Report

Drive: SSD Micron_2210_MTFDHBA512QFD

Q0:

```
job1: (g=0): rw=read, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=windowsaio, iodepth=1
fio-3.39
Starting 1 thread

job1: (groupid=0, jobs=1): err= 0: pid=1824: Mon Apr 7 01:47:18 2025
  read: IOPS=19.9k, BW=77.6MiB/s (81.4MB/s)(1024MiB/13198msec)
    slat (usec): min=10, max=5933, avg=16.71, stdev=12.61
    clat (nsec): min=321, max=2440.7k, avg=32342.83, stdev=17377.98
     lat (usec): min=33, max=6075, avg=49.05, stdev=22.11
    clat percentiles (usec):
     |  1.00th=[   25],  5.00th=[   27], 10.00th=[   27], 20.00th=[   28],
     | 30.00th=[   29], 40.00th=[   30], 50.00th=[   30], 60.00th=[   31],
     | 70.00th=[   33], 80.00th=[   35], 90.00th=[   40], 95.00th=[   45],
     | 99.00th=[   64], 99.50th=[   77], 99.90th=[  194], 99.95th=[  334],
     | 99.99th=[  766]
   bw (  KiB/s): min=61381, max=87960, per=99.98%, avg=79430.19, stdev=6158.62, samples=26
   iops        : min=15345, max=21990, avg=19857.38, stdev=1539.68, samples=26
  lat (nsec)   : 500=0.01%, 750=0.01%
  lat (usec)   : 4=0.01%, 10=0.06%, 20=0.15%, 50=97.25%, 100=2.29%
  lat (usec)   : 250=0.16%, 500=0.05%, 750=0.02%, 1000=0.01%
  lat (msec)   : 2=0.01%, 4=0.01%
  cpu          : usr=0.00%, sys=7.58%, ctx=0, majf=0, minf=0
  IO depths    : 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, >=64=0.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     issued rwts: total=262144,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1

Run status group 0 (all jobs):
   READ: bw=77.6MiB/s (81.4MB/s), 77.6MiB/s-77.6MiB/s (81.4MB/s-81.4MB/s), io=1024MiB (1074MB), run=13198-13198msec
```

Q1:

Sequential read:

```
Run status group 0 (all jobs):
   READ: bw=79.5MiB/s (83.4MB/s), 79.5MiB/s-79.5MiB/s (83.4MB/s-83.4MB/s), io=64.0MiB (67.1MB), run=805-805msec
```

Random read:

```
Run status group 1 (all jobs):
   READ: bw=28.3MiB/s (29.7MB/s), 28.3MiB/s-28.3MiB/s (29.7MB/s-29.7MB/s), io=64.0MiB (67.1MB), run=2260-2260msec
```

可以發現 sequential read 有更高的 bandwidth 和更少的 run time，雖然 SSD 不像 HDD 會有移動的讀寫頭，但在讀 page 的時候，sequential 可以照著順序讀取下去，而 random read 會需要尋找資料位址增加 overhead 或 latency。

Q2:

Sequential write:

```
Run status group 0 (all jobs):
  WRITE: bw=85.3MiB/s (89.5MB/s), 85.3MiB/s-85.3MiB/s (89.5MB/s-89.5MB/s), io=64.0MiB (67.1MB), run=750-750msec
```

Random write:

```
Run status group 1 (all jobs):
  WRITE: bw=85.2MiB/s (89.4MB/s), 85.2MiB/s-85.2MiB/s (89.4MB/s-89.4MB/s), io=64.0MiB (67.1MB), run=751-751msec
```

可以看到兩者差距十分輕微，只有 sequential 輕微的比 random 好一點點，差別很小是因為不論是 sequential 或是 random 在寫入時，都是去找 available block，經歷的流程是一樣的，因此 sequential 和 random 的差異不大。

Q3:

Forward write:

```
Run status group 0 (all jobs):
  WRITE: bw=81.8MiB/s (85.8MB/s), 81.8MiB/s-81.8MiB/s (85.8MB/s-85.8MB/s), io=64.0MiB (67.1MB), run=782-782msec
```

Backward write:

```
Run status group 1 (all jobs):
  WRITE: bw=84.5MiB/s (88.7MB/s), 84.5MiB/s-84.5MiB/s (88.7MB/s-88.7MB/s), io=64.0MiB (67.1MB), run=757-757msec
```

可以發現差距不大，如同 Q2 所說，write 的順序和效能的影響並不大，而 backward write 只是換了一個順序的 write 而已，因此結果差距會很小。

Q4_1:

Buffered read:

```
Run status group 0 (all jobs):
   READ: bw=120MiB/s (125MB/s), 120MiB/s-120MiB/s (125MB/s-125MB/s), io=64.0MiB (67.1MB), run=535-535msec
```

Nonbuffered read:

```
Run status group 1 (all jobs):
   READ: bw=92.4MiB/s (96.8MB/s), 92.4MiB/s-92.4MiB/s (96.8MB/s-96.8MB/s), io=64.0MiB (67.1MB), run=693-693msec
```

可以看到有用 buffer 的效果比較好，因為 read buffer 的速度相較於直接讀取硬碟的速度更快，且在 sequential read 時，buffer 可以做到 readahead，多讀幾個 page，因此兩者在讀取硬碟的次數就會有所差距，前者可以在多次 IO 才需要讀一次硬碟，並且大多時候都是讀取更快的 buffer，但後者卻不行，導致速度上的差異。

Q4_2:

Buffered random read:

```
Run status group 0 (all jobs):
   READ: bw=22.5MiB/s (23.6MB/s), 22.5MiB/s-22.5MiB/s (23.6MB/s-23.6MB/s), io=64.0MiB (67.1MB), run=2849-2849msec
```

Nonbuffered random read:

```
Run status group 1 (all jobs):
   READ: bw=26.7MiB/s (27.9MB/s), 26.7MiB/s-26.7MiB/s (27.9MB/s-27.9MB/s), io=64.0MiB (67.1MB), run=2401-2401msec
```

可以看到兩者是差不多的，因為 random read 沒辦法很好的發揮 buffer 的用處，random read 很可能每次的 IO 都需要取不同位置的資料，導致之前取出來在 buffer 裡的資料都沒有用，需要重新尋找資料位置，因此有沒有 buffer 的影響不大。

Q5:

```
Run status group 0 (all jobs):
   READ: bw=1246MiB/s (1306MB/s), 1246MiB/s-1246MiB/s (1306MB/s-1306MB/s), io=1024MiB (1074MB), run=822-822msec

Run status group 1 (all jobs):
   READ: bw=1739MiB/s (1823MB/s), 1739MiB/s-1739MiB/s (1823MB/s-1823MB/s), io=1024MiB (1074MB), run=589-589msec

Run status group 2 (all jobs):
   READ: bw=1762MiB/s (1848MB/s), 1762MiB/s-1762MiB/s (1848MB/s-1848MB/s), io=1024MiB (1074MB), run=581-581msec

Run status group 3 (all jobs):
   READ: bw=1759MiB/s (1845MB/s), 1759MiB/s-1759MiB/s (1845MB/s-1845MB/s), io=1024MiB (1074MB), run=582-582msec

Run status group 4 (all jobs):
   READ: bw=1652MiB/s (1732MB/s), 1652MiB/s-1652MiB/s (1732MB/s-1732MB/s), io=1024MiB (1074MB), run=620-620msec
```

可以看到 bandwidth 沒有特別的趨勢，百分比的 offset 可以視為不同的 LBA number，不像 HDD 會將 LBA 轉成 CHS 去讀寫不同區塊，SSD 是透過 table 去轉換，因此 LBA 實際對應到的 PBA 不是固定的，而速度差異的原因，可能是存放內容或是對應到的實際硬體擺放位置的差別，又或是 SSD 設計方式讓其對應至不同的區間，而形成速度差異。

Q6:

Size=64m

```
32m: (g=0):
16m: (g=1):
8m: (g=2):
4m: (g=3):
2m: (g=4):
1m: (g=5):
64k: (g=6):
16k: (g=7):
4k: (g=8):
1k: (g=9):
```

```
Run status group 0 (all jobs):
   READ: bw=1143MiB/s (1198MB/s), 1143MiB/s-1143MiB/s (1198MB/s-1198MB/s), io=64.0MiB (67.1MB), run=56-56msec

Run status group 1 (all jobs):
   READ: bw=1362MiB/s (1428MB/s), 1362MiB/s-1362MiB/s (1428MB/s-1428MB/s), io=64.0MiB (67.1MB), run=47-47msec

Run status group 2 (all jobs):
   READ: bw=1333MiB/s (1398MB/s), 1333MiB/s-1333MiB/s (1398MB/s-1398MB/s), io=64.0MiB (67.1MB), run=48-48msec

Run status group 3 (all jobs):
   READ: bw=1333MiB/s (1398MB/s), 1333MiB/s-1333MiB/s (1398MB/s-1398MB/s), io=64.0MiB (67.1MB), run=48-48msec

Run status group 4 (all jobs):
   READ: bw=1280MiB/s (1342MB/s), 1280MiB/s-1280MiB/s (1342MB/s-1342MB/s), io=64.0MiB (67.1MB), run=50-50msec

Run status group 5 (all jobs):
   READ: bw=1085MiB/s (1137MB/s), 1085MiB/s-1085MiB/s (1137MB/s-1137MB/s), io=64.0MiB (67.1MB), run=59-59msec

Run status group 6 (all jobs):
   READ: bw=771MiB/s (809MB/s), 771MiB/s-771MiB/s (809MB/s-809MB/s), io=64.0MiB (67.1MB), run=83-83msec

Run status group 7 (all jobs):
   READ: bw=327MiB/s (342MB/s), 327MiB/s-327MiB/s (342MB/s-342MB/s), io=64.0MiB (67.1MB), run=196-196msec

Run status group 8 (all jobs):
   READ: bw=89.0MiB/s (93.3MB/s), 89.0MiB/s-89.0MiB/s (93.3MB/s-93.3MB/s), io=64.0MiB (67.1MB), run=719-719msec

Run status group 9 (all jobs):
   READ: bw=23.8MiB/s (25.0MB/s), 23.8MiB/s-23.8MiB/s (25.0MB/s-25.0MB/s), io=64.0MiB (67.1MB), run=2689-2689msec
```

Q6_1:

可以看到後兩個也就是 4k 和 1k 的差距是十分大的，有可能是因為當 block size 越大時，每次可取得資料量變多，所需的次數變少，可以更快的取完所需的資料。

Q6_2:

可以看到大概在 2m 及 4m 的時候差不多就來到了最快的上限，因此我會選擇 4m。而遇到瓶頸的原因，我猜測有可能是系統還是會將大 IO 拆成小的 IO，並將多個 IO 排序合併減少 seek time，剛好在 2m 或 4m 的 IO size 可以有最好的表現，也有可能是 bandwidth 已經到達硬體的極限，因此不再上升。

Q7:

```
job: (g=0): rw=read, bs=(R) 8192KiB-8192KiB, (W) 8192KiB-8192KiB, (T) 8192KiB-8192KiB, ioengine=windowsaio, iodepth=1
fio-3.39
Starting 1 thread

job: (groupid=0, jobs=1): err= 0: pid=10348: Tue Apr 8 01:40:29 2025
  read: IOPS=183, BW=1471MiB/s (1543MB/s)(1024MiB/696msec)
    slat (usec): min=192, max=1344, avg=254.27, stdev=107.94
    clat (usec): min=4026, max=7046, avg=5176.87, stdev=309.03
     lat (usec): min=4333, max=7564, avg=5431.14, stdev=355.72
    clat percentiles (usec):
     |  1.00th=[ 4178],  5.00th=[ 4555], 10.00th=[ 5145], 20.00th=[ 5145],
     | 30.00th=[ 5145], 40.00th=[ 5145], 50.00th=[ 5211], 60.00th=[ 5211],
     | 70.00th=[ 5211], 80.00th=[ 5211], 90.00th=[ 5342], 95.00th=[ 5473],
     | 99.00th=[ 6194], 99.50th=[ 7046], 99.90th=[ 7046], 99.95th=[ 7046],
     | 99.99th=[ 7046]
   bw (  MiB/s): min= 1456, max= 1456, per=98.96%, avg=1456.00, stdev= 0.00, samples=1
   iops        : min=  182, max=  182, avg=182.00, stdev= 0.00, samples=1
  lat (msec)   : 10=100.00%
  cpu          : usr=0.00%, sys=0.00%, ctx=0, majf=0, minf=0
  IO depths    : 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, >=64=0.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     issued rwts: total=128,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1

Run status group 0 (all jobs):
   READ: bw=1471MiB/s (1543MB/s), 1471MiB/s-1471MiB/s (1543MB/s-1543MB/s), io=1024MiB (1074MB), run=696-696msec
```

根據 Q6 的實驗，我選擇 4m 的 block size，並只花了 696msec 就完成 1g 的 read。

Bonus1: 因為沒有其他種類的硬碟，因此只有分析

在 HDD 上，sequential read 比 random read 理論上會快上許多，因為 HDD 受到硬體的限制，在 read 時需要將讀取頭對到相應的 sector 上，並且磁碟是隨時處於轉動的狀態，這個過程若是 sequential read 的話可以照順序讀取，但如果是 random read 的話，則是在讀取完一區之後要接著找到下一區進行讀取，要花費較多時間 seek time 上。

Bonus2:

Capacities [4]	512GB
Seq Read (MB/s)[5]	2,200
Seq Write (MB/s)[5]	1,070

```
Run status group 0 (all jobs):
   READ: bw=1604MiB/s (1682MB/s), 1604MiB/s-1604MiB/s (1682MB/s-1682MB/s), io=4096MiB (4295MB), run=2553-2553msec

Run status group 1 (all jobs):
  WRITE: bw=147MiB/s (154MB/s), 147MiB/s-147MiB/s (154MB/s-154MB/s), io=4096MiB (4295MB), run=27833-27833msec
```

可以發現差的蠻多的，有可能是因為 SPEC 上標註的是全新的 SSD 的速度，但我測試的裝置已經使用多年了，因此會有磨損的情況發生，此外，還有可能是因為我設定的 fio job 沒有辦法完整的跑到硬體的上限，也就是說很可能有些優化或是排程沒有使用，又或是資料的實際硬體位置影響到其速度。