Table of Contents

[0. Introduction 1](#_Toc490144566)

[1. BIOS Configuration 1](#_Toc490144567)

[2. Firmware Configuration 2](#_Toc490144568)

[3. Test Steps 3](#_Toc490144569)

[4. Testing Configuration 6](#_Toc490144570)

[5. Performance Result 7](#_Toc490144571)

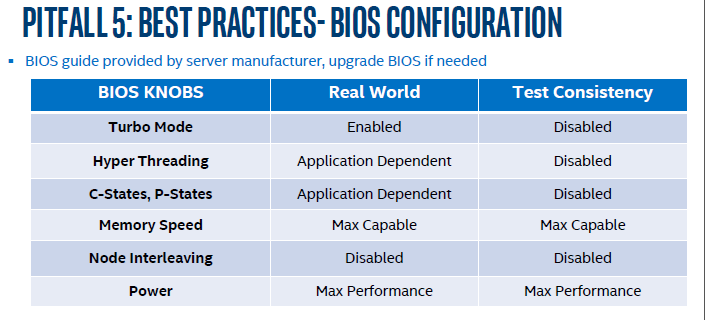
[6. Appendix 7](#_Toc490144572)

## 0. Introduction

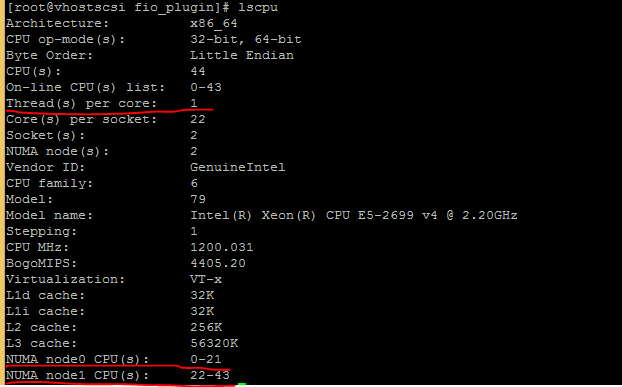
This document contains the preferred information on evaluating SPDK performance on local NVMe SSDs. Particularly it will compare the multiple SSDs case where SPDK has one core to manage several NVMe SSDs while Kernel driver has one core on each NVMe SSD. So that more cores will be used under the Kernel case.

## 1. BIOS Configuration

1. Based on below table, there are two scenarios. One is for the “Real World” scenario and the other is for the scenario under testing where the better performance data can be obtained.



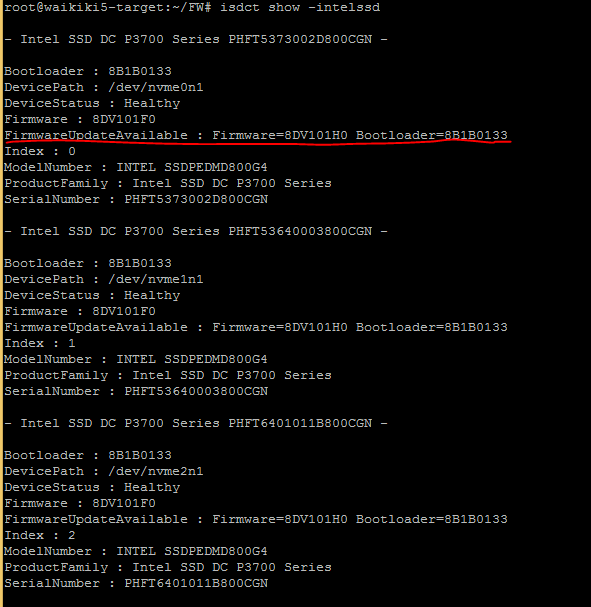
1. After making changes accordingly to the above “Test Consistency” scenario, you can check whether these configuration already takes effect as following diagram like “Threads per core = 1, NUMA node0 CPUs: 0-21, NUMA node1 CPUs: 22-43”. Note: the number of CPU cores depends on the CPU model.



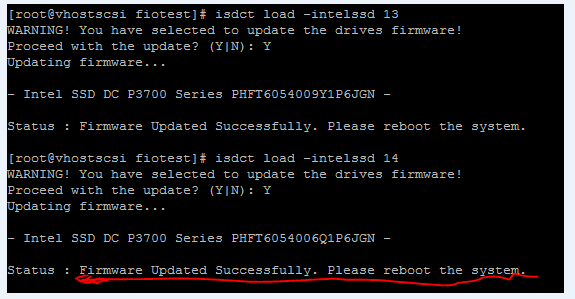
## 2. Firmware Configuration

1. Check the SSD firmware like DC P3700, use the command “isdct show -a -intelssd” to check whether there is a new version of firmware.

*\*isdct (Intel SSD Data Center Tool) refer* [*here*](https://downloadcenter.intel.com/download/26970/Intel-SSD-Data-Center-Tool?product=67709)*.*



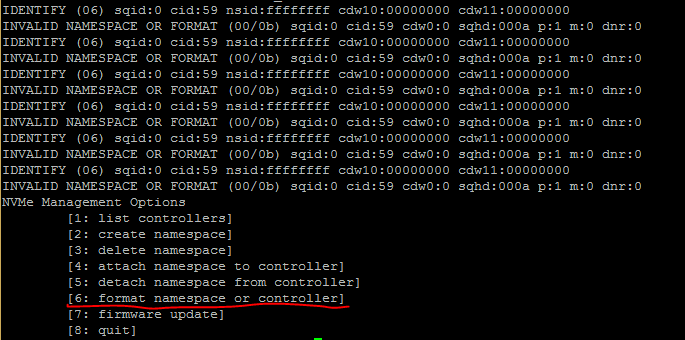
1. Use command “isdct load -intelssd x(x SSD’s index)” to load the new firmware



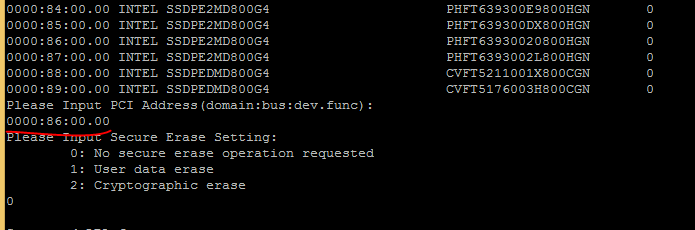
## 3. Test Steps

1. Erase and Format the SSD: if it is a new SSD, this is not needed. If not, for a used SSD, first to do this step before other operation like “precondition” to remove some old information on this SSD.

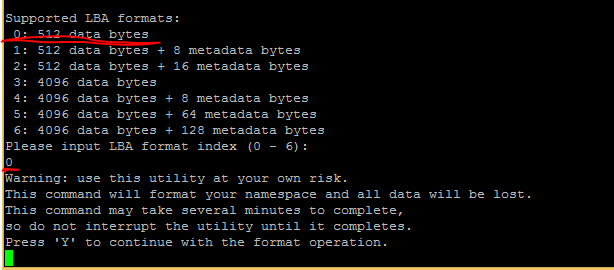
One tool is spdk/examples/nvme/nvme\_manage/nvme\_manage to conduct the “Erase and Format the SSD” as below:



* 1. Enter pci address(ssd pci address in your server) and choose option 0:

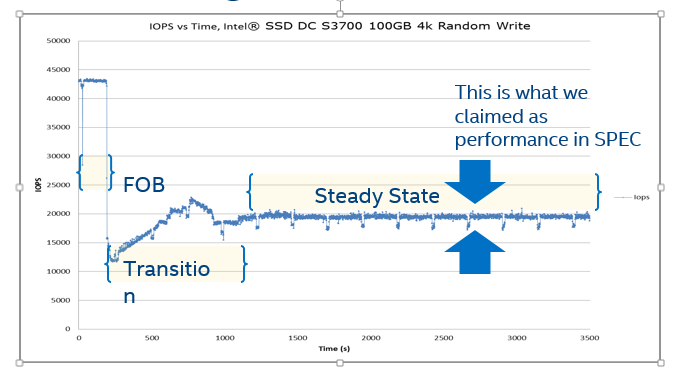


* 1. Choose option 0 and enter Y :

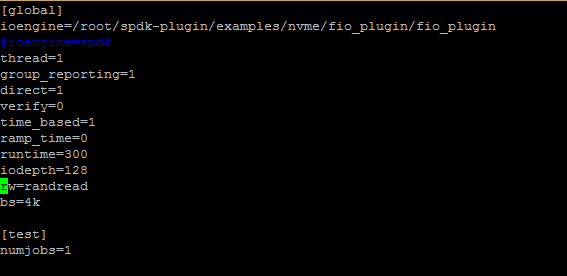


1. Precondition SSD: testing on NAND SSD needs to conduct this phase so that the write performance can enter the stable state to have a consistent performance data. In the testing initial phase, SSD’s performance (randwrite) will much better (FOB phase in below diagram). This is due to the writes are completed in the empty drive and/or with less background operation. After the precondition phase, the performance will enter the steady state and that’s the meaningful performance we tracked.

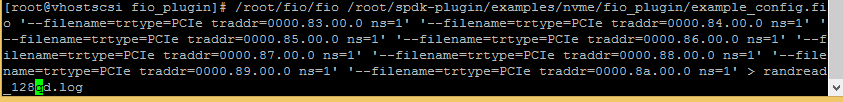
Usually for a P3700 800G SSD, first sequentially write about two hours (1.6TB data written) to cover all the space of the drive, then randomly write 1 hour (1.6TB data written). After this, it will enter the comparably steady data. The duration of sequential and random write depends on the actual SSDs involved.



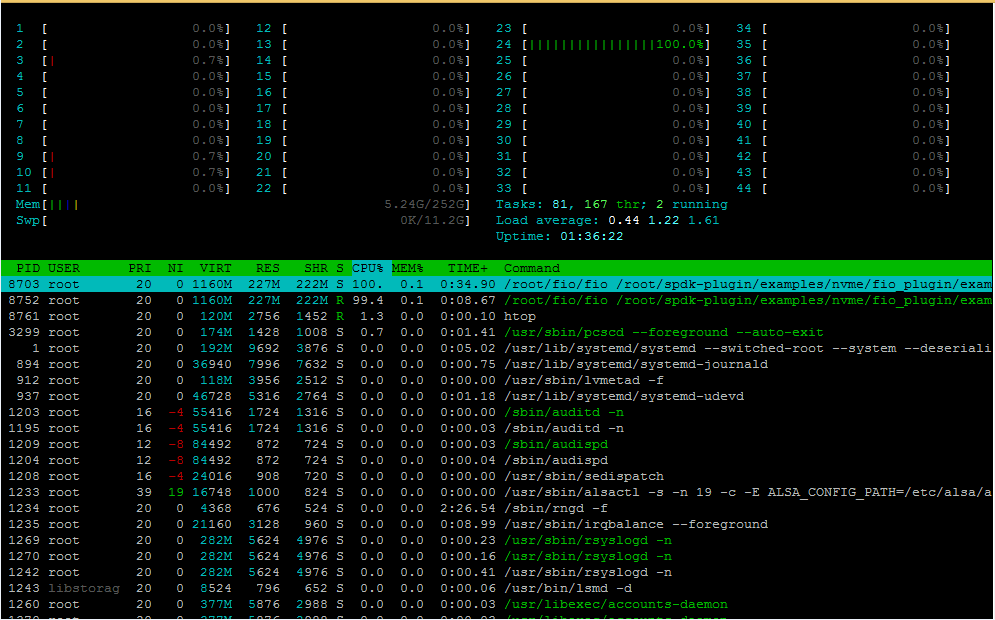
1. Use SPDK fio-plugin to test the SPDK performance, configuration from [config.fio](https://github.com/spdk/spdk/blob/master/examples/nvme/fio_plugin/example_config.fio) as following:



* 1. Use SPDK fio-plugin to test multiple SSDs



1. To confirm whether there is a single Core used in the testing, when testing is ongoing, use the “htop” command to check the CPU usage (in below example, only core24 is allocated for the testing).



## 4. Testing Configuration

|  |  |  |
| --- | --- | --- |
| Below testing involving the comparison of kernel driver and SPDK driver performance on 8 NVMe SSDs. The typical configuration as:  **Kernel:8core 8NVMe devices vs SPDK: 1core with 8NVMe devices** | |  |
| Software： | |  |
| OS: CentOS Linux release 7.2.1511 Kernel: 3.10.0-327.el7.x86\_64 |  |  |
| fio-Version: fio-2.18 | |  |
| GCC Version: 4.8.5 20150623 (Red Hat 4.8.5-11) (GCC) | |  |
| SPDK: master branch: commit 6fb1ce422e4d3e893770bd4f86d053eb5894e32c | |  |
| Hardware: | |  |
| SSD: Intel® P3700 NVMe SSD (1.6TB) x 8 | |  |
| CPU: Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz  (HT off, Intel® Turbo Boost Technology enabled)  Memory: 256GB DDR4, 8x 32GB DDR4 2400 MT/s | | |
| Testing method:  randread: runtime=300s | |  |
| randwrite: runtime=600s | |  |
| randrw: runtime=600s  queue depth: 1, 4, 16  io size: 4K | |  |

Note:

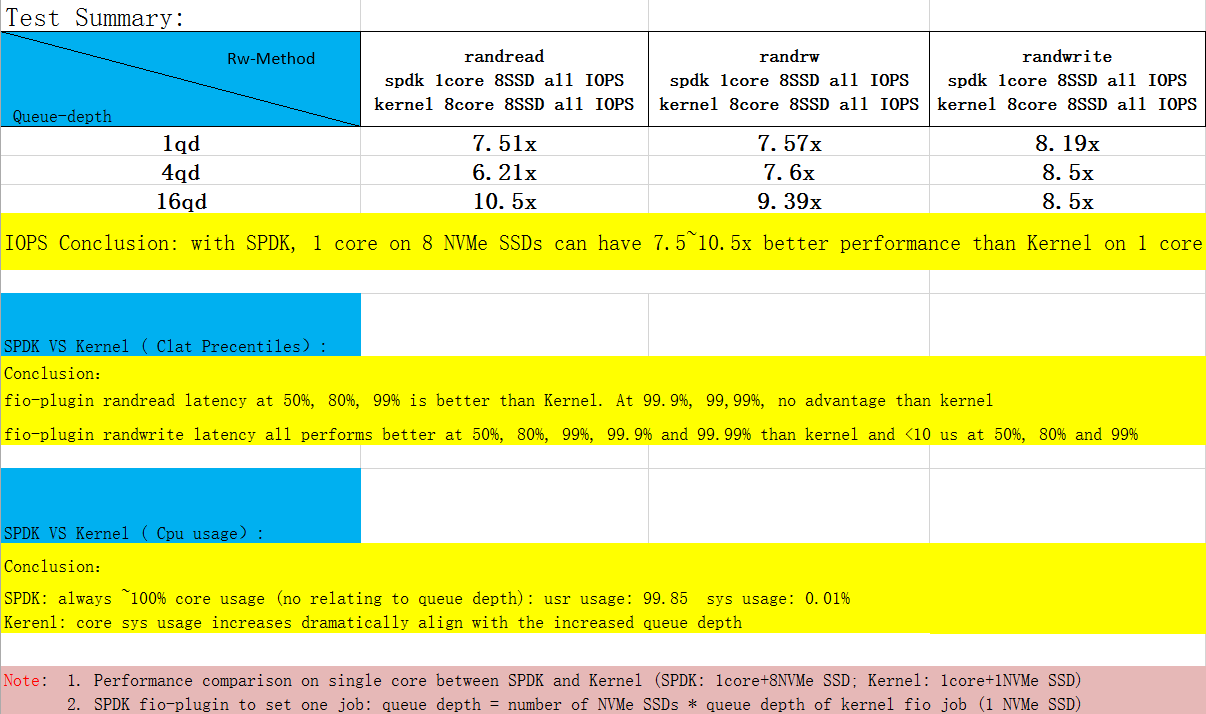
1. queue depth of one fio job is shared if multiple NVMe SSDs are used for the SPDK fio\_plugin usage. For example, if 128 is the queue depth for the one fio job and 8 NVMe SSDs connected, each SSD will have 16 queue depth. So that to compare SPDK and Kernel driver, it will be as following:

SPDK: 1 job, 128 queue depth/job; 8 NVMe SSDs, 1 core

Kernel: 8 job, 16 queue depth/job; 8 NVMe SSDs, 8 cores

1. To achieve better performance, it’s also preferred to have the hardware resource under testing on the same socket like NVMe SSD, CPU Core, Memory here. For some SPDK applications like SPDK perf, the core for IO testing can be explicitly assigned while for the FIO plugin, it is inherited from FIO application on which core to run. It may be on the different core from the NVMe SSDs.

## 5. Performance Result

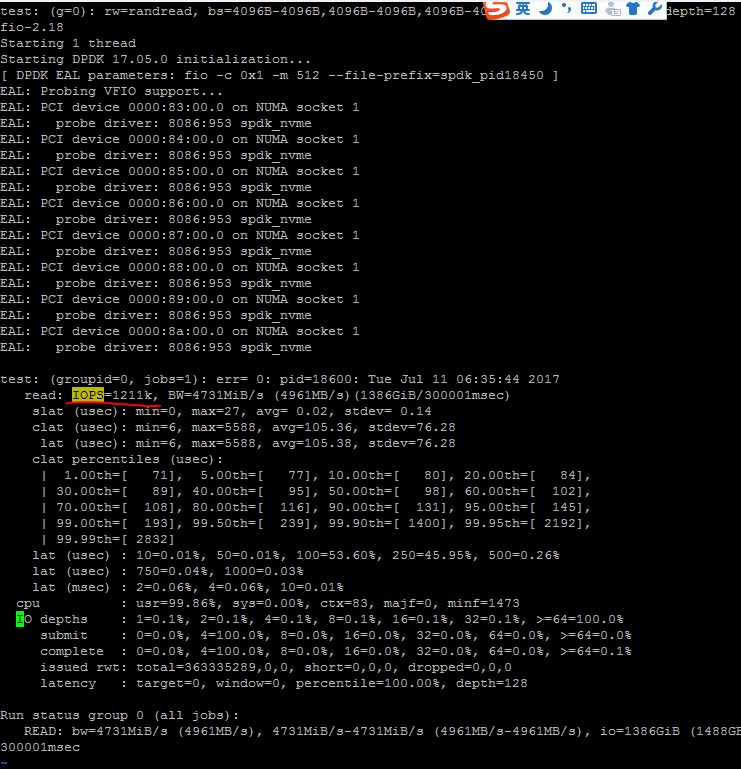


Detailed performance result at .

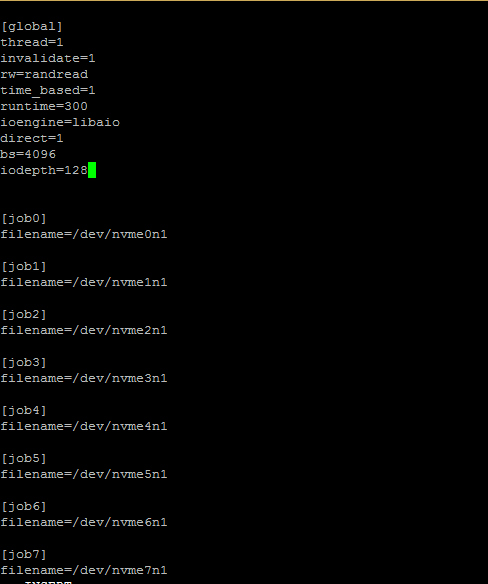
## 6. Appendix

1. Related Testing Information
   1. Result of fio-plugin: 1 core 8 SSDs.

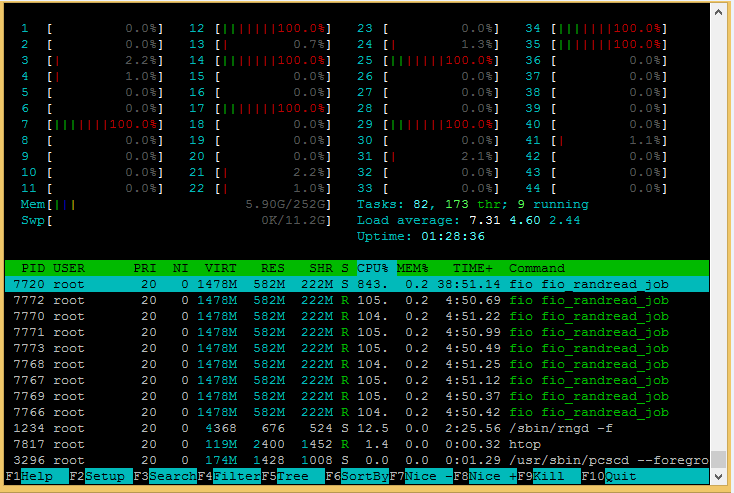
1 job with 128 queue depth. Each SSD with 16 queue depth on 4K randread. Overall IOPS 1211K:



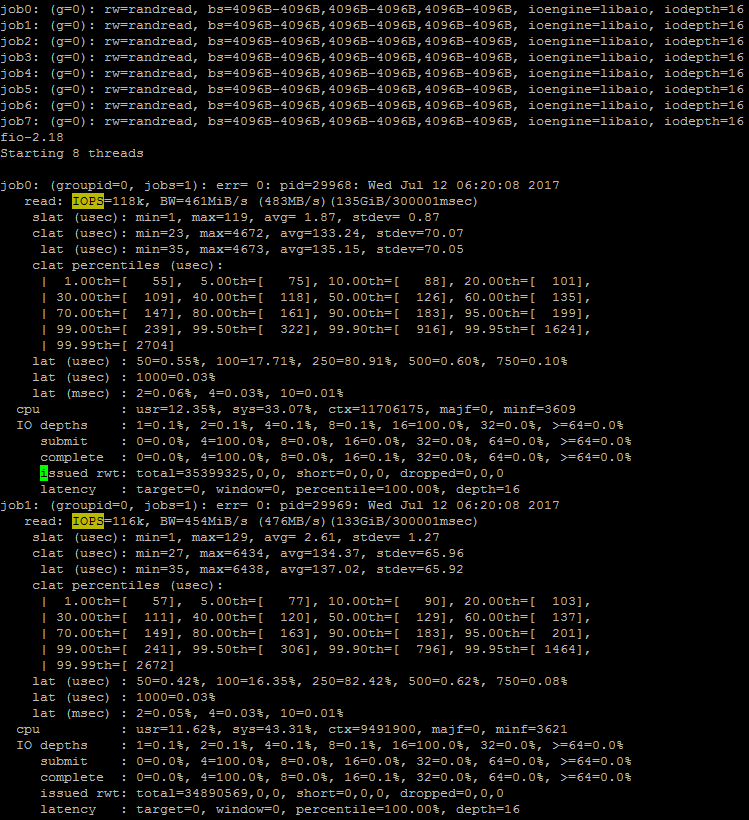
* 1. Test Kernel performance, fio configuration:



* 1. Use htop command to confirm the CPU usage on kernel case (8 fio jobs on 8 cores)

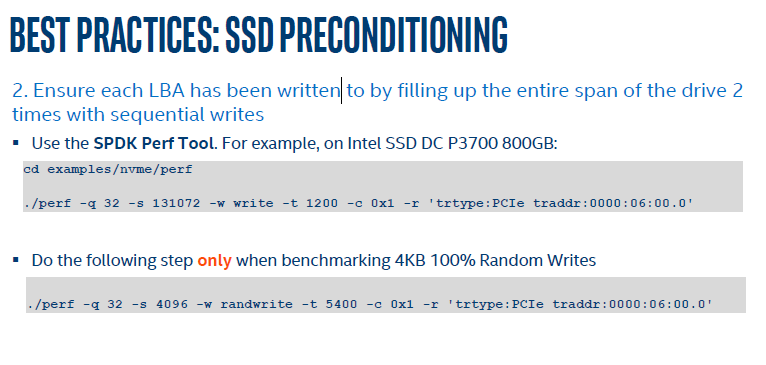


* 1. Performance result on Kernel 16 queue depth（overall IOPS 921K IOPS on 8 Jobs IOPS）:



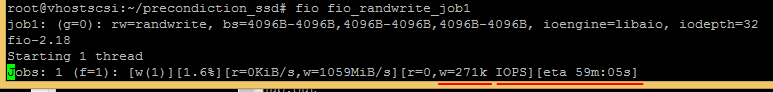
1. Perf tool testing steps

Use perf tool to do the SSD precondition:



Note:

1. Precondition sequential write and randwrite needs to be conducted continuously, no stop running after sequential write and before the randwrite. The exact time of sequential write and randwrite depends on the capacity of the SSD and the write speed. The guideline is to make the sequential write covers the whole capacity. To make sure all the space of SSD has been written, you can use the nvme smart-log /dev/nvmeXnY command to check the written data before and after the “precondition” operation.
2. After “precondition”, the continuous read/write test at the initial stage, there still may have unsteady and better performance. It’s preferred to increase the testing time to wait the SSD enter the steady state. Take P3700 as example from below diagram: the randwrite testing performance after “precondition” can achieve 271K IOPS at the very early few minutes (60minutes running time and in the first minute as following) and later enter the steady state (P3700 1.6TB randwrite SPEC 150K IOPS)：



Long run of randwrite test (60minutes in below), the IOPS reaches 128K IOPS (8 minutes testing left) which is close to SPEC (150K IOPS):

