sysbench 压测 MySQL 性能及优化

作者：valarchie
原文链接：https://ld246.com/article/1586780730391
来源网站：链滴

sysbench安装流程

下载

wget https://github.com/akopytov/sysbench/archive/master.zip

unzip安装

yum install unzip

解压

unzip master.zip

安装相关依赖

yum -y install make automake libtool pkgconfig libaio-devel vim-common

yum -y install mysql-devel

编译

进入之前解压后的文件夹

```
./autogen.sh
#如果不是yum安装的原文件地址的话，需要添加参数改
./configure
make
make install
```

检验

出现版本号即为安装成功

sysbench --version

sysbench 1.1.0

性能压测

我提前准备了一台4C8G的MySQL机器
如果没有安装MySQL的话，推荐以下教程安装《linux 安装 MySQL》

准备数据

我们准备10张表，每张表有三百万数据

```
sysbench /usr/local/share/sysbench/oltp_write_only.lua --table-size=3000000 --tables=10 --t
reads=32 --db-driver=mysql --mysql-db=testdb --mysql-host=127.0.0.1 --mysql-user=root -
mysql-password=root prepare
```

开始压测

```
sysbench /usr/local/share/sysbench/oltp_read_write.lua --table-size=3000000 --tables=10 --t
reads=32 --db-driver=mysql --mysql-db=testdb --mysql-host=127.0.0.1 --mysql-user=root -
mysql-password=root run > test.log
```

测试结果

```
SQL statistics:
    queries performed:
        read:                            67424
        write:                           19264
        other:                           9632
        total:                           96320
    transactions:                        4816   (477.58 per sec.)
    queries:                             96320  (9551.58 per sec.)
    ignored errors:                      0      (0.00 per sec.)
    reconnects:                          0      (0.00 per sec.)

Throughput:
    events/s (eps):                      477.5789
    time elapsed:                        10.0842s
    total number of events:              4816

Latency (ms):
         min:                                    5.25
         avg:                                   66.73
         max:                                  283.53
         95th percentile:                      132.49
         sum:                               321393.19

Threads fairness:
    events (avg/stddev):           150.5000/5.45
    execution time (avg/stddev):   10.0435/0.02
```

清除数据

```
sysbench /usr/local/share/sysbench/oltp_write_only.lua --table-size=3000000 --tables=10 --t
reads=32 --db-driver=mysql --mysql-db=testdb --mysql-host=127.0.0.1 --mysql-user=root -
mysql-password=root cleanup
```

参数调优

由于机器的内存有8G，所以将innodb_buffer_pool_size调整至4G。

测试结果

```
SQL statistics:
    queries performed:
        read:                            101192
        write:                           28912
        other:                           14456
        total:                           144560
    transactions:                        7228   (715.85 per sec.)
    queries:                             144560 (14317.09 per sec.)
    ignored errors:                      0      (0.00 per sec.)
    reconnects:                          0      (0.00 per sec.)

Throughput:
    events/s (eps):                      715.8545
    time elapsed:                        10.0970s
    total number of events:              7228

Latency (ms):
         min:                                    4.42
         avg:                                   44.56
         max:                                  225.83
         95th percentile:                       89.16
         sum:                               322100.40

Threads fairness:
    events (avg/stddev):           225.8750/8.69
    execution time (avg/stddev):   10.0656/0.03
```

性能大概上升50%左右

其他主要优化参数

参数的配置主要有几个方面

- 连接数、io
- buffer大小:read\sort\join
- buffer_pool:大小以及实例数量
- 刷脏页策略、日志落磁盘策略

```
#mysql最大连接数
max_connections = 800

#当我们的join是ALL,index,rang或者Index_merge的时候使用的buffer。
#实际上这种join被称为FULL JOIN
join_buffer_size = 128M

#读入缓冲区的大小，将对表进行顺序扫描的请求将分配一个读入缓冲区，
#MySQL会为它分配一段内存缓冲区
read_buffer_size = 16M

#随机读缓冲区大小，当按任意顺序读取行时（列如按照排序顺序）将分配
#一个随机读取缓冲区，进行排序查询时，MySQL会首先扫描一遍该缓冲，
```

```
#以避免磁盘搜索，提高查询速度
read_rnd_buffer_size = 32M

#缓存innodb表的索引，数据，插入数据时的缓冲，专用mysql服务器设置的大小:
#操作系统内存的70%-80%最佳
innodb_buffer_pool_size = 4G

#可以开启多个内存缓冲池，把需要缓冲的数据hash到不同的缓冲池中，
#这样可以并行的内存读写
innodb_buffer_pool_instances = 8

#ORDER BY 或者GROUP BY 操作的buffer缓存大小
innodb_sort_buffer_size = 64M

#InnoDB存储引擎在刷新一个脏页时，会检测该页所在区(extent)的所有页，
#如果是脏页，那么一起刷新。这样做的好处是通过AIO可以将多个IO写操作
#合并为一个IO操作。对于传统机械硬盘建议使用，而对于固态硬盘可以关闭。
innodb_flush_neighbors = 1


#这两个设置会影响InnoDB每秒在后台执行多少操作.大多数写IO(除了
#写InnoDB日志)是后台操作的.如果你深度了解硬件性能(如每秒可以
#执行多少次IO操作),则使用这些功能是很可取的,而不是让它闲着
innodb_io_capacity = 4000

innodb_io_capacity_max = 8000
```

技术讨论群QQ:1398880

笔者个人理解总结，如有错误恳请网友评论指正。