# MySQL2 基准测试

> 基准测试（bechmark）是针对系统设计的一种压力测试

## 基准测试的策略

> 主要有两种策略：一是针对整个系统的整体测试，另外是单独测试MySQL。也被称为集成式（full-stack）以及单组形式（single-cmponent）基准测试。

测试整个系统做集成式测试的原因：
1. 测试整个应用系统，包括 WEB服务器、应用代码、网络和数据
2. MySQL并非总是应用的瓶颈
3. 才能发现各部分之间缓存带来的影响
4. 才能揭示应用的真实表现

只测试MySQL：
1. 需要比较不同的schema或查询的性能
2. 针对应用中某个具体问题的测试
3. 做一个短期的基准测试，检测出调整后的结果

## 测试指标

> 选明确选择测试目标

### 吞吐量

吞吐量指的是单位时间内的事务处理数，经典的数据库应用测试指标。这类的基准测试主要针对在线事务处理（OLYP）的吞吐量，非常适用于多用户的交互式应用。常用测试单位是每秒事务数（TPS），每分钟事务数（TPM）

### 响应时间或者延迟

这个指标用于测试任务所需的整体时间。测试的时间单位可能是微秒、毫秒、秒或者分钟。根据不同的事件单位可以计算出平均响应时间、最小响应时间、最大响应时间和所占百分比。通常使用百分比响应时间（percentile response time）来代替最大响应时间。

### 并发性

Web服务器的并发性更准确的是在任意时间有多少同时发生的并发请求。并发性测试通常不是为了测试应用能达到的并发度，而是为了测试应用在不同并发下的性能。

### 可扩展性

给系统增加一倍的工作，在理想情况下，就能获得两倍的结果。可扩展性指标对于容量规范非常有用，它可以提供其他测试无法提供的信息，来帮助发现应用的瓶颈。

# 基准测试方法

## 获取系统性能和状态

> 在执行基准测试的时候，需要尽可能多地收集被测试系统地信息，最好为基准测试建立一个目录，并且每执行一轮测试都创建单独地子目录，将测试结果、配置文件、测试指标、脚本和其他相关说明都保存在其中。

下面是一个收集MySQL测试数据的shell脚本：

```
#! /bin/sh

INTERVAL=5
PREFIX=$INTERVAL-sec-status
RUNFILE=/home/benchmarks/running
mysql -e 'SHOW GLOBAL VARIABLES' >> mysql-variable
while test -e $RUNFILE; do
    file=$(date+ %F_%I)
    sleep=$(date + %s.%N | awk "{print $INTERVAL - (\$1 % $INTERVAL)}")   #读取输入行，并且输出 INTERVAL、第一个变量和 INTERVAL 变量
    sleep $sleep
    ts="$(date + "TS %s.%N %F %T")"
    loadavg="$(uptime)"
    echo "$ts $loadvag" >> $PREFIX-${file}-status
    mysql -e 'SHOW GLOBAL STATUS' >> $PREFIX-${file}-status &
    echo "$ts $loadvag" >> $PREFIX-${file}-innodbstatus
    mysql -e 'SHOW ENGINE INNODB STATUS\G' >> $PREFIX-${file}-innodbstatus &
    echo "$ts $loadvag" >> $PREFIX-${file}-processlist
    mysql -e 'SHOW FULL PROCESSLIST\G' >> $PREFIX-${file}-processlist &
    echo $ts
done
echo Exiting because $RUNFILE does not exist
```

下面是这个脚本的一些要点：

1. 迭代是基于固定时间间隔的，每个5秒运行一次收集的动作，注意这里sleep的时间有一个特殊的技巧，如果只是简单地在每次循环时插入一条语句“sllep 5”的指令，循环的执行间隔时间一般都会大于5秒。也可以设置为1、10、30秒，推荐使用5秒或者10秒。

2. 每个文件末都包含了该轮测试开始的日期和小时。如果太大可以存到其他地方上去。

3. 每次抓取数据都会先记录当前的时间戳，所以可以在文件中搜索某个时间点的数据，也可以写一些awk或者sed脚本来简化操作。

4. 这个脚本不会处理或者过滤收集到的数据，先收集所有的原始数据，然后再基于此做分析过滤

5. 如果需要在测试完后脚本自动退出，只需要删除 /home/bechmarks/running文件即可

## 分析数据

> 从数据中抽象出有意义的操作，依赖于如何收集数据。通常要写一些脚本来分析数据。

```
#! /bin/sh
```

```
# This script converts SHOW GLOBAL STATUS into a tabulated foemat, one line per
sample in the input, with the metrics divided by the time elapsed between
samples.

awk '
    BEGIN{
        print "#ts date time load QPS";
        fmt = " %.2f";
    }
    /^TS/{ # The timestamp lines begin with TS.
        ts = substr($2, 1, index($2, ".") - 1);
        load = NF - 2;
        diff = ts - prev_ts;
        prev_ts = ts
        printf "\n%s %s %s %s",ts, $3, $4, substr($load, 1, length($load) -1 );
    }
    /Queries/{
        printf fmt, ($2-Queries)/diff
    QUeries=$2
    }
' "$@"
```

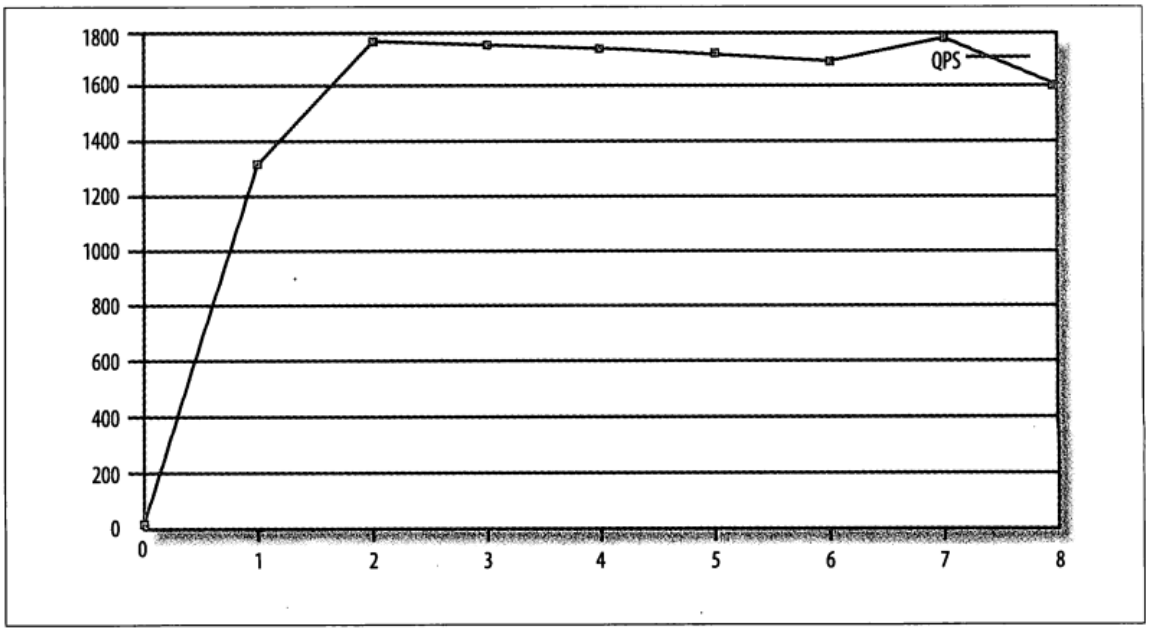
假设脚本名为 analyze，当前的脚本生成状态文件后，就可以运行脚本，可能会得到下面的结果：

```
[baron@ginger ~]$ ./analyze 5-sec-status-2011-03-20
#ts date time load QPS
1300642150 2011-03-20 17:29:10 0.00 0.62
1300642155 2011-03-20 17:29:15 0.00 1311.60
1300642160 2011-03-20 17:29:20 0.00 1770.60
1300642165 2011-03-20 17:29:25 0.00 1756.60
1300642170 2011-03-20 17:29:30 0.00 1752.40
1300642175 2011-03-20 17:29:35 0.00 1735.00
1300642180 2011-03-20 17:29:40 0.00 1713.00
1300642185 2011-03-20 17:29:45 0.00 1788.00
1300642190 2011-03-20 17:29:50 0.00 1596.40
```

第一行是列的名字，第二行的数据应该忽略，因为这是测试实际启动前的数据，接下来包含 Unix时间戳、日期、时间、系统负载、数据库的QPS（每次查询次数）五列。

## 绘图

```
gnuplot> plot "QPS-per-5-seconds" using 5 w lines title "QFS"
```




# 基准测试工具

## 集成式测试工具

- ab

  ab是apache HTTP服务器基准测试工具，它可以测试HTTP服务器每秒最多可以处理多少请求。如果是Web应用服务，这个结果可以转换成整个应用每秒可以满足多少请求。只能针对单个URL进程尽可能快的测试

- http_load

  可以通过输入文件提供多个URL，http_load在这些URL中随机选择测试。

- JMeter

  JMeter比ab和http_load都要复杂得多。它可以通过控制预热时间等参数，更加灵活地模拟真实用户的访问。还有自带的绘图接口，还可以对测试进行记录，然后离线重演测试结果。

## 单组件式测试工具

- mysqlslap

  可以模拟服务器的负载，并输出计时信息。测试时可以执行并发连接数，并指定SQL语句

- MySQL Benchmark Suite（sql-bench）

> 在MySQL的发行包中也提供了一款自己的基准测试套件。
> 这个测试套件的主要好处就是包含了大量预定义的测试，容易使用，所以可以很轻松地用于比较不同存储引擎或者不同配置的性能测试。也可以用于高层次测试，比较两个服务器的总体性能。
> 缺点： 它是单用户模式的，测试的数据集很小而且用户无法使用指定的数据，并且同一次多次运行的结果可能会相差很大。

- 等等

# 基准测试案例

## http_load

### 安装下载

```
1. 下载网址：http://www.acme.com/software/http_load/http_load-09Mar2016.tar.gz，通过ftp进行传输

2. 解压
tar -xvf http_load-09Mar2016.tar.gz

3.进入目录，下载安装
cd http_load-09Mar2016
make & make install

4. 输入 http_load出现相关信息
http_load
usage:  http_load [-checksum] [-throttle] [-proxy host:port] [-verbose] [-
timeout secs] [-sip sip_file]
            -parallel N | -rate N [-jitter]
            -fetches N | -seconds N
            url_file
One start specifier, either -parallel or -rate, is required.
One end specifier, either -fetches or -seconds, is required.
```

### 使用案例

```
1. 建立一个.txt文件，用来存储下面URL
http://www.mysqlperformanceblog.com/
http://www.mysqlperformanceblog.com/page/2/
http://www.mysqlperformanceblog.com/mysql-patches/
http://www.mysqlperformanceblog.com/mysql-performance-presentations/
http://www.mysqlperformanceblog.com/2006/09/06/slow-query-log-analyse-tools/

2. 创建一个文件夹，并且执行测试命令
# http_load -parallel 1 -seconds 10 test/urls.txt

16 fetches, 1 max parallel, 2556 bytes, in 10 seconds
159.75 mean bytes/connection
1.6 fetches/sec, 255.6 bytes/sec
msecs/connect: 285.066 mean, 1211.96 max, 199.734 min
msecs/first-response: 303.933 mean, 1144.04 max, 209.094 min
HTTP response codes:
  code 302 -- 16
```

测试结果很容易理解，只是简单的输出了请求的统计信息。下面模拟同时有五个并发用户在进行请求

```
# http_load -parallel 5 -seconds 10 test/urls.txt

74 fetches, 5 max parallel, 12126 bytes, in 10 seconds
163.865 mean bytes/connection
7.39997 fetches/sec, 1212.59 bytes/sec
msecs/connect: 355.784 mean, 2239.05 max, 199.111 min
msecs/first-response: 283.263 mean, 1462.28 max, 205.545 min
HTTP response codes:
  code 302 -- 74
```

```
# 模拟一个用户
http_load -parallel 1 -seconds 10 test/urls.txt
# 模拟五个用户
http_load -parallel 5 -seconds 10 test/urls.txt
# 预估的访问请求率（每秒5次），做压力测试
http_load -rate 5 -seconds 10 test/urls.txt
# 预估的访问请求率（每秒20次），做压力测试
http_load -rate 20 -seconds 10 test/urls.txt
```

# MySQL基准测试套件

> MySQL基准测试套件用的比较多的还是sysbench 和 mysqlslap,在这里就使用sysbench进行测试

## sysbench安装

```
1. 下载安装
curl -s
https://packagecloud.io/install/repositories/akopytov/sysbench/script.rpm.sh |
sudo bash

sudo yum -y install sysbench

2. 查看下载是否成功
sysbench --version
```

## 使用案例

```
1. 先在自己的数据库中建立sbtest数据库
mysql> create database sbtest;

2. 阿里云提供的需要生成的数据较多
# sysbench oltp_common.lua --time=300 --mysql-host=127.0.0.1 --mysql-port=3306 -
-mysql-user=root --mysql-password=yourpassword! --mysql-db=sbtest --table-
size=1000000 --tables=10 --threads=32 --events=999999999   prepare

执行如下：
Creating table 'sbtest1'...
Inserting 1000000 records into 'sbtest1'
Creating a secondary index on 'sbtest1'...
```

```
Creating table 'sbtest2'...
Inserting 1000000 records into 'sbtest2'
Creating a secondary index on 'sbtest2'...
Creating table 'sbtest3'...
Inserting 1000000 records into 'sbtest3'
Creating a secondary index on 'sbtest3'...
Creating table 'sbtest4'...
Inserting 1000000 records into 'sbtest4'
Creating a secondary index on 'sbtest4'...
Creating table 'sbtest5'...
Inserting 1000000 records into 'sbtest5'
Creating a secondary index on 'sbtest5'...
Creating table 'sbtest6'...
Inserting 1000000 records into 'sbtest6'
Creating a secondary index on 'sbtest6'...
Creating table 'sbtest7'...
Inserting 1000000 records into 'sbtest7'
Creating a secondary index on 'sbtest7'..
...
```

3. 执行测试

```
sysbench oltp_read_write.lua --time=300 --mysql-host=127.0.0.1 --mysql-port=3306
--mysql-user=root --mysql-password=yourpaaword! --mysql-db=sbtest --table-
size=1000000 --tables=10 --threads=16 --events=999999999  --report-interval=10
run
```

出现如下：

```
Running the test with following options:
Number of threads: 16
Report intermediate results every 10 second(s)
Initializing random number generator from current time


Initializing worker threads...

Threads started!

[ 10s ] thds: 16 tps: 75.95 qps: 1540.26 (r/w/o: 1081.44/305.31/153.51) lat
(ms,95%): 397.39 err/s: 0.00 reconn/s: 0.00
[ 20s ] thds: 16 tps: 96.80 qps: 1935.62 (r/w/o: 1355.94/386.08/193.59) lat
(ms,95%): 253.35 err/s: 0.00 reconn/s: 0.00
[ 30s ] thds: 16 tps: 85.64 qps: 1714.08 (r/w/o: 1198.45/344.36/171.28) lat
(ms,95%): 297.92 err/s: 0.00 reconn/s: 0.00
...
```

最终的结果为

```
SQL statistics:
    queries performed:
        read:                            364532
        write:                           104152
        other:                           52076
        total:                           520760
    transactions:                        26038  (86.68 per sec.)
    queries:                             520760 (1733.52 per sec.)
    ignored errors:                      0      (0.00 per sec.)
    reconnects:                          0      (0.00 per sec.)

General statistics:
```

```
    total time:                          300.4042s
    total number of events:              26038

Latency (ms):
         min:                                    21.00
         avg:                                   184.54
         max:                                   682.07
         95th percentile:                       297.92
         sum:                                 4804958.00

Threads fairness:
    events (avg/stddev):           1627.3750/10.17
    execution time (avg/stddev):   300.3099/0.11

4. 清理数据
sysbench oltp_read_write.lua --time=300 --mysql-host=127.0.0.1 --mysql-port=3306
--mysql-user=root --mysql-password=yourpassword! --mysql-db=sbtest --table-
size=1000000 --tables=10 --threads=16 --events=999999999  --report-interval=10
cleanup

执行结果如下:
Dropping table 'sbtest1'...
Dropping table 'sbtest2'...
Dropping table 'sbtest3'...
Dropping table 'sbtest4'...
Dropping table 'sbtest5'...
Dropping table 'sbtest6'...
Dropping table 'sbtest7'...
Dropping table 'sbtest8'...
Dropping table 'sbtest9'...
Dropping table 'sbtest10'...
```

**准备命令**

```
sysbench oltp_common.lua --time=3600 --mysql-host=127.0.0.1 --mysql-port=3306 --
mysql-user=root --mysql-password=buzhidao --mysql-db=sbtest --table-
size=10000000 --tables=64 --threads=32 --events=999999999 --report-interval
prepare
```

- `oltp_common.lua`： 执行的测试脚本，因为我们使用的是yum安装，所以需要进入到/usr/share/sysbench/ 目录下，看到sysbench自带的lua测试脚本
- `--time`：最大的执行总时间，以秒为单位，默认为10s
- `--event`：最大允许的事件个数，默认为0个，应该和 --time互相形成最大执行时间和次数
- M有SQL相关参数
  - `--mysql-host` : MySQL Server host
  - `--mysql-port` : MySQL Server port
  - `--mysql-user` : MySQL Server 账号
  - `--mysql-password` : MySQL Server 密码
  - `--mysql-db` : MySQL Server 数据库名
- `--table-size` : 表记录条数
- `--tables`： 表名
- `--threads`： 要使用的线程数，默认1个

- `--report-interval`：以秒为单位定期报告具有指定间隔的中间统计信息，默认为 0 ，表示禁用中间报告。
- `repare`：执行准备数据

**执行命令**

```
sysbench oltp_read_write.lua --time=300 --mysql-host=127.0.0.1 --mysql-port=3306 --mysql-user=root --mysql-password=yourpaaword! --mysql-db=sbtest --table-size=1000000 --tables=10 --threads=16 --events=999999999  --report-interval=10 run
```

- `oltp_read_write.lua`：执行的测试脚本。此时，我们在 `/usr/share/sysbench/` 下，寻找我们想要测试的场景。
  `oltp_read_write.lua`，表示混合读写，在一个事务中，默认比例是：
  `select:update_key:update_non_key:delete:insert = 14:1:1:1:1`。这也是为什么，我们测试出来的 TPS 和 QPS 的比例，大概在 1:18~20 左右。相当于说，一个事务中，有 18 个读写操作。
- `run`：执行测试。

### 推荐文章

```
1. 《使用 sysbench 对 mysql 压力测试》
2. 《基准测试工具 Sysbench》
```

## mysqlslap

> FROM 《MySQL压力测试工具 mysqlslap》
> mysqlslap 是一个 MySQL 官方提供的压力测试工具。

### 测试案例

```
# mysqlslap --concurrency=16,32 --iterations=3 --number-int-cols=1 --number-char-cols=2 --auto-generate-sql --auto-generate-sql-add-autoincrement --engine=innodb --number-of-queries=10000 --create-schema=sbtest2 -uroot -pMyNewPass4!

执行结果：
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
    Running for engine innodb
    Average number of seconds to run all queries: 3.626 seconds  平均延迟
    Minimum number of seconds to run all queries: 3.556 seconds
    Maximum number of seconds to run all queries: 3.678 seconds
    Number of clients running queries: 16  16个客户端
    Average number of queries per client: 625

Benchmark
    Running for engine innodb
    Average number of seconds to run all queries: 2.666 seconds
    Minimum number of seconds to run all queries: 2.590 seconds
    Maximum number of seconds to run all queries: 2.790 seconds
    Number of clients running queries: 32
```

```
    Average number of queries per client: 312
```

**测试语句**

```
mysqlslap --concurrency=16,32 --iterations=3 --number-int-cols=1 --number-char-cols=2 --auto-generate-sql --auto-generate-sql-add-autoincrement --engine=innodb --number-of-queries=10000 --create-schema=sbtest2 -uroot -pMyNewPass4!
```

- `--concurrency` ： 并发量，也就是模拟多少个客户端同时执行命令。可指定多个值，以逗号或者 `–delimiter` 参数指定的值做为分隔符
- `--iterations` ： 测试执行的迭代次数。
- `--number-int-cols` ： 自动生成的测试表中包含多少个数字类型的列，默认 1 。此处设置为 1 的原因是，因为我们上面 sysbench 我们生成了一个 int 类型的字段。
- `--number-char-cols` ： 自动生成的测试表中包含多少个字符类型的列，默认 1 。此处设置为 2 的原因是，因为我们上面 sysbench 我们生成了一个 char 类型的字段。
- `--auto-generate-sql` ： 自动生成测试表和数据。这个命令，带来的效果，就类似 sysbench 命令的 prepare 指令。
  - `--auto-generate-sql-add-autoincrement` ： 增加 auto_increment 一列。
  - 如果想看，生成的具体脚本，可以用 `–only-print` 指令，只打印测试语句而不实际执行。
- `--engine` ： 创建测试表所使用的存储引擎，可指定多个。
- `--number-of-queries` ： 总的测试查询次数(并发客户数×每客户查询次数)。
- `--create-schema` ： 测试的 schema ， MySQL中 schema 也就是 database 数据库名。
- `-uroot -pMyNewPass4!` ： 设置 MySQL 账号和密码。

## 推荐书籍

- 《MySQL 性能测试经验》
- 《MySQL 高性能压力测试》
- 《mysqlslap 使用总结》
- 《MySQL 性能测试&压力测试 - mysqlslap》