
# ./ezfio.py -d /dev/nvme0n1 -u 1 -yes
```

```
*****
*****
```

ezFio test parameters:

```

        Drive: /dev/nvme0n1
        Model: NVMe HUSPR1616AHP301 P670
        Serial: STM0001A3778
    AvailCapacity: 1490 GiB
    TestedCapacity: 14 GiB
        CPU: Intel Xeon CPU E5-2640 0 @ 2.50GHz
        Cores: 24
        Frequency: 2500
```

```
Test Description                                                     BW (MB/s) ...
-----
```

```
---Sequential Preconditioning---
```

```
Sequential Preconditioning Pass 1                                   DONE...
Sequential Preconditioning Pass 2                                   DONE...
```

```
---Sustained Multi-Threaded Sequential Read Tests by Block Size---
```

```

Sustained Multi-Threaded Sequential Read Tests by Block Size, BS=512      48.26...
Sustained Multi-Threaded Sequential Read Tests by Block Size, BS=1024     143.87...
Sustained Multi-Threaded Sequential Read Tests by Block Size, BS=2048     417.54...
Sustained Multi-Threaded Sequential Read Tests by Block Size, BS=4096    1,083.22...
Sustained Multi-Threaded Sequential Read Tests by Block Size, BS=8192    1,618.88...
Sustained Multi-Threaded Sequential Read Tests by Block Size, BS=16384    2,040.36...
Sustained Multi-Threaded Sequential Read Tests by Block Size, BS=32768    2,457.75...
...
```

When the tests are completed, the script will display the generated output spreadsheet name:

```

...
Sustained Multi-Threaded Random Write Tests by Block Size, BS=65536      861.94...
Sustained Multi-Threaded Random Write Tests by Block Size, BS=131072      863.75...
```

COMPLETED!

Spreadsheet file:

```
/root/ezfio/ezfio_results_1600GB_24cores_2500MHz_nvme0n1_tm33_2016-01-27_12-39-34.ods
```

## 4. Test Sequence

### 4.1 Overview

The test sequence is documented in the function “DefineTests,” present in the Windows PowerShell and Linux Python scripts. The general test flow is divided into Sequential and Random tests.

### 4.2 Sequential

1. Pass 1: Sequential Fill for full addressable size of device, 128K blocks.
2. Pass 2: Sequential Fill, 128K blocks. (Required to ensure any overprovisioning area is also filled.)
3. Sequential Read tests by block size, since the device is sequentially preconditioned.
4. Random Read tests by block size. These tests are performed at this stage because the sequential preconditioning, by using a 128K block size, should allow for full random performance for all block sizes 128K and smaller. If this were done after 4K random write preconditioning, read blocks larger than 4K would actually read data from multiple smaller 4K units and show lower, incorrect performance.
5. 512-byte Sequential Write tests by queue depth.
6. Sequential Write tests by block size, 512-bytes to 128KB.

### 4.3 Random

1. Pass 1: Random Fill, writes the entire usable capacity of drive randomly. Notices that as the pattern is truly random, this does guarantee that every 4K block will be touched by this write; however, because all blocks on the device were written by the sequential passes, it guarantees that the written data is read in later tests.
2. Pass 2: Random Fill, similar to the prior step. (This ensures overprovisioning areas are utilized.)
3. 4K Random Read performance versus queue depth.
4. 4K Mixed Read / Write performance versus queue depth.
5. Long term performance stability test; 4K Random Mixed Read / Write.
6. 4K Random Write performance.
7. Sustained Random Write performance by block size.



## 5. Interpreting the Test Results

### 5.1 Overview

The spreadsheet output will consist of three sheets: Graphs, Tests and Timeseries. The focus of this section is the Graphs sheet, as it provides a visual summary of the values recorded in the remaining sheets.

### 5.2 System and DUT Summary

The top of the sheet summarizes the system and device under test.

	A	B	C	D	E	F	G	H
1	Drive	\\.\PHYSICALDRIVE6						
2	Tested	2070 GB						
3	Model	NVME1						
4	Serial	NVME0200						
5	CPU	Intel(R) Xeon(R) CPU E5-2640 0 @ 2.50GHz						
6	OS	Microsoft Windows Server 2012 R2 Datacenter Evaluation - Build 9600 - ServicePack 0.0						

Figure 4: System and DUT Summary

### 5.3 Sustained Mixed Read-Write Test

The next graph will show the results of the Sustained Mixed Read / Write test. This is a time-series graph and captures the IOPS delivered, second-by-second, by the device over a period of 20 minutes, as well as the numerical average and variance. Large variance or “choppiness” in the graph indicates that the applications are stalling while waiting for I/O.

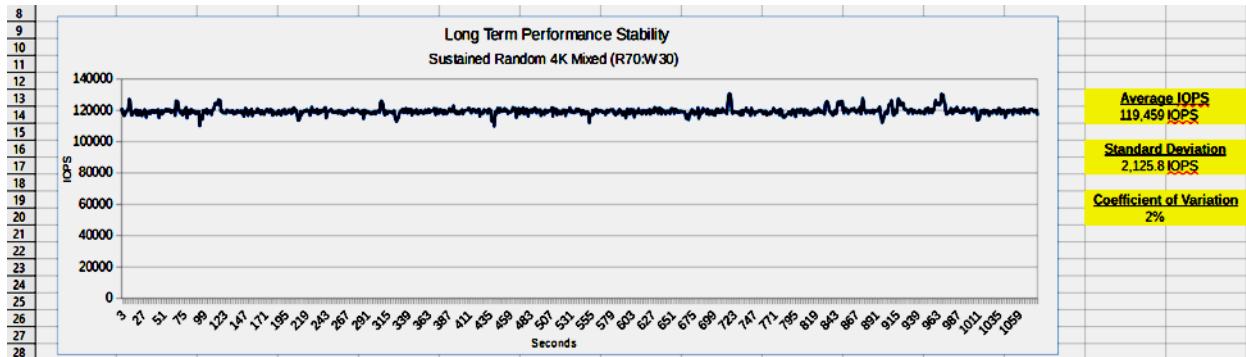


Figure 5: Sustained Mixed Read-Write Test

## 5.4 Queue Depth

The next graph shows how the queue depth affects the Sustained Mixed Read / Write performance. Whereas the previous time-series graph provides a “best case” sustained performance by allowing for a high queue depth, the “IOPS vs. Queue Depth” graph will identify how an application with a lower number of I/Os in flight will perform.

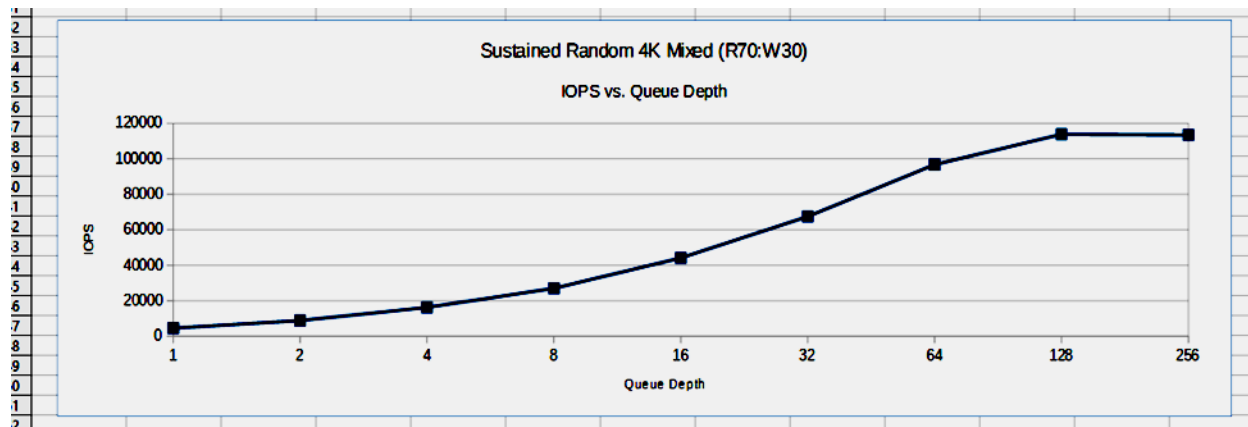


Figure 6: IOPS vs. Queue Depth

## 5.5 4K Random Read Performance with Varying Queue Depths

The next two graphs shows the 4K Random Read performance over varying queue depths. The IOPS and total application I/O latency are charted. The latency graph often resembles a “hockey-stick”, with the point of inflection occurring when the device reaches peak performance at the minimum queue depth. The increase of queue depth after this point often results in an exponential increase in I/O latencies, as the underlying device is fully utilized.

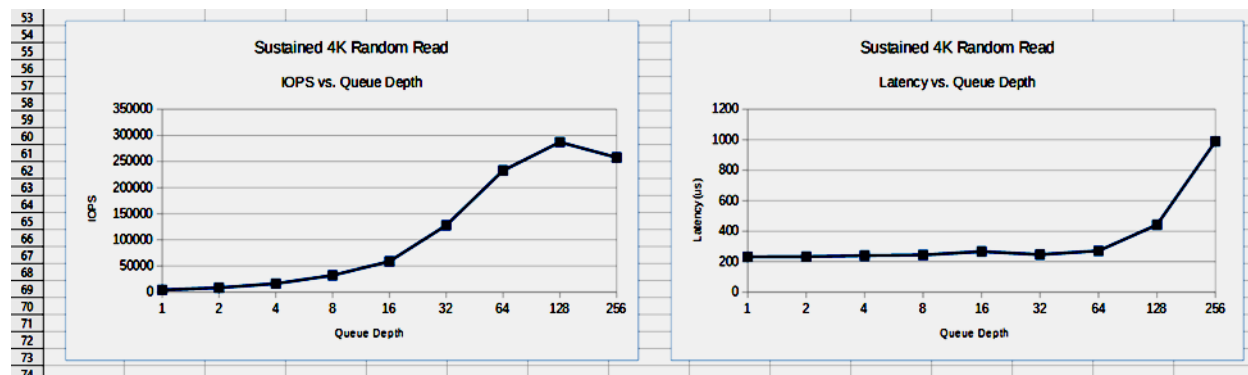


Figure 7: 4K Random Read, Queue Depth and Latency

## 5.6 4K Pure Random Writes

The next two graphs are similar, but depict pure 4K Random Writes.

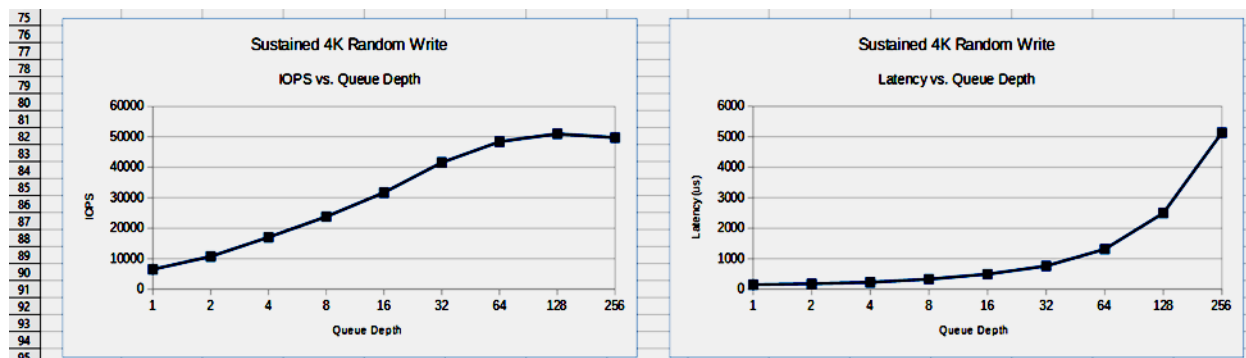


Figure 8: 4K Pure Random Writes

## 5.7 Random Read Performance / Sequential Reads / Identical Block Sizes

The next two graphs are useful for understanding how Random Reads perform compared to Sequential Reads of the identical block size. Usually, due to architectural implementation made by the NVMe device manufacturer, there are profound differences in Sequential (or streaming) versus fully Random performance.

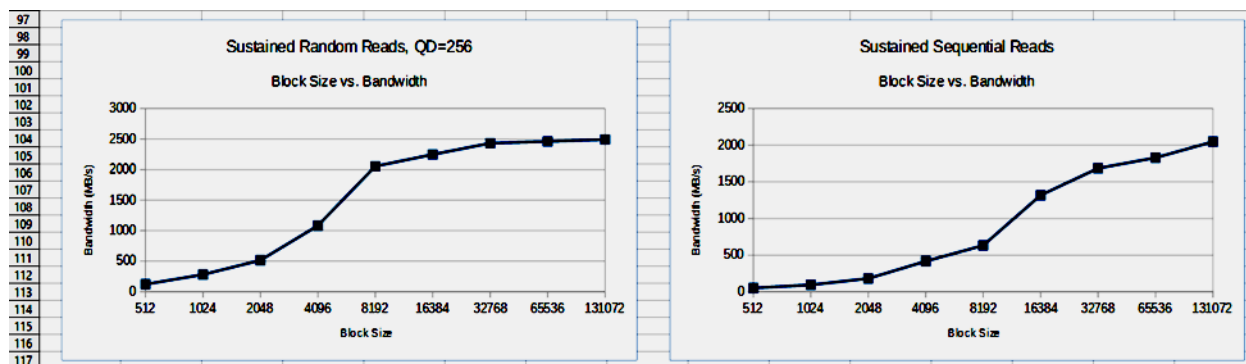


Figure 9: Sequential vs. Random Read Performance by Block Size

## 5.8 Full Write Workloads with Varying Block Sizes

The next two graphs are similar, but depict full Write workloads as the block size varies.

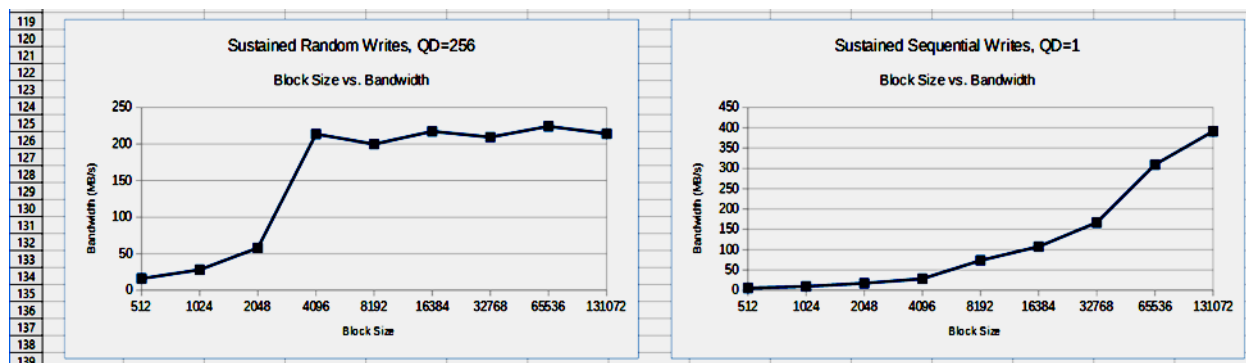


Figure 10: Sequential vs. Random Write Performance by Block Size

## 5.9 Raw FIO Output

The raw FIO output for each test, and the command line executed to generate the results, are found in a timestamped directory named `details_*`. The individual `*.ods` files may be opened in a text editor or spreadsheet and parsed.

- ✓ The FIO output results will be in “terse” format due to the implementation of script processing. The user should consult the FIO documentation for the description of each semicolon-delimited field.
- ✓ The resulting performance of any test run that is run independently may not match the performance reported by ezFIO under full sustained mode operation due to the preconditioning and background garbage collection processes.

# Index

## Contributions

ezFIO .....	5
GitHub .....	5
GitHub branching .....	5
GitHub pull request .....	5

## ezFIO

CLI	
-d parameter .....	14
ezfio.ps1 .....	12
ezfio.py .....	14
Linux .....	14
listing devices .....	12
running .....	13
sdparm .....	14
Windows .....	12

## GUI

Disk Management .....	10
Drive Selection Window .....	10
Open Graphs Spreadsheet .....	12
Start Test .....	11
Test Progress Window .....	11
Windows .....	10

## Installation

Linux	
ezfio.py .....	9
tarball archive .....	9
original.ods template .....	9
Windows	
ezfio.bat .....	9
ezfio.ps1 .....	9
ZIP archive .....	9

## Introduction

ezFIO .....	8
FIO .....	8
Microsoft Excel .....	8
OpenDocument	
OpenOffice Calc .....	8

spreadsheet .....	8
preconditioning .....	8
scripting languages .....	8
Sustained Mode .....	8

## Licensing

FIO Moral License .....	5
GNU Public License .....	5

## System Requirements

Linux	
FIO .....	9
libaio .....	9
Python .....	9
raw device access .....	9
root privileges .....	9
Windows	
.Net Framework 4.5 .....	9
Administrator .....	9
computer settings .....	9
FIO 2.2 .....	9
Server .....	9

## Test Results

4K Pure Random Writes .....	19
4K Random Read, Queue Depths .....	18
Full Write Workloads .....	19
interpreting .....	17
Queue Depth .....	18
Random Reads	
identical block sizes .....	19
Raw FIO Output .....	20
Sequential Reads	
identical block sizes .....	19
Sustained Mixed Read-Write Test .....	17
System / DUT Summary .....	17

## Test Sequence

Random .....	16
Sequential .....	16

© 2016 HGST, Inc. All rights reserved.

HGST, a Western Digital company  
3403 Yerba Buena Road  
San Jose, CA 95135  
Produced in the United States

United States: 800-801-4618 (Toll-Free)  
International: 408-717-6000  
FAX: 408-717-5000

February 19, 2016

ezFIO is distributed under the GNU Public License V2.0 or later. The HGST logo is a trademark of HGST, Inc. and its affiliates in the United States and/or other countries. HGST trademarks are authorized for use in countries and jurisdictions in which HGST has the right to use, market and advertise the brands. HGST shall not be held liable to third parties for unauthorized use of HGST trademarks.

All other trademarks and registered trademarks are the property of their respective owners.

References in this publication to HGST products, programs or services do not imply that HGST intends to make these available in all countries in which HGST operates. Product information is provided for information purposes only and does not constitute a warranty. Information is true as of the date of publication and is subject to change. Actual results may vary.

This publication is for general guidance only. Photographs may show design models.