Bottom-Up Database Hardware Benchmarking

Greg Smith

2ndQuadrant US

04/13/2011
About this presentation

- The master source for these slides is: http://projects.2ndquadrant.com
- Source code to automate testing available there too
- Slides are released under the Creative Commons Attribution 3.0 United States License: http://creativecommons.org/licenses/by/3.0/us
Why should you always benchmark your hardware?

- Many useful tests will only run when the server isn’t being used yet
- Software stacks are complicated
- Spending money on upgrades only helps if you upgrade the right thing usefully
- Vendors lie
Systematic Benchmarking

- Memory
- CPU
- Disk
- Database server
- Application
Databases and the CPU

- MySQL and PostgreSQL use only a single CPU per query
- Queries executing against cached data will bottleneck on CPU
- Both CPU and memory need to be fast for individual queries to be fast
STREAM Benchmarking

http://www.advancedclustering.com/company-blog/

Greg Smith  Bottom-Up Database Hardware Benchmarking
Oracle Calling Center OLTP Benchmark

Oracle Calling Circle
Transactions per second, higher is better

<table>
<thead>
<tr>
<th>Processor Type</th>
<th>Transactions per second</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual Xeon X5670</td>
<td>580</td>
</tr>
<tr>
<td>Dual Xeon X5570 2.93</td>
<td>464</td>
</tr>
<tr>
<td>Single Xeon X5670 2.93</td>
<td>443</td>
</tr>
<tr>
<td>Dual Opteron 2435 2.6</td>
<td>321</td>
</tr>
<tr>
<td>Dual Opteron 2389 2.9</td>
<td>260</td>
</tr>
<tr>
<td>Dual Xeon E5450 3.0</td>
<td>228</td>
</tr>
<tr>
<td>Dual Xeon X5365 3.0</td>
<td>209</td>
</tr>
<tr>
<td>Dual Opteron 2356 2.3</td>
<td>207</td>
</tr>
<tr>
<td>Dual Xeon L5320 1.86</td>
<td>156</td>
</tr>
<tr>
<td>Dual Xeon 5080 3.73</td>
<td>100</td>
</tr>
</tbody>
</table>

stream-scaling memory read test

git clone git://github.com/gregs1104/stream-scaling.git
cd stream-scaling
./stream-scaling
Memory Speeds - DDR2 Era

### DDR2 Era Chart

- **T7200**: DDR2/667
- **E5420**: DDR2/667
- **4 x 8347**: DDR2/667
- **E2180**: DDR2/800
- **x2 4600+**: DDR2/800
- **2 x 280**: DDR2/800
- **Q6600**: DDR2/800
- **8 x 8431**: DDR2/800
Memory Speeds - DDR3 Era

[Graph showing memory speeds for DDR3 era with different configurations and core counts.]
Sources for slow memory results

- Single channel RAM/slot mistakes
- Incorrect SPD/timing/voltage
- Bad RAM/CPU multiplier combination
- Poor quality RAM
- BIOS setup error
CPU Tests

- Synthetic CPU benchmarks don’t work anymore
- Use an in-memory, CPU intensive database test instead
- Heavy trivial SELECT statements work well
Sources for slow CPU results

- Slow memory
- Power management throttling
- Linux: /proc/cpuinfo shows 1000MHz suggests you need to adjust the CPUFreq Governor to “performance”
Disk Tests

- Sequential write: INSERT, Bulk loading (when not CPU limited)
- Sequential read: SELECT * FROM and similar table sequential scans
- Seeks: SELECT using index, UPDATE
- Commit fsync rate: INSERT, UPDATE
Compute 2X the size of your RAM in 8KB blocks
blocks = 250,000 * gigabytes of RAM

```
time sh -c "dd if=/dev/zero of=bigfile bs=8k count=X && sync"
time dd if=bigfile of=/dev/null bs=8k
```

Watch `vmstat` and/or `iostat` during disk tests
`vmstat`'s `bi` and `bo` will match current read/write rate
Note the CPU percentage required to reach the peak rate
./bonnie++

bon_csv2html

- Ignore the per-character and create results, look at the block output/input ones
- Random Seeks:
- The test runs SeekProcCount processes (default 3) in parallel, doing a total of 8000 random seek reads to locations in the file. In 10% of cases, the block read is changed and written back.
bonnie++ ZCAV

./zcav -f/dev/sda > t500

- Must get a recent version of bonnie++ for ZCAV to scale properly for TB drives (1.03e works)
- ZCAV on experimental branch (1.96) gave useless results for me
- Download somewhat broken gnuplot script sample and typical results from:

unset autoscale x
set autoscale xmax
unset autoscale y
set autoscale ymax
set xlabel "Position GB"
set ylabel "MB/s"
set key right bottom
set terminal png
set output "zcav.png"
plot "raid0" title "7200RPM RAID 0 3 Spindles",
    "single" title "7200RPM Single Drive"
Seagate Momentus 7200.4 Laptop

7200RPM Laptop ST9320423AS Max/Min=92/49
Scaling of 3-Disk RAID0 with 7200RPM SATA Disks

RAID0 Speedup: 3 Spindles

RAID0 3 Spindles Max/Min=230/110
7200RPM WD1600AAJS Max/Min=77/37
3-Disk Short-Stroked RAID0, larger 7200RPM SATA

RAID0 Speedup: 3 Spindles, Fastest Half Short-Stroked

1/2 Short-Stroked RAID0 3 Spindles Max/Min/Avg=343/102/285
7200RPM WD6400AACS Max/Min/Avg=116/55/85
THREADS=1

GB=10

MODE=rndrd

OPTIONS="--test=fileio --num-threads=$THREADS
--file-block-size=8K --file-test-mode=$MODE
--file-num=$GB --file-total-size=${GB}G
--file-fsync-freq=0 --file-fsync-end=no"

sysbench prepare $OPTIONS

sysbench run --max-time=60 $OPTIONS

sysbench cleanup $OPTIONS
Sample sysbench random read results

Read 78.125Mb Written 0b
Total transferred 78.125Mb (1.0059Mb/sec)
128.75 Requests/sec executed

▶ That’s 128.75 seeks/second over 10GB, resulting in a net throughput of 128.75 * 8KB/s = 1.01MB/s
▶ Consider both the size of the disk used and the number of clients doing seeks
More customizable seek tests

- bonnie++ experimental (currently at 1.96)
- iozone
- fio
- Windows: HD Tune does everything but commit rate
Sources for slow disk results

- Poor mapping to underlying hardware
- Buggy driver
- Insufficient bandwidth to storage
- Bottlenecking at CPU/memory limits
- Bad performing filesystem or filesystem misaligned with stripe sizes
- Writes faster than reads? Probably low read-ahead settings somewhere.
- Vibration: don’t shout at your JBODs! They don’t like it!

http://www.youtube.com/watch?v=tDacjrSCeq4
sysbench --test=fileio --file-fsync-freq=1 --file-num=1
--file-total-size=16384 --file-test-mode=rndwr run
| grep "Requests/sec"

- Database insert-only tests
Solid State Drives

- Writes are batched to block size by caching small ones
- There must be a write cache for good speed and to reduce wear
- Look for the battery, capacitor, or super-capacitor to allow flushing writes when power is lost
- Ask what happens when someone trips over the power cord
- Manufacturer doesn’t say? Assume your data is toast.
Good and bad drives

- Bad: Intel X25-M, X25-E, and most cheap consumer drives
- Good: OCZ Vertex 2 Pro, Intel 320 series
- Enterprise SSD models usually get this right, sometimes with weird downsides
- Run diskchecker.pl and pull the plug yourself: http://brad.livejournal.com/2116715.html
- Pull the plug on write-heavy database tests, too
Sample laptop disk specification

- ST9320423AS Momentus 7200.4 320GB
- 7200 RPM
- 16MB Cache
- Average seek: 11ms read/13ms write
- Average rotational latency: 4.17ms
Computed parameters

- Rotational latency = $\frac{1}{\text{RPM}} \div 60 \div 2$
- IOPS = $\frac{1}{(\text{latency} + \text{seek})}$
- IOPS = $\frac{1}{\left(\left(\frac{1}{(\text{RPM}/60)})/2\right) + S\right)}$
- IOPS = $\frac{1}{(4.17\text{ms} + 11\text{ms})} = 65.9$ IOPS
Sample disk results

<table>
<thead>
<tr>
<th>Disk Count</th>
<th>Seq Rd</th>
<th>Seq Wr</th>
<th>bonnie++ Seq seeks</th>
<th>Read-only Seq seeks</th>
<th>Commit Rate</th>
<th>Drive Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>71</td>
<td>58</td>
<td>232 @ 4GB</td>
<td>194 @ 4GB</td>
<td>105/s</td>
<td>7200.4</td>
</tr>
<tr>
<td>1</td>
<td>59</td>
<td>54</td>
<td>177 @ 16GB</td>
<td>56 @ 100GB</td>
<td>10212/s</td>
<td>WD160</td>
</tr>
<tr>
<td>3</td>
<td>125</td>
<td>119</td>
<td>371 @ 16GB</td>
<td>60 @ 100GB</td>
<td>10855/s</td>
<td>RAID0</td>
</tr>
<tr>
<td>1</td>
<td>254</td>
<td>147</td>
<td>3935@ 32GB</td>
<td>3417@100GB</td>
<td>5005/s</td>
<td>SSD</td>
</tr>
</tbody>
</table>

- Commit rate for 7200.4 laptop drive is 1048/s with unsafe volatile write cache
- Non-laptop spinning drives include a 256MB battery-backed write cache, Linux SW RAID
Using sysbench for database comparisons

- Originally targeted MySQL database testing
- Use current source code from development repo: https://launchpad.net/sysbench
- Now tests PostgreSQL fairly for OLTP read-only transactions
- Standard OLTP tests quietly fail on PostgreSQL due to transaction isolation differences
apt-get install bzr

bzr checkout https://code.launchpad.net/\sysbench-developers/sysbench/0.4

cd 0.4

./autogen.sh

./configure --with-pgsql

make
Server configuration for sysbench results

- Quad-Core Intel i870, 8 Hyper-Threaded Cores
- 16GB DDR3-1600 RAM
- Areca ARC-1210 SATA II PCI-e x8 RAID controller, 256MB write cache
- DB: 3x640GB Western Digital SATA disks, short-stroked, Linux software RAID-0
- WAL: 160GB Western Digital SATA disk
- Ubuntu 10.04, Linux Kernel 2.6.32-26-generic x86_64
- OS on separate disk
- XFS filesystem
- Default database configurations
sysbench with MySQL

echo "create database sysbench;" | mysql -h localhost -u root
sysbench --mysql-user=root --db-driver=mysql
  --mysql-table-engine=innodb --mysql-db=sysbench
  --test=oltp prepare
sysbench ...
  --oltp-read-only=on --oltp-test-mode=simple
  --init-rng --max-requests=0
  --max-time=$TIME --num-threads=$THREADS run
sysbench ...
  cleanup
sysbench with PostgreSQL

```bash
sudo su - postgres -c "createdb sysbench"
sudo su - postgres -c "psql -c \"alter user postgres with password 'password';\""
sysbench --pgsql-user=postgres --pgsql-password=password
  --pgsql-db=sysbench --pgsql-host=localhost
  --db-driver=pgsql --test=oltp prepare
sysbench ... --oltp-read-only=on --oltp-test-mode=simple
  --init-rng --max-requests=0
  --max-time=$TIME --num-threads=$THREADS run
sysbench ... cleanup
```
sysbench read-only size scaling, 10,000 rows
Simple PostgreSQL Configuration - 2GB or more of RAM

- `shared_buffers = 512MB`
- `checkpoint_segments = 32`
- `wal_buffers = 16MB`
- `http://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server`
- To set these parameters and the also important `work_mem`:
  - `apt-get install pgtune`
  - `https://github.com/gregs1104/pgtune`
Serious database application tests

- Include read and write transactions
- Track latency as well as transactions/second throughput
- Note size of database relative to RAM
- Make sure load generator isn’t the bottleneck
- Only real way to test subtle tuning like I/O scheduling
What should you do?

- Trust no one
- Don’t start on application benchmarks until you’ve proven basic performance
- Don’t spend too long on basic performance if you can switch to application benchmarks
- Vendors alternate among lying, misunderstanding what you want, and trying to make you feel dumb
- Use simple, standard tools whenever possible to minimize vendor disputes
- Be prepared to translate to your vendor’s language and subvert their agenda
- Never spend real money on hardware unless you can return it if it sucks
Performance tuning of PostgreSQL 8.1 through 9.0, from hardware to scaling via replication

And lots of hardware, OS tuning, and monitoring
Questions?

- The BOFs await...