31 | 套路篇：磁盘 I/O 性能優化的幾個思路

小明

2020.08.27

PIXNET

回顧

- 性能指標：文件系統、磁盤I/O
- 核心 I/O 性能觀測工具
- 如何快速分析 I/O 性能問題
- 從瓶頸到優化

IOPS

文件系統

延遲

你、我、他
如何才能適合？

場景

吞吐量

物理磁盤

fio - Flexible I/O Tester

可客製化

裸盤

不同塊大小

最被常用

文件系統

不同 I/O 引擎

緩存支援

https://github.com/axboe/fio

安裝 fio

```
# Ubuntu
apt-get install -y fio

# CentOS
yum install -y fio
```

fio 參數

PIXNET

```
# 隨機讀
fio -name=randread -direct=1 -iodepth=64 -rw=randread -ioengine=libaio -bs=4k
-size=1G -numjobs=1 -runtime=1000 -group_reporting -filename=/dev/sdb

# 隨機寫
fio -name=randwrite -direct=1 -iodepth=64 -rw=randwrite -ioengine=libaio -bs=4k
-size=1G -numjobs=1 -runtime=1000 -group_reporting -filename=/dev/sdb

# 順序讀
fio -name=read -direct=1 -iodepth=64 -rw=read -ioengine=libaio -bs=4k -size=1G
-numjobs=1 -runtime=1000 -group_reporting -filename=/dev/sdb

# 順序寫
fio -name=write -direct=1 -iodepth=64 -rw=write -ioengine=libaio -bs=4k -size=1G
-numjobs=1 -runtime=1000 -group_reporting -filename=/dev/sdb
```

fio 參數說明

PIXNET

- direct - 1 表不用系統緩存，0 表使用系統緩存
- iodepth - 於 Asynchronous I/O，AIO 時，同時發出的 I/O 請求上限
- rw - I/O 模式。read、write、randread、randwrite
- ioengine - I/O 引擎。sync、libaio、mmap、net 等
- bs - I/O 大小。默認值 4K
- filename - 文件路徑，亦可以是磁盤路徑

fio 結果

PIXNET

```
read: (g=0): rw=read, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio, iodepth=64
fio-3.1
Starting 1 process
Jobs: 1 (f=1): [R(1)][100.0%][r=16.7MiB/s,w=0KiB/s][r=4280,w=0 IOPS][eta 00m:00s]
read: (groupid=0, jobs=1): err= 0: pid=17966: Sun Dec 30 08:31:48 2018
   read: IOPS=4257, BW=16.6MiB/s (17.4MB/s)(1024MiB/61568msec)
    slat (usec): min=2, max=2566, avg= 4.29, stdev=21.76
    clat (usec): min=228, max=407360, avg=15024.30, stdev=20524.39
    lat (usec): min=243, max=407363, avg=15029.12, stdev=20524.26
    clat percentiles (usec):
     |  1.00th=[   498],  5.00th=[  1020], 10.00th=[  1319], 20.00th=[  1713],
     | 30.00th=[  1991], 40.00th=[  2212], 50.00th=[  2540], 60.00th=[
     | 70.00th=[  5407], 80.00th=[ 44303], 90.00th=[ 45351], 95.00th=[
     | 99.00th=[ 46924], 99.50th=[ 46924], 99.90th=[ 48497], 99.95th=[
     | 99.99th=[404751]
   bw (  KiB/s): min= 8208, max=18832, per=99.85%, avg=17005.35, stdev=998.94, samples=123
   iops        : min= 2052, max= 4708, avg=4251.30, stdev=249.74, samples=123
  lat (usec)   : 250=0.01%, 500=1.03%, 750=1.69%, 1000=2.07%
```

lat avg = 15029.12
slat ave + clat ave = 15028.59

吞吐量：17005.35 KiB/s ~ 16.6 MiB/s
IOPS：5251.30

fio 結果

PIXNET

```
  lat (msec)   : 2=25.64%, 4=37.58%, 10=2.08%, 20=0.02%, 50=29.86%
  lat (msec)   : 100=0.01%, 500=0.02%
  cpu          : usr=1.02%, sys=2.97%, ctx=33312, majf=0, minf=75
  IO depths    : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=0.1%, >=64=100.0%
     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=0.1%, >=64=0.0%
     issued rwt: total=262144,0,0, short=0,0,0, dropped=0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=64

Run status group 0 (all jobs):
   READ: bw=16.6MiB/s (17.4MB/s), 16.6MiB/s-16.6MiB/s (17.4MB/s-17.4MB/s), io=1024MiB (1074MB),
run=61568-61568msec

Disk stats (read/write):
  sdb: ios=261897/0, merge=0/0, ticks=3912108/0, in_queue=3474336, util=90.09%
```

fio 結果說明

- slat - 指從 I/O 提交到實際執行 I/O 的時長（Submission latency）
- clat - 指從 I/O 提交到 I/O 完成的時長（Completion latency）
- lat - 從 fio 創建 I/O 到 I/O 完成的總時長

若為 sync I/O 則 I/O 提交 = I/O 完成
slat = I/O 完成時間，clat = 0

若為 async I/O 則
lat ~ slat + clat

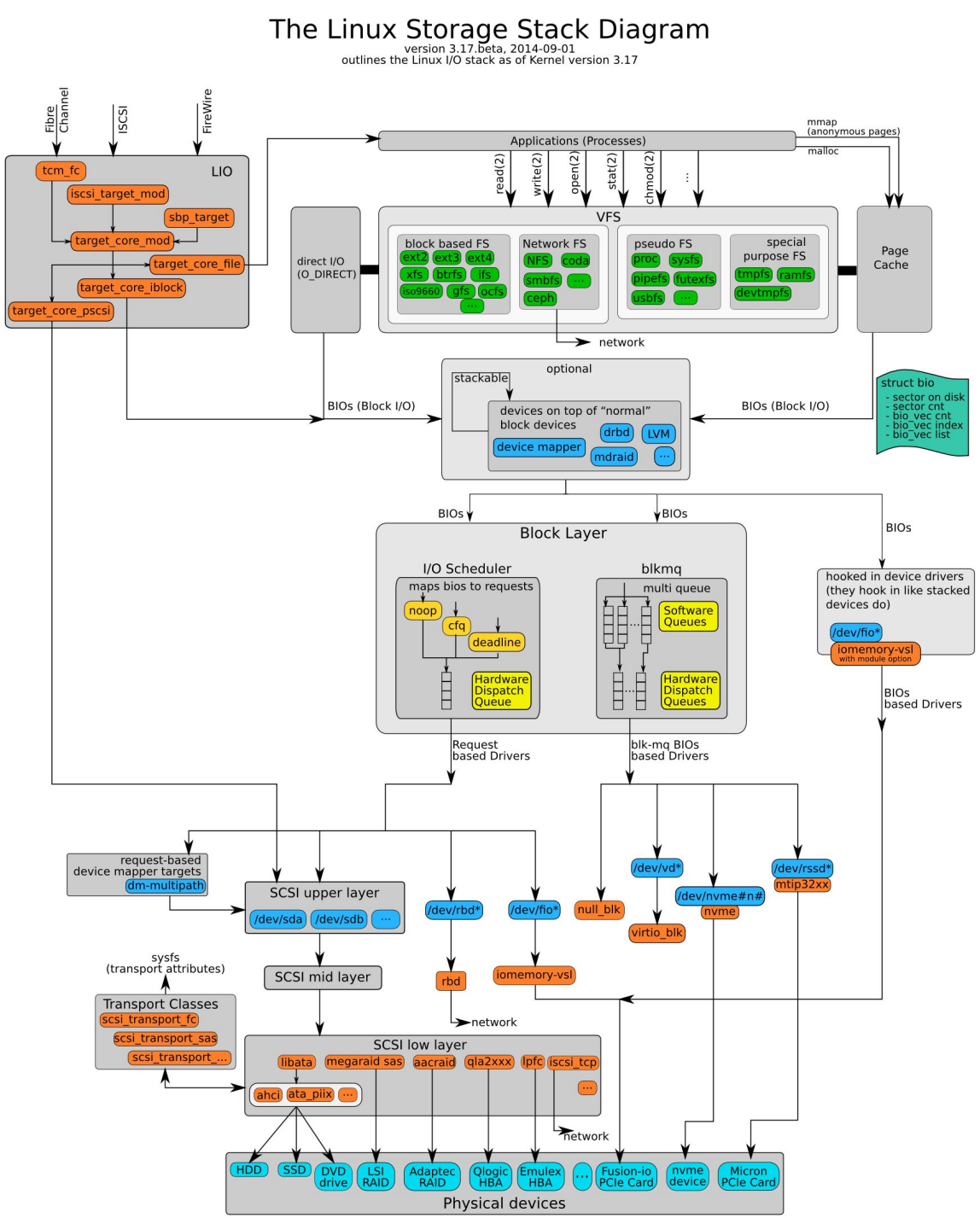
fio + blktrace

```
# 使用blktrace跟踪磁盘I/O，注意指定应用程序正在操作的磁盘
$ blktrace /dev/sdb

# 查看blktrace记录的结果
# ls
sdb.blktrace.0  sdb.blktrace.1

# 将结果转化为二进制文件
$ blkparse sdb -d sdb.bin

# 使用fio重放日志
$ fio --name=replay --filename=/dev/sdb --direct=1 --read_iolog=sdb.bin
```

fio + blktrace demo


应用程序
库函数
用户空间
内核空间
系统调用
页
缓存
虚拟文件系统（VFS）
目录项
缓存
索引节点
缓存
文件系统
Ext4
XFS
ZFS
...
NFS
缓冲区（Buffer）
本地磁盘设备
网络存储

The Linux Storage Stack Diagrom



https://www.cnblogs.com/shaohef/p/3952402.html

透過系統調用，來調整 I/O 模式

1. 以追加寫（O_APPEND）代替隨機寫
2. 借助緩存 I/O 利用系統緩存以降低 I/O 次數
3. 於應用程式內建構緩存，或用 Redis 外部緩存系統

fopen, fread 有用標準庫的緩存，而 open, read 則用系統的頁緩存和緩衝區，但無標準庫的緩存

4. 以 mmap 取代 read/write
5. 以 fsync() 取代 O_SYNC
6. 建議用 cgroups 的 I/O 子系統，限制進程 / 進程組 IOPS 及吞吐量

1. 選擇合適的文件系統
   xfs 可支持 > 16TB 磁盘及更多的文件數量
   ext4 可方便縮放文件系統
2. 優化文件系統配置選項。亦可用 tune2fs 調整文件系統
   ext_attr, dir_index, journal, ordered, writeback, noatime
3. 優化文件系統的緩存
   dirty_expire_centisecs, dirty_writeback_centisecs,
   dirty_background_ratio, drity_ratio
4. 優化內核回收目錄緩存和節點緩存
   vfs_cache_pressure
5. 使用 tmpfs

1. 使用 SSD
2. 採用 RAID
3. SSD 採 noop 調度算法，而 database 則用 deadline 算法
4. 讓 I/O 重者有單獨的磁盤
5. 增加磁盤預讀數據
   /sys/block/sdb/queue/read_ahead_kb
   利用 blockdev 工具，blockdev --setra 8192 /dev/sdb (512B)
6. 優化內容塊設備 I/O 選項：/sys/block/sdb/queue/nr_requests

磁盤優化 - 注意

1. 磁盤硬件故障
2. dmesg
3. badblocks
4. smartctl
5. e2fsck
6. fsck

1. 整理常見文件系統、磁盤 I/O 性能優化思路和方法
2. 先不要急著優化
   a. 先找出最大、最重要的效能問題
   b. 從 I/O 棧找合適方式解決
3. 磁盤和文件系統的 I/O 都是最慢，除了優化，可借助更快內存、網路、CPU等減少 I/O 調用
4. 充分利用 Buffer、Cache 或應用程序內部緩存
5. 利用 Redis 等外部緩存系統

- 除了此章所提之工具或方法，還有其他的嗎？
- 有什麼優化經驗可以分享？

THANK YOU