NVME性能测试与分析

1、背景

Intel NVME产品P4500 4T盘，Specification中稳态标称写性能1800MB/s，测试参数（bs=64KiB、iodepth=128、rw=write）。而CDS同学私下不严肃的测试中，经常发现一段时间后就有巨大的Performance Drop。不同参数测试结果波动很大，甚至同样参数在不同时间测出的结果差异不小。可能与盘当时的使用率、碎片化程度有关。测试一段时间后，内部触发频繁的gc，数据搬迁导致性能骤降。

这一切都与FTL有关，这是每个厂商的秘密，对我们是黑盒，只能通过测试摸索规律并指导存储系统设计研发。

2、测试

2.1 顺序与随机写

理论上，顺序写比随机写性能要好。随机写会造成gc代价更大（数据搬迁），特别是小块写。

测试命令如下：

测试命令

```
[root@szwg-cds-ssd-857073-test.szwg01.baidu.com tmp]# fio -filename=/dev
/nvme7n1 -direct=1 -iodepth=8 -thread -rw=randwrite  -ioengine=libaio -
bs=1M  -numjobs=1 -runtime=1800  -group_reporting  -name=mytest --debug=io
> write_offset.cici2
```

2.1.1 测试方法与结果

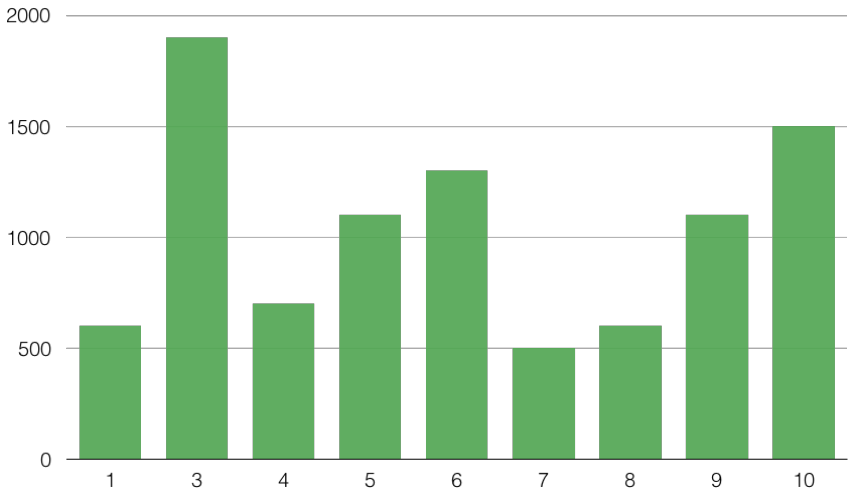
采用一块测试过多次（稳态）的盘，直接测裸盘，避免文件系统影响。先测顺序、接着测随机、最后测顺序写。

测试步骤如下：

1. 顺序写1M，刚开始性能达到1.9GB/s，进度20%的时候降到200+MB/s，进度70%的时候回升到300+MB/s，最终性能是600+MB/s。
2. trim整个盘。
3. 多次顺序写1M，稳定达到1.9G/s。
4. 随机写1M，刚开始性能1.9GB/s，一段时间后也降到200+MB/s，最终性能700+MB/s。
5. 第二次随机写1M，刚开始性能700+MB/s，较短一段时间后回升到1.9GB/s，随后又下降到400+MB/s，最终性能1.1GB/s。
6. 多次随机写1M，稳定达到1.3+GB/s。
7. 4K对齐（-blockalign=4k）随机写1M，刚开始性能达到1.4GB/s，一段时间后降到400+MB/s，进度50%之后降到200-MB/s，最终性能500+MB/s。
8. 顺序写1M，现象同1，最终性能600+MB/s。
9. 第二次顺序写1M，现象同5
10. 多次顺序写1M，现象同6，但是性能表现更好点，稳定达到1.3+GB/s，甚至1.5+GB/s。
11. 在逻辑地址[0，2G)内随机写1M，稳定达到1887MB/s。
12. 在逻辑地址[0，20G)内随机写1M，稳定达到1877MB/s。
13. 在逻辑地址[0，200G)内随机写1M，稳定达到1883MB/s。
14. 在逻辑地址[0，2000G)内随机写1M，一段时间后下降到300+MB/s，最终性能706MB/s。

2.1.2 结果分析

各步骤测试性能如下表所示。

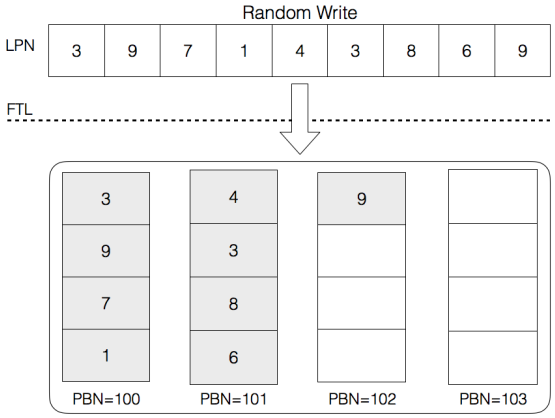


步骤1测试结果符合以往经验，一段时间后下降到200+MB/s，观察iostate发现单个io请求的排队时间增加10倍。

步骤3达到标称性能，说明Intel所谓稳态测试其实是之前不能有随机写，而且发挥SSD极致写性能只能靠顺序写。

步骤4符合预期，说明SSD对随机写并不友好，大概只能发挥标称性能的40%。

步骤5、6不符合预期，为什么多次随机写性能反而越来越好？首先通过fio代码确定的是，fio下发io请求的offset默认是按bs=1MB对齐的，而CDS场景即使是512K/1M的大写，offset却不一定按1MB对齐，碎片化会更加严重，在步骤7中再次测试该场景。其次猜测fio的随机写offset是伪随机生成，而SSD FTL倾向于把随机写和顺序写放到不同的存储区域，同时随机写也会变为顺序写，从而在物理介质上形成连续的存储块，如下图所示。



第二次随机写又将这些连续的存储块变为无效，从而降低gc的代价，减少数据的搬迁。通过fio的调试模式（debug=io）确认两次测试的io请求offset其实是一模一样的。

步骤7通过（-blockalign=4k）进行offset 4K对齐的随机写，到后期性能比1M对齐的更差。这也是模拟CDS单机引擎原地修改，证明该模型写性能并不理想。

步骤8、9、10说明碎片化严重的盘即使顺序写，性能也是较差的，因为存储块上存放的是逻辑地址随机的数据，顺序写将导致存储块上出现很多无效的空洞，gc势必造成数据的搬迁。但是多次顺序写之后，最终性能会回升上去，说明SSD FTL倾向于把顺序写放到连续的存储区域。

步骤11~14说明逻辑地址空间只要限定在一个较小的范围内，随机写也能达到峰值性能。因为在很短的时间内，随机写会将该地址空间一次次写满，在物理上连续存放的物理块会一起失效，从而避免了数据搬迁。这从本质上说明SSD达到峰值性能并不是因为顺序写，而是因为相同生命周期的数据在物理块上聚集存放，同时失效，避免了数据搬迁。

2.2 顺序与并发写

顺序写比随机写性能高很多且稳定，而CDS系统往往是多个写入流并发写，即使每个写入流是顺序的，到达SSD内部的时候会不会交织在一起变成随机写了？

2.1.1 测试方法与结果

在SSD上划分6个zone，每个zone大小为10GB/500GB。zone之间逻辑地址无重叠，每个zone配置一个fio job，该job只在本zone顺序循环写，命令如下：

测试命令

```
[root@szwg-cds-ssd-857073-test.szwg01.baidu.com tmp]# fio dev.fio
[root@szwg-cds-ssd-857073-test.szwg01.baidu.com tmp]# cat dev.fio
[global]
ioengine=libaio
iodepth=128
direct=1
group_reporting
runtime=1800
bs=4k
filename=/dev/nvme7n1
[job0]
rw=write
zonesize=500g
[job2]
rw=write
zonesize=500g
offset=1000g
[job4]
rw=write
zonesize=500g
offset=2000g
```

测试步骤如下：

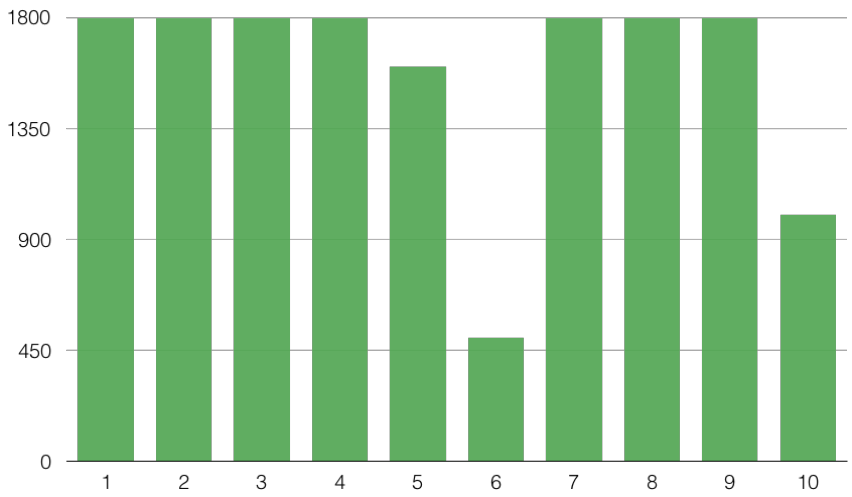
1. 在Range[0, 10GB）内顺序写1M，性能稳定达到1.8GB/s.
2. 在Range[0, 10GB)、[100GB, 110GB)、[200GB, 210GB)、[300GB, 310GB)、[400GB, 410GB)、[500GB, 510GB) 6个区域内并发顺序写1M，多次测试性能稳定达到1.8GB/s.
3. 在Range[0, 500GB)、[500GB, 1000GB)、[1000GB, 1500GB)、[1500GB, 2000GB)、[2000GB, 2500GB)、[2500GB, 3000GB) 6个区域内并发顺序写1M，多次测试性能稳定达到1.8GB/s.
4. 在Range[0, 500GB)、[500GB, 1000GB)、[1000GB, 1500GB)、[1500GB, 2000GB)、[2000GB, 2500GB)、[2500GB, 3000GB) 6个区域内并发顺序写4K，性能稳定达到1.8GB/s.
5. 在Range[0, 500GB)、[1000GB, 1500GB)、[2000GB, 2500GB) 3个区域内并发顺序写1M，刚开始性能1.9GB/s，一段时间后也降到600+MB/s，后面回升到1.8GB/s，最终性能1.6GB/s。
6. 创建1个文件，在Range[0, 10GB）内顺序写4K，性能在500+MB/s左右。
7. 创建1个文件，在Range[0, 10GB）内顺序写1M，性能稳定达到1.8GB/s。
8. 创建6个文件，各文件在[0, 10GB）内顺序写1M，并发写性能稳定达到1.8GB/s。
9. 创建300个文件，一半文件顺序写1M，另一半文件顺序写64K，并发写性能稳定达到1.8GB/s。
10. 创建300个文件，通过（fallocate=none）使得文件之间物理地址空间交织，一半文件顺序写1M，另一半文件顺序写64K，并发写性能在1GB/s左右。

测试命令

```
[root@szwg-cds-ssd-857073-test.szwg01.baidu.com tmp]# fio file.fio
[root@szwg-cds-ssd-857073-test.szwg01.baidu.com tmp]# cat file.fio
[global]
ioengine=libaio
direct=1
group_reporting
runtime=2400
time_based
fallocate=none
[job1]
bs=4K
iodepth=128
rw=write
size=500g
filesize=10g
nrfiles=50
filename_format=/home/ssd7/$jobname.$jobnum.$filenum
```

2.2.2 结果分析

各步骤测试性能如下表所示。



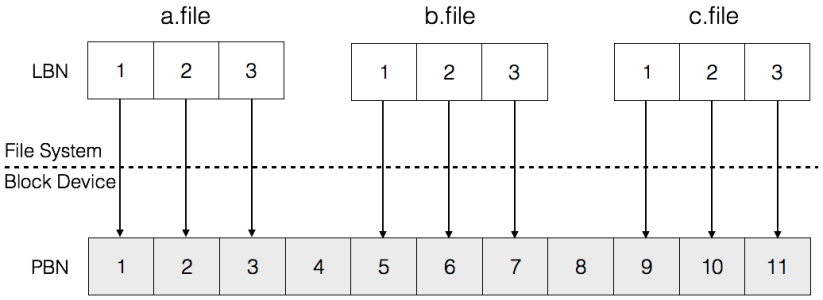
步骤1测试结果表明，虽然频繁的覆盖写，由于是顺序写，导致gc代价很低，性能好且稳定。

步骤2、3测试结果表明，逻辑地址区间隔离的并发写入流在SSD内部处理较好，性能好且稳定。那么对于ext4这样的文件系统，6个文件并发写入是什么情况呢？见步骤8、9、10.

步骤4测试结果表明，顺序写对于大小写无区别，在SSD内部应该都是以4K为单位处理并存储的。

步骤5测试结果表明，逻辑地址区间隔离的并发写入流在SSD内部并不是物理上分块隔离的，所以同一个Block上既有valid也有invalid数据，给gc带来不小开销。

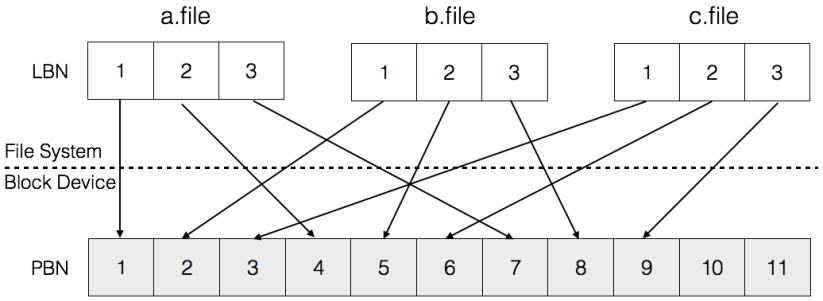
步骤6、7测试结果表明，文件系统对于大块顺序写，性能和块设备一样好，对于小块顺序写，性能只有块设备的1/4.



步骤8、9为了在文件系统层面验证多个文件并发写入流是否有影响，实测没有影响。通过filefrag分析所有数据文件映射的物理地址范围，没有重叠，如上图所示。继而定位是fio默认的fallocate模式导致的，怀疑和这个有关。

步骤9、10一半文件写1M、一半文件写64K，是为了模拟CDS内部的数据和元数据更新，元数据属于热数据，其gc的频率较高，如果和冷数据存一起，会导致gc时候的数据搬迁。

步骤10关闭fio的fallocate模式，通过filefrag检查数据文件映射的物理地址范围，呈现一段交织一段（1M、2M、4M、8M）的现象，如下图所示。测试结果表明，性能下降近一半，和随机写类似。不同文件重写速率不同，重写过的文件对于SSD来说已经是无效的数据块，无效和有效的数据块交织在一起，会导致gc的数据搬迁，从而影响写性能。



2.3 使用率与写放大

SSD不同使用率对写放大影响不小，对比测试70-%和80+%使用率下写放大有差异。然而，停了测试之后，nand_bytes_written还在持续增长，大概10s增加1、2的样子（1代表32MB数据）。过了一晚上（超过12小时）还在持续增长，说明盘一直在做defrag。这样统计写放大就需要更宽的时间范围，否则会失真。

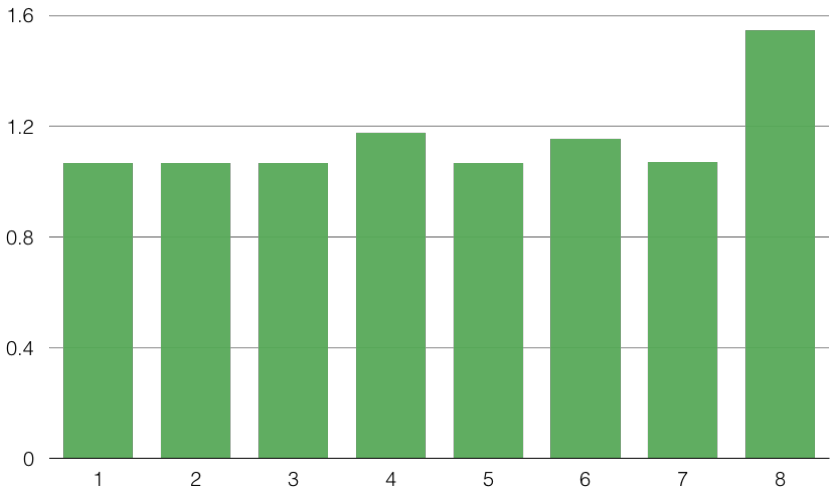
2.3.1 测试方法与结果

在SSD上划分6个zone，3个zone大小为300GB，3个zone大小为600GB，整体占比67.5%。zone之间逻辑地址无重叠，每个zone配置一个fio job，该job只在本zone顺序循环写。测试步骤如下1~4。trim之后，划分7个zone，3个zone大小为300GB，4个zone大小为600GB，整体占比82.5%。测试步骤如下5~8：

1. 6个zone并发顺序写1MB，性能稳定达到：1953MB/s，写放大：1.065；
2. 3个600GB的zone并发顺序写1MB，性能稳定达到1955MB/s，写放大：1.065；
3. 6个zone并发顺序写1MB，性能稳定达到：1949MB/s，写放大：1.071；
4. 3个300GB的zone并发顺序写1MB，性能1632MB/s，写放大：1.175
5. 7个zone并发顺序写1MB，性能稳定达到1968MB/s，写放大：1.065
6. 4个600GB的zone并发顺序写1MB，性能1694MB/s，写放大：1.154
7. 7个zone并发顺序写1MB，性能稳定达到1943MB/s，写放大：1.070
8. 3个300GB的zone并发顺序写1MB，性能1029MB/s，写放大：1.544

2.3.2 结果分析

各步骤测试写放大如下表所示。



从测试结果来看，性能和写放大是反相关的，写放大与盘使用率正相关。对比测试4和8，使用率高（80+%）比使用率低时（70%−），写放大高了31.4%，性能降低了37%。

2.4 写区间与性能

SSD的逻辑写区间和相应的物理存储区间并无严格对应关系，从上面的测试可知随机写性能很差，但如果将写区间缩小到10G甚至1G呢？是否随机写和顺序写性能也并无区别？从上面测试也可知顺序写性能极好，但如果将写区间缩小到500G、100G甚至10G，每次挑选一个区间来写，区间内顺序写，是不是也能发挥极致性能？测试命令如下。

测试命令

```
[root@szwg-cds-ssd-857073-test.szwg01.baidu.com wangxinxing]# cat rand.fio
[global]
ioengine=libaio
iodepth=128
direct=1
group_reporting
runtime=2400
bs=1m
filename=/dev/nvme7n1
[job0]
rw=randwrite
zonesize=10g
size=3000g
[root@szwg-cds-ssd-857073-test.szwg01.baidu.com wangxinxing]# cat dev.fio3
[global]
ioengine=libaio
iodepth=128
direct=1
group_reporting
runtime=2400
bs=1m
filename=/dev/nvme7n1
[job0]
rw=write
offset=100g
zonesize=100g
zoneskip=100g
size=3000g
```

2.4.1 测试方法与结果

1. 在Range[0, 1G)内随机写，性能稳定达到1.8GB/s.
2. 在Range[0, 10G)内随机写，性能稳定达到1.8GB/s.
3. 在Range[0, 100G)内随机写，性能稳定达到1.8GB/s.
4. 在Range[0, 500G)内随机写，性能波动较大0.7~1.8GB/s，最终性能1.2GB/s.
5. trim整块盘，单路顺序写1MB，性能稳定达到1.8GB/s.
6. 在Range[0, 500GB)、[1000GB, 1500GB)、[2000GB, 2500GB) 3个区域中每次选一个进行顺序写1M，多次测试性能稳定达到1.8GB/s.
7. 在Range[500GB, 1000GB)、[1500GB, 2000GB)、[2500GB, 3000GB) 3个区域中每次选一个进行顺序写1M，多次测试性能稳定达到1.8GB/s.
8. 在Range[0, 100GB)、[200GB, 300GB)、.....[2800GB, 2900GB) 15个区域中每次选一个进行顺序写1M，性能稳定达到1.8GB/s.
9. 在Range[100GB, 200GB)、[300GB, 400GB)、.....[2900GB, 3000GB) 15个区域中每次选一个进行顺序写1M，性能稳定达到1.8GB/s.
10. 在Range[0, 100GB)、[300GB, 400GB)、.....[2700GB, 2800GB) 10个区域中每次选一个进行顺序写1M，性能稳定达到1.8GB/s.
11. 在Range[100GB, 200GB)、[400GB, 500GB)、.....[2800GB, 2900GB) 10个区域中每次选一个进行顺序写1M，性能稳定达到1.8GB/s.
12. 在Range[0, 10GB)、[20GB, 30GB)、.....[2980GB, 2990GB) 150个区域中每次选一个进行顺序写1M，性能达到1.2GB/s.
13. 在Range[10GB, 20GB)、[30GB, 40GB)、.....[2990GB, 3000GB) 150个区域中每次选一个进行顺序写1M，性能达到1.8GB/s.
14. 在Range[0, 10GB)、[20GB, 30GB)、.....[2980GB, 2990GB) 150个区域中每次选一个进行顺序写1M，性能达到1.8GB/s.
15. 在Range[0, 10GB)、[30GB, 40GB)、.....[2970GB, 2980GB) 100个区域中每次选一个进行顺序写1M，性能达到1.4GB/s.
16. 在Range[10GB, 20GB)、[40GB, 50GB)、.....[2980GB, 2990GB) 100个区域中每次选一个进行顺序写1M，性能达到1.5GB/s.
17. 在Range[0, 1GB)、[2GB, 3GB)、.....[2998GB, 2999GB) 1500个区域中每次选一个进行顺序写1M，性能达到0.7GB/s.
18. 在Range[1, 2GB)、[3GB, 4GB)、.....[2999GB, 3000GB) 1500个区域中每次选一个进行顺序写1M，性能达到1.4GB/s.
19. 在Range[0, 1GB)、[2GB, 3GB)、.....[2998GB, 2999GB) 1500个区域中每次选一个进行顺序写1M，性能达到1.8GB/s.
20. 在Range[0, 1GB)、[3GB, 4GB)、.....[2997GB, 2998GB) 1000个区域中每次选一个进行顺序写1M，性能达到0.6GB/s.
21. 在Range[1, 2GB)、[4GB, 5GB)、.....[2998GB, 2999GB) 1000个区域中每次选一个进行顺序写1M，性能达到0.4GB/s.
22. 在Range[0, 3TB)区域内按100GB划分n个格子，每次随机选一个格子顺序写满，性能达到1896MB/s.
23. 在Range[0, 3TB)区域内按50GB划分n个格子，每次随机选一个格子顺序写满，性能达到1804MB/s.
24. 在Range[0, 3TB)区域内按30GB划分n个格子，每次随机选一个格子顺序写满，性能达到1712MB/s.
25. 在Range[0, 3TB)区域内按20GB划分n个格子，每次随机选一个格子顺序写满，性能达到1398MB/s.
26. 在Range[0, 3TB)区域内按10GB划分n个格子，每次随机选一个格子顺序写满，性能达到841MB/s.
27. 在Range[0, 3TB)区域内按24MB划分n个格子，每次随机选一个格子顺序写满，性能达到325MB/s.

2.4.2 结果分析

从测试1~4来看，随机写在一个较小的地址范围内（<100GB）和顺序写性能是没区别的。从测试6~11来看，保证一个较大区间（100GB+）内的顺序写，区间之间无须保证顺序性也可以达到极致性能。从测试17~21来看，在较小区间（1GB-）内的顺序写，区间之间不保证顺序性，性能下降较明显。

2.5 Optane 与 QLC

Optane+QLC 是替代 TLC 的一种新机型，用于节约成本。从性能方面来看，Optane 采用3D Xpoint，不同于 SSD NAND 写前需要擦除，其可以原地修改，随机读写延迟非常低。QLC 性能则相对 TLC 有所降低，具体情况用测试说明。

2.5.1 Optane测试结果

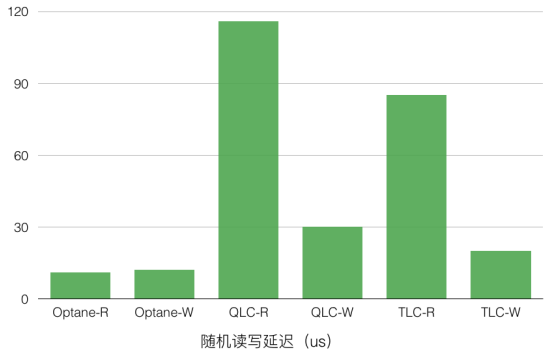
1. Random/Seq. Write 1M 8QD：2172/2279 MB/s
2. 4K对齐（-blockalign=4k）随机写1M，稳定达到2279MB/s
3. Seq. Write 4K 1/8/128QD：345/1207/1245 MB/s，IOPS：86k/311k/319k
4. Random Write 4K 1/8/128QD：341/1145/1150 MB/s，IOPS：85k/288k/286k
5. Seq. Read 4K 1/8/128QD：371/1280/1200 MB/s，IOPS：93k/328k/333k
6. Random Latency R/W：11/12 us
7. Sequential Latency R/W：11/12 us

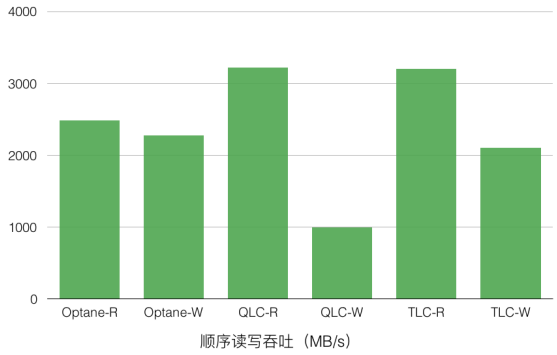
2.5.2 QLC测试结果

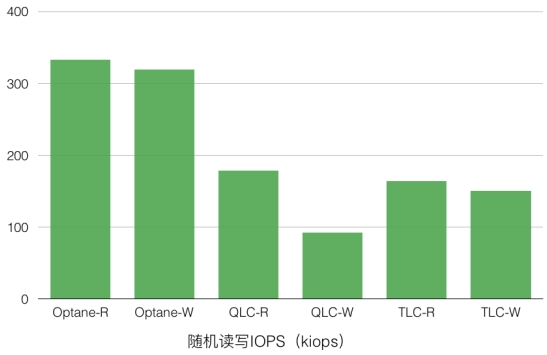
1. Random/Seq. Write 1M 8QD：296/993 MB/s；
2. Random R/W 1M 8QD：1973/296 MB/s
3. Random latency R/W：116/30 us
4. Random R/W 4K 256QD：178k/92k IOPS

2.5.3 结果分析

Optane/QLC/TLC 4K 随机读写延迟、1M 顺序读写吞吐、4K 随机读写 IOPS 分别如下图所示。







从延迟来看，Optane 不管是顺序还是随机读写延迟都很低而且稳定，TLC 表现居中，QLC 表现最差。

从吞吐来看，Optane 读写都超过2GB/s，QLC 和 TLC 读吞吐超过 3GB/s，但是 QLC 写吞吐只有 TLC 的一半。

从 IOPS 来看，Optane 表现最好，是TLC的两倍，QLC读IOPS和TLC相当，写IOPS只有TLC一半。

Optane 对顺序、随机写不敏感，延迟相比 QLC/TLC 有比较大的优势，吞吐方面则优势有限。

3、结论

结论一：SSD顺序写性能比随机写好很多（1.8GB/s vs 600+MB/s）。

结论二：顺序写能稳定达到标称写性能1.8GB/s，随机写无法达到，而且波动很大，波动区间[1.8GB/s，200+MB/s]，如果offset不是1MB对齐而是4KB对齐，最低性能还不到200MB/s，沦落成HDD了。

结论三：对于裸设备顺序写，大块写和小块写（1MB vs 4KB）性能一致；对于文件系统（ext4），大块顺序写和裸设备性能一致，小块顺序写只有裸设备的1/4.

结论四：逻辑地址区间隔离的并发写入流在SSD内部处理较好，性能好且稳定，文件系统利用fallocate预分配也能达到同样效果。

结论五：无效和有效数据交织存放，从而给GC造成数据搬迁，是影响SSD性能的关键因素。

结论六：SSD达到峰值性能的本质并不是因为顺序写，而是因为相同生命周期的数据在物理块上聚集存放，同时失效，避免了数据搬迁。

结论七：存储系统设计应该考虑，采用顺序写避免随机写，按冷热划分多个数据区，将生命周期相近的数据顺序存放，以减少GC开销。然而，区分冷热的策略只有支持set-associative、multi-streaming或者是open-channel的SSD才有效。

结论八：所有达到标称性能的测试方案均未在测试中主动发起trim，证明trim不是达到最佳性能的必要条件。