Practical sysbench

Peter Boros
Consultant @ Percona
FOSDEM 2014
This talk...

• Is beginner level
• Probably you will benefit the most if you have very little knowledge about sysbench
Agenda

• Benchmarking in general
• Benchmarking disk IO with sysbench fileio
• Benchmark MySQL with sysbench
• Some tips on processing sysbench data
A benchmark ...

- is Synthetic
- does not represent a real-world workload normally
- is good for comparing
- is easily repeatable and deterministic
Compiling sysbench

- Always use trunk
- On red hat or fedora, use Frederic Descamps's packages (lefred.be), which is usually close to trunk.

```
# cd /opt
# bzr co lp:sysbench
# ./autogen.sh
# ./configure
# make
# make install
```
Sysbench fileio
Creating files for the benchmark

```
petya@ptp:~/practical_sysbench$ sysbench --test=fileio --file-total-size=32G --file-num=32 prepare
sysbench 0.5: multi-threaded system evaluation benchmark

32 files, 1048576Kb each, 32768Mb total
Creating files for the test...
Extra file open flags: 0
Creating file test_file.0
Creating file test_file.1
Creating file test_file.2```
Fileio parameters we will use

- --file-block-size (16k)
- --file-total-size (32G)
- --file-num (32)
  --file-extra-flags (direct)
- --rand-init (on)
- --num-threads (?)
Fileio parameters we will use II.

- --file-io-mode (sync, async)
- --file-test-mode
  - rndwr, rndrd, rndwr
  - seqrd, seqwr, seqrewr
- --max-requests (0)
- --max-time (?)
- --report-interval (1)
Interpreting the output

Sample

Read throughput

fsync rate

Write throughput

Response time
Does number of threads matter?

Sync

ASync
Does number of threads matter?

Sync

ASync
Sysbench demo

```bash
cat sb_demo.sh
#!/bin/bash
sysbench --test=fileio \
    --file-block-size=16384 \
    --file-total-size=32G \
    --file-num=32 \
    --file-extra-flags=direct \
    --file-fsync-freq=0 \
    --rand-init=on \
    --num-threads=1 \
    --file-io-mode=sync \
    --file-test-mode=rndwr \
    --max-requests=0 \
    --max-time=120 \
    --report-interval=1 \
run
```
Graphing results

Asynchronous write throughput (16k random asynchronous writes)

throughput (MB/s)

time (sec)

storage ssd
Graphing results

Asynchronous write response time (16k random asynchronous writes)
Graphing results

Asynchronous write response time (16k random asynchronous writes)
Graphing results

Asynchronous write throughput (16k random asynchronous writes)
Graphing results

Asynchronous read throughput (16k random asynchronous reads)
Graphing results

Asynchronous read response time (16k random asynchronous reads)
Graphing results

Asynchronous mixed throughput (16k random asynchronous reads and writes)
Graphing results

Asynchronous mixed response time (16k random asynchronous reads and writes)
Sync vs async IO

So sync io doesn't matter?
Graphing results

Synchronous write throughput (16k random synchronous writes)
Graphing results
Graphing results
Graphing results

Synchronous read response time (16k random synchronous reads)

threads

95th percentile response time (ms)

storage ssd
Graphing results

Synchronous mixed throughput (16k random synchronous reads and writes)
Graphing results

Synchronous mixed response time (16k random synchronous reads and writes)
Benchmarking MySQL with Sysbench
Preparing test database

create database sysbench;
grant all on sbtest.*
to 'sbtest'@'localhost'
identified by 'sbtest';

sysbench
--test=/opt/sysbench/sysbench/tests/db/parallel_prepare.lua \
   --oltp-table-size=100000 \
   --oltp-tables-count=16 \
   --num-threads=32 \
   --mysql-user=sbtest \
   --mysql-password=sbtest \
   --mysql-host=127.0.0.1 \
run
Doing one benchmark iteration

```bash
sysbench \
--test=/opt/sysbench/sysbench/tests/db/oltp.lua \
--oltp-table-size=100000 \
--oltp-tables-count=16 \
--num-threads=32 \
--mysql-user=sbtest \
--mysql-password=sbtest \
--mysql-host=127.0.0.1 \
--max-requests=0 \
--report-interval=1 \
run
```
Modifying lua
Default parameters for oltp

- oltp_table_size = oltp_table_size or 10000
- oltp_range_size = oltp_range_size or 100
- oltp_tables_count = oltp_tables_count or 1
- oltp_point_selects = oltp_point_selects or 10
- oltp_simple_ranges = oltp_simple_ranges or 1
- oltp_sum_ranges = oltp_sum_ranges or 1
- oltp_order_ranges = oltp_order_ranges or 1
- oltp_distinct_ranges = oltp_distinct_ranges or 1
- oltp_index_updates = oltp_index_updates or 1
- oltp_non_index_updates = oltp_non_index_updates or 1
Reconnecting workload
Reconnecting update_index

- Just the update transaction by primary key
- If we modify it to call `db_disconnect()` in the event function, it stops after a few seconds
- With persistent connections, we seem to be ok
Fixing update_index

- The issue is tcp/ip source ip:port pair exhaustion
- Lots of connections in TIME_WAIT
- One solution is to set tcp_max_tw_buckets to a more reasonable value
Thanks!