



PERCONA
Performance Consulting Experts

High Performance from a Boring Architecture

Baron Schwartz

Percona Performance Conference

April 2009

Oooh, buzzwords!

- Sexy topics and buzzword-compliant architectures are all the rage at conferences
 - “sharding”
 - memcached
 - how <hot startup> does it
 - SSD
 - scale-out
 - cloud computing
- I do not dismiss these; they are valid, but not the only way
 - Let's talk about the kinds of things you can do for your average company, with average people

For average users

- Average = perhaps a few people
- Wearing all hats
 - architect
 - coder
 - DBA
 - sysadmin
- Thousands of \$\$ for whole system
 - Not upper tens of thousands

Context

- The system I helped build, which inspires this talk, was eventually sharded
- But the key word here is *eventually*
- Had we not done what I'll talk about, we'd have hit the wall years sooner
- How much can you win?
 - I have a client who's using *tens of times* more hardware for a similar workload that is *much smaller* and costs *hundreds of times* more hardware!
 - i.e. it is possible to make a small application seem big

Why Shard?

- Why do you ***need*** to shard?
 - Usually because you cannot get enough write I/O capacity on a single server
 - Reads can be scaled easily via other means
 - And even writes can be optimized *significantly*
 - Look for a balance between performance and cost

Make it measurable

- Arm yourself with information to direct your efforts
- Get Monitoring and Trending!
 - <http://code.google.com/p/mysql-cacti-templates/>
 - Nagios
- Build profiling into your application
 - Measure time and resources consumed for everything
- Capture metrics in an easy-to-analyze way

Identify the winners and losers

- Which parts of your architecture cost the most?
 - are the most labor-intensive?
 - break the most?
 - are most risky?
 - $\text{exposure} = \text{likelihood} * \text{severity}$
 - are hardest to replace?
 - are hardest to understand or test?
 - require the most specialized skills?
 - are hardest to scale?
 - of course, which ones Just Work?

Do it smarter, not harder

- Spreadsheets are your friend for the previous slide
- Focus your efforts at every level in the stack
 - Profile your application code
 - Profile the database (to the extent possible)
 - Most important: measure, monitor, and profile the user experience! (“user” could mean the application)

The bleeding edge

- There's a reason it's called “bleeding”
- Brittle, incomplete, awkward technologies may cost
 - Ask for advice if you're not familiar with it

Go commodity

- Use commodity hardware, software, tools, techniques, skills, etc etc.
- Commodity does not mean “cheap junk”
 - “...a product that is the same no matter who produces it, such as petroleum, notebook paper, or milk.” - Wikipedia
- “Pioneers Often Die With Arrows In Their Backs”

Archive & Purge Your Data

- Big, fast, cheap: pick any two!
- The hardest part about this is the human element
- You do cost/benefit analysis, and so should users

Sidestep tough problems

- Real-time processing is a big energy suck
- All models are wrong; some models are useful
- Avoid gold-plating
- Remember: perfect is the enemy of good
 - (OK, that's really a note to myself.)

Ask for help

- Myth: “no one else could understand my problems”
 - So you're terminally unique, eh?
- Don't just “tune via Google”

Take a Scientific Approach

- Guesswork will cost you more than anything.
- That's my guess anyway.

Use the right technology

- MySQL might not be the answer
- Ever heard of BerkeleyDB?
 - How about Tokyo Cabinet?
- How about the good old filesystem?
- Use the right storage engine

Use Replication if You Can

- Offload reads
- Offload large jobs
- Keep transactions small to reduce slave lag
 - This is *not* uniquely a MySQL problem

Minimize exposure

- Don't put MySQL on the critical path
- Don't do work in the database if you can do it elsewhere

Benchmark Components

- Have you tested your disk subsystem performance?
 - If not, how do you know you're not getting 100x worse performance than you could?

Assemble Simple Components

- Don't reinvent, reuse
- Simpler is generally better
- Leave yourself more time for solving real problems

Be Boring

- Axioms of building a boring system:
 - If it worked 10, 20, 30 years ago, it is worth considering
 - If a crisis makes you come alive, you have your priorities wrong
 - If you are working on urgent system problems, it's not a boring architecture.
 - You should be working on exciting new things that serve your customers and are otherwise unimportant.