

# 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

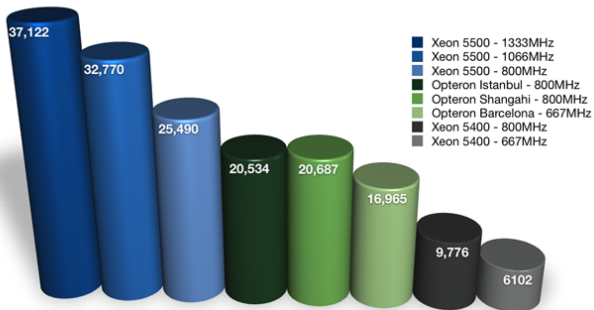
# Systematic Benchmarking

- ▶ Memory
- ▶ CPU
- ▶ Disk
- ▶ Database server
- ▶ Application

# Databases and the CPU

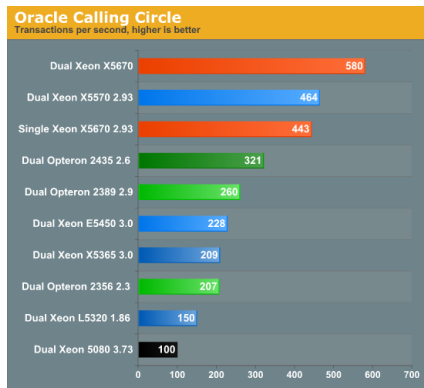
- ▶ MySQL and PostgreSQL use only a single CPU per query
- ▶ Queries executing against cached data will bottleneck on CPU
- ▶ Both CPU and memory need to be fast for individual queries to be fast

## Stream Memory Bandwidth MB/s (Triad)



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

# Oracle Calling Center OLTP Benchmark

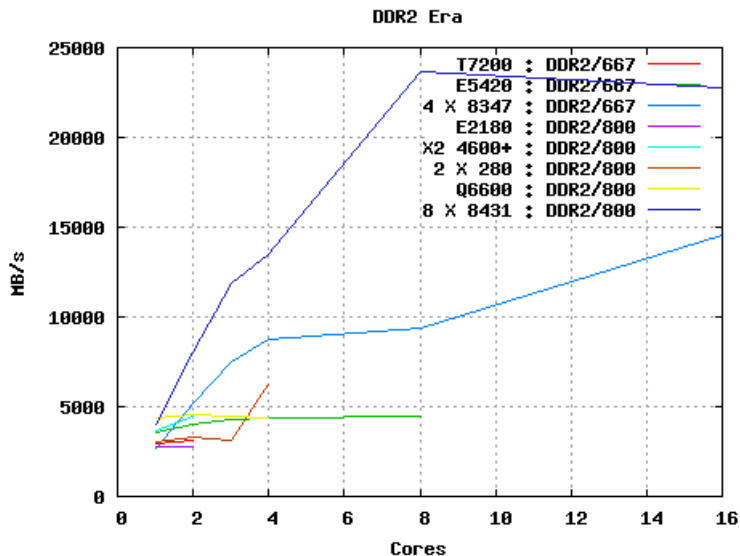


<http://it.anandtech.com/IT/showdoc.aspx?i=3769&p=4>

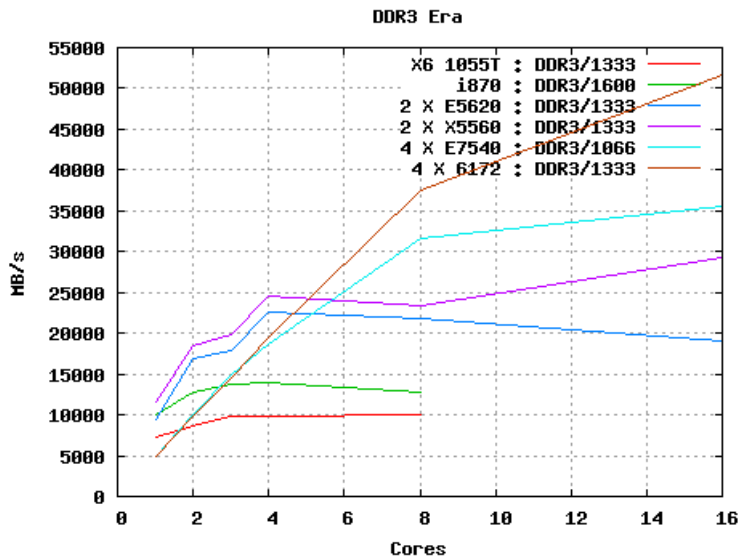
# stream-scaling memory read test

```
git clone git://github.com/gregs1104/stream-scaling.git
cd stream-scaling
./stream-scaling
```

# Memory Speeds - DDR2 Era



# Memory Speeds - DDR3 Era



# 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

- ▶ 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”

- ▶ 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
- ▶  $\text{blocks} = 250,000 * \text{gigabytes of RAM}$

```
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.

```
./zcav -f/dev/sda > t500
```

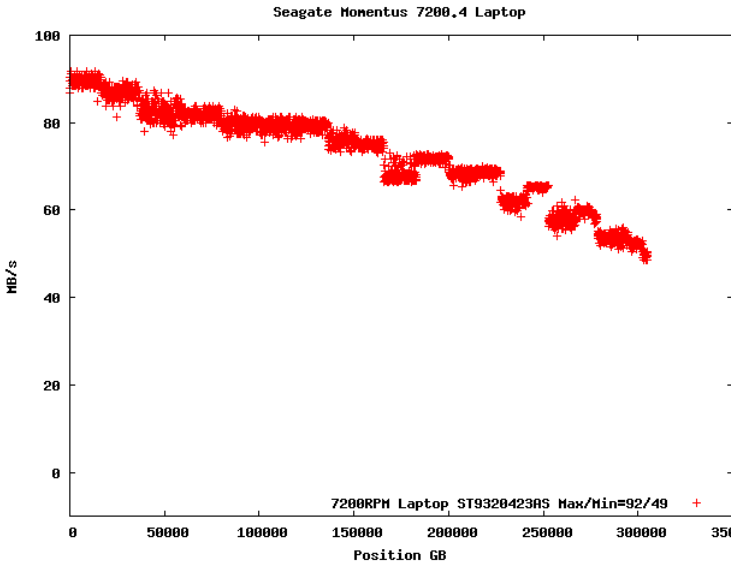
- ▶ Must get a recent version of bonnie++ for ZCAV to scale properly for TB drives (1.03e works)
- ▶ ZCAV on experimental branch (1.96) gave useless results for me
- ▶ Download somewhat broken gnuplot script sample and typical results from:

<http://www.coker.com.au/bonnie++/zcav/results.html>

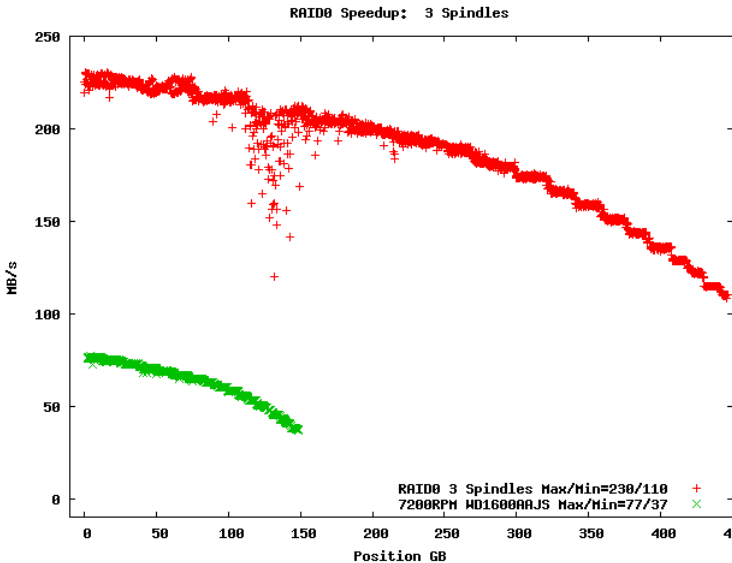
# Improved bonnie++ ZCAV gnuplot script

```
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"
```

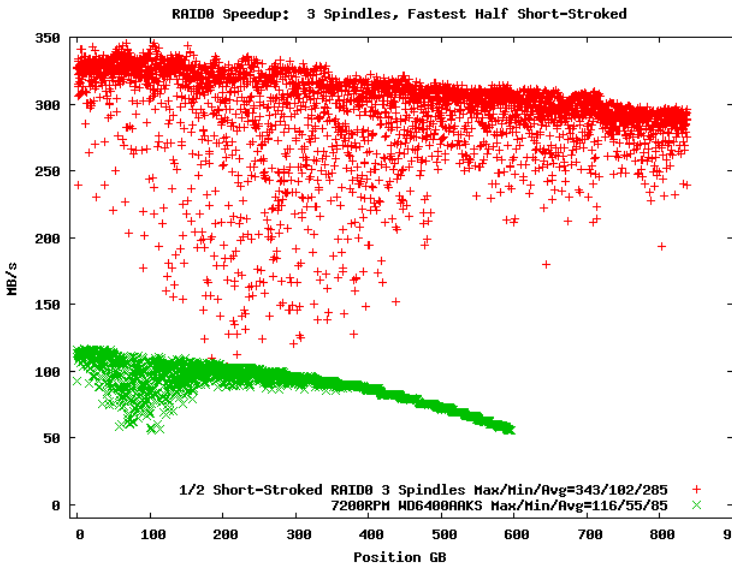
# bonnie++ ZCAV: Laptop 7200RPM Disk



# Scaling of 3-Disk RAID0 with 7200RPM SATA Disks



# 3-Disk Short-Stroked RAID0, larger 7200RPM SATA



# Read seeks/second - sysbench

```
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  $128.75 * 8\text{KB/s} = 1.01\text{MB/s}$
- ▶ 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://it.toolbox.com/blogs/database-soup/the-problem-with-iscsi-30602>

<http://blog.endpoint.com/2008/09/filesystem-io-what-we-presented.html>

<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

- ▶ 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 =  $1 / \text{RPM} / 60 / 2$
- ▶  $\text{IOPS} = 1 / (\text{latency} + \text{seek})$
- ▶  $\text{IOPS} = 1 / (((1 / (\text{RPM} / 60)) / 2) + S)$
- ▶  $\text{IOPS} = 1 / (4.17\text{ms} + 11\text{ms}) = 65.9 \text{ IOPS}$

# IOPS Calculators and Info

<http://www.wmarow.com/strcalc/>

[http://www.dbasupport.com/oracle/ora10g/disk\\_IO\\_02.shtml](http://www.dbasupport.com/oracle/ora10g/disk_IO_02.shtml)

<http://storageadvisors.adaptec.com/2007/03/20/sata-iops-measurement/>

# Sample disk results

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

- ▶ 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 bzip2
bzip2 -d <code>https://code.launchpad.net/\
  sysbench-developers/sysbench/0.4</code>
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

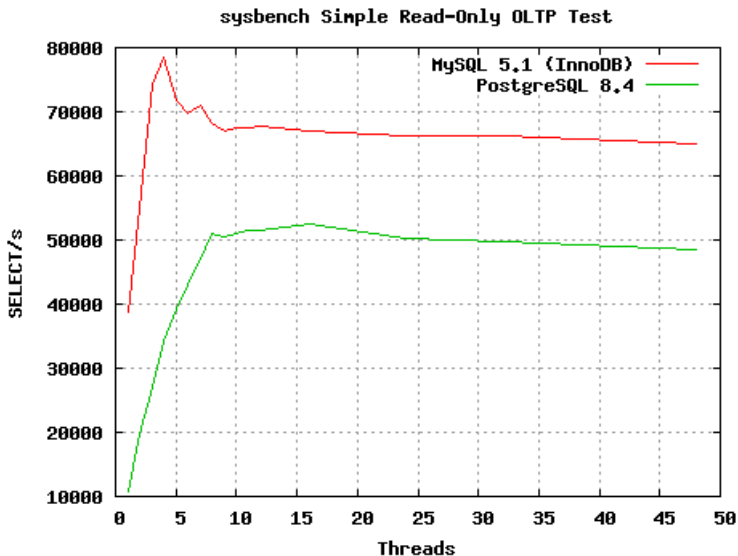
sysbench ... --oltp-read-only=on --oltp-test-mode=simple
--init-rng --max-requests=0
--max-time=$TIME --num-threads=$THREADS run

sysbench ... cleanup
```

# sysbench with PostgreSQL

```
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](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>

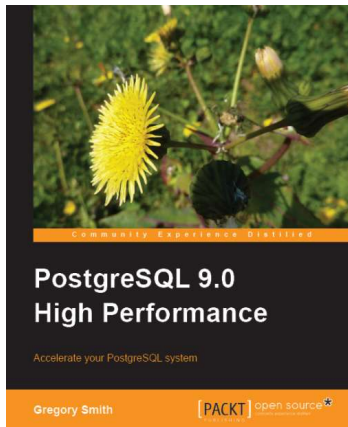
# Serious database application tests

- ▶ 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

For more information...



- ▶ Performance tuning of PostgreSQL 8.1 through 9.0, from hardware to scaling via replication
- ▶ And lots of hardware, OS tuning, and monitoring

2ndQuadrant   
Professional PostgreSQL



# Questions?

- ▶ The BOFs await...