Preface:

I first tried running the experiments on a virtual machine (VM), but no matter what kind of storage I used—NVMe, SSD, or HDD—the VM showed them all as QEMU virtual disks. This caused some weird performance issues, and the I/O results didn't really reflect how the actual hardware would behave.

```
ben@memory:~$ lsblk -o NAME,MODEL,SIZE
NAME                      MODEL          SIZE
sda                       QEMU HARDDISK  128G
├─sda1                                     1M
├─sda2                                     2G
└─sda3                                   126G
  └─ubuntu--vg-ubuntu--lv                 63G
sr0                       QEMU DVD-ROM     3G
```

To get around the virtualization problem, I switched to a Raspberry Pi 4 and connected SATA hard drive using a USB 3.0 to SATA adapter. This way, I could work directly with the physical disk and get more accurate results based on the drive's real performance.

```
ben@raspberrypi:~ $ lsblk -o NAME,MODEL,SIZE
NAME        MODEL                          SIZE
sda         Hitachi HTS543225L9A300 232.9G
├─sda1                                      50M
├─sda2                                   232.3G
└─sda3                                     546M
```

So, for the rest of the assignment, I ran all the tests in this physical setup.

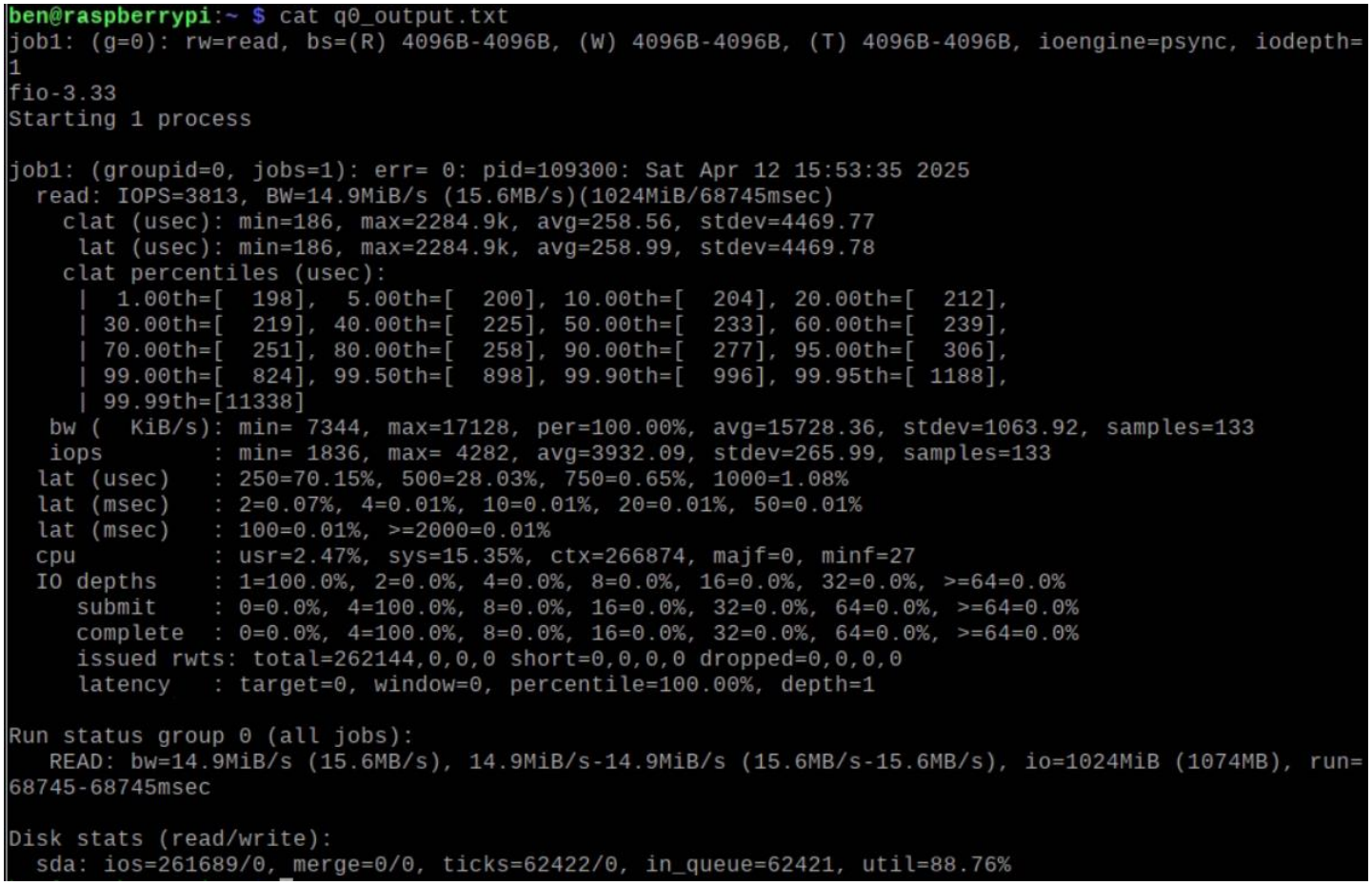
Device used for testing Q1-Q7:

- **Model:** Hitachi HTS543225L9A300 (HDD)
- **Connection:** USB 3.0 to SATA adapter
- **Interface:** /dev/sda

Device used for testing Bonus:

- **Model:** INTEL SSDSC2KW512G8 (SSD)
- **Connection:** USB 3.0 to SATA adapter
- **Interface:** /dev/sdb

Q0. Example



I ran a sequential read test on my Hitachi HTS543225L9A300 (HDD), which is just a basic 2.5" SATA hard drive that spins at 5400 RPM. Here's what I got:

- **Bandwidth:** 14.9 MiB/s (about 15.6 MiB/s)
- **IOPS:** 3813
- **Average Latency:** around 258.99 µs
- **99.99th Percentile Latency:** 11338 µs
- **Disk Usage:** about 88.76%

These results are what I'd expect from a traditional hard drive. Because it's a mechanical device, the performance is limited by physical movement—like the time it takes for the disk to spin and for the read/write head to seek the right position. That's why the IOPS and bandwidth are relatively low, and the latency numbers are quite high and spread out.

The high disk utilization (almost 89%) also suggests that the drive was working at full capacity during the test, which makes sense given the mechanical overhead.

In short, the results align for this type of drive, and they confirm that my Raspberry Pi 4 setup with direct disk access is capturing real hardware behavior accurately.

Q1. Read vs. Randread

Is there any significant difference between read and randread? Why or why not? Please justify your answer in brief.

```
seq_read: (groupid=0, jobs=1): err= 0: pid=102410: Sat Apr 12 11:38:06 2025
  read: IOPS=2607, BW=10.2MiB/s (10.7MB/s)(64.0MiB/6284msec)
    clat (usec): min=186, max=2282.3k, avg=380.44, stdev=17830.18
     lat (usec): min=187, max=2282.3k, avg=380.80, stdev=17830.19
    clat percentiles (usec):
     |  1.00th=[  196],  5.00th=[  198], 10.00th=[  200], 20.00th=[  204],
     | 30.00th=[  210], 40.00th=[  217], 50.00th=[  225], 60.00th=[  233],
     | 70.00th=[  245], 80.00th=[  253], 90.00th=[  265], 95.00th=[  285],
     | 99.00th=[  816], 99.50th=[  881], 99.90th=[  979], 99.95th=[ 1074],
     | 99.99th=[24511]
   bw (  KiB/s): min=11048, max=17104, per=100.00%, avg=15790.38, stdev=2047.47, samples=8
   iops        : min= 2762, max= 4276, avg=3947.38, stdev=511.78, samples=8
  lat (usec)   : 250=74.30%, 500=24.11%, 750=0.45%, 1000=1.06%
  lat (msec)   : 2=0.06%, 4=0.01%, 10=0.01%, 50=0.01%, >=2000=0.01%
  cpu          : usr=1.45%, sys=9.09%, ctx=16536, majf=0, minf=26
  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=16384,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1
```

```
rand_read: (groupid=1, jobs=1): err= 0: pid=102414: Sat Apr 12 11:38:06 2025
  read: IOPS=135, BW=542KiB/s (555kB/s)(64.0MiB/120879msec)
    clat (usec): min=218, max=56677, avg=7359.41, stdev=4601.30
     lat (usec): min=219, max=56679, avg=7360.96, stdev=4601.33
    clat percentiles (usec):
     |  1.00th=[  269],  5.00th=[  318], 10.00th=[ 1156], 20.00th=[ 3458],
     | 30.00th=[ 4686], 40.00th=[ 5997], 50.00th=[ 7308], 60.00th=[ 8586],
     | 70.00th=[ 9896], 80.00th=[11076], 90.00th=[12387], 95.00th=[13042],
     | 99.00th=[20055], 99.50th=[32113], 99.90th=[35914], 99.95th=[40633],
     | 99.99th=[43779]
   bw (  KiB/s): min=  408, max= 1280, per=99.23%, avg=538.19, stdev=77.25, samples=241
   iops        : min=  102, max=  320, avg=134.52, stdev=19.29, samples=241
  lat (usec)   : 250=0.12%, 500=9.61%, 750=0.18%, 1000=0.05%
  lat (msec)   : 2=0.47%, 4=13.98%, 10=46.77%, 20=27.83%, 50=0.99%
  lat (msec)   : 100=0.01%
  cpu          : usr=0.39%, sys=1.57%, ctx=16803, majf=0, minf=24
  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=16384,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1
```

Metric	Sequential Read	Random Read	Difference / Note
Bandwidth	10.7 MiB/s	555 KB/s	~**19x slower** in random read
IOPS	2607	135	~**20x fewer** operations
Avg Latency	381 µs	7361 µs	Random read is ~**19x slower**
99.99% Latency	24,511 µs	43,779 µs	Much higher latency tail in randread

There's a dramatic difference between sequential and random reads. When reading sequentially, the disk head moves smoothly and doesn't need to jump around, so everything is faster. But with random reads, the head has to keep repositioning, which adds a lot of delay.

These results match what I'd expect from a mechanical hard drive — random access is just way more expensive due to physical movement.

Q2. Write vs. Randwrite

Is there any significant difference between write and randwrite? Why or why not? Please justify your answer in brief.

```
write: (groupid=0, jobs=1): err= 0: pid=105721: Sat Apr 12 13:48:42 2025
  write: IOPS=1031, BW=4125KiB/s (4224kB/s)(64.0MiB/15889msec); 0 zone resets
    clat (usec): min=437, max=47251, avg=963.43, stdev=1231.95
     lat (usec): min=437, max=47251, avg=964.36, stdev=1231.95
    clat percentiles (usec):
     |  1.00th=[  498],  5.00th=[  502], 10.00th=[  510], 20.00th=[  562],
     | 30.00th=[  578], 40.00th=[  619], 50.00th=[  635], 60.00th=[  693],
     | 70.00th=[  709], 80.00th=[  734], 90.00th=[  840], 95.00th=[ 4293],
     | 99.00th=[ 4621], 99.50th=[ 4686], 99.90th=[ 5080], 99.95th=[11994],
     | 99.99th=[38011]
   bw (  KiB/s): min= 3735, max= 4512, per=100.00%, avg=4129.00, stdev=190.29, samples=31
   iops        : min=  933, max= 1128, avg=1032.16, stdev=47.68, samples=31
  lat (usec)   : 500=1.79%, 750=82.20%, 1000=7.07%
  lat (msec)   : 2=0.57%, 4=0.03%, 10=8.28%, 20=0.04%, 50=0.02%
  cpu          : usr=0.87%, sys=8.16%, ctx=16748, majf=0, minf=25
  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=0,16384,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1
```

```
randwrite: (groupid=1, jobs=1): err= 0: pid=105733: Sat Apr 12 13:48:42 2025
  write: IOPS=862, BW=3449KiB/s (3532kB/s)(64.0MiB/19002msec); 0 zone resets
    clat (usec): min=598, max=140133, avg=1150.89, stdev=1695.45
     lat (usec): min=599, max=140136, avg=1152.05, stdev=1695.45
    clat percentiles (usec):
     |  1.00th=[  644],  5.00th=[  685], 10.00th=[  701], 20.00th=[  725],
     | 30.00th=[  783], 40.00th=[  816], 50.00th=[  824], 60.00th=[  840],
     | 70.00th=[  857], 80.00th=[  881], 90.00th=[ 1004], 95.00th=[ 4490],
     | 99.00th=[ 4817], 99.50th=[ 4883], 99.90th=[ 5604], 99.95th=[14877],
     | 99.99th=[47973]
   bw (  KiB/s): min= 2240, max= 3720, per=99.89%, avg=3445.84, stdev=230.91, samples=37
   iops        : min=  560, max=  930, avg=861.35, stdev=57.70, samples=37
  lat (usec)   : 750=24.67%, 1000=65.23%
  lat (msec)   : 2=1.70%, 4=0.04%, 10=8.28%, 20=0.04%, 50=0.04%
  lat (msec)   : 250=0.01%
  cpu          : usr=1.15%, sys=6.13%, ctx=16981, majf=0, minf=23
  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=0,16384,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1
```

Metric	Sequential Write	Random Write	Difference / Note
Bandwidth	4124 KB/s	3449 KB/s	~16% lower in random write
IOPS	1032	861	Slightly lower in random write
Avg Latency	964 µs	1152 µs	Random write is ~20% slower
99.99% Latency	38,011 µs	47,973 µs	Higher latency spikes in random write

It's **not as dramatic as what we saw in the read tests**.

Even if we use direct=1 to turn off the OS-level cache, **hard drives still have their own built-in write cache**. This means they can **temporarily hold random write data**, and then **rearrange it into a smoother, more sequential order**, which helps keep performance from dropping too much.

Also, **disk controllers are smart** — they can **merge nearby write operations** so the disk head doesn't have to move as much. This further **hides the cost of random writes**.

On the other hand, **random reads can't really be optimized** like that — the disk head has to **physically jump to each location**, which causes much higher delay and lower speed.

If you want to check whether drive's internal write cache is turned on, you can use this command:

```
ben@raspberrypi:~ $ sudo hdparm -W /dev/sda

/dev/sda:
 write-caching =  1 (on)
```

Q3: Forward write vs. Backward write

Is there any significant difference between forward and backward write? Why or why not? Please justify your answer in brief.

```
forward: (groupid=0, jobs=1): err= 0: pid=105952: Sat Apr 12 13:56:40 2025
  write: IOPS=1033, BW=4134KiB/s (4233kB/s)(64.0MiB/15853msec); 0 zone resets
    clat (usec): min=437, max=48798, avg=961.26, stdev=1242.32
     lat (usec): min=437, max=48798, avg=962.21, stdev=1242.32
    clat percentiles (usec):
     |  1.00th=[  498],  5.00th=[  502], 10.00th=[  510], 20.00th=[  562],
     | 30.00th=[  578], 40.00th=[  603], 50.00th=[  635], 60.00th=[  660],
     | 70.00th=[  693], 80.00th=[  717], 90.00th=[  840], 95.00th=[ 4359],
     | 99.00th=[ 4686], 99.50th=[ 4752], 99.90th=[ 5276], 99.95th=[10290],
     | 99.99th=[36439]
   bw (  KiB/s): min= 3824, max= 4512, per=100.00%, avg=4140.10, stdev=170.55, samples=31
   iops        : min=  956, max= 1128, avg=1034.94, stdev=42.72, samples=31
  lat (usec)   : 500=1.82%, 750=84.94%, 1000=4.19%
  lat (msec)   : 2=0.65%, 4=0.05%, 10=8.29%, 20=0.04%, 50=0.02%
  cpu          : usr=0.74%, sys=8.30%, ctx=16727, majf=0, minf=25
  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=0,16384,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1
```

```
backward: (groupid=1, jobs=1): err= 0: pid=105963: Sat Apr 12 13:56:40 2025
  write: IOPS=950, BW=3801KiB/s (3892kB/s)(64.0MiB/17241msec); 0 zone resets
    clat (usec): min=552, max=146920, avg=1045.80, stdev=1937.30
     lat (usec): min=552, max=146921, avg=1046.77, stdev=1937.37
    clat percentiles (usec):
     |  1.00th=[   594],  5.00th=[   619], 10.00th=[   627], 20.00th=[   660],
     | 30.00th=[   676], 40.00th=[   685], 50.00th=[   701], 60.00th=[   709],
     | 70.00th=[   725], 80.00th=[   750], 90.00th=[   898], 95.00th=[  4359],
     | 99.00th=[  4752], 99.50th=[  4817], 99.90th=[  6259], 99.95th=[ 12518],
     | 99.99th=[130548]
   bw (  KiB/s): min= 2752, max= 4208, per=100.00%, avg=3803.71, stdev=269.71, samples=34
   iops        : min=  688, max= 1052, avg=950.88, stdev=67.43, samples=34
  lat (usec)   : 750=79.52%, 1000=11.25%
  lat (msec)   : 2=0.85%, 4=0.01%, 10=8.27%, 20=0.05%, 50=0.02%
  lat (msec)   : 250=0.01%
  cpu          : usr=0.81%, sys=5.74%, ctx=16826, majf=0, minf=25
  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=0,16384,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1
```

Metric	Forward Write	Backward Write	Difference / Note
Bandwidth	4134 KiB/s	3801 KiB/s	~8% lower in backward write
IOPS	1034	951	Slightly lower in backward write
Avg Latency	962 µs	1047 µs	Backward write is ~9% slower
99.99% Latency	10,290 µs	12,518 µs	Slightly higher latency tail in backward write

There's **no big difference** between forward write and backward write in this test.

Even though the order is reversed (like writing from block 10 to 1 instead of 1 to 10), it's still basically a **sequential write pattern**. That means the data is written in a smooth, continuous way on the disk.

Since there's **no random jumping around** (like in random write), the disk head doesn't need to move much, so the performance stays **pretty much the same**.

Q4: Buffered read vs. Nonbuffered read

1. Is there any significant difference between buffered and nonbuffered sequential read? Why or why not? Please justify your answer in brief.

```
buffered_seqread: (groupid=0, jobs=1): err= 0: pid=106145: Sat Apr 12 14:06:24 2025
  read: IOPS=2310, BW=9243KiB/s (9465kB/s)(64.0MiB/7090msec)
    clat (usec): min=2, max=2916.4k, avg=431.77, stdev=30955.30
     lat (usec): min=2, max=2916.4k, avg=431.92, stdev=30955.31
    clat percentiles (usec):
     |  1.00th=[      3],  5.00th=[      3], 10.00th=[      3],
     | 20.00th=[      3], 30.00th=[      3], 40.00th=[      3],
     | 50.00th=[      3], 60.00th=[      3], 70.00th=[      4],
     | 80.00th=[      4], 90.00th=[      6], 95.00th=[      6],
     | 99.00th=[     10], 99.50th=[     15], 99.90th=[    147],
     | 99.95th=[    693], 99.99th=[2298479]
   bw (  KiB/s): min=10912, max=50541, per=100.00%, avg=27345.50, stdev=18149.88, samples=4
   iops        : min= 2728, max=12635, avg=6836.25, stdev=4537.34, samples=4
  lat (usec)   : 4=81.52%, 10=17.54%, 20=0.66%, 50=0.10%, 100=0.05%
  lat (usec)   : 250=0.05%, 500=0.01%, 750=0.02%
  lat (msec)   : 4=0.01%, 20=0.01%, 50=0.01%, 500=0.01%, 2000=0.01%
  lat (msec)   : >=2000=0.01%
  cpu          : usr=0.25%, sys=2.50%, ctx=736, majf=0, minf=27
  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=16384,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1
```

```
nonbuffered_seqread: (groupid=1, jobs=1): err= 0: pid=106152: Sat Apr 12 14:06:24 2025
  read: IOPS=3848, BW=15.0MiB/s (15.8MB/s)(64.0MiB/4257msec)
    clat (usec): min=191, max=279993, avg=253.01, stdev=2229.52
     lat (usec): min=191, max=279995, avg=253.28, stdev=2229.54
    clat percentiles (usec):
     |  1.00th=[  198],  5.00th=[  198], 10.00th=[  200], 20.00th=[  202],
     | 30.00th=[  204], 40.00th=[  206], 50.00th=[  210], 60.00th=[  219],
     | 70.00th=[  239], 80.00th=[  253], 90.00th=[  260], 95.00th=[  277],
     | 99.00th=[  807], 99.50th=[  881], 99.90th=[ 1029], 99.95th=[ 1074],
     | 99.99th=[55313]
   bw (  KiB/s): min= 6657, max=17120, per=100.00%, avg=15482.12, stdev=3616.59, samples=8
   iops        : min= 1664, max= 4280, avg=3870.38, stdev=904.21, samples=8
  lat (usec)   : 250=76.84%, 500=21.62%, 750=0.47%, 1000=0.97%
  lat (msec)   : 2=0.09%, 4=0.01%, 100=0.01%, 500=0.01%
  cpu          : usr=1.15%, sys=13.39%, ctx=16621, majf=0, minf=28
  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=16384,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1
```

Metric	Buffered Seq Read	Nonbuffered Seq Read	Difference / Note
Bandwidth	9243 KiB/s (~9.0 MiB/s)	15482 KiB/s (~15.1 MiB/s)	~67% faster in nonbuffered read
IOPS	2310	3848	Higher in nonbuffered read
Avg Latency	432 µs	253 µs	Lower latency in nonbuffered read
99.99% Latency	~2.29s	~553 ms	Much better tail latency in nonbuffered read

Yes, there is a significant difference. Nonbuffered sequential read is much faster than buffered read in this test.

- **Buffered read overhead:**
  Buffered read involves extra work, such as:
  - Memory copy between kernel and user space
  - Page cache checks
- **Nonbuffered read benefit:**
  In contrast, nonbuffered read (using direct=1) accesses the disk directly without going through the operating system's cache. This bypasses the additional overhead, resulting in higher bandwidth and lower latency.

This might seem counterintuitive, since buffered reads are usually expected to be faster due to the benefits of page cache and readahead mechanisms provided by most operating systems. However, in our case, several factors likely influenced the outcome:

- **Higher overhead in buffered reads:** Buffered mode introduces extra memory copy between user and kernel space, cache page lookup, and management overhead, especially with small block sizes and low concurrency.
- **I/O engine limitations:** The test used ioengine=psync, which is synchronous and single-threaded. This restricts parallelism and limits throughput, particularly in buffered mode.
- **Direct read advantages:** The nonbuffered test (direct=1) bypassed the OS cache and directly accessed the disk, reducing overhead. The block size used aligned well with the device's physical block boundaries, which may have further enhanced performance.
- **Hardware constraints:** The Raspberry Pi's modest CPU and memory bandwidth may have further amplified the overhead associated with buffered reads.

In summary, **buffered reads are not always faster**—especially on low-power systems or when using suboptimal I/O settings. Our experiment shows that **I/O engine, block size, and concurrency** have a significant impact, and careful tuning is critical to achieving optimal performance.

2. Replace **sequential read** with **random read**. Is there any significant difference between buffered and nonbuffered random read? Why or why not? Please justify your answer in brief.

```
buffered_randread: (groupid=2, jobs=1): err= 0: pid=106155: Sat Apr 12 14:06:24 2025
  read: IOPS=135, BW=541KiB/s (554kB/s)(64.0MiB/121132msec)
    clat (usec): min=236, max=161054, avg=7373.74, stdev=4719.20
     lat (usec): min=237, max=161056, avg=7375.23, stdev=4719.22
    clat percentiles (usec):
     |  1.00th=[  277],  5.00th=[  334], 10.00th=[ 1663], 20.00th=[ 3523],
     | 30.00th=[ 4817], 40.00th=[ 6063], 50.00th=[ 7308], 60.00th=[ 8586],
     | 70.00th=[ 9765], 80.00th=[11076], 90.00th=[12387], 95.00th=[13042],
     | 99.00th=[20055], 99.50th=[32113], 99.90th=[35390], 99.95th=[40109],
     | 99.99th=[43779]
   bw (  KiB/s): min=  400, max= 1466, per=99.62%, avg=539.29, stdev=96.90, samples=242
   iops        : min=  100, max=  366, avg=134.71, stdev=24.19, samples=242
  lat (usec)   : 250=0.07%, 500=9.55%, 750=0.25%, 1000=0.05%
  lat (msec)   : 2=0.40%, 4=13.46%, 10=47.89%, 20=27.33%, 50=0.99%
  lat (msec)   : 250=0.01%
  cpu          : usr=0.42%, sys=1.80%, ctx=16897, majf=0, minf=25
  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=16384,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1
```

```
nonbuffered_randread: (groupid=3, jobs=1): err= 0: pid=106203: Sat Apr 12 14:06:24 2025
  read: IOPS=135, BW=544KiB/s (557kB/s)(64.0MiB/120581msec)
    clat (usec): min=218, max=43999, avg=7338.93, stdev=4562.35
     lat (usec): min=219, max=44001, avg=7340.33, stdev=4562.37
    clat percentiles (usec):
     |  1.00th=[  269],  5.00th=[  314], 10.00th=[ 1549], 20.00th=[ 3458],
     | 30.00th=[ 4686], 40.00th=[ 5997], 50.00th=[ 7308], 60.00th=[ 8586],
     | 70.00th=[ 9765], 80.00th=[11076], 90.00th=[12387], 95.00th=[12911],
     | 99.00th=[19530], 99.50th=[32113], 99.90th=[34866], 99.95th=[35390],
     | 99.99th=[42206]
   bw (  KiB/s): min=  423, max= 1213, per=98.99%, avg=538.34, stdev=65.30, samples=240
   iops        : min=  105, max=  303, avg=134.57, stdev=16.32, samples=240
  lat (usec)   : 250=0.28%, 500=9.45%, 750=0.11%, 1000=0.05%
  lat (msec)   : 2=0.57%, 4=13.95%, 10=47.14%, 20=27.44%, 50=0.99%
  cpu          : usr=0.40%, sys=1.57%, ctx=16899, majf=0, minf=23
  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=16384,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1
```

Metric	Buffered Rand Read	Nonbuffered Rand Read	Difference / Note
Bandwidth	541 KiB/s	544 KiB/s	Almost the same
IOPS	135	135	Identical
Avg Latency	~7373 µs	~7340 µs	No significant change
99.99% Latency	~44 ms	~35 ms	Slightly better in nonbuffered, but similar

There is **no significant difference** between buffered and nonbuffered **random reads**.

- This is because random reads are dominated by **physical disk seek time**.
- **Caching provides little benefit**, since each request targets a different disk location.
- Therefore, whether the OS cache is used or bypassed has **minimal impact**.

Q5: LBA

Is there any bandwidth trend between these jobs? Why or why not? Please justify your answer in brief. You can also experiment with the size of the file.

```
job1: (groupid=0, jobs=1): err= 0: pid=106469: Sat Apr 12 14:13:52 2025
  read: IOPS=27, BW=27.4MiB/s (28.7MB/s)(1024MiB/37384msec)
    clat (msec): min=26, max=2356, avg=36.47, stdev=73.85
     lat (msec): min=26, max=2356, avg=36.47, stdev=73.85
    clat percentiles (msec):
     |  1.00th=[   28],  5.00th=[   28], 10.00th=[   28], 20.00th=[   28],
     | 30.00th=[   28], 40.00th=[   28], 50.00th=[   29], 60.00th=[   31],
     | 70.00th=[   32], 80.00th=[   39], 90.00th=[   50], 95.00th=[   72],
     | 99.00th=[   85], 99.50th=[  106], 99.90th=[  116], 99.95th=[ 2366],
     | 99.99th=[ 2366]
   bw (  KiB/s): min=12312, max=38989, per=100.00%, avg=29727.11, stdev=7678.31, samples=70
   iops        : min=   12, max=   38, avg=28.99, stdev= 7.51, samples=70
  lat (msec)   : 50=90.14%, 100=9.28%, 250=0.49%, >=2000=0.10%
  cpu          : usr=0.16%, sys=1.97%, ctx=1180, majf=0, minf=281
  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=1024,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1
```

```
job2: (groupid=1, jobs=1): err= 0: pid=106486: Sat Apr 12 14:13:52 2025
  read: IOPS=39, BW=39.1MiB/s (41.0MB/s)(1024MiB/26207msec)
    clat (usec): min=22630, max=58159, avg=25553.56, stdev=2626.69
     lat (usec): min=22632, max=58161, avg=25555.42, stdev=2626.70
    clat percentiles (usec):
     |  1.00th=[22938],  5.00th=[22938], 10.00th=[24773], 20.00th=[24773],
     | 30.00th=[24773], 40.00th=[24773], 50.00th=[24773], 60.00th=[24773],
     | 70.00th=[25822], 80.00th=[26084], 90.00th=[26608], 95.00th=[26870],
     | 99.00th=[35914], 99.50th=[42206], 99.90th=[46924], 99.95th=[57934],
     | 99.99th=[57934]
   bw (  KiB/s): min=36864, max=43094, per=100.00%, avg=40045.53, stdev=1603.46, samples=51
   iops        : min=   36, max=   42, avg=39.06, stdev= 1.57, samples=51
  lat (msec)   : 50=99.90%, 100=0.10%
  cpu          : usr=0.14%, sys=2.95%, ctx=1109, majf=0, minf=279
  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=1024,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1
```

```
job3: (groupid=2, jobs=1): err= 0: pid=106500: Sat Apr 12 14:13:52 2025
  read: IOPS=39, BW=39.5MiB/s (41.5MB/s)(1024MiB/25904msec)
    clat (usec): min=20020, max=88775, avg=25256.70, stdev=9039.67
     lat (usec): min=20022, max=88777, avg=25258.86, stdev=9039.68
    clat percentiles (usec):
     |  1.00th=[20055],  5.00th=[20055], 10.00th=[20055], 20.00th=[20317],
     | 30.00th=[22152], 40.00th=[22152], 50.00th=[22152], 60.00th=[22152],
     | 70.00th=[23725], 80.00th=[23987], 90.00th=[33162], 95.00th=[44303],
     | 99.00th=[66323], 99.50th=[76022], 99.90th=[86508], 99.95th=[88605],
     | 99.99th=[88605]
   bw (  KiB/s): min=20480, max=49250, per=99.82%, avg=40406.06, stdev=8080.73, samples=51
   iops        : min=   20, max=   48, avg=39.41, stdev= 7.88, samples=51
  lat (msec)   : 50=96.88%, 100=3.12%
  cpu          : usr=0.20%, sys=2.95%, ctx=1112, majf=0, minf=278
  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=1024,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1
```

```
job4: (groupid=3, jobs=1): err= 0: pid=106511: Sat Apr 12 14:13:52 2025
  read: IOPS=47, BW=47.5MiB/s (49.8MB/s)(1024MiB/21574msec)
    clat (usec): min=18633, max=66584, avg=21029.99, stdev=2991.54
     lat (usec): min=18635, max=66586, avg=21031.82, stdev=2991.71
    clat percentiles (usec):
     |  1.00th=[19006],  5.00th=[19006], 10.00th=[19006], 20.00th=[19006],
     | 30.00th=[19268], 40.00th=[20579], 50.00th=[20841], 60.00th=[21103],
     | 70.00th=[21103], 80.00th=[22152], 90.00th=[22414], 95.00th=[22676],
     | 99.00th=[32113], 99.50th=[38536], 99.90th=[43254], 99.95th=[66323],
     | 99.99th=[66323]
   bw (  KiB/s): min=45146, max=51302, per=100.00%, avg=48641.40, stdev=1784.37, samples=42
   iops        : min=   44, max=   50, avg=47.43, stdev= 1.78, samples=42
  lat (msec)   : 20=31.93%, 50=67.97%, 100=0.10%
  cpu          : usr=0.23%, sys=3.42%, ctx=1124, majf=0, minf=278
  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=1024,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1
```

```
job5: (groupid=4, jobs=1): err= 0: pid=106520: Sat Apr 12 14:13:52 2025
  read: IOPS=47, BW=47.3MiB/s (49.6MB/s)(1024MiB/21629msec)
    clat (usec): min=17185, max=86647, avg=21098.18, stdev=6092.74
     lat (usec): min=17187, max=86648, avg=21100.25, stdev=6092.72
    clat percentiles (usec):
     |  1.00th=[17957],  5.00th=[17957], 10.00th=[17957], 20.00th=[18220],
     | 30.00th=[18220], 40.00th=[19530], 50.00th=[20055], 60.00th=[20055],
     | 70.00th=[20055], 80.00th=[21365], 90.00th=[23987], 95.00th=[31065],
     | 99.00th=[43779], 99.50th=[53216], 99.90th=[73925], 99.95th=[86508],
     | 99.99th=[86508]
   bw (  KiB/s): min=36864, max=55296, per=100.00%, avg=48528.14, stdev=4297.25, samples=43
   iops        : min=   36, max=   54, avg=47.35, stdev= 4.19, samples=43
  lat (msec)   : 20=62.70%, 50=36.52%, 100=0.78%
  cpu          : usr=0.13%, sys=3.35%, ctx=1175, majf=0, minf=278
  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=1024,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1
```

Job	Offset (%)	Bandwidth (MiB/s)	IOPS	Avg Latency (µs)
job1	80%	27.4	28	36,470
job2	60%	39.1	39	25,555
job3	40%	39.5	39	25,257
job4	20%	47.5	47	21,038
job5	0%	47.3	47	21,108

The lower the LBA (Logical Block Address), the higher the bandwidth and lower the latency. As the offset increases (from job5 → job1), bandwidth **decreases** and latency **increases**.

On most hard disk drives (HDDs), **outer tracks (lower LBA regions) have more sectors per track** and therefore:

- **Higher data density**
- **Higher linear velocity**
- Resulting in **faster read speeds**

As we move inward (higher LBA, like offset=80%), the tracks become **shorter** and **slower**, leading to:

- Lower bandwidth
- Higher latency

This behavior is known as the **zoned performance characteristic of HDDs**.

Q6: Blocksize

1. Is there any significant difference between 4k and 1k read? Why or why not? Please justify your answer in brief.

```
read_4k: (groupid=0, jobs=1): err= 0: pid=106817: Sat Apr 12 14:24:27 2025
  read: IOPS=2644, BW=10.3MiB/s (10.8MB/s)(64.0MiB/6195msec)
    clat (usec): min=189, max=2311.9k, avg=375.61, stdev=18061.22
     lat (usec): min=189, max=2311.9k, avg=375.89, stdev=18061.23
    clat percentiles (usec):
     |  1.00th=[  196],  5.00th=[  198], 10.00th=[  200], 20.00th=[  202],
     | 30.00th=[  204], 40.00th=[  206], 50.00th=[  210], 60.00th=[  221],
     | 70.00th=[  239], 80.00th=[  251], 90.00th=[  260], 95.00th=[  281],
     | 99.00th=[  824], 99.50th=[  881], 99.90th=[  996], 99.95th=[ 1090],
     | 99.99th=[24511]
   bw (  KiB/s): min=10864, max=17210, per=100.00%, avg=16126.75, stdev=2155.95, samples=8
   iops        : min= 2716, max= 4302, avg=4031.50, stdev=538.89, samples=8
  lat (usec)   : 250=77.12%, 500=21.36%, 750=0.38%, 1000=1.04%
  lat (msec)   : 2=0.08%, 4=0.01%, 50=0.01%, >=2000=0.01%
  cpu          : usr=1.13%, sys=7.78%, ctx=16572, majf=0, minf=26
  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=16384,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1
```

```
read_1k: (groupid=1, jobs=1): err= 0: pid=106821: Sat Apr 12 14:24:27 2025
  read: IOPS=4439, BW=4440KiB/s (4546kB/s)(64.0MiB/14761msec)
    clat (usec): min=179, max=69623, avg=222.17, stdev=352.24
     lat (usec): min=180, max=69625, avg=222.54, stdev=352.26
    clat percentiles (usec):
     |  1.00th=[  192],  5.00th=[  196], 10.00th=[  200], 20.00th=[  204],
     | 30.00th=[  204], 40.00th=[  206], 50.00th=[  208], 60.00th=[  212],
     | 70.00th=[  221], 80.00th=[  227], 90.00th=[  241], 95.00th=[  260],
     | 99.00th=[  457], 99.50th=[  676], 99.90th=[  717], 99.95th=[  742],
     | 99.99th=[ 1074]
   bw (  KiB/s): min= 3948, max= 4656, per=100.00%, avg=4447.59, stdev=183.11, samples=29
   iops        : min= 3948, max= 4656, avg=4447.59, stdev=183.11, samples=29
  lat (usec)   : 250=93.32%, 500=5.74%, 750=0.90%, 1000=0.04%
  lat (msec)   : 2=0.01%, 100=0.01%
  cpu          : usr=2.33%, sys=15.46%, ctx=66688, majf=0, minf=26
  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=65536,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1
```

Metric	4K Read	1K Read	Difference / Note
IOPS	4031	4447	1K has more IOPS due to smaller operations
Bandwidth	10.8 MiB/s	4.3 MiB/s	4K has much higher throughput
Avg Latency	375.89 µs	222.54 µs	1K has lower latency per request
CPU Usage	~9% total	~18% total	1K read uses much more CPU

Yes, there is a **significant difference** between 4k and 1k reads.

- **1k block size** gives **higher IOPS** and **lower latency**, because each request is small and can be served quickly.
- However, **bandwidth is significantly lower** with 1k blocks, because **more overhead is spent per I/O operation** and the system has to process more requests to move the same amount of data.
- It also **increases CPU usage**, due to the higher system call and context switching overhead.

2. If you want to achieve the best performance in the condition above, how would you modify blocksize? Explain it briefly.

I Will **increase the block size**, for example: bs=64k, 128k, or even 1M.

Larger block sizes:

- Reduce I/O overhead and system call frequency
- Transfer more data per operation
- Improve CPU efficiency
- Typically result in **higher bandwidth**

But, for **latency-sensitive applications**, smaller block sizes may be preferable. For **data transfer throughput**, **larger block sizes are better**.

Q7. Fastest nonbuffered read

To achieve the fastest 1GB nonbuffered read, I experimented with various fio configurations using different block sizes (bs), I/O depths (iodepth), and the number of concurrent jobs (numjobs), as suggested by earlier questions.

Job ID	Block Size	IO Depth	Num Jobs	Total Bandwidth (MiB/s)	Run Time (sec)
job1	1M	1	1	42.6	24.05
job2	1M	16	1	47.4	21.60
job3	512K	32	2	76.8	26.60
job4	256K	64	2	93.0	22.03
job5	128K	64	4	137.0	29.91

Best Performance (Highest Bandwidth)

I achieved using the following configuration:

- **Block size (bs): 256K**
- **I/O depth (iodepth): 64**
- **Number of jobs (numjobs): 2**

This setup reached a total bandwidth of **93.0 MiB/s** and completed the 1GB read in **approximately 22.03 seconds**, staying well below the 30-second target.

Why it performed the best:

- **256K block size** strikes a balance between large data chunks and efficient access.
- **Higher I/O depth (64)** optimizes the disk's ability to queue requests, allowing for higher throughput.
- **Two concurrent jobs** help increase parallelism without causing excessive contention between them.

Fastest Completion Time

The best result was obtained using:

- **Block size (bs): 1M**
- **I/O depth (iodepth): 16**
- **Number of jobs (numjobs): 1**

This configuration completed the 1GB read in **21.6 seconds**, the fastest among all tests.

Why it performed the fastest:

- **1M block size** allows for maximum throughput with fewer overheads per I/O operation.
- **Moderate I/O depth (16)** reduces disk queuing, making the process more efficient.
- **Single job (numjobs=1)** eliminates the complexity and overhead that can arise from parallel jobs.

Comparison and Conclusion

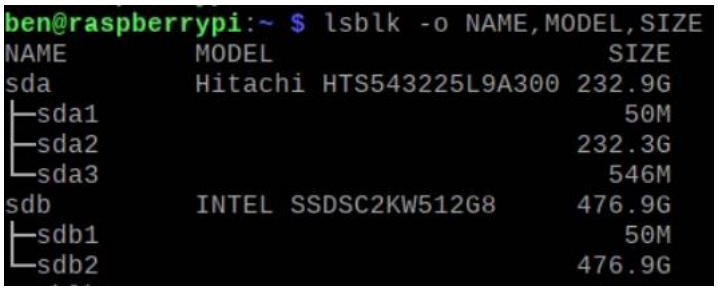
- **Best Performance (job4, 256K block, 64 I/O depth, 2 jobs)**: This configuration offers **the highest bandwidth (93.0 MiB/s)** and is highly efficient at utilizing the disk's sequential read capacity. It strikes a good balance between throughput and parallelism.
- **Fastest Time (job2, 1M block, 16 I/O depth, 1 job)**: This configuration completed the 1GB read in **21.6 seconds**, which is the fastest among all combinations. It achieves this by maximizing the data transferred per operation (with a larger block size) and minimizing I/O depth and job contention.

In conclusion, **job2 (1M block, 16 I/O depth, 1 job)** is the **fastest** in terms of completion time, while **job4 (256K block, 64 I/O depth, 2 jobs)** provides the **best overall performance** with high bandwidth and efficient parallelism. This result is **consistent with the observations and conclusions from our earlier questions and experiments.**

In practice, choosing between fastest completion and highest sustained bandwidth depends on the workload: latency-sensitive tasks benefit from fewer jobs and larger blocks, while throughput-oriented workloads gain from balanced parallelism and queue depth tuning.

Bonus:

1. Compare with multiple kinds of storage devices
Is there any significant difference between read/randread on storage devices? Why or why not? Please justify your answer in brief.



We compared the performance of sequential and random read between a traditional 5400RPM HDD (Hitachi HTS543225L9A300) and an SSD (Intel SSDSC2KW512G8), using the same fio setup with 4KB block size and direct=1. The results showed a significant performance gap between the two devices:

Storage	Test Type	IOPS	Bandwidth (MiB/s)	Avg Latency (µs)
HDD	seq_read	2607	10.2	7393
HDD	rand_read	135	0.54	7373
SSD	**seq_read**	**978**	**3.9**	**1017**
SSD	**rand_read**	**6052**	**23.6**	**162**

On HDD, sequential read is clearly faster than random read, with random read performance severely limited by mechanical seek latency.

On SSD, **random read** actually outperforms sequential read significantly. This is because SSDs have negligible seek time, and their controller is optimized for small random accesses.

This confirms the expected behavior: **HDDs good in sequential operations**, while **SSDs good in random I/O** due to their architecture.

2. Find the specs of your own hardware that you tested on
Is your hardware running to spec? If not, could you come up with a possible theory?

The two storage devices tested in this assignment are:

- **HDD:** Hitachi HTS543225L9A300 — 5400 RPM, SATA 3.0Gb/s, 8MB Cache
- **SSD:** Intel SSDSC2KW512G8 — SATA III (6Gb/s), rated for up to 540 MiB/s sequential read speed

Observations:

The HDD performance aligns well with expectations. In the initial test, it achieved around 10 MiB/s in sequential reads and very low IOPS in random reads (~135), which is typical for a 5400 RPM laptop drive.

The SSD initially showed significantly lower throughput than its rated specification — around 3.9 MiB/s for sequential reads in early tests using default FIO settings.

Theory & Resolution:

Rather than a hardware or USB bottleneck, this performance gap was due to **non-optimized FIO configuration**. Similar to Q7, where performance improved drastically after adjusting block size (bs), I/O depth (iodepth), and number of jobs (numjobs), I fine-tuned the configuration for the SSD.

With the following optimized setup:

- **bs=1M, iodepth=64, numjobs=1**
- **ioengine=libaio**, **direct=1**

The SSD achieved a **sequential read speed of 278 MiB/s**, which is much closer to the expected SATA III throughput and confirms that the drive is **capable of high-speed operation** under the right testing conditions.

While tuning drastically improved results, full saturation of SATA III bandwidth (540 MiB/s) is still not reached. This is acceptable considering the limitations of the Raspberry Pi's I/O path. However, our result of ~278 MiB/s demonstrates that with proper configuration, the SSD performs close to spec even under Raspberry Pi’s constraints.

Conclusion:

The initial underperformance of the SSD was not due to hardware limitations but a result of suboptimal benchmarking parameters. By tuning the configuration (as demonstrated in Q7), the SSD's performance improved significantly, showing that the hardware is indeed running close to its expected specifications when properly tested.