Insights on sysbench and custom benchmarks

Alexey Kopytov, Martín Arrieta
Agenda

• what is sysbench?
• new features in recent and upcoming releases
• live demo!
Who are we?

sysbench maintainer since 2004. In the past: MySQL, Percona Server, XtraBackup, XtraDB Cluster developer.

Lead Database Consultant at Pythian, Open Source enthusiast, developer by hobby.
What is sysbench?

load generation tool

Default targets:
- MySQL / PostgreSQL
- filesystem / disk
- CPU
- RAM

extensible with Lua scripts
What is sysbench?

easy to install and setup:

```bash
$ apt -y install sysbench
$ sysbench oltp_read_write prepare
$ sysbench oltp_read_write run
```

used by:

- leading MySQL performance experts
- QA teams
- myself
What all benchmarks do

\[
T_{\text{start}} = \text{get\_time()}
\]

\[
\text{do\_something()}
\]

\[
T_{\text{delta}} = \text{get\_time()} - T_{\text{start}}
\]

\[
++i > MAX?
\]

\[
\text{aggregate()}
\]
Workloads in Lua

users define their workloads by implementing Lua hooks

sysbench does all heavy lifting: threads, random numbers, stats collection, aggregation, ...

```lua
function prepare()
  c = c or sysbench.sql.driver():connect()
  c:query("CREATE TABLE t (a INT)"
  c:query("INSERT INTO t VALUES (1)"
end

function event()
  c = c or sysbench.sql.driver():connect()
  c:query("UPDATE t SET a = a + " .. sb_rand(1, 1000))
end

function cleanup()
  c = c or sysbench.sql.driver():connect()
  c:query("DROP TABLE t")
end
```
Why Lua?

the "speed queen" of dynamic languages

designed to be embedded into C/C++ applications

simple and elegant, but powerful

LuaJIT = minimal overhead + FFI library
Lua ecosystem

Libraries for all popular databases, network protocols, file formats, ...

LuaRocks — package manager similar to RubyGems, Pip, Npm, PEAR, CPAN, ...

```
$ luarocks install http

request = require("http.request")
r = request.new_from_uri("http://example.com")
function event()
    r:go()
end
```
sysbench 1.0

MySQL Application of the Year 2017

Versatility

- scalable to modern hardware
- extended, more flexible API
- ability to use external Lua libraries
Custom benchmarks

arbitrarily complex scenarios are now possible with extended API

new sysbench-tpcc benchmark by Percona

more in the works:
  • both SQL and NoSQL (MongoDB, Tarantool)
sysbench-tpcc

TPC-C-like workload in sysbench Lua API by Percona

“industry-standard benchmark”

for those who care 😎
Why implement your own benchmark?

No benchmark is perfect

Simulating your specific applications/workload will give more relevant results

Can benchmark any part of the stack (SQL/HTTP/application servers or microbenchmarks)
Microbenchmark example

syscall.lua

```lua
ffi.cdef("long syscall(long, long, long, long);")
function event()
    ffi.C.syscall(0, 0, 0, 0)
end
```

$ sysbench syscall.lua run

Running the test with following options:
Number of threads: 1
Initializing random number generator from current time

Initializing worker threads...

Threads started!

Throughput:
  events/s (eps): 5046112.7918
  time elapsed: 10.0002s
  total number of events: 50462252
load doubles during holiday season

change this configuration parameter

move database to new hardware / cloud instance

upgrade database version

migrate to another DB vendor
sysbench tune

automated OS/hardware tuning for stable, consistent and reproducible results

$ sysbench tune list

mysqlbench Tune system for MySQL benchmarks

$ sysbench tune apply --profile=mysqlbench

Applying profile mysqlbench...

vm.swappiness = 1
net.ipv4.tcp_max_syn_backlog = 4096
net.core.somaxconn = 4096
kernel.sched_autogroup_enabled = 0
kernel.sched_minGranularityNs = 5000000
kernel numa_balancing = 0
kernel.randomize_va_space = 0

...
SysbenchRocks: http://rocks.sysbench.io

repository of sysbench-specific rocks (packages)

based on LuaRocks

tpcc and tune extensions are already there!

more extensions and libraries will be published in future
sysbench 1.0: customizable reports

<table>
<thead>
<tr>
<th>Timings</th>
<th>[8s]</th>
<th>[9s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>thds</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>tps</td>
<td>11580.79</td>
<td>11703.11</td>
</tr>
<tr>
<td>qps</td>
<td>232597.61</td>
<td>234551.37</td>
</tr>
<tr>
<td>lat (ms,95%)</td>
<td>4.10</td>
<td>3.96</td>
</tr>
<tr>
<td>err/s</td>
<td>52.99</td>
<td>35.01</td>
</tr>
<tr>
<td>reconnect/s</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

SQL statistics:

<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
<th>per sec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>read</td>
<td>1678180</td>
<td>11926.57</td>
</tr>
<tr>
<td>write</td>
<td>478000</td>
<td>339334.51</td>
</tr>
<tr>
<td>other</td>
<td>239239</td>
<td>239334.51</td>
</tr>
<tr>
<td>total</td>
<td>2395419</td>
<td></td>
</tr>
<tr>
<td>transactions</td>
<td>119369</td>
<td></td>
</tr>
<tr>
<td>queries</td>
<td>2395419</td>
<td></td>
</tr>
<tr>
<td>ignored errors</td>
<td>501</td>
<td>50.06</td>
</tr>
<tr>
<td>reconnects</td>
<td>0</td>
<td>0.00</td>
</tr>
</tbody>
</table>

General statistics:

<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>total time</td>
<td>10.0069s</td>
</tr>
<tr>
<td>total number of events</td>
<td>119369</td>
</tr>
</tbody>
</table>

Latency (ms):

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>min</td>
<td>1.42</td>
</tr>
<tr>
<td>avg</td>
<td>2.68</td>
</tr>
<tr>
<td>max</td>
<td>15.78</td>
</tr>
<tr>
<td>95th percentile</td>
<td>4.10</td>
</tr>
<tr>
<td>sum</td>
<td>319811.19</td>
</tr>
</tbody>
</table>

hard to parse into machine-readable format
function sysbench.hooks.report_intermediate(stat)
    local seconds = stat.time_interval
    print(string.format([[
        "time": %4.0f,
...
    }],[], stat.time_total, stat.threads_running, stat.events / seconds, (stat.reads + stat.writes + stat.other) / seconds, stat.reads / seconds, stat.writes / seconds, stat.other / seconds, stat.latency_pct * 1000, stat.errors / seconds, stat.reconnects / seconds))
end

{
    "time": 7,
    "threads": 32,
    "tps": 12003.44,
    "qps": {
        "total": 240990.88,
        "reads": 168816.22,
        ....
    }
}
What's next?

- loadable modules
  - sysbench playback
  - sysbench report
  - sysbench faker

- NoSQL databases
  - Tarantool
  - MongoDB

- More workloads
  - LinkBench, iibench, YCSB, ...

- https://sysbench.io
  - user + API documentation
Live Demo!
Thanks!

try creating custom workloads with sysbench!

questions? proposals? patches? let me know!

http://github.com/akopytov/sysbench

slides: http://kaamos.me/talks/dataops18

Questions?