The Evolution of the [REDACTED] Architecture

I Love LAMP, 2010 Edition

Me

- WUSTL 2008 with hilariously unrelated majors
- Yahoo! Ops Tools 2007-2009
- Flickr 2009-2010
- Now at SimpleGeo

Not Me

photo by Matt Biddulph

Not Me

photo by George Oates

(these are all smart people, I promise)

History

Codebase Origins

- 2002: Game Never Ending Development Starts
  - Stable PHP version: 4.2; Stable MySQL version: 4.0
- 2004: Flickr is born of GNE internals
- 2005: Yahoo! acquires Flickr
- EPIC GROWTH AT BREAKNECK PACE
  - Backend staff between 5 and 8 people at all times
- 2010: Spaghetti

Constraints & Numbers

5 billion photos, ish

Tens of millions of active users

(the autoincrement value doesn't count)

100kqps at peak

100,000 queries per second. Yes.

Also note that "peak" is not 12:00-12:05 ala Gilt Groupe.

"Peak" is "Monday afternoon."

~15 TB of distinct data
in MySQL

roughly 50/50 index/data

A Little Philosophy

seriously, a tiny little bit

“PHP - Training Wheels Without the Bike”

Unknown

Code Quality is Subjective

“7 people to make bugs. 5 to fix them.”

Then:

Now:

Then:

Now:

Doing the dumbest thing that works as fast as you can for 8 years has consequences.

Even if you're really really smart

“The coding style is idiosyncratic and will stay that way. [...] All functions in a library must start with the library name, globals too. We don't (often) use constants. An underscore at the start of a function means it's library-private, same with globals. Function names are all lowercase, split with underscores. We don't use objects.”

http://github.com/exflickr/flamework

“How do you feel about making sausage.. and watching sausage get made?”

The Flickr way writing PHP code didn't scale

in my kind-of-humble opinion

What I find amazing is how much got done in this way.

“Just go read
lib_geotagged_shapefiles_donutholes.gne
and
lib_yahoo_woe_xmlrpc_params.gne
and you’ll totally get it”

“The code is all in there...”

Testing without Objects

- Checking globals at the top of functions - test code outside of tests
- lib_mock_database.gne
- It's as unpleasant as it sounds

(Cal would probably interject that unit tests are stupid)

Introducing... objects?...

Are objects awesome?

- Most people say yes
- Some people say no
- It seems really hard to hire new programmers who can remain happy while writing and reading what looks like C without pointers

Objects were probably not the only way to improve things

But they sure made a lot of sense

objects vs Objects

class UserDelegateFactory

STOP IT!

Writing PHP as if it is another language seems like a good idea on the surface

but it leads to unhappiness

- Step 1: add objects
- Step 2: ???
- Step 3: Profit.

My Approach

- Use classes and objects in all new code
- Agree on some very loose, general standards
- Start converting things that intuitively feel like a THING to use objects
  - class User, class Photo etc (this hasn't happened yet)

My case study: OLT

```
# wc -l include/lib_offline_tasks_perform.gne
6434 include/lib_offline_tasks_perform.gne
```

A task as an object

- Insert
- Run
- Finish (succeed, fail, reschedule)
- Nice and easy

Minimizing Boilerplate

- Each task only defines one method: run()
- If a run() returns, it must have finished
  - no return ‘finished’; and return ‘error’;
- olt::insert()’s first argument is the task type
- The rest of the arguments are derived from run() using ReflectionMethod

A note on elegance

- It was actually going to be olt_foo::insert() with the exact same arguments as run()
- This requires late static binding
- We weren't on PHP 5.3 at the time
- Some of my coworkers liked the way it ended up better anyway, but to me it felt less clean (see: subjective!)

Results

- People seemed to like Hasselhoffline more than the old system
- Tasks are currently being converted from the old system
  - Unfortunately, changing old poorly written tasks to the new style does not automatically fix them :(

A quick word on Exceptions

Exceptions

- Were not used in the Flickr codebase until sometime around May or June
  - Attempt to minimize the probability of a failing query preventing the whole page from rendering
- This means that you basically had to check the return value of EVERY function
- Result: misleading error messages

MySQL

A Quick Aside on NoSQL!!!!

NoSQL is AWESOME

- Diversity of ideas an paradigms
- Diversity of usecases
- Diversity of tools
- Awesome spike in interest in storage
- I am working on a NoSQL data store now
  - (PLUG) Go see Malone's talk tomorrow

NoSQL is not The Silver Bullet

“Nothing user-facing uses a relational database because they’re too slow”

Michael Bryzek, CTO Gilt Groupe


ORLY?!!!!
T-minus
Wed

That kind of talk is not productive...

...yet it's everywhere

@ericflo
Eric Florenzano

How much do you want to bet this Foursquare downtime has to do with MongoDB?

4 Oct via TweetDeck ☆ Favorite ↻ Retweet ↩ Reply

Retweeted by evilhackerdude and 2 others

You can hang yourself with any kind of rope

"This was an important migration because it allowed Foursquare to keep all of their check-in data in RAM, which is essential for maintaining acceptable performance"

"Foursquare outage post mortem", mongdb-users

(pause for effect)

This isn't a MongoDB problem.

It's an "It's NoSQL, so I don't have to think about it" problem.

MySQL is a **BEAST** if you give it 132 GB memory

A database is a database is a database

The Questions are Always the Same

- How much data are you going to write
- How quickly does it need to be available
- How much of it can you afford to just lose

(CAP)

Potato vs Potato

- You have to know what it is you need and what your limits are
- You have to monitor for those limits
- You have to have a plan for what you're gonna do to continue avoiding those limits

(by the way.. you're gonna need a DBA, even if they're not called that)

Flickr is sticking to MySQL. Crazy, right?

- We know our usecases
- We know what our limits are
- We monitor those limits
  - it's really easy to monitor
- We know what we're gonna do when we get close to them

“InnoDB is a pretty good key-value store”

Making MySQL Easy

Use the tools

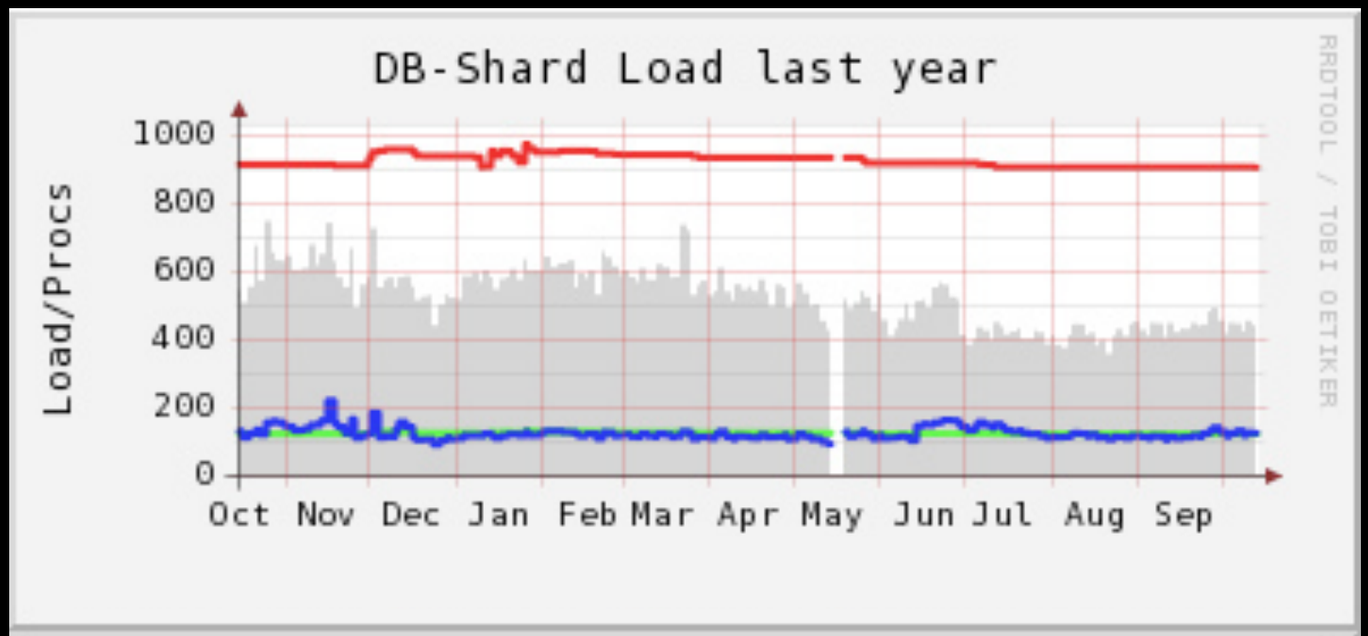
- Tons of AMAZING tools out there
- mk-query-digest is a must

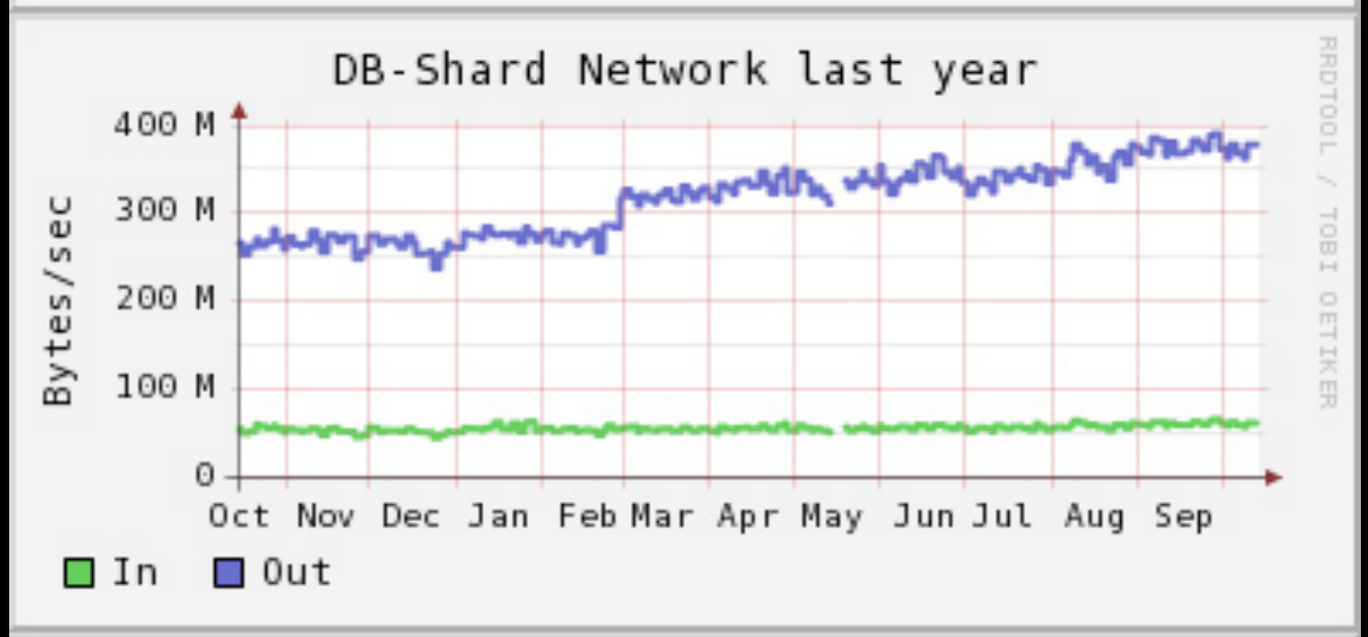
ishmael a UI for mk-query-digest

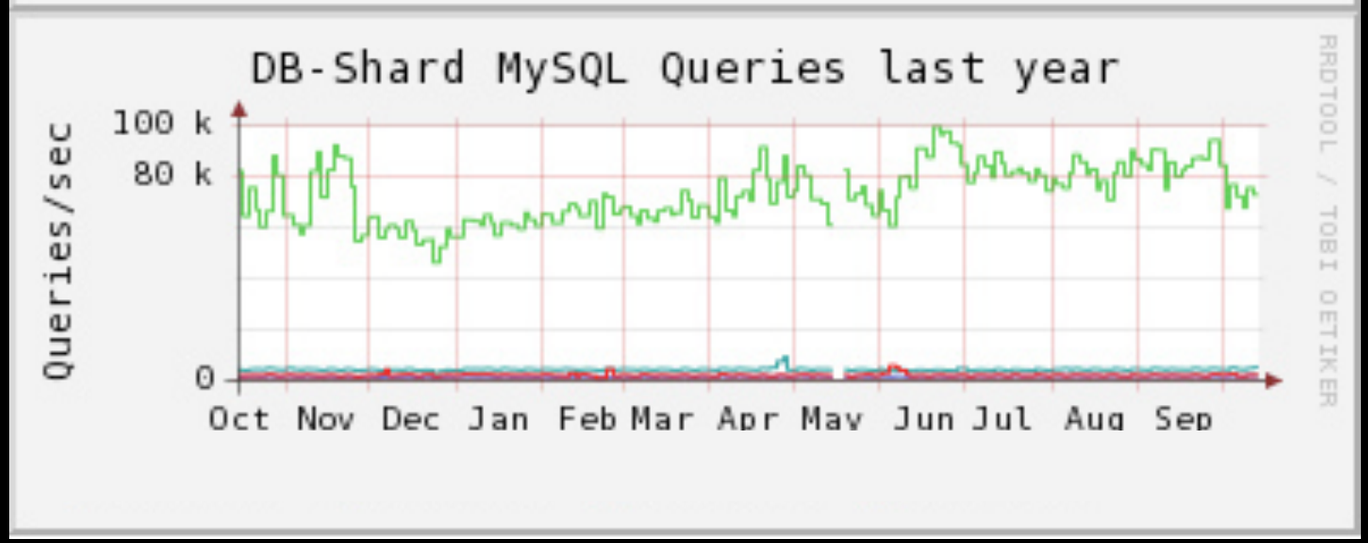
last 24 hours on db49 go

Rank	% time	% queries	Ratio	sample query		
1	19.73	23.68	0.83		explain	more
2	13.61	13.78	0.99		explain	more
3	11.77	9.42	1.25		explain	more
4	10.74	4.50	2.39		explain	more
5	5.37	8.95	0.60		explain	more
6	3.25	3.00	1.08		explain	more
7	2.48	2.48	1.00		explain	more
8	2.04	2.11	0.97		explain	more
9	1.96	2.81	0.70		explain	more
10	1.68	1.45	1.15		explain	more
11	1.60	1.13	1.43		explain	more
12	1.27	1.50	0.84		explain	more
13	1.22	1.73	0.70		explain	more

(actual queries obscured)


DB-Shard Load last year
Load/Procs
1000
800
600
400
200
0
Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep
RRDTOOL / TOBI OETIKER


DB-Shard Network last year
Bytes/sec
400 M
300 M
200 M
100 M
0
Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep
In
Out
RRDTOOL / TOBI OETIKER


DB-Shard MySQL Queries last year
Queries/sec
100 k
80 k
0
Oct Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep
RRDTOOL / TOBI OETIKER

As far as I can tell, the amount of effort spent making various datasets fit NoSQL databases is equivalent to the time it takes to get good at MySQL

(It’s just not that hard)

Things MySQL is Bad At

MySQL bad for

- **Queues** - the old offline tasks system used an InnoDB table - would start to eat itself if it got backed up to ~1 million rows
- **Counters** - Concurrent counter+=1 type writes cause deadlocks pretty quickly

Redis is good at these things

Redis

- O(1) RPUSH and LPOP (appears to hold up so far, and why wouldn't it)
  - Now backs the new offline tasks system with great success
- Much faster for counters
  - limited, largely unscientific testing shows several orders of magnitude improvement

Redis makes everyone look like a genius

... when used for the right things

Important Notes

- The durability and recovery story with Redis isn't as well defined
  - This is something we felt worked within our constraints, but still something we thought about quite a bit
- I don't think Redis is actually being used for counters yet

Java

Java

- Used as a middleware to do connection pooling
- Hopefully, eventually replication (think lots and lots of little queues)
- Used for any periodic data crunching jobs
- I wish I knew more about it so I could tell you

Hadoop

- Starting to utilize the massive Hadoop clusters that Yahoo! runs and the teams that wield them
  - What for is super duper secret, but you can probably make some pretty good guesses

Front End Performance

(story time)

FIN