SSD Performance Report

Yunhua Fang

October 2024

Introduction

In order to gain first-hands experience on profiling the performance of modern SSDs. We apply the Flexible IO tester (FIO), which is available at *https://github.com/axboe/fio*, to profile the performance of the SSD. FIO is a widely used tool in the storage industry for benchmarking and profiling the performance of storage devices such as hard drives, solid-state drives (SSDs), and other I/O subsystems. Originally developed by Jens Axboe, FIO allows users to simulate various workloads by adjusting key parameters like block size, read/write ratios, I/O queue depths, and more. It provides in-depth insights into performance metrics such as IOPS (Input/Output Operations Per Second), throughput, and latency under specific conditions. Supporting a wide range of I/O engines and platforms, including Linux, Windows, and macOS, FIO is an essential tool for developers, system administrators, and engineers seeking to evaluate storage performance across different configurations and workloads. Therefore, we design a set of experiments based on FIO to measure the SSD performance (latency and throughput) under different combinations of the following parameters: data access size, read vs. write intensity ratio, and I/O queue depth. We could observe a clear trade-off between access latency and throughput as revealed by queueing theory.

Methods

In the set of experiments, all the operations are applied on a 1GB partition of SSD. We use 4KB, 32KB, 128KB, and 512KB as data access sizes to reveal the differences between small access size and large access size. Note that throughput is typically represented in terms of IOPS (IO per second) for small access size (e.g. 64KB and below), and represented in terms of MB/s

r/w ratio \ queue depth	64	512	1024
only read	161	168	175
only write	0	0	0
5:5	72.4	71.3	71.1
7:3	108	105	109

Table 1: Read Throughput (Kilo IOPS) of 4KB Data Access Size

r/w ratio \ queue depth	64	512	1024
only read	327.96	1831.15	3173.94
only write	0	0	0
5:5	301.19	1930.31	3801.79
7:3	302.62	1857.84	3390.31

Table 2: Read Average Latency (ms) of 4KB Data Access Size

for large access size (e.g. 128KB and above). Experiments are executed under 3 queue depths: 64, 512, and 1024. By comparing the performance of deep and shallow queue depth, the experiments clearly reveal the does the data follows the Queueing theory. The read/write ratio in the workload is classified to only read, only write, 70% read and 30% write, and 50% read and 50% write. The impact of read operations and write operations to SSD could be both captured and visualized from our data.

Evaluation

Table 1 and Table 2 record the performance of reading tasks with 4KB data access size. When only operating read requests, the throughput is maximized. Although it is not apparent, the throughput also slightly increases with longer queue length. However, we can observe that the latency severely increases from hundreds of microseconds to thousands of microseconds. The phenomenon seems consistent with the Little's theorem in queueing theory. The system handles more requests, but the latency of requests also increases.

Table 3 and Table 4 record the performance of writing requests with 4KB data access size. Write operations have less throughput and less latency to respond than read operations. From the data, we obtain the same conclusion as we

r/w ratio \ queue depth	64	512	1024
only read	0	0	0
only write	112	114	114
5:5	72.5	71.3	71.1
7:3	46.2	45.1	46.9

Table 3: Write Throughput (Kilo IOPS) of 4KB Data Access Size

get from read requests. The latency grievously increases to approximately 20 times the original latency when we compare the queue with 64 depth and with 1024 depth.

r/w ratio \ queue depth	64	512	1024
only read	0	0	0
only write	289.42	2268.58	4519.26
5:5	207.51	1831.60	3698.13
7:3	172.84	1699.98	3255.19

Table 4: Write Average Latency (ms) of 4KB Data Access Size

Table 5, 6, 7, 8 record the latency and throughput of the program with 32KB data access size. By comparing them with previous tables, the throughput decreases because of the larger data access size. Since the queue size is fixed, it only allows a certain amount of data to transfer. The size of each chunk of data read from or written to the storage device during the test becomes larger than before leading less operation executions at a time.

The data in tables show that When the queue depth increases from 64 to 1024 on an SSD, the latency of operations rises and throughput decreases due to the inherent characteristics of SSDs. SSDs are designed to handle multiple I/O requests in parallel, benefiting from features like multiple NAND flash channels and parallel data paths. At lower queue depths (e.g., 64), SSDs can efficiently distribute and manage incoming requests across these channels, optimizing both throughput and latency. However, when the queue depth increases significantly (e.g., to 1024), the number of queued requests begins to exceed the device's ability to process them efficiently. The internal resources, such as flash memory, controller bandwidth, and buffers, become saturated, leading to contention. As a result, I/O requests spend more time

waiting in the queue before being serviced, which increases latency. Additionally, managing a large number of simultaneous requests adds overhead for the SSD controller, reducing its efficiency and causing throughput to decrease. This degradation in performance reflects the SSD's inability to take full advantage of increased concurrency beyond a certain threshold, where internal bottlenecks limit its capacity to process data efficiently. Thus, while SSDs are optimized for parallelism, excessively high queue depths lead to diminishing returns.

r/w ratio \ queue depth	64	512	1024
only read	33.3	87.4	78.2
only write	0	0	0
5:5	31.2	29.1	27.2
7:3	48.9	42.4	41.3

Table 5: Read Throughput (Kilo IOPS) of 32KB Data Access Size

r/w ratio \ queue depth	64	512	1024
only read	1911.70	5628.61	12838.15
only write	0	0	0
5:5	645.96	1808.06	11465.11
7:3	762.87	2293.80	10874.12

Table 6: Read Average Latency (ms) of 32KB Data Access Size

r/w ratio \ queue depth	64	512	1024
only read	0	0	0
only write	83	84.9	86.2
5:5	31.1	29	27.2
7:3	20.9	18	17.6

Table 7: Write Throughput (Kilo IOPS) of 32KB Data Access Size

Table 9, 10, 11, 12 include the performance with 128KB data access size. We can also see that the throughput and latency increases compare to previous data access sizes. It corresponds to the Little's theorem we discussed before. Both throughput and latency increase with the queue depth increases, which

r/w ratio \ queue depth	64	512	1024
only read	0	0	0
only write	536.63	5521.99	11068.75
5:5	1383.56	14073.43	25894.34
7:3	1247.93	22822.04	32067.89

Table 8: Write Average Latency (ms) of 32KB Data Access Size

r/w ratio \ queue depth	64	512	1024
only read	2350	3085	2811
only write	0	0	0
5:5	1011	1129	1132
7:3	1788	1553	1650

Table 9: Read Throughput (MB/s) of 128KB Data Access Size

means the number of requests that are handled increases. The average number of items in a queue (L) is equal to the arrival rate (λ) multiplied by the average time spent in the system (W), or $L = \lambda \times W$. The function keeps consistent on our data.

We also sample some trend graphs to clearly show the comparison between these attributes. We selected some distinctive samples from our test results to observe because some of them were replicated. Figure 1 shows that the latency continuously increases with a larger access size. More data needs to be accessed, which means more tasks are needed to handle, so the latency increases. Figure 2 proves the statement with increasing throughput. More tasks are handled during higher throughput, which follows the Little's theorem.

r/w ratio \ queue depth	64	512	1024
only read	3529.50	20899.89	44997.52
only write	0	0	0
5:5	1238.41	7460.42	36419.98
7:3	1270.03	9840.12	37447.28

Table 10: Read Average Latency (ms) of 128KB Data Access Size

r/w ratio \ queue depth	64	512	1024
only read	0	0	0
only write	2848	2833	3345
5:5	1034	1155	1158
7:3	787	683	726

Table 11: Write Throughput (MB/s) of 128KB Data Access Size

r/w ratio \ queue depth	64	512	1024
only read	0	0	0
only write	2923.11	23415.33	50320.52
5:5	5970.79	50412.56	78862.06
7:3	7730.49	75175.44	97940.73

Table 12: Write Average Latency (ms) of 128KB Data Access Size

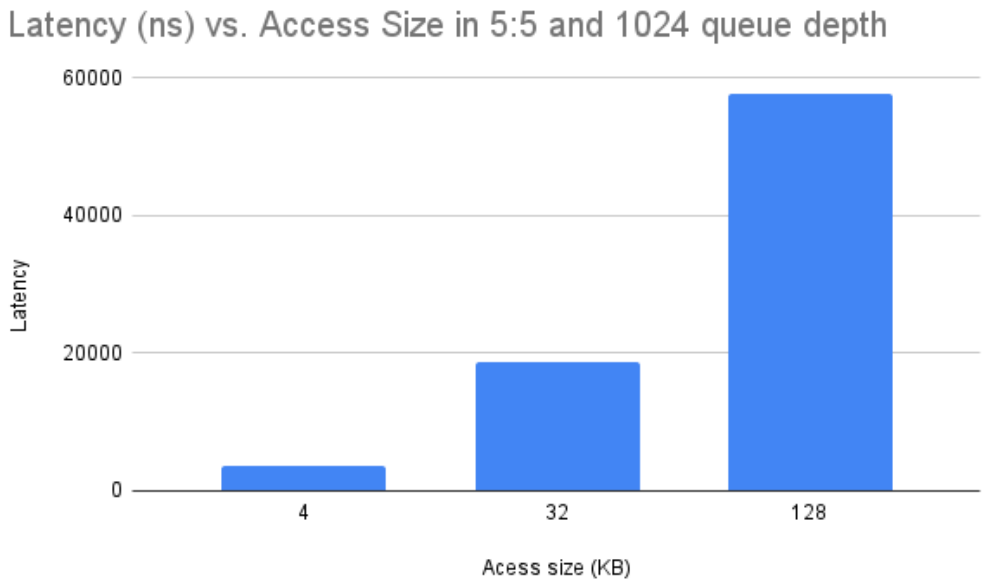

Figure 1: Latency (ns) vs. Access Size in 5:5 and 1024 queue depth

Figure 3 proves that the SSD takes more time to handle read tasks then write tasks. One possible reason is the nature of the workload being tested; if the read operations involve accessing data that is not cached or is scat-

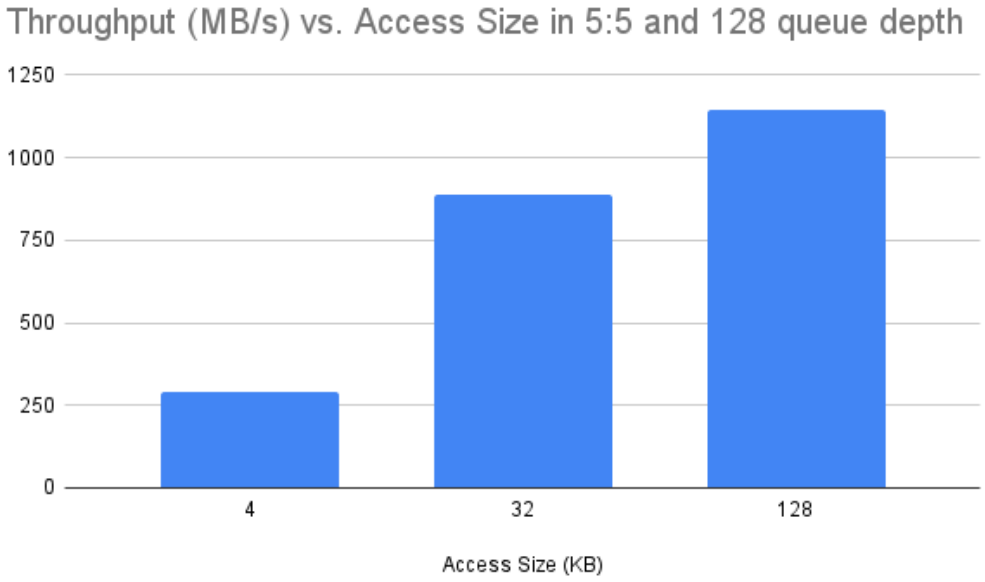

Figure 2: Throughput (MB/s) vs. Access Size in 5:5 and 1024 queue depth

tered across the drive, the SSD controller may need to perform additional operations to retrieve the data, thereby increasing latency. Additionally, the SSD's firmware and controller algorithms might prioritize write operations to maintain data integrity and wear leveling, inadvertently causing read tasks to experience higher latency. Another factor could be the type of NAND flash memory used; for instance, TLC (Triple-Level Cell) or QLC (Quad-Level Cell) NAND tends to have slower read speeds compared to SLC (Single-Level Cell) NAND, especially under heavy load conditions. High I/O depths, as indicated in the FIO test results, can also lead to contention and increased latency for read requests if the SSD is simultaneously handling a large number of write operations.

Figure 5 shows the trend of increasing queue depth under 50% ratio of read tasks and 4KB access size. A longer queue size means that the SSD controller must manage a higher number of simultaneous I/O requests, which can lead to contention for internal resources such as memory channels and processing units. As the queue grows, the controller may struggle to efficiently schedule and prioritize these requests, resulting in longer wait times before each task is processed. Additionally, higher queue depths can overwhelm the SSD's parallelism capabilities; while SSDs are designed to handle multiple operations concurrently, there is a limit to how effectively they can distribute and

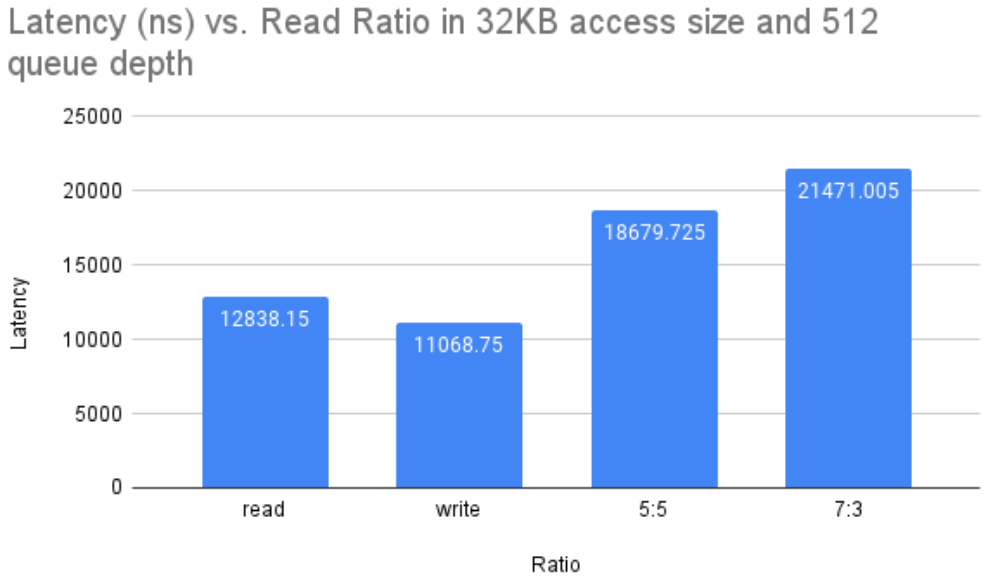

Figure 3: Latency (ns) vs. Read Ratio in 32KB access size and 512 queue depth

execute these tasks in parallel. This saturation can cause delays as the controller becomes a bottleneck, unable to maintain optimal throughput for each individual request. Furthermore, increased queue sizes can exacerbate issues related to garbage collection and wear leveling, processes that are essential for maintaining SSD performance and longevity but can introduce additional overhead when managing numerous pending tasks. Thermal throttling may also come into play, as higher workloads generate more heat, prompting the SSD to reduce performance to prevent overheating. All these factors collectively contribute to longer latency when SSDs are subjected to tasks with extended queue sizes, highlighting the importance of balancing queue depth with the drive's architectural capabilities to maintain optimal performance.

Intel Data Center NVMe SSD D7-P5600 (1.6TB) lists a random write-only 4KB IOPS of 130K. On our 1GB of Kingston NV2 PCIe 4.0 NVMe SSD, we have approximately 114K IOPS. The performance difference between them highlights their distinct target applications and hardware capabilities. The Intel D7-P5600 which is optimized for high-performance, multi-queue workloads typical in enterprise environments where consistent and high throughput under heavy loads is critical. This SSD likely benefits from advanced features like a robust controller, more NAND flash channels, and better power-

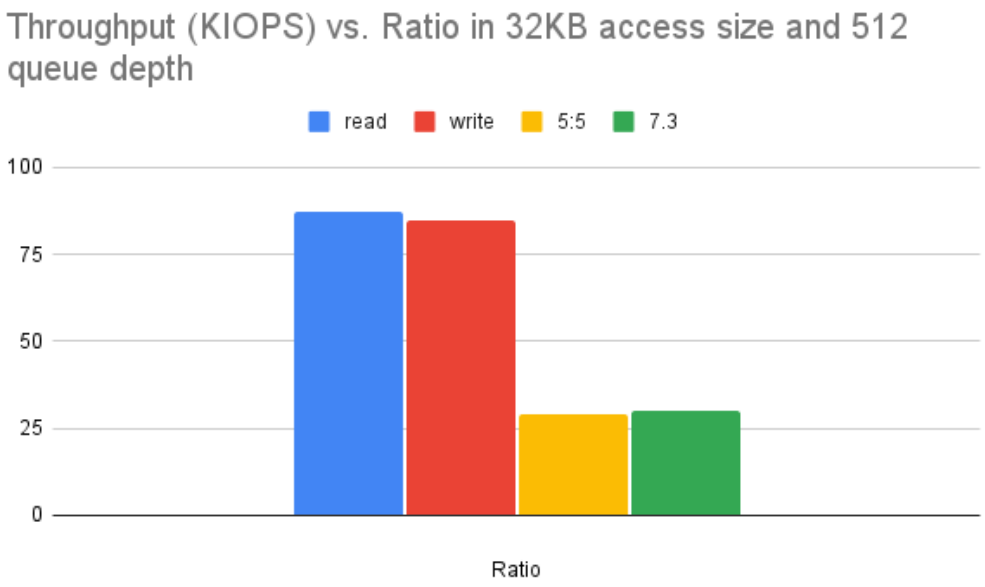

Figure 4: Throughput (KIOPS) vs. Ratio in 32KB access size and 512 queue depth

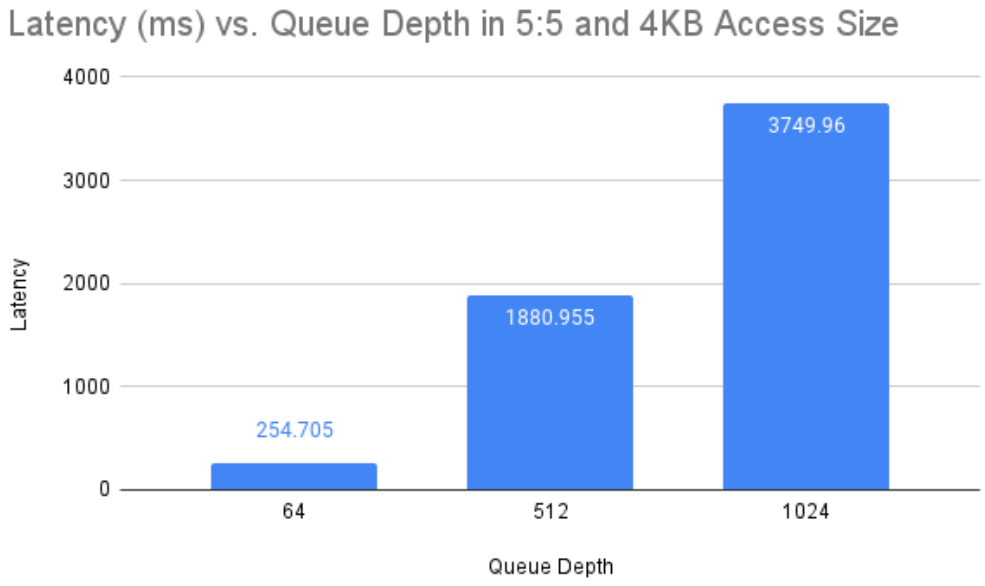

Figure 5: Latency (ms) vs. Queue Depth in 5:5 and 4KB Access Size

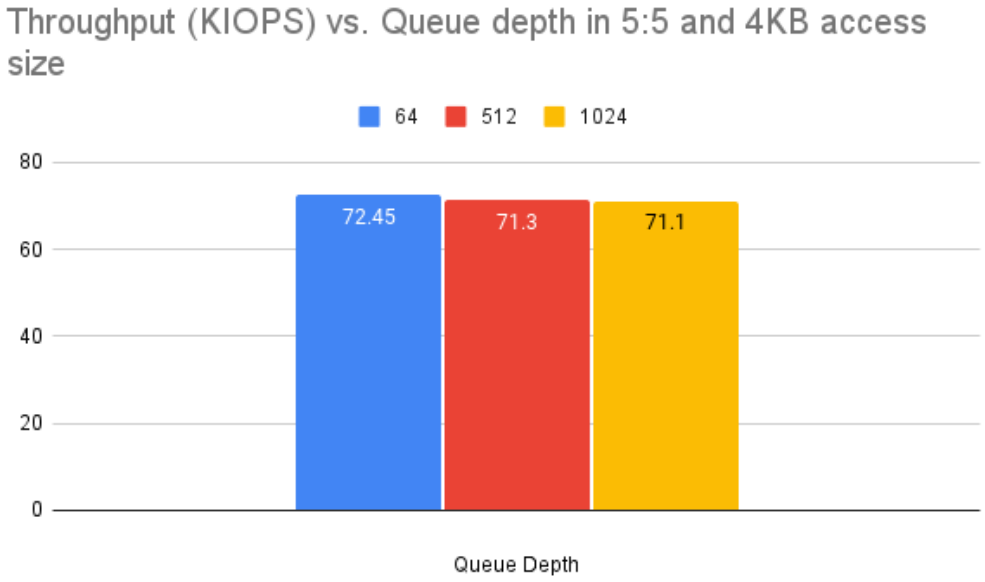

Figure 6: Throughput (KIOPS) vs. Queue depth in 5:5 and 4KB access size

loss protection, making it ideal for tasks such as high-frequency database operations, virtualization, and large-scale web applications that demand sustained I/O performance. On the other hand, the Kingston NV2 PCIe 4.0 NVMe SSD is a consumer-grade drive intended for general-purpose computing. While it performs well in everyday tasks such as booting an operating system or loading applications quickly, it may struggle under heavier, data-center-like workloads due to limited internal resources like fewer flash channels and a simpler controller design. Therefore, while the Kingston NV2 offers solid performance for standard consumer use, it lacks the consistent high-speed capabilities required for enterprise-level, I/O-intensive tasks where the Intel D7-P5600 excels.