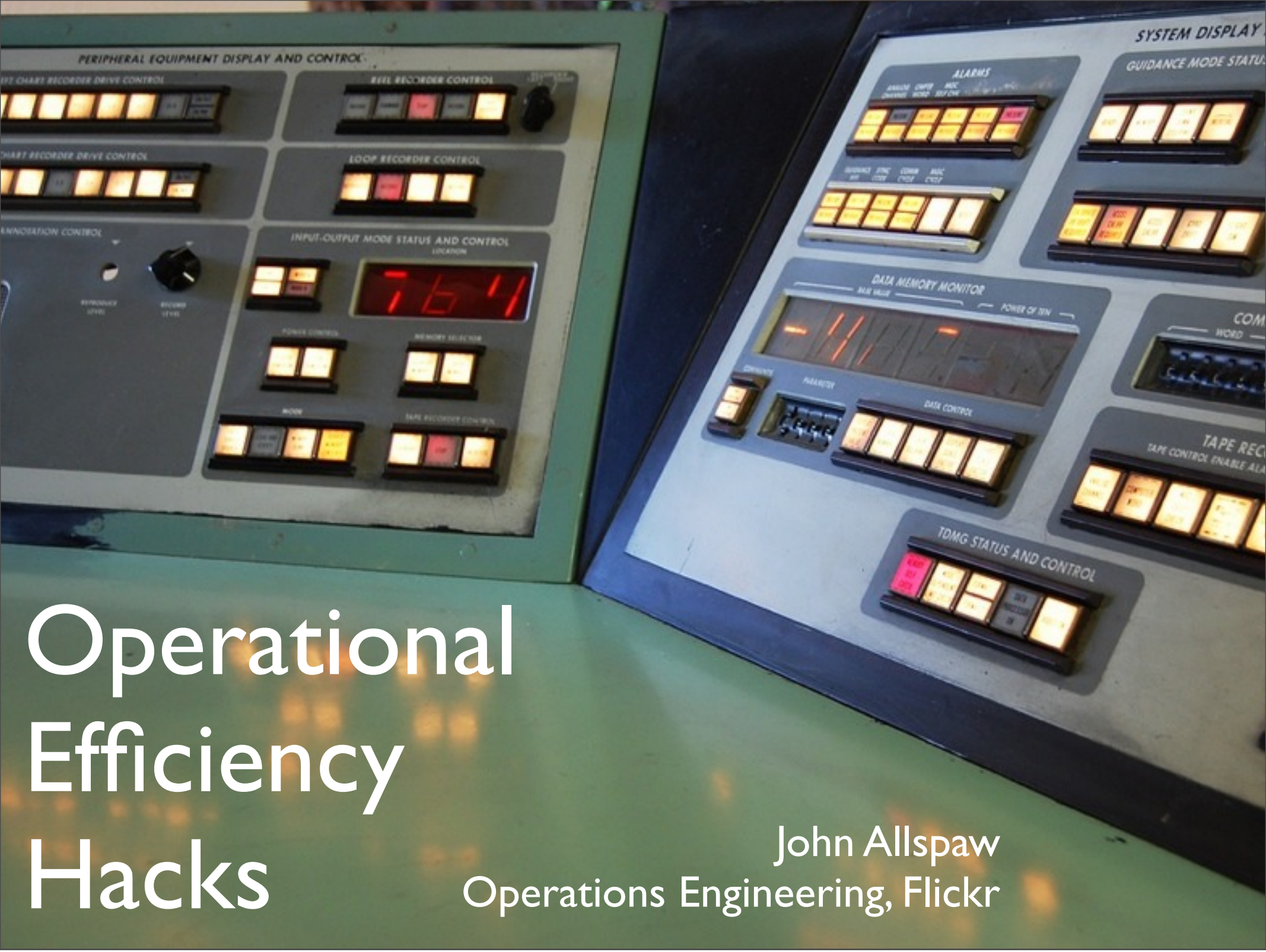


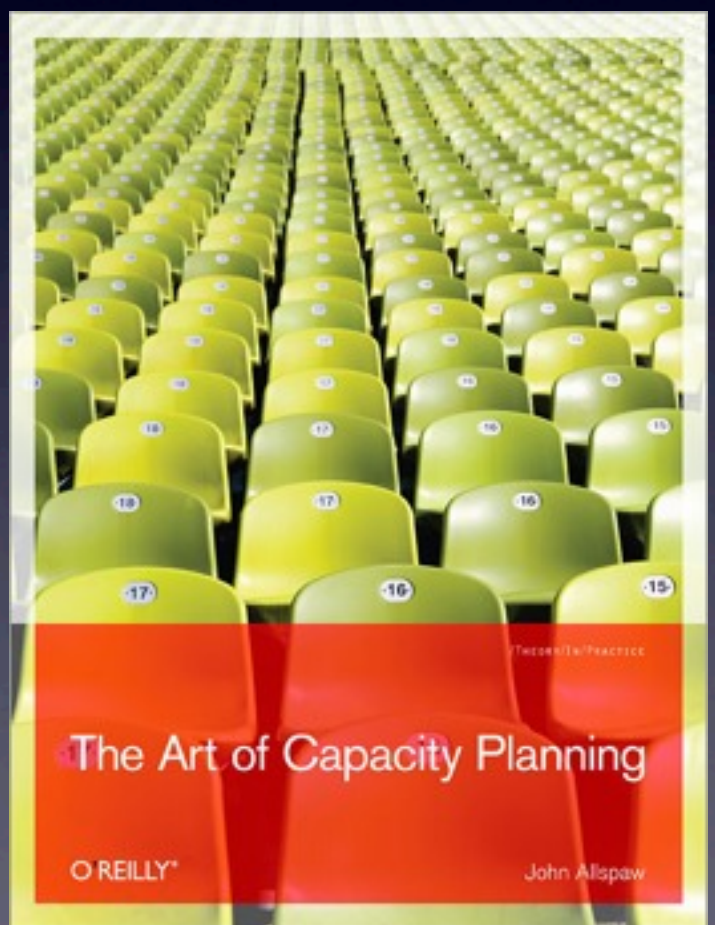
Operational Efficiency Hacks

John Allspaw
Operations Engineering, Flickr

who am I?

Manage the Flickr Operations group

Wrote a geeky book:




“Efficiencies”

“Efficiencies”

Doing more with the robots you’ve got


"Efficiencies"
Doing more with the robots you've got
Doing more with the humans you've got

Some optimization “rules”

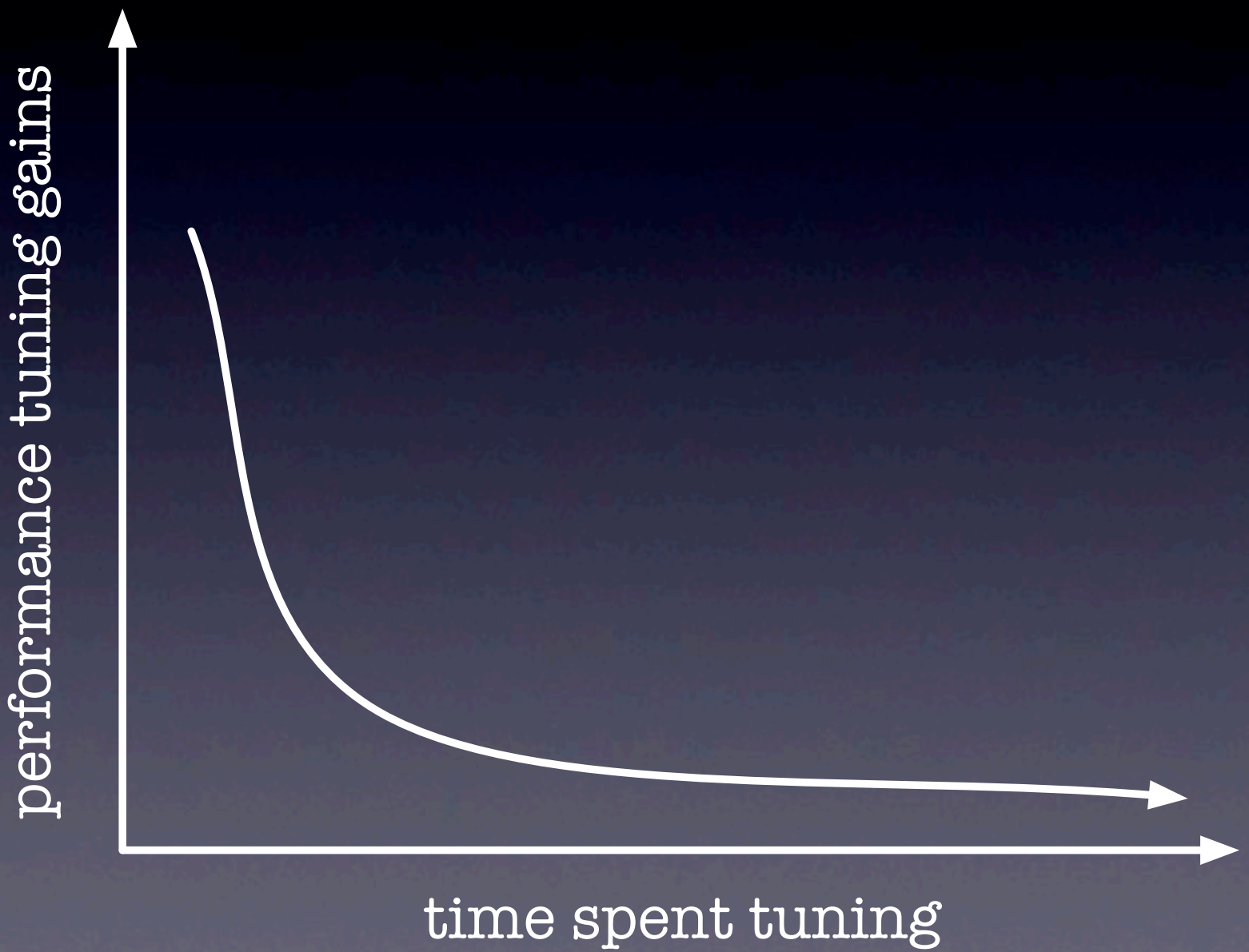
Some optimization "rules"

- 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"



Optimization "rules"

"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.

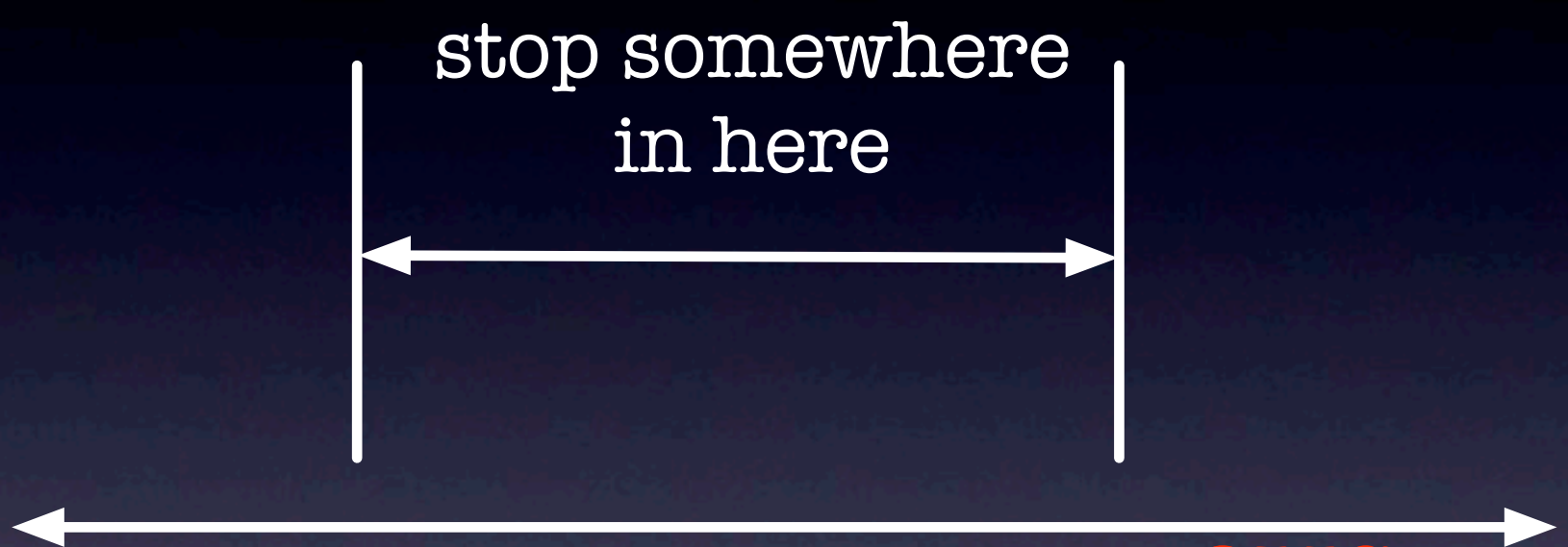
We lean on the experience of people in the community for evidence that tuning(s) might be a worthwhile thing to do.

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

Our Context

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

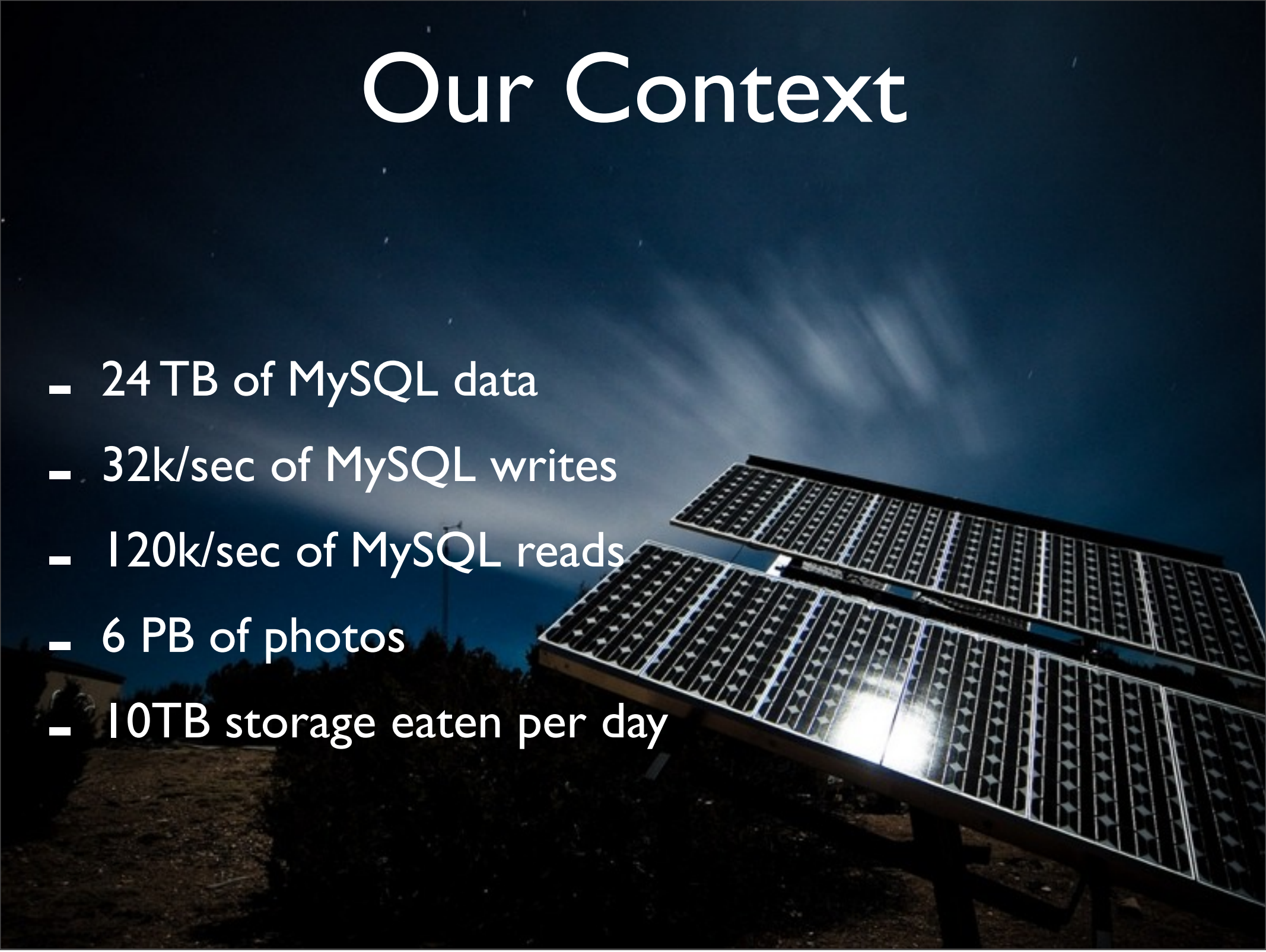
Our Context

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

Our Context

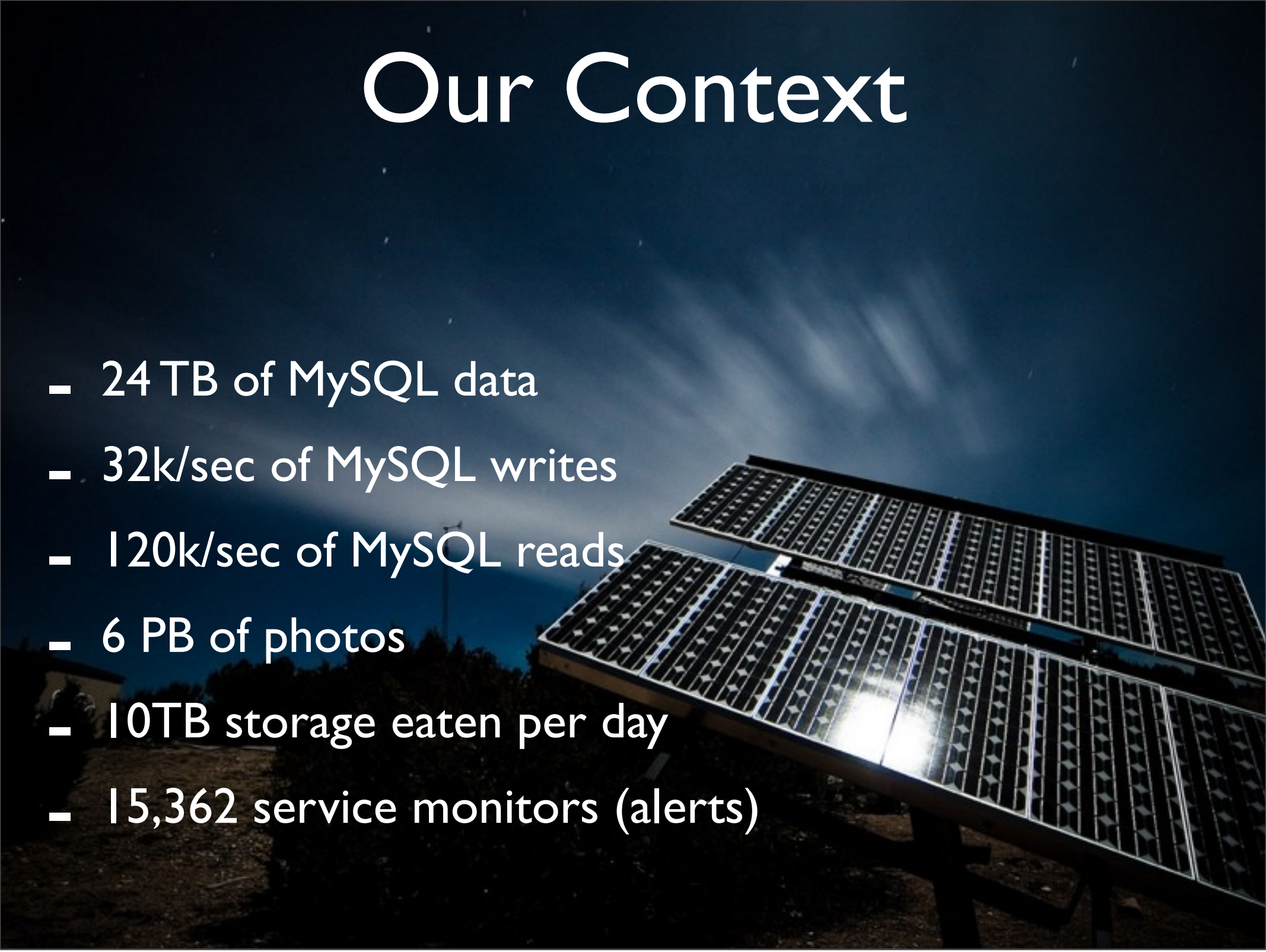
- 24 TB of MySQL data
- 32k/sec of MySQL writes
- 120k/sec of MySQL reads
- 6 PB of photos

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

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
- 15,362 service monitors (alerts)

Infrastructure Hacks

Infrastructure Hacks

- Examples of what changing software can do

  (plain old-fashioned performance tuning)

Infrastructure Hacks

- Examples of what changing software can do

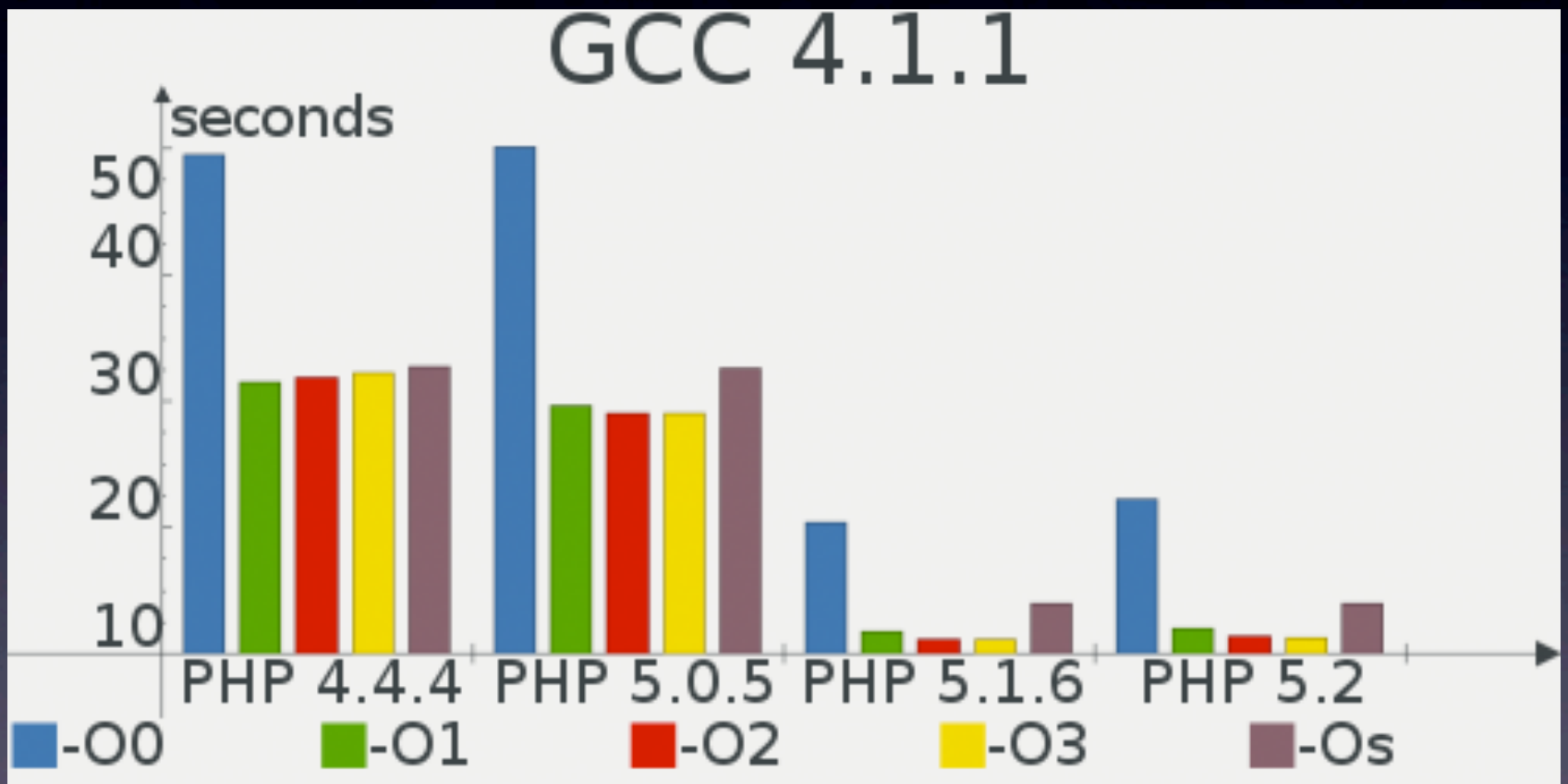
(plain old-fashioned performance tuning)

- Examples of what changing hardware can do

(yay for Mr. Moore!)

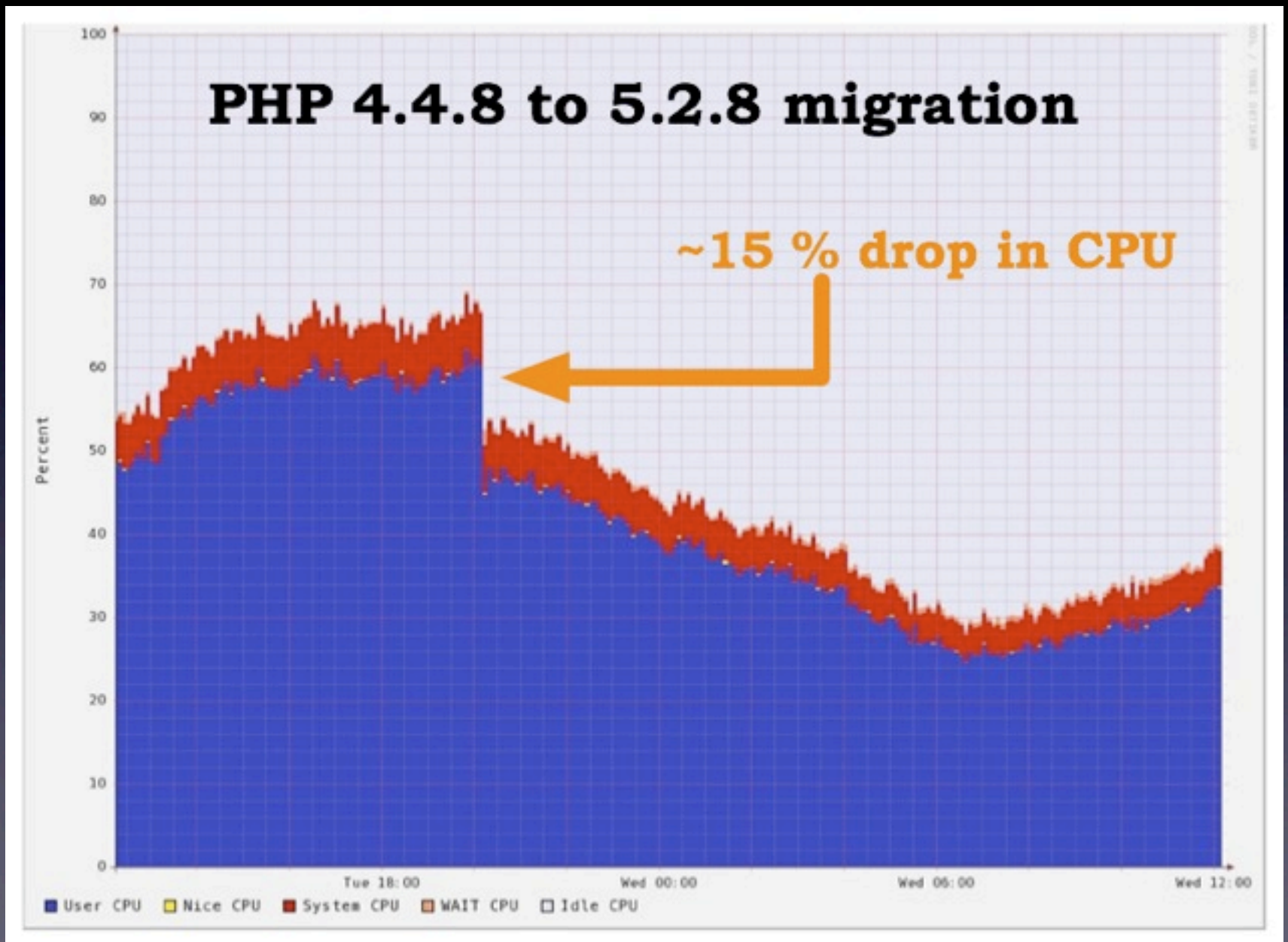
Leaning on compilers

(synthetic PHP benchmarks, not real-world)

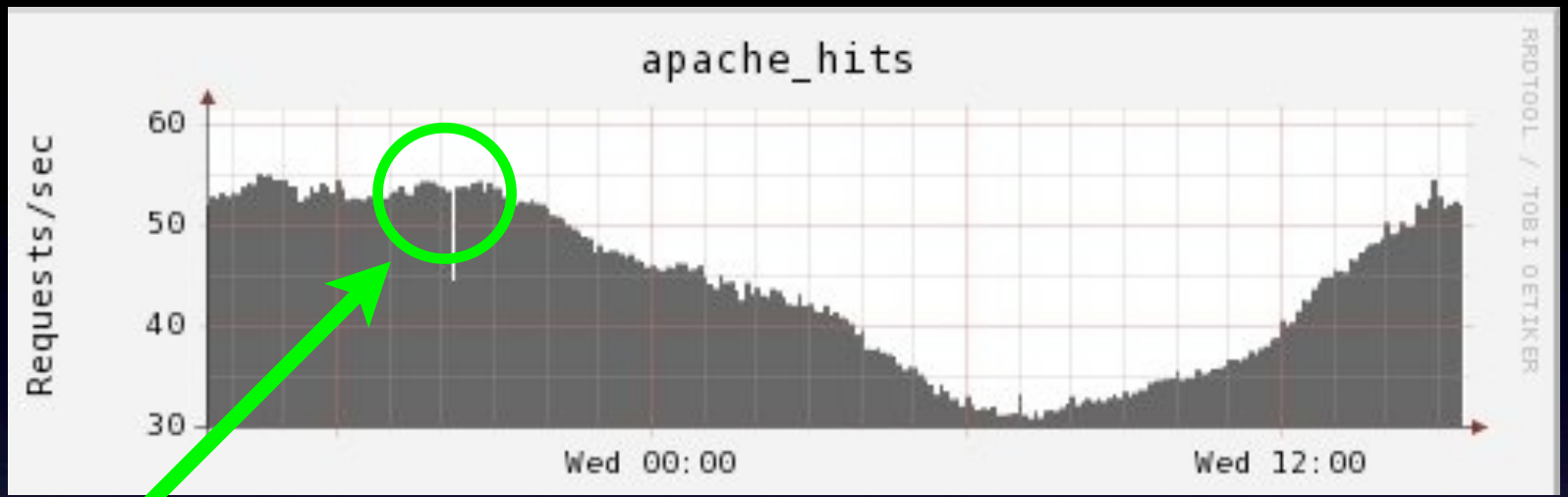


(http://sebastian-bergmann.de/archives/634-PHP-GCC-ICC-Benchmark.html)

PHP *(real-world)*



Can now handle more with less



same taste, less filling

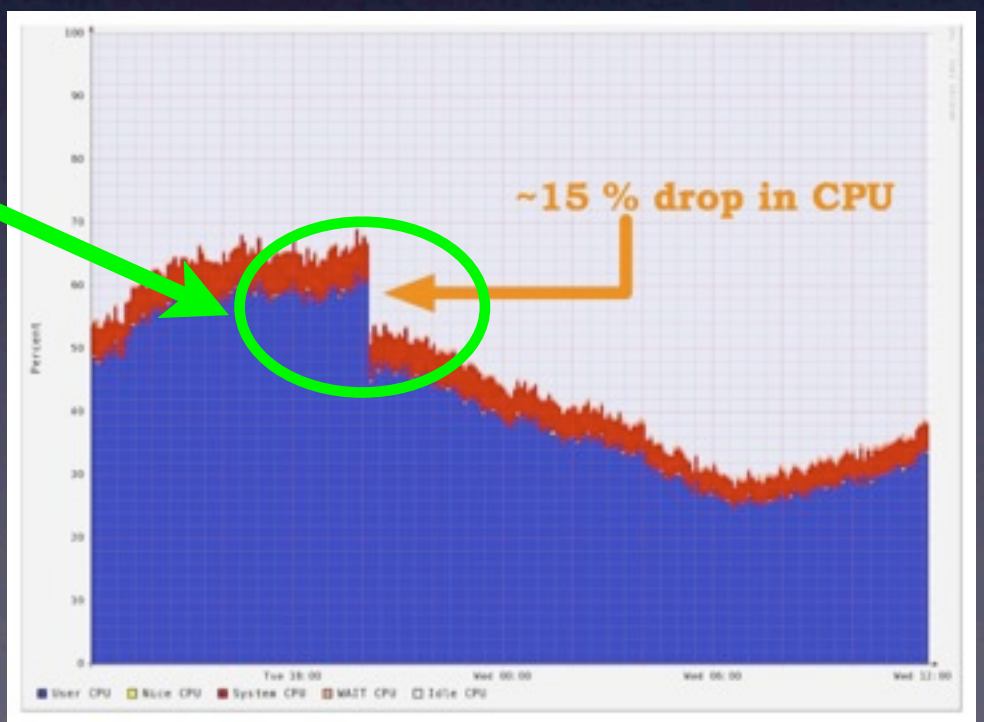


Image Processing

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)

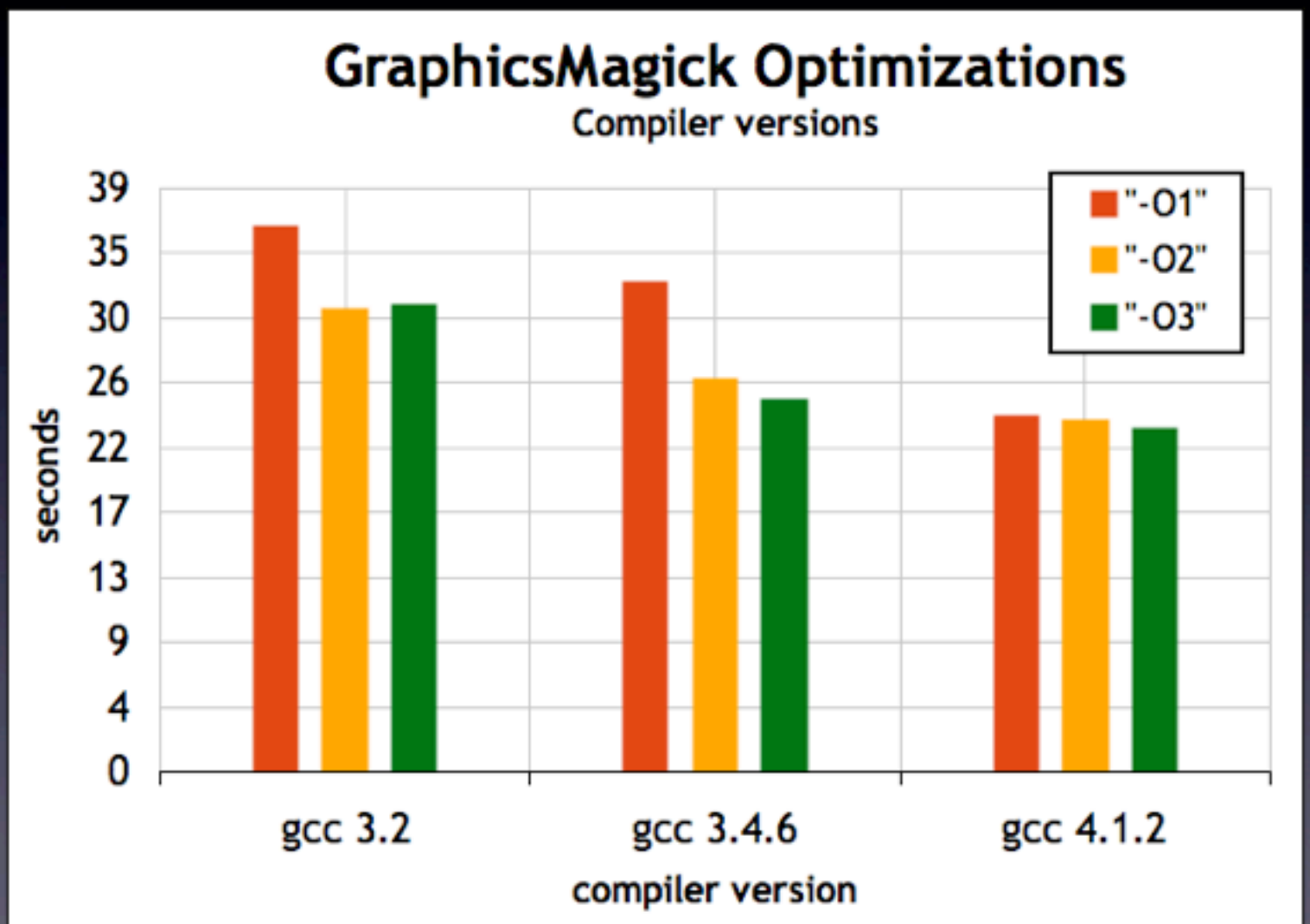
  - Allows parallelization of processing jobs, using multiple cores working on the same image
  - Some algorithms have more parallelization than others

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

Image Processing

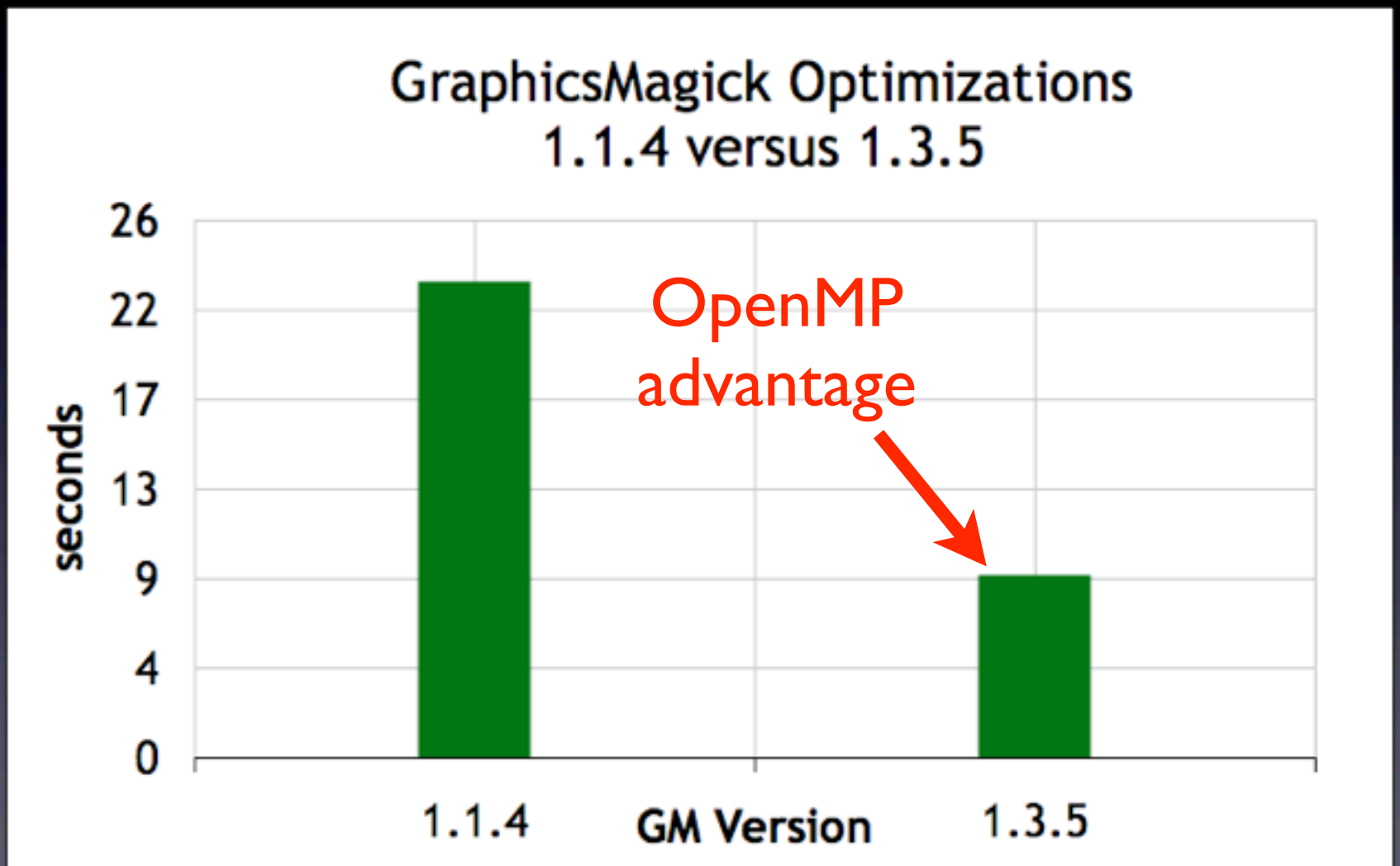
compiler differences



(GM version 1.1.14, non-OpenMP)

Image Processing

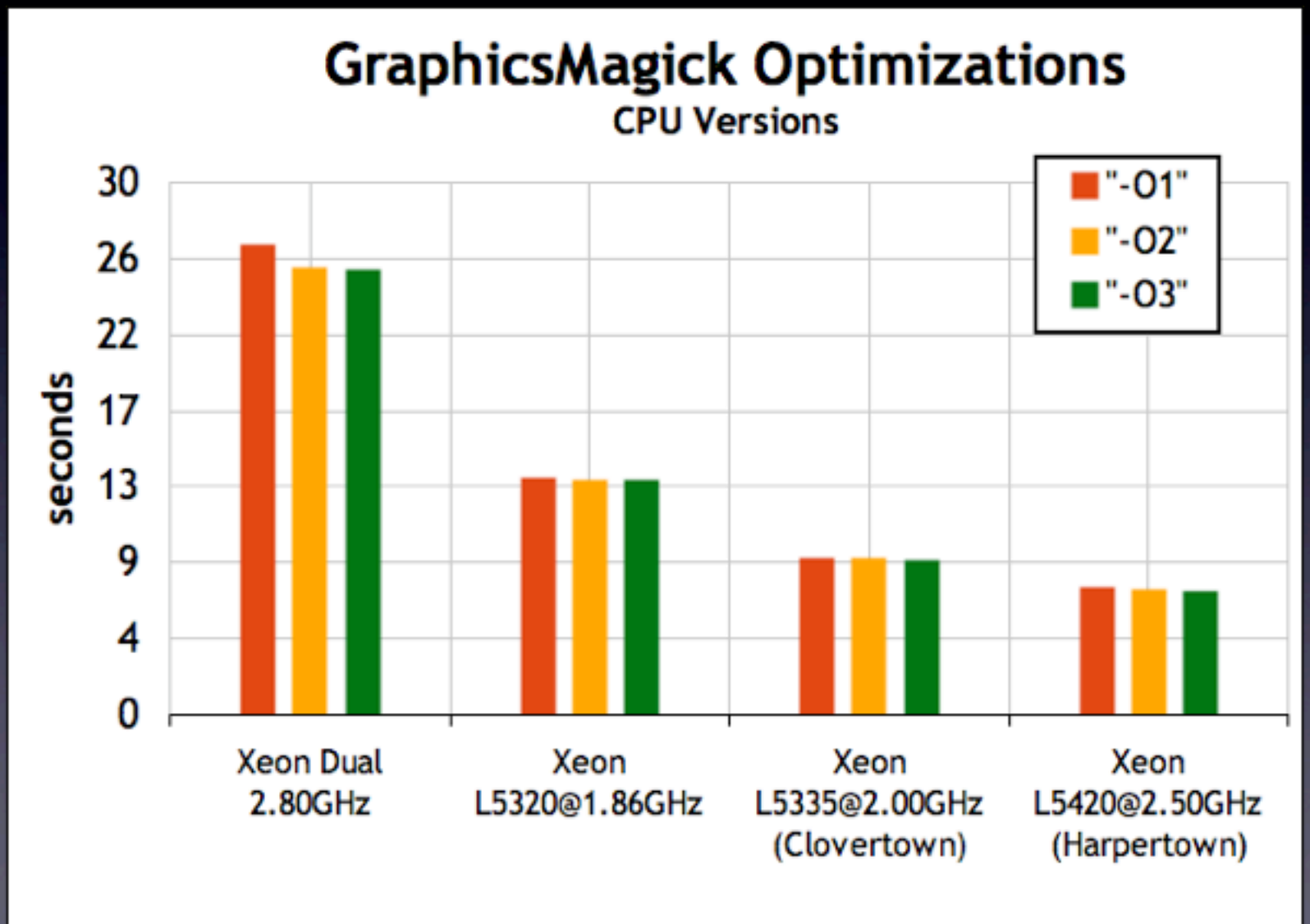
OpenMP differences



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

Image Processing

CPU differences



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” webservers with 18 “new” :

Diagonal Scaling

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

servers	CPUs per server	RAM per server	drives per server	total power (W) @60% peak
67	2	4GB	1x80GB	8763.6
18	8	4GB	1x146GB	2332.8

Diagonal Scaling

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

servers	CPUs per server	RAM per server	drives per server	total power (W) @60% peak
67	2	4GB	1x80GB	8763.6
18	8	4GB	1x146GB	2332.8

~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"

server	photos/min	rack	total power (W) @60% peak
23	1035	23	3008.4
8	1120	8	1036.8

Diagonal Scaling

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

server	photos/min	rack	total power (W) @60% peak
23	1035	23	3008.4
8	1120	8	1036.8

~75% FASTER

15U LESS rack space

65% LESS power

Diagonal Scaling

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

server	photos/min	rack	total power (W) @60% peak
23	1035	23	3008.4
8	1120	8	1036.8

~75% FASTER

15U LESS rack space

65% LESS power

Diagonal Scaling

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

server	photos/min	rack	total power (W) @60% peak
23	1035	23	3008.4
8	1120	8	1036.8

~75% FASTER

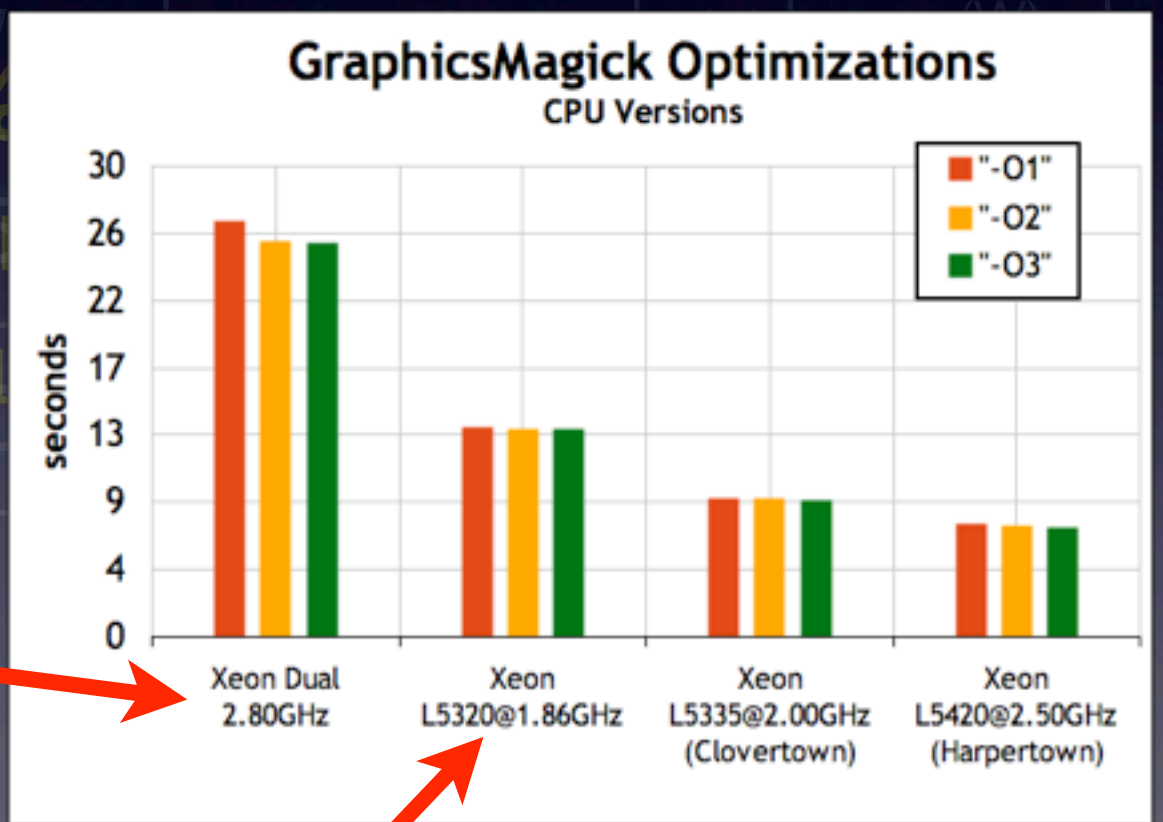
15U LESS rack space

65% LESS power

Diagonal Scaling

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

server	photos/
23	150
8	65%



from this

to this

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:

  http://code.flickr.com/blog/2008/09/26/flickr-engineers-do-it-offline/

Offline Tasks

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

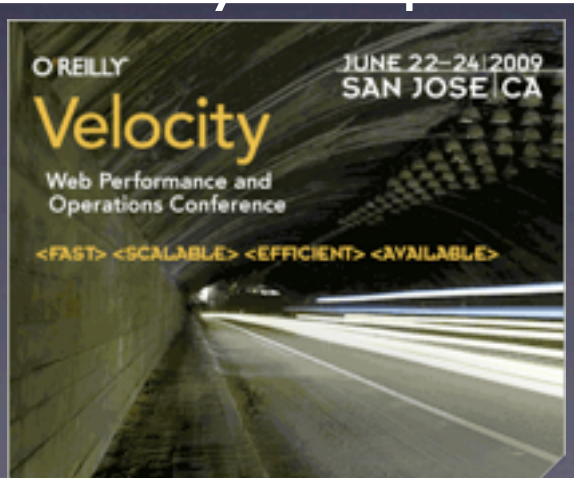
  http://code.flickr.com/blog/2008/09/26/flickr-engineers-do-it-offline/
- See Myles Grant talk about it more here:

Offline Tasks

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

  http://code.flickr.com/blog/2008/09/26/flickr-engineers-do-it-offline/
- See Myles Grant talk about it more here:

  http://en.oreilly.com/velocity2009/public/schedule/detail/7552



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

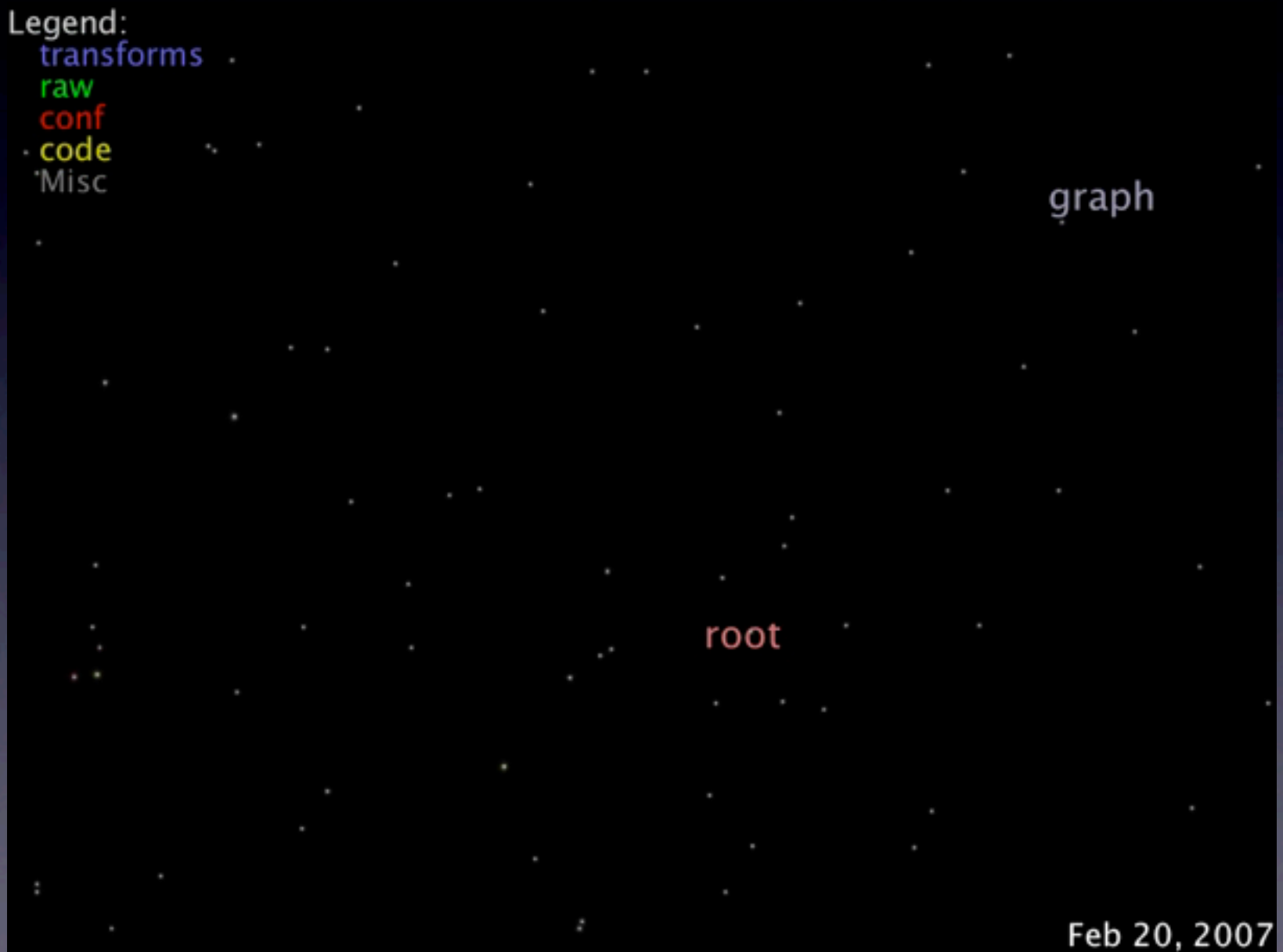
Automated Infrastructure

- 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:
  - Opscode/Chef *(http://opscode.com/)*
  - Puppet *(http://reductivelabs.com/products/puppet/)*
  - System Imager/Configurator *(http://wiki.systemimager.org)*

Conguration Management Codeswarm



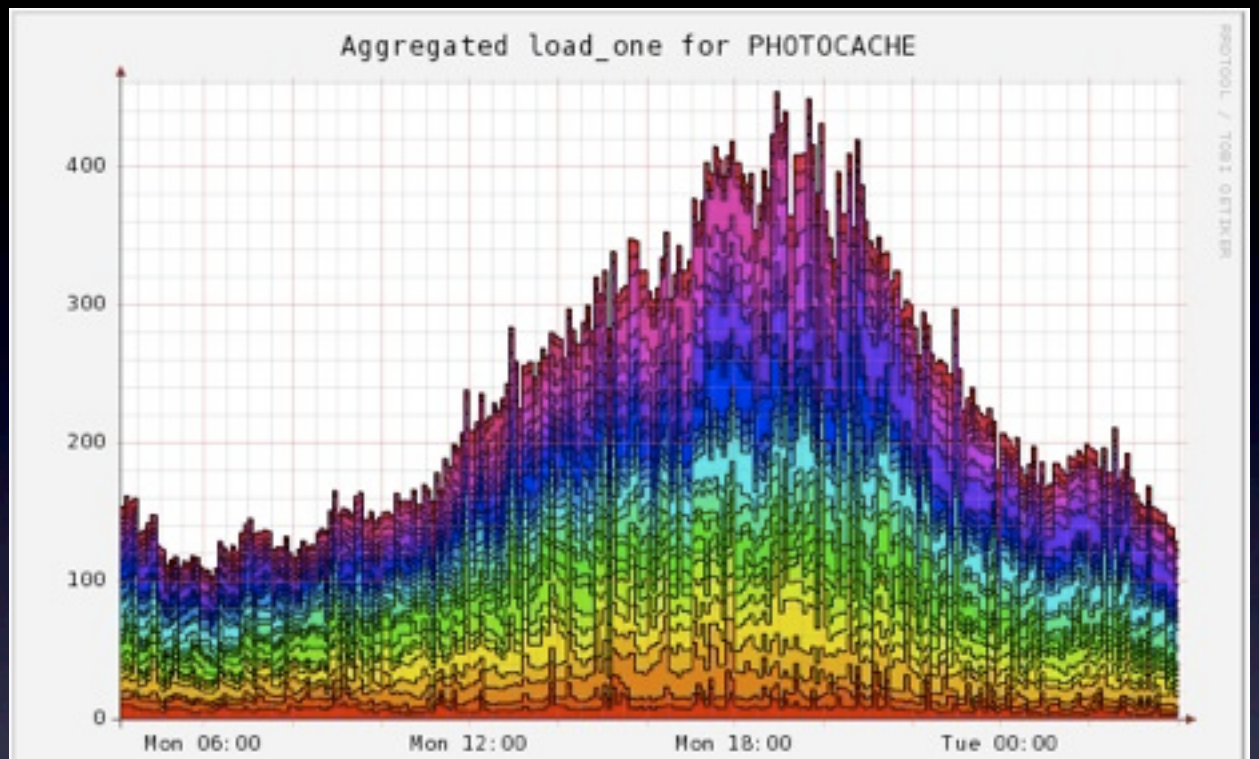
Time

Machine time is cheaper than human time.

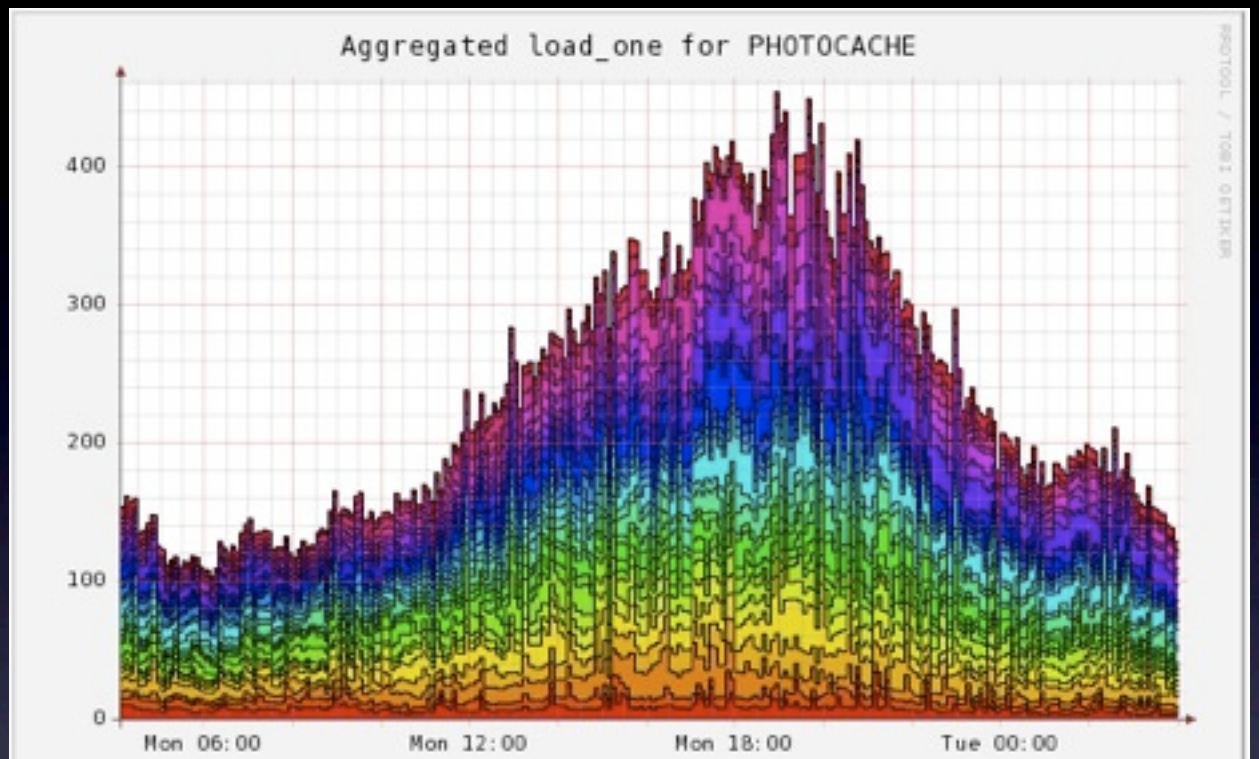
If a failure results in some commands being run to ‘fix’ it, make the machines do it.

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

Aggregate Monitoring



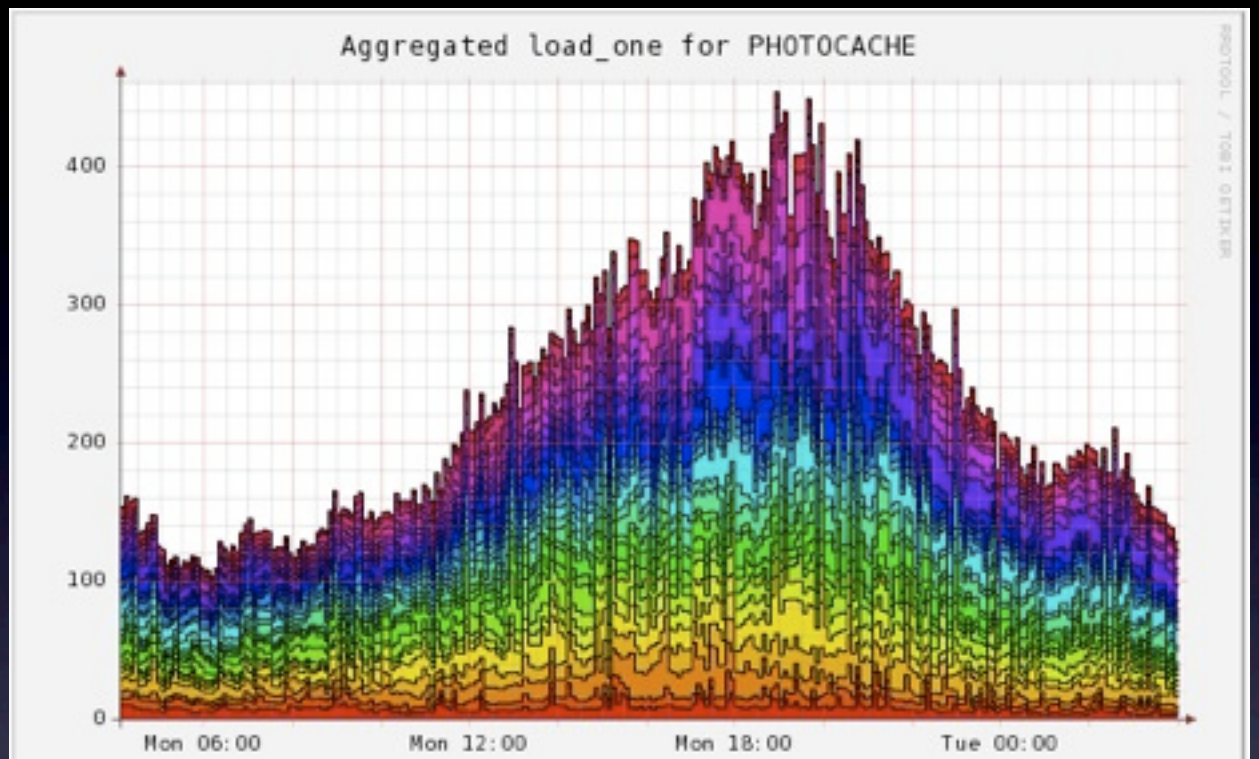
Aggregate Monitoring



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)*

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)*
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

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

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

Communications

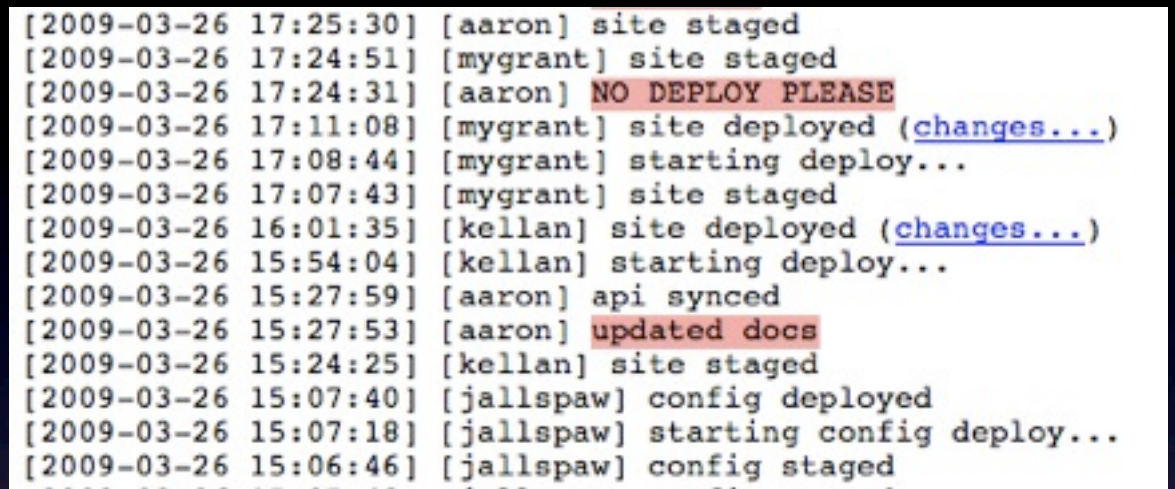
- Internal IRC
  - For ongoing discussions
  - Logged, so “infinite” scrollback

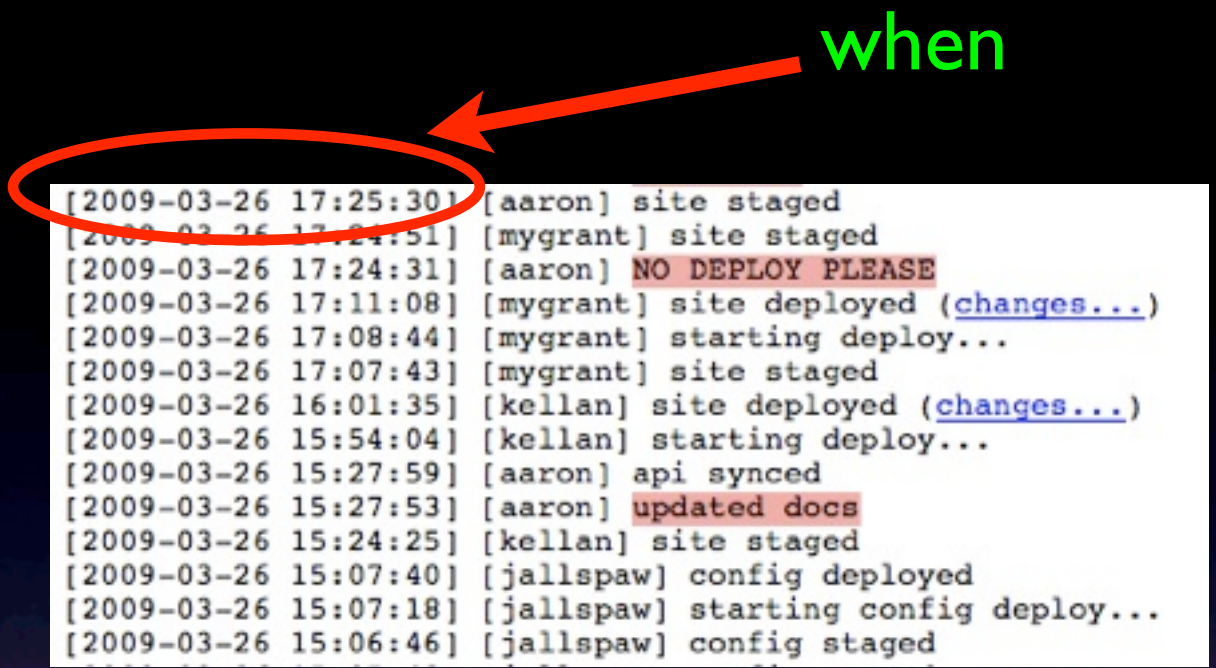
Communications

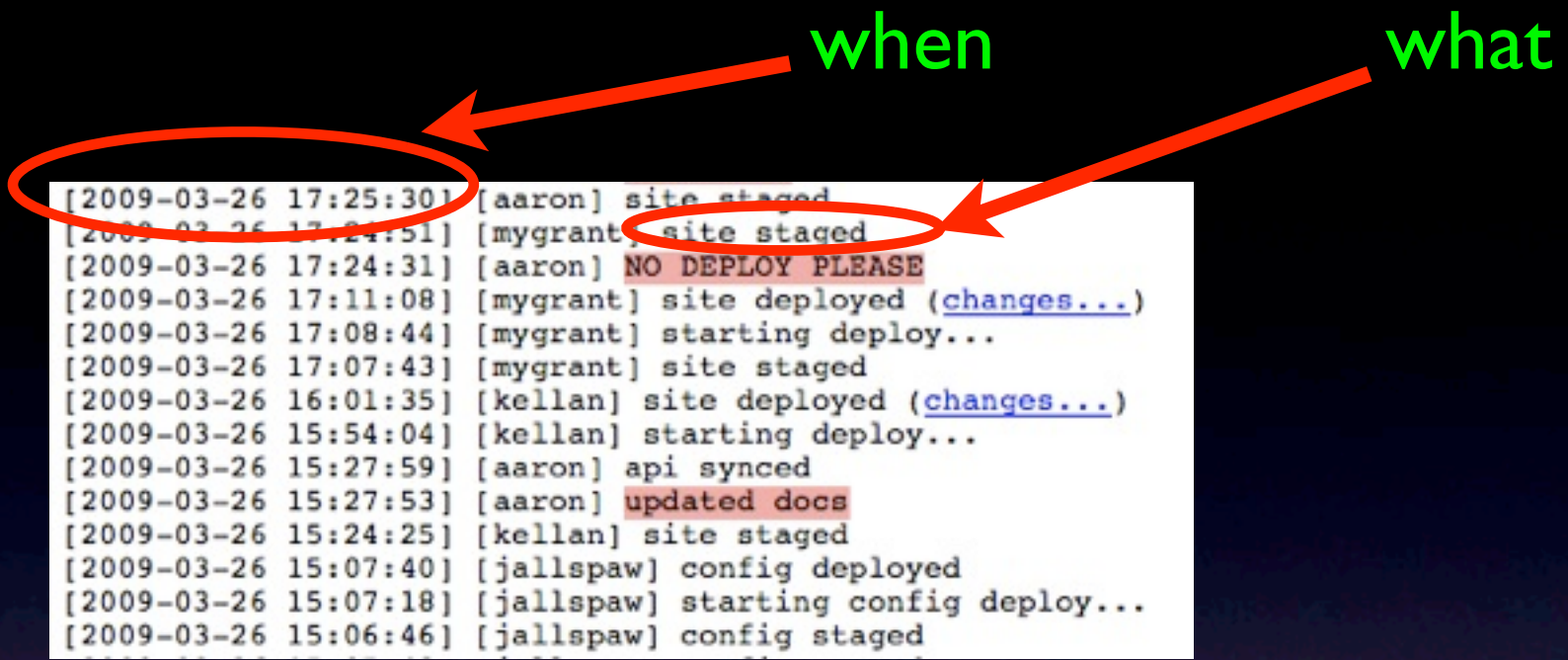
- Internal IRC
  - For ongoing discussions
  - Logged, so “infinite” scrollback
- IM Bot *(built on libyahoo2.sf.net)*
  - For production changes
  - Broadcasts all to all contacts
  - Logged, and injected into IRC
  - IM Status = who is in primary/secondary on-call

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


[2009-03-26 17:25:30] [aaron] site staged
[2009-03-26 17:24:51] [mygrant] site staged
[2009-03-26 17:24:31] [aaron] NO DEPLOY PLEASE
[2009-03-26 17:11:08] [mygrant] site deployed (changes...)
[2009-03-26 17:08:44] [mygrant] starting deploy...
[2009-03-26 17:07:43] [mygrant] site staged
[2009-03-26 16:01:35] [kellan] site deployed (changes...)
[2009-03-26 15:54:04] [kellan] starting deploy...
[2009-03-26 15:27:59] [aaron] api synced
[2009-03-26 15:27:53] [aaron] updated docs
[2009-03-26 15:24:25] [kellan] site staged
[2009-03-26 15:07:40] [jallspaw] config deployed
[2009-03-26 15:07:18] [jallspaw] starting config deploy...
[2009-03-26 15:06:46] [jallspaw] config staged

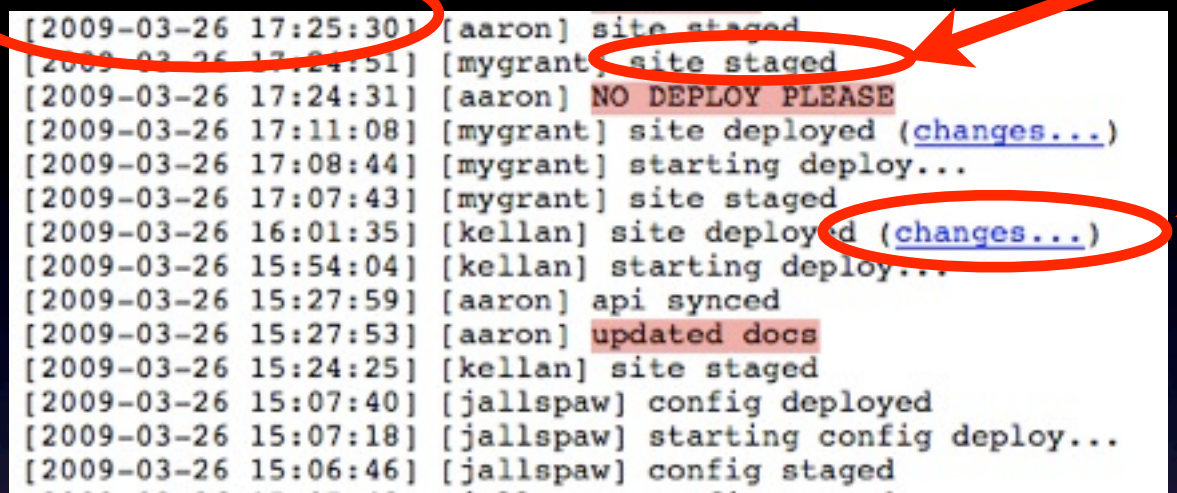

when
[2009-03-26 17:25:30] [aaron] site staged
[2009-03-26 17:24:51] [mygrant] site staged
[2009-03-26 17:24:31] [aaron] NO DEPLOY PLEASE
[2009-03-26 17:11:08] [mygrant] site deployed (changes...)
[2009-03-26 17:08:44] [mygrant] starting deploy...
[2009-03-26 17:07:43] [mygrant] site staged
[2009-03-26 16:01:35] [kellan] site deployed (changes...)
[2009-03-26 15:54:04] [kellan] starting deploy...
[2009-03-26 15:27:59] [aaron] api synced
[2009-03-26 15:27:53] [aaron] updated docs
[2009-03-26 15:24:25] [kellan] site staged
[2009-03-26 15:07:40] [jallspaw] config deployed
[2009-03-26 15:07:18] [jallspaw] starting config deploy...
[2009-03-26 15:06:46] [jallspaw] config staged

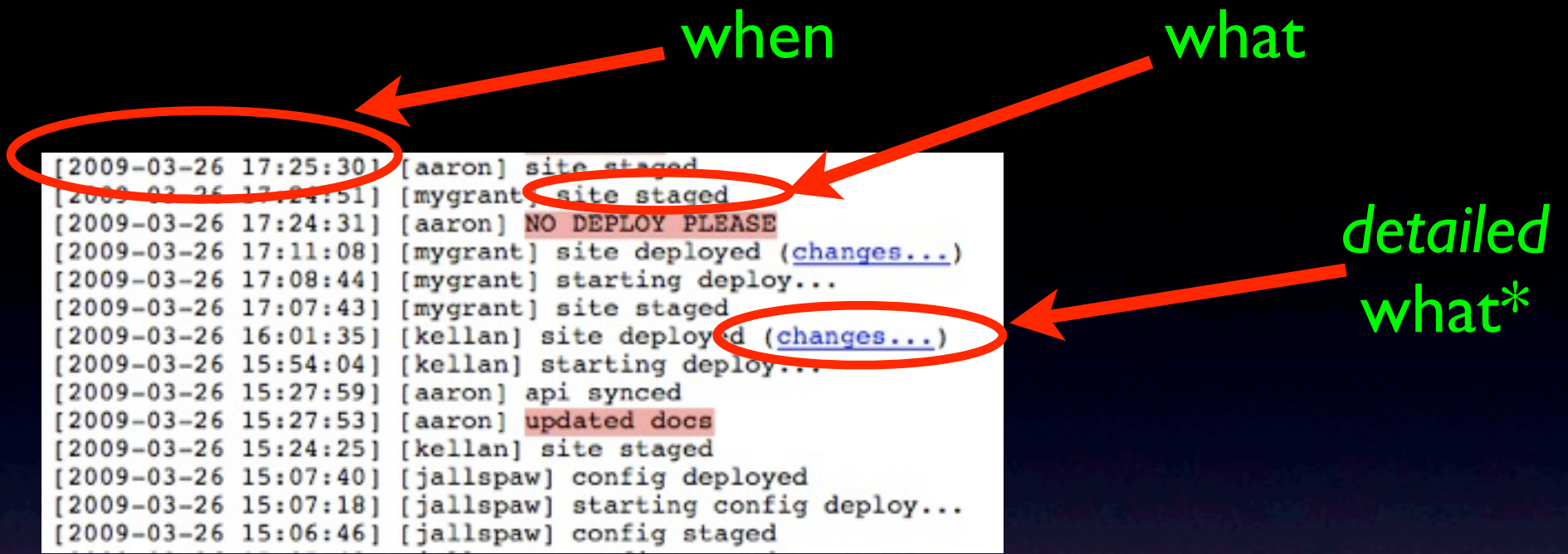

when
what
[2009-03-26 17:25:30] [aaron] site staged
[2009-03-26 17:24:51] [mygrant] site staged
[2009-03-26 17:24:31] [aaron] NO DEPLOY PLEASE
[2009-03-26 17:11:08] [mygrant] site deployed (changes...)
[2009-03-26 17:08:44] [mygrant] starting deploy...
[2009-03-26 17:07:43] [mygrant] site staged
[2009-03-26 16:01:35] [kellan] site deployed (changes...)
[2009-03-26 15:54:04] [kellan] starting deploy...
[2009-03-26 15:27:59] [aaron] api synced
[2009-03-26 15:27:53] [aaron] updated docs
[2009-03-26 15:24:25] [kellan] site staged
[2009-03-26 15:07:40] [jallspaw] config deployed
[2009-03-26 15:07:18] [jallspaw] starting config deploy...
[2009-03-26 15:06:46] [jallspaw] config staged

when

what

detailed what*

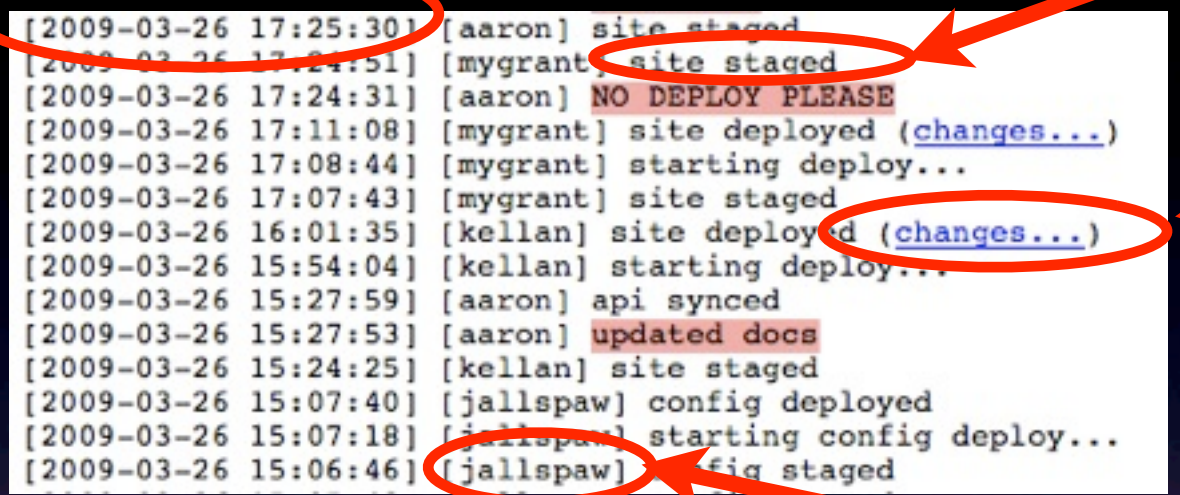




*also points to what commands should be used to back out the changes

when

what



detailed
what*

who

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

when

what

detailed what*

who

```
[2009-03-26 17:25:30] [aaron] site staged
[2009-03-26 17:24:51] [mygrant] site staged
[2009-03-26 17:24:31] [aaron] NO DEPLOY PLEASE
[2009-03-26 17:11:08] [mygrant] site deployed (changes...)
[2009-03-26 17:08:44] [mygrant] starting deploy...
[2009-03-26 17:07:43] [mygrant] site staged
[2009-03-26 16:01:35] [kellan] site deployed (changes...)
[2009-03-26 15:54:04] [kellan] starting deploy...
[2009-03-26 15:27:59] [aaron] api synced
[2009-03-26 15:27:53] [aaron] updated docs
[2009-03-26 15:24:25] [kellan] site staged
[2009-03-26 15:07:40] [jallspaw] config deployed
[2009-03-26 15:07:18] [jallspaw] starting config deploy...
[2009-03-26 15:06:46] [jallspaw] config staged
```

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

when

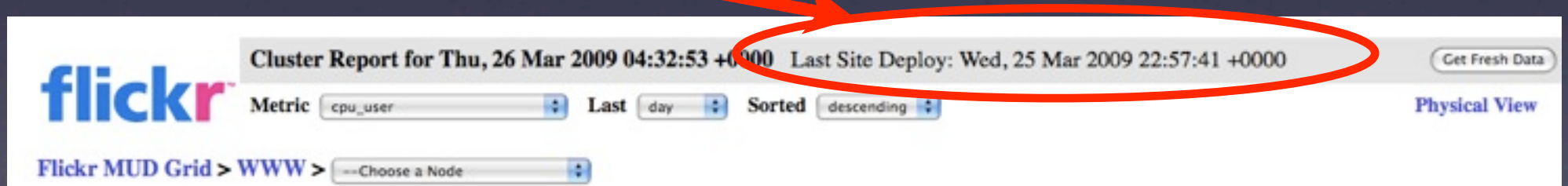
what

detailed what*

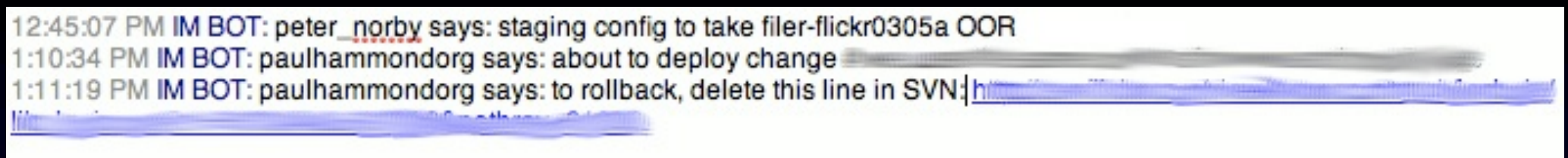
who

```
[2009-03-26 17:25:30] [aaron] site staged
[2009-03-26 17:24:51] [mygrant] site staged
[2009-03-26 17:24:31] [aaron] NO DEPLOY PLEASE
[2009-03-26 17:11:08] [mygrant] site deployed (changes...)
[2009-03-26 17:08:44] [mygrant] starting deploy...
[2009-03-26 17:07:43] [mygrant] site staged
[2009-03-26 16:01:35] [kellan] site deployed (changes...)
[2009-03-26 15:54:04] [kellan] starting deploy...
[2009-03-26 15:27:59] [aaron] api synced
[2009-03-26 15:27:53] [aaron] updated docs
[2009-03-26 15:24:25] [kellan] site staged
[2009-03-26 15:07:40] [jallspaw] config deployed
[2009-03-26 15:07:18] [jallspaw] starting config deploy...
[2009-03-26 15:06:46] [jallspaw] config staged
```

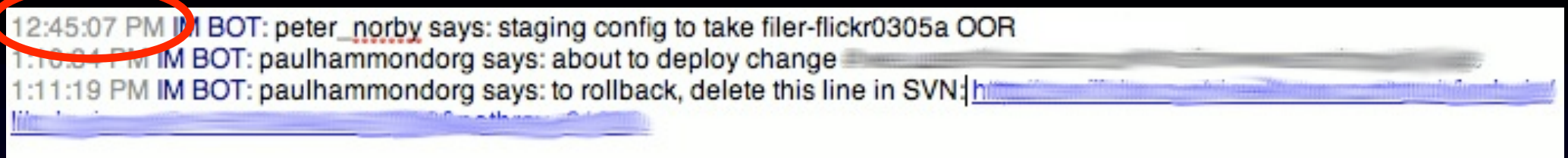
time of last deploy at top of ganglia



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


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:

IM Bot (timestamps help correlation)



IM Bot (timestamps help correlation)

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

filer-flickr0305a Find!

who: anyone relevancy sort 3 lines scope

after YYYY-MM-DD: 1969-12-31 before YYYY-MM-DD: 2046-12-31

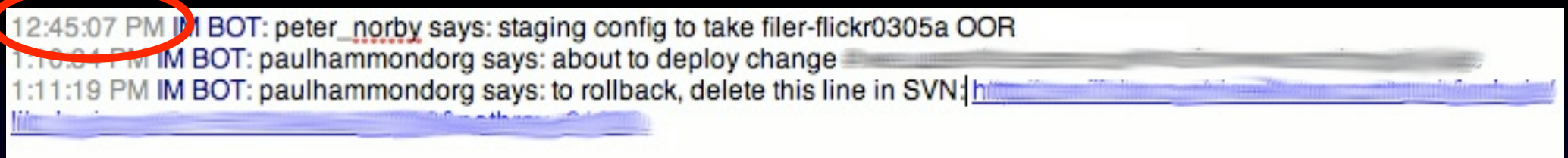
kevbob (1) peter_norby (1)

TOTAL HITS: 2

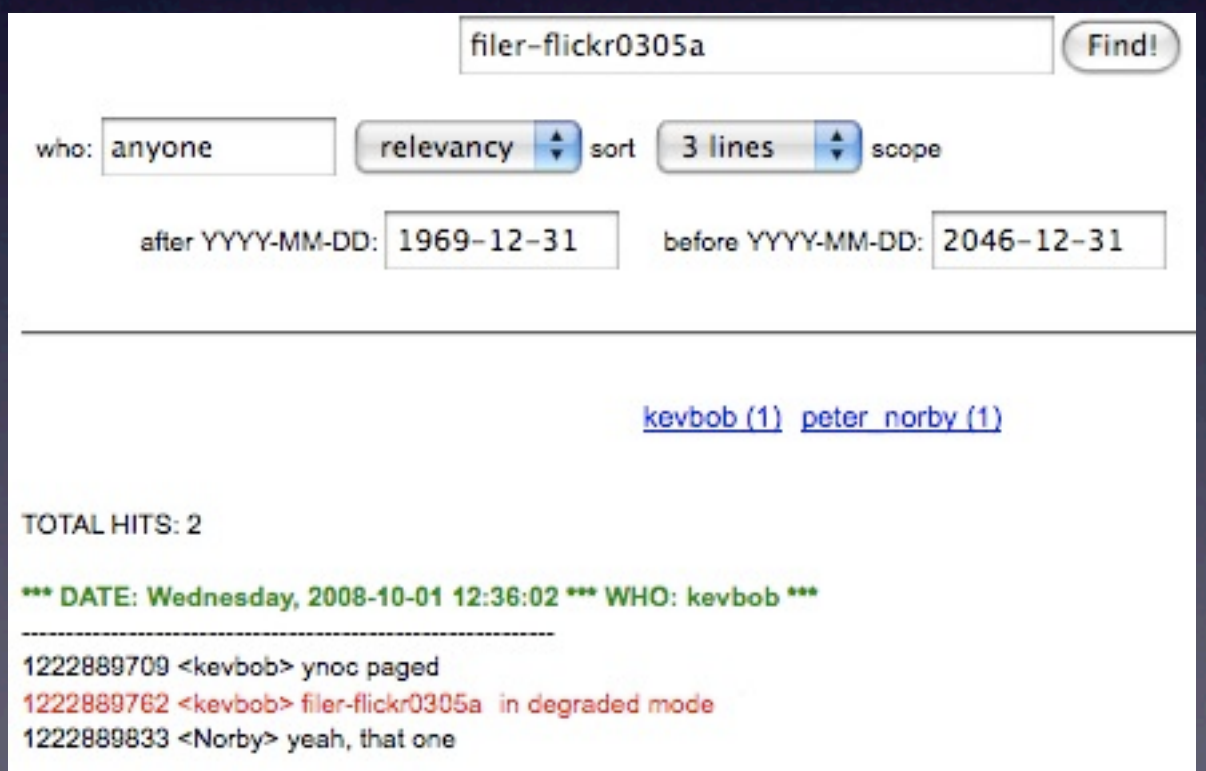
*** DATE: Wednesday, 2008-10-01 12:36:02 *** WHO: kevbob ***

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

IM Bot (timestamps help correlation)



all IRC, IM bot into searchable history



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:

http://jprall.vox.com/library/post/85-operations-rules-to-live-by.html

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/191063346/
http://www.flickr.com/photos/telstar/2861103147/
http://www.flickr.com/photos/norby/2309046043/
http://www.flickr.com/photos/allysonk/201008992/