Operational Efficiency Hacks

John Allspaw
Operations Engineering, Flickr

Wednesday, April 8, 2009
who am I?

Manage the Flickr Operations group

Wrote a geeky book:

The Art of Capacity Planning

Wednesday, April 8, 2009
“Efficiencies”
“Efficiencies”
Doing more with the robots you’ve got

Wednesday, April 8, 2009
“Efficiencies”

Doing more with the robots you’ve got
Doing more with the humans you’ve got

Wednesday, April 8, 2009
Some optimization
“rules”
Some optimization “rules”

- Don’t rely on being able to tweak anything.
Some optimization “rules”

- Don’t rely on being able to tweak anything.
- Don’t waste too much time tuning when you have no evidence it’ll matter.
Optimization “rules”

Performance tuning gains vs. time spent tuning

Wednesday, April 8, 2009
“We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil.”

Knuth, (or Hoare)
however...
Optimization “rules”
Optimization “rules”

That doesn’t give us an excuse to be lazy and inefficient.
Optimization “rules”

That doesn’t give us an excuse to be lazy and inefficient.
Optimization “rules”

That doesn’t give us an excuse to be lazy and inefficient.

We lean on the experience of people in the community for evidence that tuning(s) might be a worthwhile thing to do.
Optimization “rules”

“Yet we should not pass up our opportunities in that critical 3 percent.”

Knuth, (or Hoare)
So...

stop somewhere in here

obvious tuning wins

OMG
I'm wasting !@$# time
for no reason
Our Context
Our Context

- 24 TB of MySQL data
- 24 TB of MySQL data
- 32k/sec of MySQL writes
Our Context

- 24 TB of MySQL data
- 32k/sec of MySQL writes
- 120k/sec of MySQL reads
Our Context

- 24 TB of MySQL data
- 32k/sec of MySQL writes
- 120k/sec of MySQL reads
- 6 PB of photos
Our Context

- 24 TB of MySQL data
- 32k/sec of MySQL writes
- 120k/sec of MySQL reads
- 6 PB of photos
- 10TB storage eaten per day
24 TB of MySQL data
- 32k/sec of MySQL writes
- 120k/sec of MySQL reads
- 6 PB of photos
- 10TB storage eaten per day
- 15,362 service monitors (alerts)
Infrastructure Hacks

- Examples of what changing software can do
  (plain old-fashioned performance tuning)
Infrastructure Hacks

- Examples of what changing software can do
  (plain old-fashioned performance tuning)
- Examples of what changing hardware can do
  (yay for Mr. Moore!)
Leaning on compilers
(synthetic PHP benchmarks, not real-world)

(http://sebastian-bergmann.de/archives/634-PHP-GCC-ICC-Benchmark.html)
PHP (real-world)

PHP 4.4.8 to 5.2.8 migration

~15% drop in CPU
Can now handle more with less

same taste, less filling

~15% drop in CPU
Image Processing
- 2004, Flickr was using ImageMagick for image processing (version 6.1.9)
Image Processing

- 2004, Flickr was using ImageMagick for image processing (version 6.1.9)
- Changed to GraphicsMagick, about 15% faster at the time (version 1.1.5)
Image Processing

- 2004, Flickr was using ImageMagick for image processing (version 6.1.9)
- Changed to GraphicsMagick, about 15% faster at the time (version 1.1.5)
- Only need a subset of ImageMagick features anyway for our purposes
Image Processing

- OpenMP support
  (http://en.wikipedia.org/wiki/Openmp)
  - Allows parallelization of processing jobs, using multiple cores working on the same image
  - Some algorithms have more parallelization than others
Image Processing

- Test script
  - 7 large-ish DSLR photos
  - Cascade resizing each to 6 smaller sizes, semi-typical for Flickr’s workload
  - Each resize processed serially

Wednesday, April 8, 2009
Image Processing

compiler differences

GraphicsMagick Optimizations

Compiler versions

seconds

GCC 3.2          GCC 3.4.6          GCC 4.1.2

"-O1"          "-O2"          "-O3"

(GM version 1.1.14, non-OpenMP)
Image Processing

OpenMP differences

GraphicsMagick Optimizations
1.1.4 versus 1.3.5

OpenMP advantage

(gcc 4.1.2, on quad core Xeon L5335 @ 2.00GHz)

Wednesday, April 8, 2009
Image Processing

CPU differences

![Graph showing CPU Optimizations](image-url)

- Xeon Dual 2.80GHz
- Xeon L5320@1.86GHz
- Xeon L5335@2.00GHz (Clovertown)
- Xeon L5420@2.50GHz (Harpertown)
Diagonal Scaling
Diagonal Scaling

- Vertically scaling your *already* horizontally-scaled nodes
Diagonal Scaling

- Vertically scaling your *already* horizontally-scaled nodes
- *a.k.a.* “tech refresh”
Diagonal Scaling

- Vertically scaling your *already* horizontally-scaled nodes
- *a.k.a.* “tech refresh”
- *a.k.a.* “Moore’s Law Surfing”
Diagonal Scaling
Diagonal Scaling

We replaced 67 “old” web servers with 18 “new”: Wednesday, April 8, 2009
Diagonal Scaling

We replaced 67 “old” webservers with 18 “new”:

<table>
<thead>
<tr>
<th>servers</th>
<th>CPUs per server</th>
<th>RAM per server</th>
<th>drives per server</th>
<th>total power (W) @60% peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>67</td>
<td>2</td>
<td>4GB</td>
<td>1x80GB</td>
<td>8763.6</td>
</tr>
<tr>
<td>18</td>
<td>8</td>
<td>4GB</td>
<td>1x146GB</td>
<td>2332.8</td>
</tr>
</tbody>
</table>

Wednesday, April 8, 2009
Diagonal Scaling

We replaced 67 "old" websockets with 18 "new":

<table>
<thead>
<tr>
<th>servers</th>
<th>CPUs per server</th>
<th>RAM per server</th>
<th>drives per server</th>
<th>total power (W) @60% peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>67</td>
<td>2</td>
<td>4GB</td>
<td>1x80GB</td>
<td>8763.6</td>
</tr>
<tr>
<td>18</td>
<td>8</td>
<td>4GB</td>
<td>1x146GB</td>
<td>2332.8</td>
</tr>
</tbody>
</table>

~70% LESS power

49U LESS rack space
Diagonal Scaling
Diagonal Scaling

We replaced 23 “old” image processing boxes with 8 “new”
Diagonal Scaling

We replaced 23 “old” image processing boxes with 8 “new”

<table>
<thead>
<tr>
<th>server</th>
<th>photos/min</th>
<th>rack</th>
<th>total power (W) @60% peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>1035</td>
<td>23</td>
<td>3008.4</td>
</tr>
<tr>
<td>8</td>
<td>1120</td>
<td>8</td>
<td>1036.8</td>
</tr>
</tbody>
</table>
## Diagonal Scaling

We replaced 23 "old" **image processing** boxes with 8 "new"

<table>
<thead>
<tr>
<th>server</th>
<th>photos/min</th>
<th>rack</th>
<th>total power (W) @60% peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>1035</td>
<td>23</td>
<td>3008.4</td>
</tr>
<tr>
<td>8</td>
<td>1120</td>
<td>8</td>
<td>1036.8</td>
</tr>
</tbody>
</table>

- **~75%** FASTER
- **15U** LESS rack space
- **65%** LESS power
Diagonal Scaling

We replaced 23 “old” image processing boxes with 8 “new”.

<table>
<thead>
<tr>
<th>server</th>
<th>photos/min</th>
<th>rack</th>
<th>total power (W) @60% peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>~75% FASTER</td>
<td>15U</td>
<td>3008.4</td>
</tr>
<tr>
<td>8</td>
<td>65% LESS rack space</td>
<td>1120</td>
<td>1036.8</td>
</tr>
</tbody>
</table>

Wednesday, April 8, 2009
Diagonal Scaling

We replaced 23 “old” image processing boxes with 8 “new”

<table>
<thead>
<tr>
<th>server</th>
<th>photos/min</th>
<th>rack</th>
<th>total power (W) @60% peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>~75% FASTER</td>
<td>15U</td>
<td>3008.4</td>
</tr>
<tr>
<td>8</td>
<td>65% LESS</td>
<td>1120</td>
<td>1036.8</td>
</tr>
<tr>
<td></td>
<td>rack space</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LESS power</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Diagonal Scaling

We replaced 23 “old” image processing boxes with 8 “new” server photos/min.

<table>
<thead>
<tr>
<th>server</th>
<th>photos/min</th>
<th>total power (W) @60% peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>1035</td>
<td>6311</td>
</tr>
<tr>
<td>8</td>
<td>1120</td>
<td>1036.8</td>
</tr>
</tbody>
</table>

~75% FASTER

from this to this

Wednesday, April 8, 2009
What do you do with old/slow machines?
What do you do with old/slow machines?

- Liquidate
What do you do with old/slow machines?

- Liquidate
- Re-purpose as dev/staging/etc
What do you do with old/slow machines?

- Liquidate
- Re-purpose as dev/staging/etc
- “offline” tasks
Offline Tasks
Offline Tasks

- Out-of-band/asynchronous queuing and execution system, for non-realtime tasks
Offline Tasks

- Out-of-band/asynchronous queuing and execution system, for non-realtime tasks
- See here:
Offline Tasks

- Out-of-band/asynchronous queuing and execution system, for non-realtime tasks
- See here:

Offline Tasks

- Out-of-band/asynchronous queuing and execution system, for non-realtime tasks

- See here:

- See Myles Grant talk about it more here:
Offline Tasks

- Out-of-band/asynchronous queuing and execution system, for non-realtime tasks

- See here:
  

- See Myles Grant talk about it more here:
  
Runbook Hacks

“WTF HAPPENED LAST NIGHT?!”
Why?
Why?

As infrastructure grows, try to keep the Humans:Machines ratio from getting out of hand.
Why?

As infrastructure grows, try to keep the Humans:Machines ratio from getting out of hand

Some of the How:
Why?

As infrastructure grows, try to keep the Humans:Machines ratio from getting out of hand

Some of the How:

- teach machines to build themselves
Why?

As infrastructure grows, try to keep the Humans:Machines ratio from getting out of hand

Some of the How:

- teach machines to build themselves
- teach machines to watch themselves
Why?

As infrastructure grows, try to keep the Humans:Machines ratio from getting out of hand

Some of the How:

- teach machines to build themselves
- teach machines to watch themselves
- teach machines to fix themselves
Why?

As infrastructure grows, try to keep the Humans: Machines ratio from getting out of hand.

Some of the How:

- teach machines to build themselves
- teach machines to watch themselves
- teach machines to fix themselves
- reduce MTTR by streamlining
Automated Infrastructure
Automated Infrastructure

- If there is only **one** thing you do, automatic configuration and deployment management should be it.
Automated Infrastructure

- If there is only **one** thing you do, automatic configuration and deployment management should be it.

- See:
  - System Imager/Configurator ([http://wiki.systemimager.org](http://wiki.systemimager.org))
Conguration Management
Codeswarm

Legend:
- transforms
- raw
- conf
- code
- Misc

graph
root

Feb 20, 2007
Time

Machine time is cheaper than human time.

If a failure results in some commands being run to ‘fix’ it, make the machines do it.

(i.e., don’t wake people up for stupid things!)
Aggregate Monitoring
Aggregate Monitoring

Don’t care about single nodes, only care about delta change of metrics/faults

- Warn (email) on X % change
- Page (wake up) on Y % change
Aggregate Monitoring

Don’t care about single nodes, only care about delta change of metrics/faults

- Warn (email) on X % change
- Page (wake up) on Y % change

High and low water marks for some metrics
Self-Healing
Self-Healing

Make service monitoring fix common failure scenarios, notify us later about it.
Self-Healing

Make service monitoring fix common failure scenarios, notify us later about it.
Self-Healing

Make service monitoring fix common failure scenarios, notify us later about it.

Daemons/processes run on machines, will take corrective action under certain conditions, and report back with what they did.
Self-Healing

Make service monitoring fix common failure scenarios, notify us later about it.

Daemons/processes run on machines, will take corrective action under certain conditions, and report back with what they did.
Self-Healing

Make service monitoring fix common failure scenarios, notify us later about it.

Daemons/processes run on machines, will take corrective action under certain conditions, and report back with what they did.

Can greatly reduce your mean time to recovery (MTTR)
Self-Healing

Make service monitoring fix common failure scenarios, notify us later about it.

Daemons/processes run on machines, will take corrective action under certain conditions, and report back with what they did.

Can greatly reduce your mean time to recovery (MTTR)
Basic Apache Example
Basic Apache Example

1. Webserver not running?
Basic Apache Example

1. Webserver not running?

2. Under certain conditions, try to start it, and email that this happened. (*I’ll read it tomorrow*)
1. Webserver not running?

2. Under certain conditions, try to start it, and email that this happened. *(I’ll read it tomorrow)*

3. Won’t start? Assume something’s really wrong, so don’t keep trying *(email that, too)*
MySQL Self-Healing
MySQL Self-Healing

Some MySQL Issues “fixed” by the machines
MySQL Self-Healing

Some MySQL Issues “fixed” by the machines
MySQL Self-Healing

Some MySQL Issues “fixed” by the machines

- Kill long-running SELECT queries (marked safe to kill)
MySQL Self-Healing

Some MySQL Issues “fixed” by the machines

- Kill long-running SELECT queries (marked safe to kill)
- Queries not safe to kill are marked by the application as “NO KILL” in comments
MySQL Self-Healing

Some MySQL Issues “fixed” by the machines

- Kill long-running SELECT queries (marked *safe* to kill)
- Queries not safe to kill are marked by the application as “NO KILL” in comments
- Run EXPLAIN on killed queries, and report the results

Wednesday, April 8, 2009
MySQL Self-Healing

Some MySQL Issues “fixed” by the machines

- Kill long-running SELECT queries (marked safe to kill)
- Queries not safe to kill are marked by the application as “NO KILL” in comments
- Run EXPLAIN on killed queries, and report the results
- Keep track of the query types and databases that need the most killing, produce a “DBs that Suck” report
MySQL Self-Healing

Some MySQL Replication issues “fixed” by the machines, by error
MySQL Self-Healing

Some MySQL Replication issues “fixed” by the machines, by error

- Skip errors that can safely be skipped and restart slave threads
MySQL Self-Healing

Some MySQL Replication issues “fixed” by the machines, by error

- Skip errors that can safely be skipped and restart slave threads

- Force refetch of replication binlogs on:
  - 1064 (ER_PARSE_ERROR)
MySQL Self-Healing

Some MySQL Replication issues “fixed” by the machines, by error

- Skip errors that can safely be skipped and restart slave threads

- Force refetch of replication binlogs on:
  - 1064 (ER_PARSE_ERROR)

- Re-run queries on:
  - 1205 (ER_LOCK_WAIT_TIMEOUT)
  - 1213 (ER_LOCK_DEADLOCK)
Troubleshooting
Code and Config
Deploy Logs
Code and Config
Deploy Logs

1. ESSENTIAL
Code and Config
Deploy Logs

1. ESSENTIAL
2. MANDATORY
Communications

- Internal IRC
  - For ongoing discussions
  - Logged, so “infinite” scrollback
Communications

- Internal IRC
  - For ongoing discussions
  - Logged, so “infinite” scrollback
- IM Bot (*built on libyahoo2.sf.net*)
  - For production changes
  - Broadcasts all to all contacts
  - Logged, and injected into IRC
  - IM Status = who is in primary/secondary on-call
Communications

- Internal IRC
  - For ongoing discussions
  - Logged, so “infinite” scrollback
- IM Bot (*built on libyahoo2.sf.net*)
  - For production changes
  - Broadcasts all to all contacts
  - Logged, and injected into IRC
  - IM Status = who is in primary/secondary on-call
- All of IRC and IM Bot slurped into a search index
<table>
<thead>
<tr>
<th>timestamp</th>
<th>username</th>
<th>event</th>
<th>details</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009-03-26</td>
<td>aaron</td>
<td>site staged</td>
<td></td>
</tr>
<tr>
<td>2009-03-26</td>
<td>mygrant</td>
<td>site staged</td>
<td></td>
</tr>
<tr>
<td>2009-03-26</td>
<td>aaron</td>
<td>NO DEPLOY PLEASE</td>
<td></td>
</tr>
<tr>
<td>2009-03-26</td>
<td>mygrant</td>
<td>site deployed (changes...)</td>
<td></td>
</tr>
<tr>
<td>2009-03-26</td>
<td>mygrant</td>
<td>starting deploy...</td>
<td></td>
</tr>
<tr>
<td>2009-03-26</td>
<td>mygrant</td>
<td>site staged</td>
<td></td>
</tr>
<tr>
<td>2009-03-26</td>
<td>kellan</td>
<td>site deployed (changes...)</td>
<td></td>
</tr>
<tr>
<td>2009-03-26</td>
<td>kellan</td>
<td>starting deploy...</td>
<td></td>
</tr>
<tr>
<td>2009-03-26</td>
<td>aaron</td>
<td>api synced</td>
<td></td>
</tr>
<tr>
<td>2009-03-26</td>
<td>aaron</td>
<td>updated docs</td>
<td></td>
</tr>
<tr>
<td>2009-03-26</td>
<td>kellan</td>
<td>site staged</td>
<td></td>
</tr>
<tr>
<td>2009-03-26</td>
<td>jallspaw</td>
<td>config deployed</td>
<td></td>
</tr>
<tr>
<td>2009-03-26</td>
<td>jallspaw</td>
<td>starting config deploy...</td>
<td></td>
</tr>
<tr>
<td>2009-03-26</td>
<td>jallspaw</td>
<td>config staged</td>
<td></td>
</tr>
<tr>
<td>Timestamp</td>
<td>User</td>
<td>Action Description</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>------------</td>
<td>----------------------------------</td>
<td></td>
</tr>
<tr>
<td>2009-03-26 17:25:30</td>
<td>aaron</td>
<td>site staged</td>
<td></td>
</tr>
<tr>
<td>2009-03-26 17:24:51</td>
<td>mygrant</td>
<td>site staged</td>
<td></td>
</tr>
<tr>
<td>2009-03-26 17:24:31</td>
<td>aaron</td>
<td><strong>NO DEPLOY PLEASE</strong></td>
<td></td>
</tr>
<tr>
<td>2009-03-26 17:08:44</td>
<td>mygrant</td>
<td>starting deploy</td>
<td></td>
</tr>
<tr>
<td>2009-03-26 17:07:43</td>
<td>mygrant</td>
<td>site staged</td>
<td></td>
</tr>
<tr>
<td>2009-03-26 16:01:35</td>
<td>kellan</td>
<td>site deployed (changes...)</td>
<td></td>
</tr>
<tr>
<td>2009-03-26 15:54:04</td>
<td>kellan</td>
<td>starting deploy...</td>
<td></td>
</tr>
<tr>
<td>2009-03-26 15:27:59</td>
<td>aaron</td>
<td>api synced</td>
<td></td>
</tr>
<tr>
<td>2009-03-26 15:27:53</td>
<td>aaron</td>
<td>updated docs</td>
<td></td>
</tr>
<tr>
<td>2009-03-26 15:24:25</td>
<td>kellan</td>
<td>site staged</td>
<td></td>
</tr>
<tr>
<td>2009-03-26 15:07:40</td>
<td>jallspaw</td>
<td>config deployed</td>
<td></td>
</tr>
<tr>
<td>2009-03-26 15:07:18</td>
<td>jallspaw</td>
<td>starting config deploy...</td>
<td></td>
</tr>
<tr>
<td>2009-03-26 15:06:46</td>
<td>jallspaw</td>
<td>config staged</td>
<td></td>
</tr>
</tbody>
</table>
when

what

detailed
what

*also points to what commands should be used to back out the changes

Wednesday, April 8, 2009
*also points to what commands should be used to back out the changes*
*also points to what commands should be used to back out the changes
time of last deploy at top of ganglia

when

what

detailed what*

who

*also points to what commands should be used to back out the changes

Wednesday, April 8, 2009
12:45:07 PM IM BOT: peter_norby says: staging config to take filer-flickr0305a OOR
1:10:34 PM IM BOT: paulhammondorg says: about to deploy change
1:11:19 PM IM BOT: paulhammondorg says: to rollback, delete this line in SVN

$LastChangedFile: $
IM Bot (timestamps help correlation)

12:45:07 PM IM BOT: peter_norby says: staging config to take filer-flickr0305a OOR
1:10:01 PM IM BOT: paulhammondorg says: about to deploy change
1:11:19 PM IM BOT: paulhammondorg says: to rollback, delete this line in SVN
IM Bot (timestamps help correlation)

12:45:07 PM IM BOT: peter_norby says: staging config to take filer-flickr0305a OOR
1:10:31 PM IM BOT: paulhammondorg says: about to deploy change
1:11:19 PM IM BOT: paulhammondorg says: to rollback, delete this line in SVN
IM Bot (timestamps help correlation)

all IRC, IM bot into searchable history

TOTAL HITS: 2

*** DATE: Wednesday, 2008-10-01 12:36:02 *** WHO: kevbob ***

1222889709 <kevbob> ynoc paged
1222889762 <kevbob> filer-flickr0305a in degraded mode
1222889833 <Norby> yeah, that one

Wednesday, April 8, 2009
Morals of Our Stories
Morals of Our Stories

- Optimizations can be a Very Good Thing™
Morals of Our Stories

- Optimizations can be a Very Good Thing™
- Weigh time spent optimizing against expected gains
Morals of Our Stories

- Optimizations can be a Very Good Thing™
- Weigh time spent optimizing against expected gains
- Lean on others for how much “expected gains” mean for different scenarios
Morals of Our Stories

- Optimizations can be a Very Good Thing™
- Weigh time spent optimizing against expected gains
- Lean on others for how much “expected gains” mean for different scenarios
- Plain old-fashioned intuition
Some Wisdom Nuggets

Jon Prall’s 85 WebOps Rules:
Questions?

http://www.flickr.com/photos/ebarney/3348965637/
http://www.flickr.com/photos/dgmiller/1606071911/
http://www.flickr.com/photos/dannyboyster/60371673/
http://www.flickr.com/photos/bright/189338394/
http://www.flickr.com/photos/nickwheeleroz/2475011402/
http://www.flickr.com/photos/dramaqueennorma/191060346/
http://www.flickr.com/photos/telstar/2861103147/
http://www.flickr.com/photos/norby/2309046043/
http://www.flickr.com/photos/allysonk/201008992/