MySQL DBA学习笔记----------美河学习在线 www.eimhe.com 仅学习参考

MySQL学习笔记（Day019：磁盘测试）

MySQL学习

一. 磁盘调度算法介绍

1. CFQ

> CFQ把I/O请求 按照进程 分别放入进程对应的队列中，所以A进程和B进程发出的I/O请求会在两个队列中。而各个队列内部仍然采用 合并和排序 的方法，区别仅在于，每一个提交I/O请求的进程都有自己的I/O队列。
> CFQ的"公平"是针对进程而言的，它以时间片算法为前提，轮转调度队列，默认从当前队列中取4个请求处理，然后处理下一个队列的4个请求。这样就可以确保每个进程享有的I/O资源是均衡的。
> CFQ的缺点是先来的IO请求不一定能被及时满足，可能出现 饥饿 的情况。
> CFQ Wiki

2. Deadline

> 同CFQ一样，除了维护一个拥有合并和排序功能的请求队列以外，还额外维护了两个队列，分别是 读请求队列 和 写请求队列 ，它们都是 带有超时的FIFO队列 。当新来一个I/O请求时，会被同时插入普通队列和读/写队列，然后处理普通队列中的请求。当调度器发现读/写请求队列中的请求超时的时候，会优先处理这些请求，保证尽可能不产生请求饥饿
> 在DeadLine算法中，每个I/O请求都有一个超时时间，默认 读请求是500ms ， 写请求是5s 。
> Deadline Wiki

3. Noop

> Noop做的事情非常简单，它不会对I/O请求排序也不会进行任何其它优化（除了合并）。Noop除了对请求合并以外，不再进行任何处理，直接以类似FIFO的顺序提交I/O请求。
> Noop面向的不是普通的块设备，而是随机访问设备（例如SSD），对于这种设备，不存在传统的寻道时间，那么就没有必要去做那些多余的为了减少寻道时间而采取的事情了。
> Noop Wiki

二. iostat（下）

- **rrqm/s 和 wrqm/s**
  - `Merge` 将若干个连续地址的IO请求进行合并。来提高IO的效率
  - `rrqm/s` 是每秒读（read）请求合并的次数
  - `wrqm/s` 是每秒写（write）请求合并的次数
- **r/s和w/s**
  - 在 `合并之后（after merge）` IO请求的次数
  - `r/s` 合并之后每秒读IO的次数
  - `w/s` 合并之后每秒写IO的次数
  - **`r/s + w/s = IOPS`**
- **rsec/s（rKB/s、rMB/s）和 wsec/s（wKB/s、wMB/s）**
  - `sec` 是 `Sector（扇区）`，为 `512Byte`
  - `KB` 和 `MB` 是通过扇区的 `512Byte` 进行的换算
- **avgrq-sz**
  - 一块磁盘可能存储数据的同时还存储日志，所以请求的IO大小是不一样的
  - 该参数就是平均的请求数，注意，该值需要 * 512Byte 才是最终的结果，因为该值是以扇区为单位的
- **avgqu-sz**
  - 请求的IO队列的平均长度 `（比较重要）`
  - HDD可能在4左右，SSD可以达到30左右
- **await、r_await、w_await**
  - IO请求平均等待的时间，单位是ms
  - `r_await` 和 `w_await` 分别对应 `读IO请求的等待` 和 `写IO请求的等待`
- **svctm**
  - 服务于IO请求的平均时间
  - man文档中提示不要相信该值，以后会被移除
- **%util**
  - 磁盘是否空闲；不能简单的等同于IO的使用率；该值可以解释为磁盘是否繁忙
  - 如果该值100% 不能简单的等同于磁盘的负载满了，达到了瓶颈
  - 需要综合 `avgqu-sz` 、 `await` 等其他指标进行综合判断磁盘是否达到瓶颈

三. MySQL的IO使用情况

1. iotop

```
shell> iotop -u mysql  # -u 表示监控哪个user的进程，所以前提是你的mysql服务是用mysql用户启动的
```

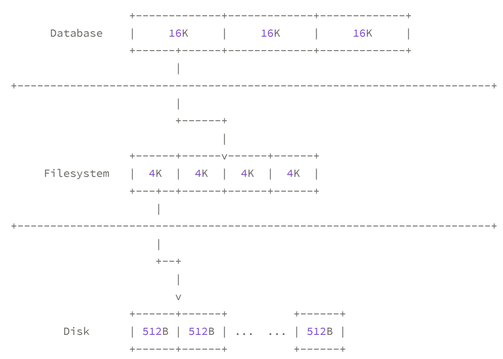
> **注意：**
> 上述命令只能看到MySQL的线程ID（Thread ID）

2. performance_schema.threads

```
mysql> use performance_schema;
Database changed

mysql> desc threads;
+---------------------+---------------------+------+-----+---------+-------+
| Field               | Type                | Null | Key | Default | Extra |
+---------------------+---------------------+------+-----+---------+-------+
| THREAD_ID           | bigint(20) unsigned | NO   |     | NULL    |       | -- MySQL内部线程ID
| NAME                | varchar(128)        | NO   |     | NULL    |       |
| TYPE                | varchar(10)         | NO   |     | NULL    |       |
| PROCESSLIST_ID      | bigint(20) unsigned | YES  |     | NULL    |       |
| PROCESSLIST_USER    | varchar(32)         | YES  |     | NULL    |       |
| PROCESSLIST_HOST    | varchar(60)         | YES  |     | NULL    |       |
| PROCESSLIST_DB      | varchar(64)         | YES  |     | NULL    |       |
| PROCESSLIST_COMMAND | varchar(16)         | YES  |     | NULL    |       |
| PROCESSLIST_TIME    | bigint(20)          | YES  |     | NULL    |       |
| PROCESSLIST_STATE   | varchar(64)         | YES  |     | NULL    |       |
| PROCESSLIST_INFO    | longtext            | YES  |     | NULL    |       |
| PARENT_THREAD_ID    | bigint(20) unsigned | YES  |     | NULL    |       |
| ROLE                | varchar(64)         | YES  |     | NULL    |       |
| INSTRUMENTED        | enum('YES','NO')    | NO   |     | NULL    |       |
| HISTORY             | enum('YES','NO')    | NO   |     | NULL    |       |
| CONNECTION_TYPE     | varchar(16)         | YES  |     | NULL    |       |
| THREAD_OS_ID        | bigint(20) unsigned | YES  |     | NULL    |       | -- 操作系统的线程ID
+---------------------+---------------------+------+-----+---------+-------+
17 rows in set (0.00 sec)

mysql> select name,type,thread_id,thread_os_id from threads;
+---------------------------------------+------------+-----------+--------------+
| name                                  | type       | thread_id | thread_os_id |
+---------------------------------------+------------+-----------+--------------+
| thread/sql/main                       | BACKGROUND |         1 |         2481 |
| thread/sql/thread_timer_notifier      | BACKGROUND |         2 |         2482 |
| thread/innodb/io_read_thread          | BACKGROUND |         3 |         2486 |
| thread/innodb/io_read_thread          | BACKGROUND |         4 |         2487 |
| thread/innodb/io_read_thread          | BACKGROUND |         5 |         2488 |
| thread/innodb/io_write_thread         | BACKGROUND |         6 |         2489 |
| thread/innodb/io_write_thread         | BACKGROUND |         7 |         2490 |
| thread/innodb/io_write_thread         | BACKGROUND |         8 |         2491 |
| thread/innodb/io_write_thread         | BACKGROUND |         9 |         2492 |
| thread/innodb/page_cleaner_thread     | BACKGROUND |        10 |         2493 |
| thread/innodb/io_read_thread          | BACKGROUND |        11 |         2485 |
| thread/innodb/io_log_thread           | BACKGROUND |        12 |         2484 |
| thread/innodb/io_ibuf_thread          | BACKGROUND |        13 |         2483 |
| thread/innodb/srv_master_thread       | BACKGROUND |        15 |         2501 |  -- 主线程
| thread/sql/background                 | BACKGROUND |        16 |         2502 |
| thread/innodb/srv_purge_thread        | BACKGROUND |        17 |         2502 |
| thread/sql/background                 | BACKGROUND |        18 |         2503 |
| thread/innodb/srv_monitor_thread      | BACKGROUND |        19 |         2500 |
| thread/innodb/srv_error_monitor_thread | BACKGROUND |        20 |         2499 |
| thread/sql/background                 | BACKGROUND |        21 |         2504 |
| thread/sql/background                 | BACKGROUND |        22 |         2505 |
| thread/innodb/srv_lock_timeout_thread | BACKGROUND |        23 |         2498 |
| thread/innodb/dict_stats_thread       | BACKGROUND |        24 |         2507 |
| thread/innodb/buf_dump_thread         | BACKGROUND |        25 |         2506 |
| thread/sql/signal_handler             | BACKGROUND |        26 |         2510 |
| thread/sql/compress_gtid_table        | FOREGROUND |        27 |         2511 |
| thread/sql/one_connection             | FOREGROUND |        28 |         2514 |  -- FOREGROUND前台线程
+---------------------------------------+------------+-----------+--------------+
27 rows in set (0.00 sec)

-- thread/sql/one_connection 就是我连接的线程

mysql> select name,thread_id,thread_os_id,processlist_id from threads;  -- 查看processlist_id
+---------------------------------------+-----------+--------------+----------------+
| name                                  | thread_id | thread_os_id | processlist_id |
+---------------------------------------+-----------+--------------+----------------+
| thread/sql/main                       |         1 |         2481 |           NULL |
| thread/sql/thread_timer_notifier      |         2 |         2482 |           NULL |
| thread/innodb/io_read_thread          |         3 |         2486 |           NULL |
| thread/innodb/io_read_thread          |         4 |         2487 |           NULL |
| thread/innodb/io_read_thread          |         5 |         2488 |           NULL |
| thread/innodb/io_write_thread         |         6 |         2489 |           NULL |
| thread/innodb/io_write_thread         |         7 |         2490 |           NULL |
| thread/innodb/io_write_thread         |         8 |         2491 |           NULL |
| thread/innodb/io_write_thread         |         9 |         2492 |           NULL |
| thread/innodb/page_cleaner_thread     |        10 |         2493 |           NULL |
| thread/innodb/io_read_thread          |        11 |         2485 |           NULL |
| thread/innodb/io_log_thread           |        12 |         2484 |           NULL |
| thread/innodb/io_ibuf_thread          |        13 |         2483 |           NULL |
| thread/innodb/srv_master_thread       |        15 |         2501 |           NULL |
| thread/sql/background                 |        16 |         2502 |           NULL |
| thread/innodb/srv_purge_thread        |        17 |         2502 |           NULL |
| thread/sql/background                 |        18 |         2503 |           NULL |
| thread/innodb/srv_monitor_thread      |        19 |         2500 |           NULL |
| thread/innodb/srv_error_monitor_thread |        20 |         2499 |           NULL |
| thread/sql/background                 |        21 |         2504 |           NULL |
| thread/sql/background                 |        22 |         2505 |           NULL |
| thread/innodb/srv_lock_timeout_thread |        23 |         2498 |           NULL |
| thread/innodb/dict_stats_thread       |        24 |         2507 |           NULL |
| thread/innodb/buf_dump_thread         |        25 |         2506 |           NULL |
| thread/sql/signal_handler             |        26 |         2510 |           NULL |
| thread/sql/compress_gtid_table        |        27 |         2511 |              1 |
| thread/sql/one_connection             |        28 |         2514 |              2 |
+---------------------------------------+-----------+--------------+----------------+
27 rows in set (0.00 sec)

-- processlist_id 对应的就是 show processlist中的id
mysql> show processlist;
+----+------+-----------+--------------------+---------+------+----------+------------------+
| Id | User | Host      | db                 | Command | Time | State    | Info             |
+----+------+-----------+--------------------+---------+------+----------+------------------+
|  2 | root | localhost | performance_schema | Query   |    0 | starting | show processlist |
+----+------+-----------+--------------------+---------+------+----------+------------------+
1 row in set (0.00 sec)

mysql> select connection_id();   -- 查看当前connection的id
+-----------------+
| connection_id() |
+-----------------+
|               2 |
+-----------------+
1 row in set (0.00 sec)
```

> **通过 `threads`表 中的信息，结合 `iotop -u mysql` 的输出，就可以知道某个线程的io使用情况**
>
> MySQL 5.6 版本中没有 `thread_os_id` 这个列。
> 作业一：如何将iotop中的Thread ID和MySQL5.6中的threads表中的信息对应起来。

3. 存储结构对应关系



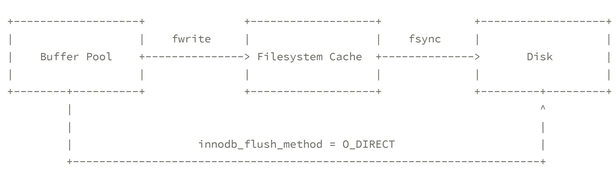
> **SSD扇区的大小一般为4K或者8K。但是为了兼容HDD，SSD通过Flash Translation Layer (FTL)的方式转换成512B**

4. O_DIRECT

- **fwrite / fsync**
  - `fwrite` 是把数据写入文件系统层（Filesystem）（可能有cache），并不能保证写入Disk
  - `fsync` 可以保证把数据写入到Disk（数据落盘）

> 只通过 `fwrite` 写入数据特别快（因为有缓存），但随后调用 `fsync` 就会很慢，这个速度取决于磁盘的 `IOPS`
> 如果不手工执行 `fysnc` ，当Filesystem的 `cache` 小于 `10%` 时，操作系统才会将数据刷入磁盘。所以可能存在数据丢失的风险，比如掉电

- **O_DIRECT**



> **`O_DIRECT` 的设置参数是告诉系统 `直接将数据写入磁盘` ，跳过文件系统的缓存。等同于使用 `裸设备` 的效果**

四. sysbench

1. 安装

建议安装 `sysbenh-0.5` 的版本

```
shell> https://github.com/akopytov/sysbench.git # 通过git clone得到源码
shell> cd sysbench
shell> ./autogen.sh
shell> ./configure  --with-mysql-includes=/usr/local/mysql56/include/  --with-mysql-libs=/usr/local/mysql56/lib/  # 关联mysql的头文件和库
##
## 注意，如果我这里使用mysql5.7.9 的include和lib ，提示我 /usr/bin/ld: cannot find -lmysqlclient_r
##

shell> make -j 2 # -j 2 表示用几个cpu核心进行编译
shell> make install # 默认安装到 /usr/local/bin ，如果有自定义目录，configure增加参数 --prefix=自定义目录
shell> echo "export LD_LIBRARY_PATH=/usr/local/mysql56/lib/:$LD_LIBRARY_PATH" >> ~/.bashrc # 添加LD_LIBRARY_PATH
shell> source ~/.bashrc
shell> sysbench --version
sysbench 0.5
```

3. 测试

```
#
# 生成测试文件
#
shell> sysbench --test=fileio \               # File IO测试
                --file-num=4 \                # 测试文件数是4个
                --file-block-size=8K \        # block size是8K
                --file-total-size=1G \        # 4个文件的总大小是1G
                --file-test-mode=rndrd \       # 测试方法是随机读
                --file-extra-flags=direct \   # direct io，跳过缓存
                --max-requests=0 \            # 一共发起多少请求，0表示任意
                --max-time=3600 \             # 测试3600s
                --num-threads=4 \             # 使用4个线程
                prepare #  run or cleanup     # prepare：生成文件
                                              # run：开始测试
                                              # cleanup：删除测试文件

## 其他说明  sysbench --test=fileio help
# --file-num=N                  创建文件数
  # --file-block-size=N           block size大小
  # --file-total-size=SIZE        文件数的大小总和
  # --file-test-mode=STRING       测试模式 {seqwr, seqrewr, seqrd, rndrd, rndwr, rndrw} （顺序写，顺序读写，顺序读，随机读，随机写，随机读写）
  # --file-io-mode=STRING         文件操作方式 {sync,async,mmap}
  # --file-extra-flags=STRING     打开文件的额外标志 {sync,dsync,direct} []
  # --file-fsync-freq=N           多少请求后执行fsync。默认是0，不执行
  # --file-fsync-all=[on|off]     是否每次操作后都执行fsync
  # --file-fsync-end=[on|off]     测完成后执行fsync，默认是on
  # --file-fsync-mode=STRING      同步的方法 {fsync, fdatasync}默认是 [fsync]
  # --file-merged-requests=N      最多多少IO请求被合并，默认为0，不合并
  # --file-rw-ratio=N             读写比例默认是 [1.5]，即 3:2



#
# 开始测试
#
shell> sysbench --test=fileio \
                --file-num=4 \
                --file-block-size=8K \
                --file-total-size=1G \
                --file-test-mode=rndrd \
                --file-extra-flags=direct \
                --max-requests=0 \
                --max-time=30 \             # 简单测试，测试30秒
                --num-threads=4 \
                --report-interval=3 \       # 每3秒产生报告
                run
sysbench 0.5:  multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 4
Report intermediate results every 3 second(s)
Random number generator seed is 0 and will be ignored


Extra file open flags: 3
4 files, 256Mb each
1Gb total file size
Block size 8Kb
Number of IO requests: 0
Read/Write ratio for combined random IO test: 1.50
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing random read test
Threads started!

[   3s] reads: 1.70 MB/s writes: 0.00 MB/s fsyncs: 0.00/s response time: 54.416ms (95%)
[   6s] reads: 1.78 MB/s writes: 0.00 MB/s fsyncs: 0.00/s response time: 55.469ms (95%)
[   9s] reads: 1.75 MB/s writes: 0.00 MB/s fsyncs: 0.00/s response time: 55.253ms (95%)
[  12s] reads: 1.66 MB/s writes: 0.00 MB/s fsyncs: 0.00/s response time: 52.120ms (95%)
[  15s] reads: 1.76 MB/s writes: 0.00 MB/s fsyncs: 0.00/s response time: 51.840ms (95%)
[  18s] reads: 1.79 MB/s writes: 0.00 MB/s fsyncs: 0.00/s response time: 50.933ms (95%)
[  21s] reads: 1.78 MB/s writes: 0.00 MB/s fsyncs: 0.00/s response time: 54.858ms (95%)
[  24s] reads: 1.88 MB/s writes: 0.00 MB/s fsyncs: 0.00/s response time: 50.857ms (95%)
[  27s] reads: 1.75 MB/s writes: 0.00 MB/s fsyncs: 0.00/s response time: 56.238ms (95%)
[  30s] reads: 1.61 MB/s writes: 0.00 MB/s fsyncs: 0.00/s response time: 64.097ms (95%)
Operations performed:  6709 reads, 0 writes, 0 Other = 6709 Total
Read 52.414Mb  Written 0b  Total transferred 52.414Mb  (1.7462Mb/sec)
  223.51 Requests/sec executed  # 这个就是IOPS

General statistics:
    total time:                          30.0160s
    total number of events:              6709
    total time taken by event execution: 120.0223s
    response time:
         min:                                  0.13ms
         avg:                                 17.89ms
         max:                                254.62ms
         approx.  95 percentile:              54.97ms

Threads fairness:
    events (avg/stddev):           1677.2500/28.16
    execution time (avg/stddev):   30.0056/0.01

##
## 上述测试随机读的速度在1.7MB/s左右，
## （1.7MB/s * 1024 / 8KB =217）换算后得到的值就是IOPS，约等于上面的223。
##
```

测试完成后执行 `cleanup`
如果是真实的测试 `max-time` 设置成一周的时间
`run` 期间可以使用 `iotop` 或者 `iostat` 进行观察