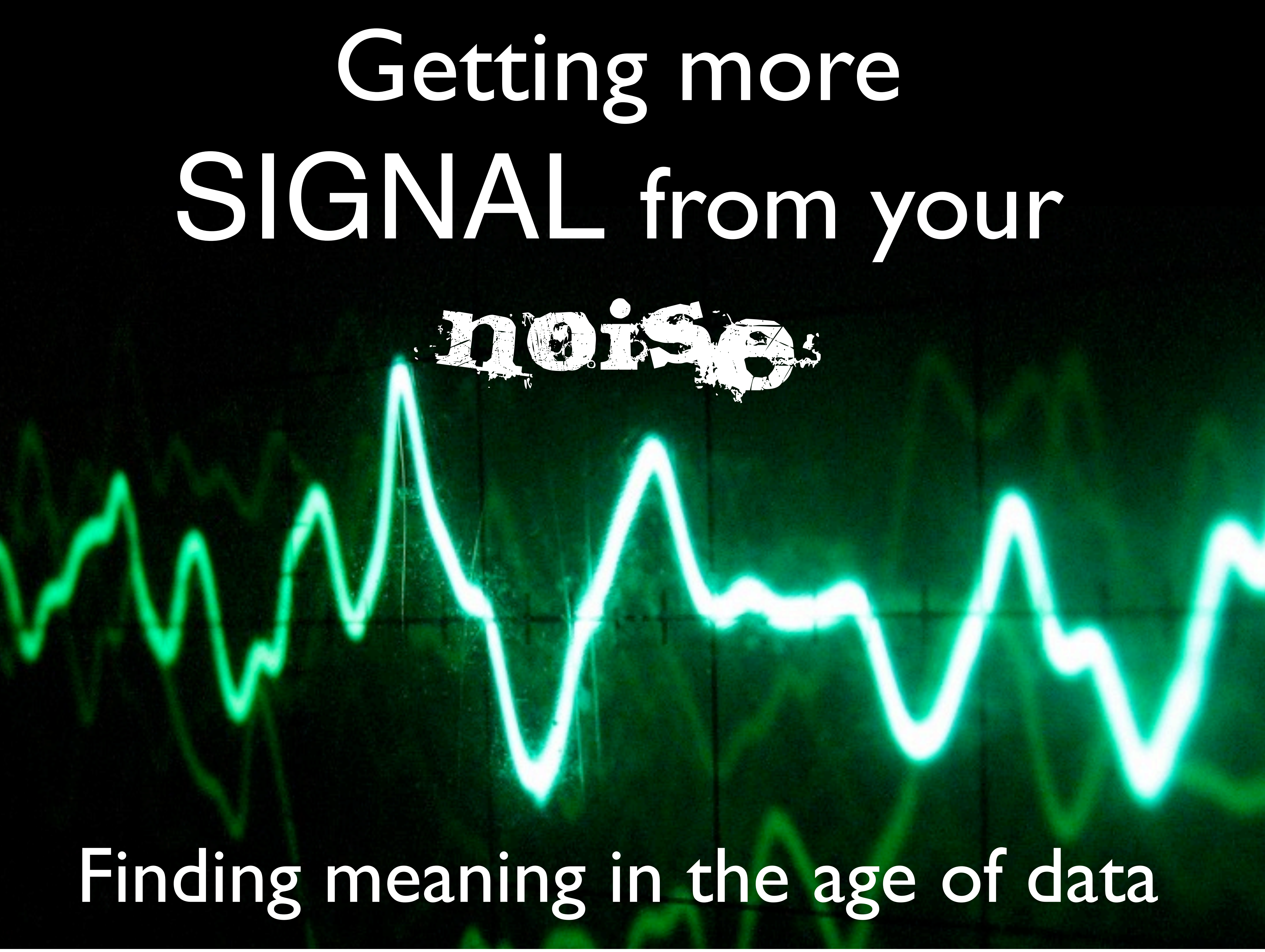
Getting more SIGNAL from your noise



Finding meaning in the age of data

Why are you talking?

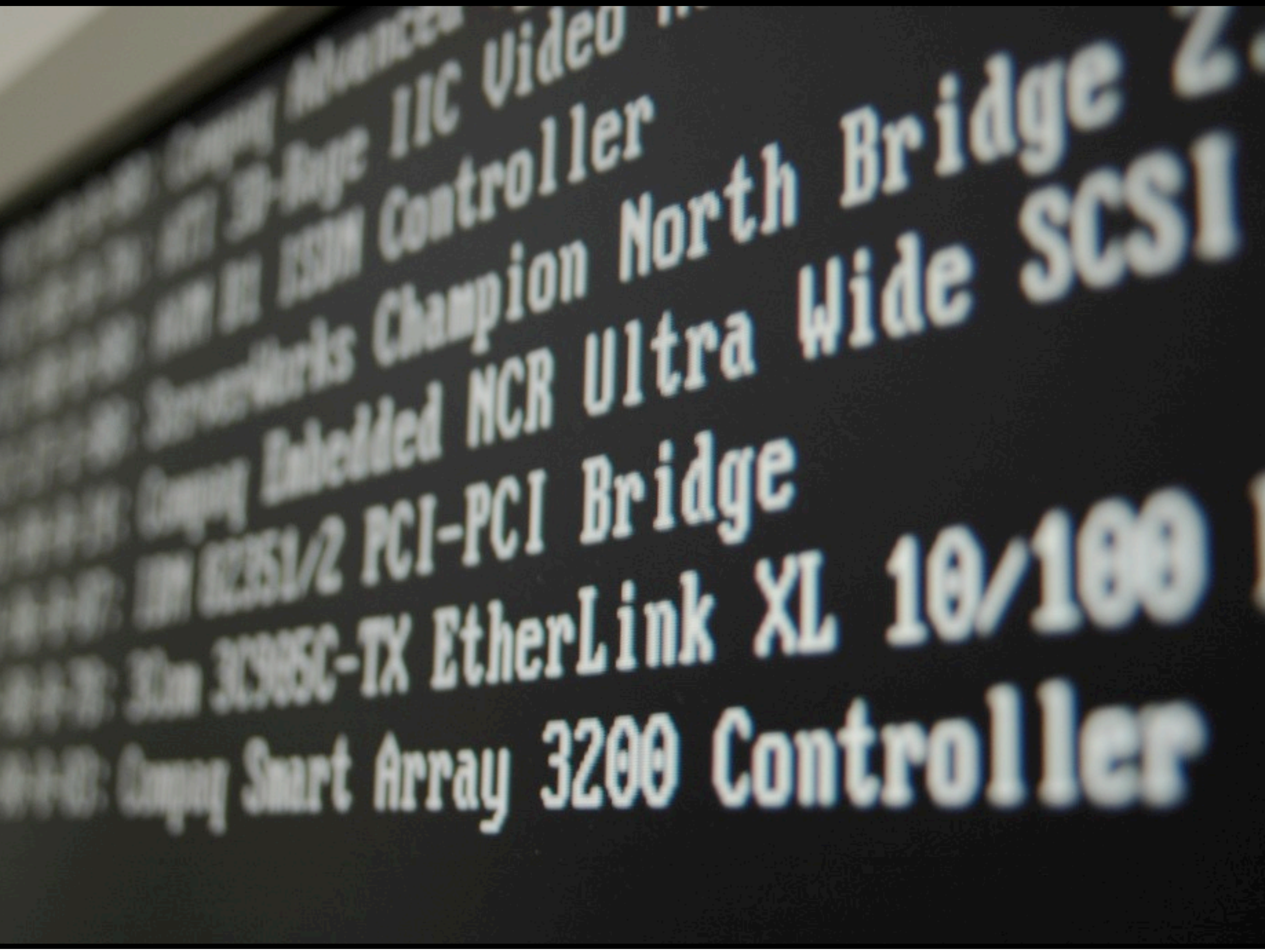
Data → Decisions

Have been turning data into decisions for my whole career
From UCLA's retail ->
3G development (ns timing between devices matters) ->
Real-time robotics (motion streaming) to a
lot of years running lots and lots and lots of servers (real + virtual)

Still deeply a student in this world but I spend a lot of time reading/testing/thinking about it.

NOISE

Define your terms
What do you mean by noise



```
Compaq Advanced
ATI 3D-Rage IIC Video
ISDN Controller
ServerWorks Champion North Bridge 2.
Compaq Embedded NCR Ultra Wide SCSI
IBM 82351/2 PCI-PCI Bridge
3Com 3C905C-TX EtherLink XL 10/100
Compaq Smart Array 3200 Controller
```

- Modern systems (can) generate a ton
- Server stats
    - (CPU, memory, network, (links and flows), each FS, disk and tier of disk, RAID, SSD, etc, swap, kernel counters, power utilization)
    - System stats (OS/Kernel/Packages/etc)
- Network stats (between devices, in devices)
- Daemon stats + Logging (everything running in init.d can generate data)
    - Trivial to be collecting more from (say) mysqld than the server that mysqld is running on

- size and complexity of systems are exploding
    - multiple servers
    - multiple providers/clouds
    - multiple tiers (cdn, memcached, SQL, NoSQL, web server, app server, ...)

Not JUST mysql but MMM replicated, DRBD, Pacemaker, ...

External data
    Health checks/Performance monitoring

Your business's data
    Sales numbers
    Hardware costs
    Twitter chatter

Lots to measure
Lots that arguably should be measured
Lots more things have an API of some kind that we can rip #'s from

> “In a data deluge–era sensing system, the number and resolution of the sensors grow to the point that the performance bottleneck moves to the sensor data processing, communication, or storage subsystem.”

Dr. Richard G. Baraniuk

‘Science Magazine’ 02/2011

“In a data deluge-era sensing system, the number and resolution of the sensors grow to the point that the performance bottleneck moves to the sensor data processing, communication, or storage subsystem.”

It used to be hard to measure things.
Now it’s TOO easy. It’s hard to get manage it all and also to get any true knowledge from it.

SIGNAL

What is Signal?

- Actionable Data
- The kind of information you can ask questions about and make decisions based on

INFORMATION

+

ANALYSIS

+

PRESENTATION

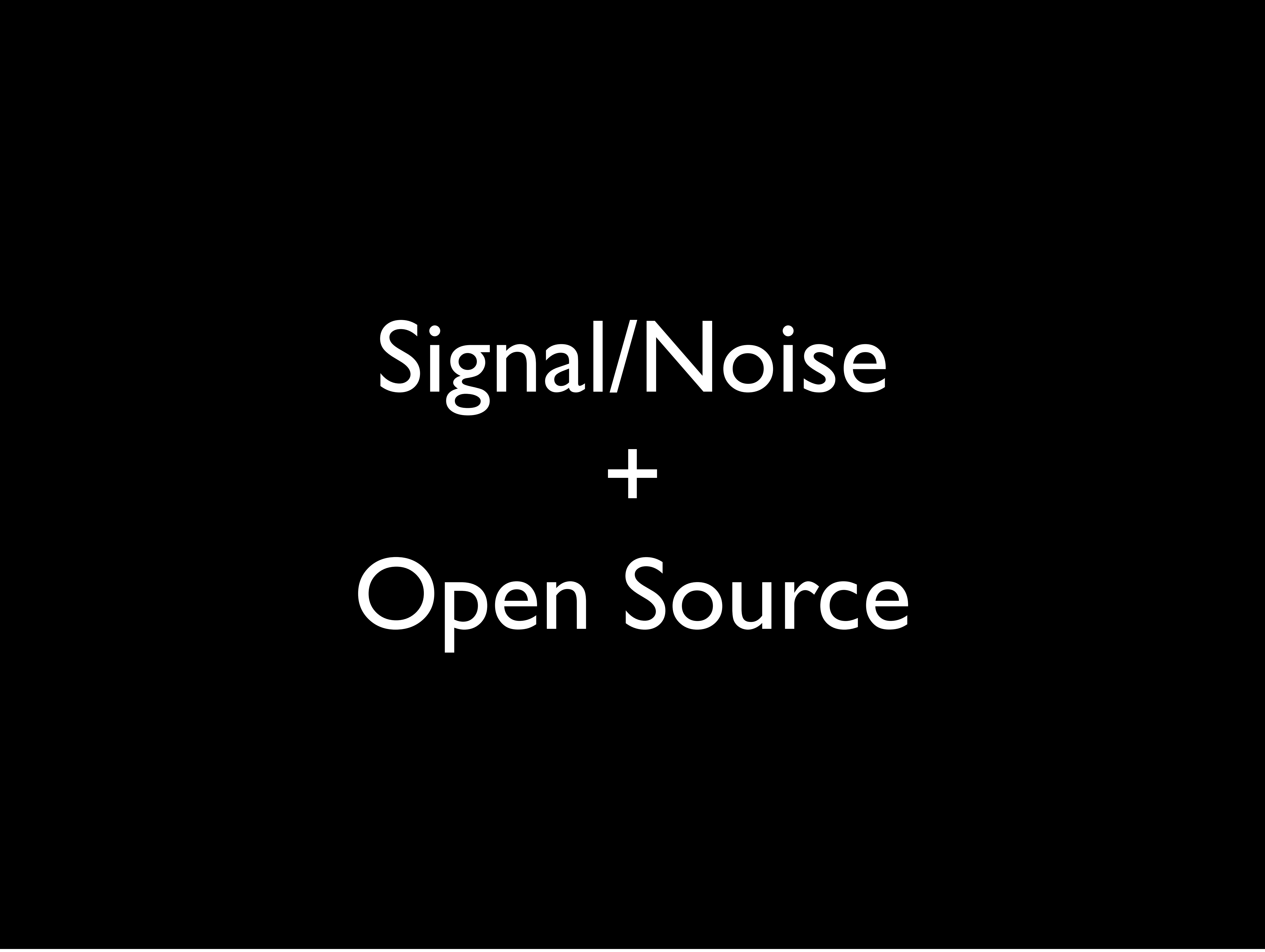
> At StumbleUpon, we have found this system tremendously helpful to:
>
> Get real-time state information about our infrastructure and services.
>
> Understand outages or how complex systems interact together.
>
> Measure SLAs (availability, latency, etc.)
>
> Tune our applications and databases for maximum performance.
>
> Do capacity planning.

Benoit "tsuna" Sigoure

(OpenTSDB Creator)

Signal looks a lot more like
"your customer-facing latency is being caused by the database table cache being too small"
"you need to buy more servers by the 12th"
“switching mail daemons cut our disk bandwidth utilization by 100%”
Than
"the load average is 28 on host 7"

We have so many cases where minutes matter.
Sometimes you have 5 minutes to prevent an outage when something goes critical
When you have an outage already, knowing “it’s the BLAH” immediately means you can skip the “detect”, “diagnose” stages and go right to “FIX IT!”



Signal/Noise + Open Source

For a long time Open Source didn't have much help for us
Perl, Gnuplot, RRD
There has been an explosion for both "Big" and "not quite Big" data
- Techniques, algorithms, and open source software

Even an ORA conference (Strata)
Lots of startups and established businesses will sell you software and services to handle these needs (probably in many cases built around the open source software we’ll be discussing today)

Goals

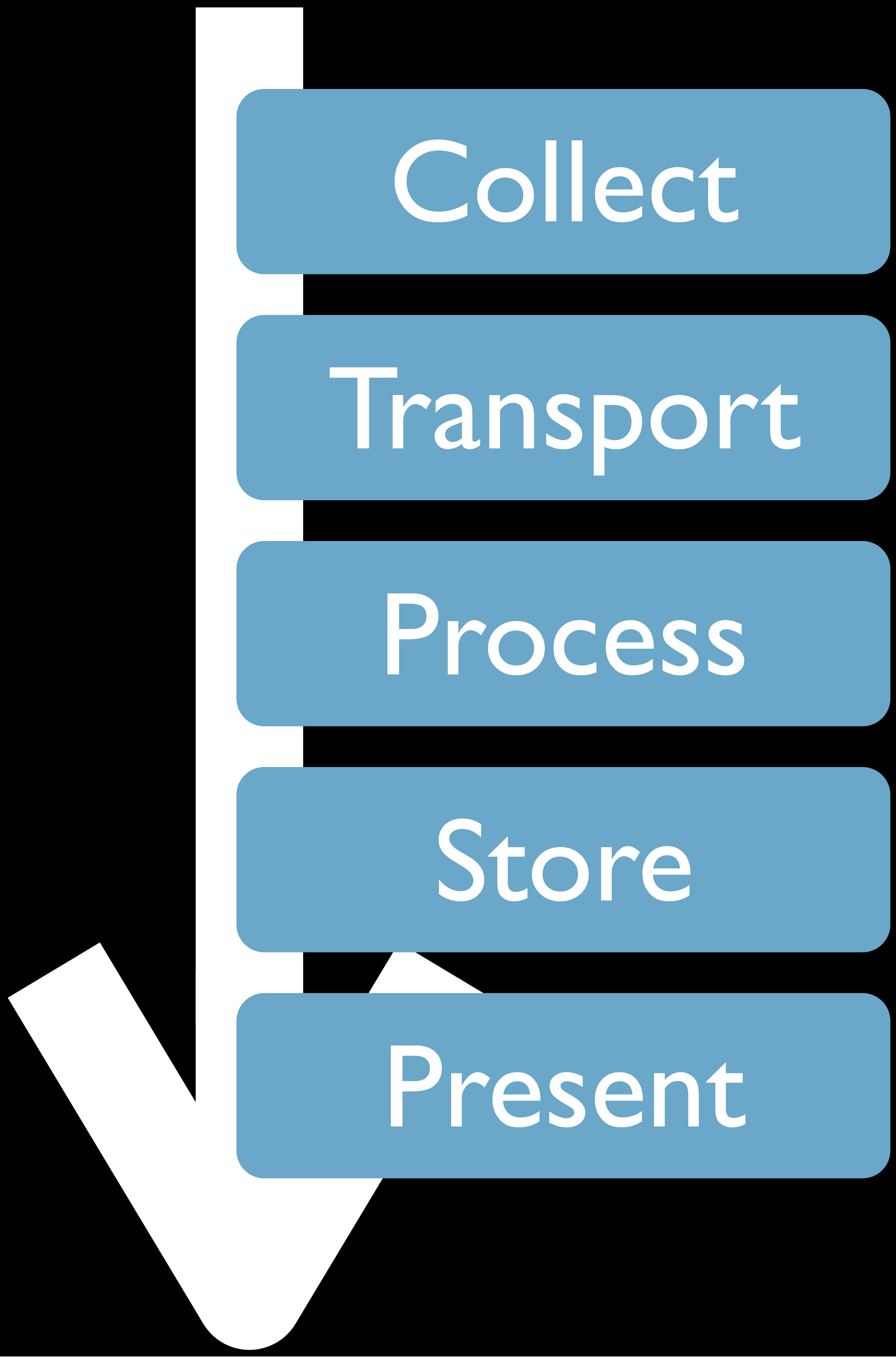
Talk Goals

- framework for evaluating/understanding the solutions out there
  (lots of mix and match possible once you understand the tiers)
- introduction to an underpromoted favorite, graphite

Desirable features

- cheap to add monitorables for all users/applications (developers especially)
- free form queries are possible (historical)
- dashboards (query real time data and map back into quick TTD/TTR views)

Avoid the “now we have 2 problems” scenario
Complexity of monitoring infra ~= infra itself



Collect

Transport

Process

Store

Present

It’s useful to think of the available software as fitting in these roles

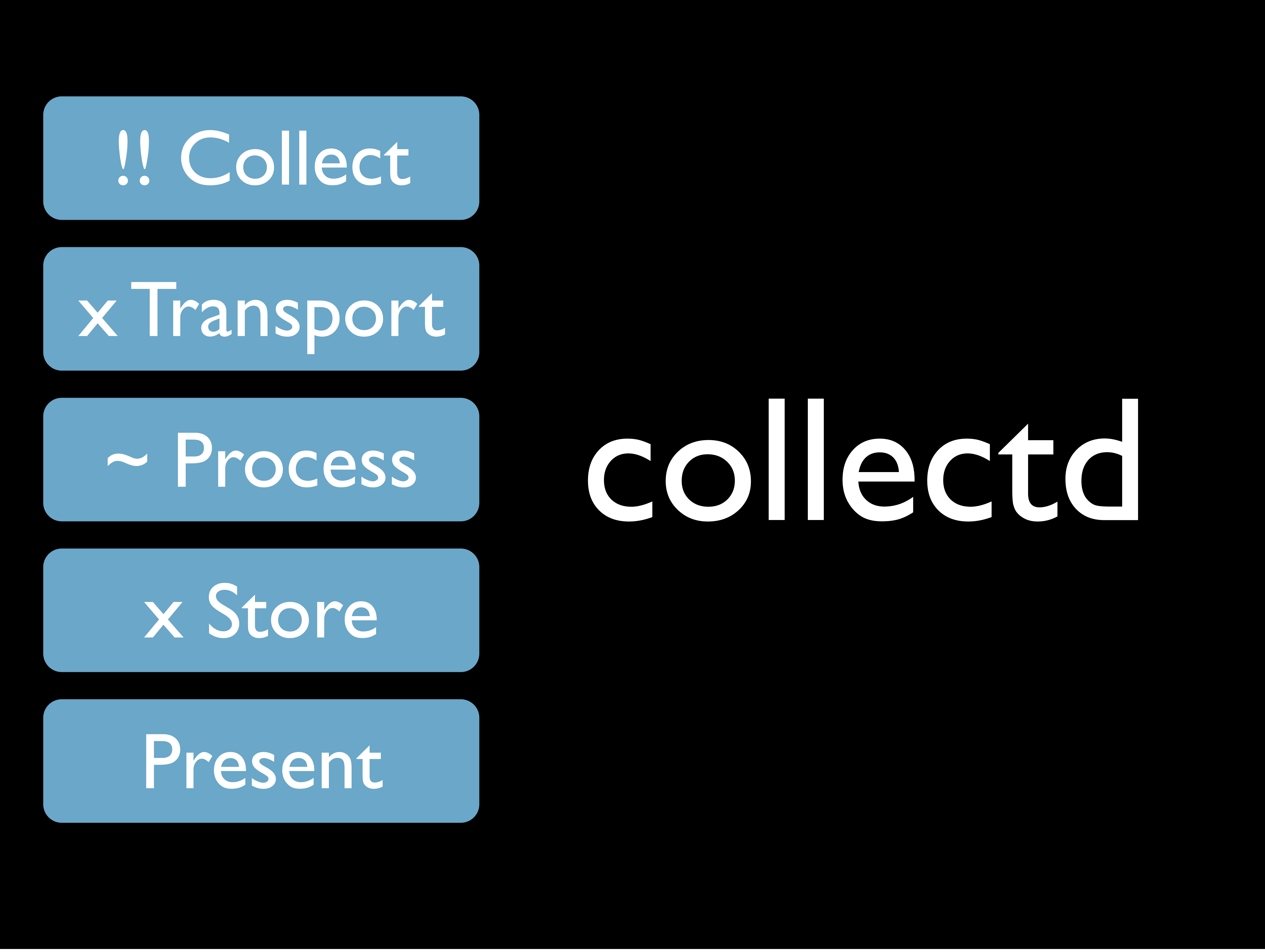
A given project or framework will provide a subset of them
Some specialize in just one

Overall status of the world
You better bring some dev to your ops here
Still need to have some code going
Layered model is a good tool
 For a given tool
 What functions does it provide
 How do they match to what I want to have "be signal"
 What interfaces can I plug into at the borders where it stops being functional?
 Most tools in this space (bless them) have language-independent interfaces

On the ‘horizontal’ axis you can look at the kind of data it’s capable of dealing with:
Time-Series?
Numeric, Text, or Events?

Project Spotlight

Tons of projects worth knowing about
I will put a more detailed catalog on my site
A few that I think more people should know about than do



!! Collect

x Transport

~ Process

x Store

Present

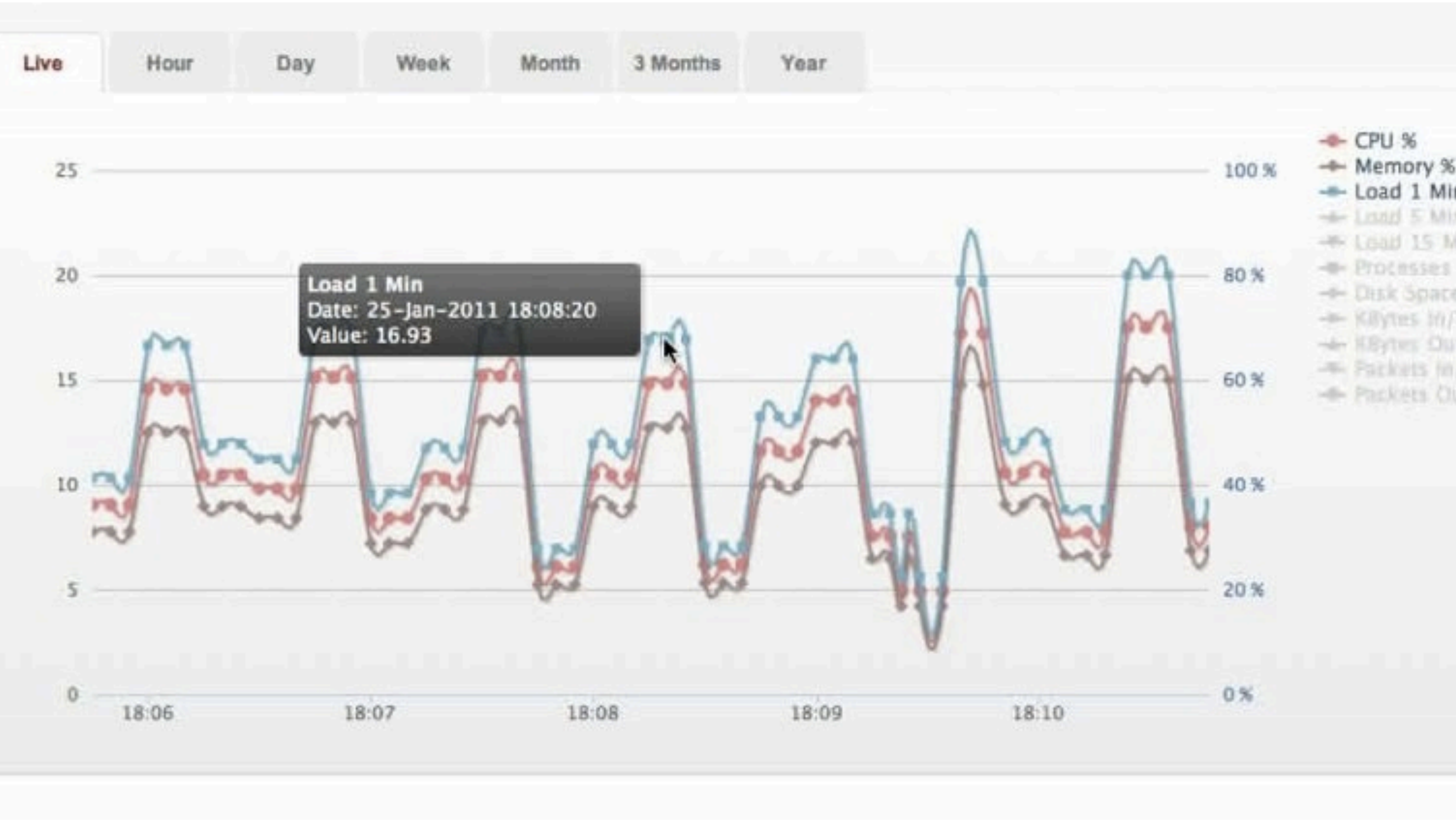
collectd

Collectd is awesome, and it kills in the ‘collect’ role (as you might guess from the name)

High performance swiss army knife
- Polls systems fast, lightweight
- Flexible plugin arch, 90 out there already
- Extract huge amounts of data from (mysql, apache, linux servers,
- Embedded perl and python interpreters for low overhead plugins

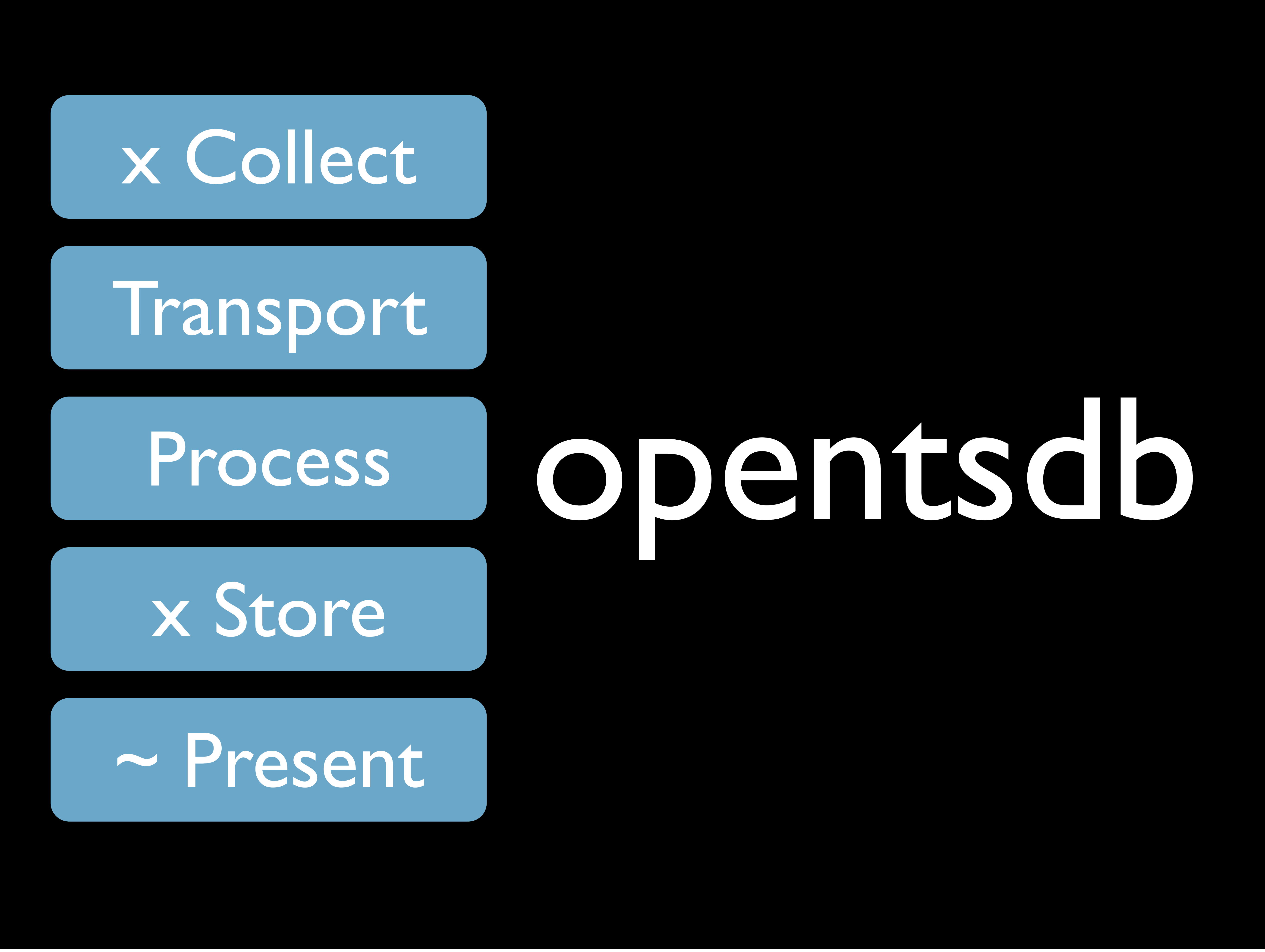
Networking amazing
- “data routing” -- send this via multicast, this via unicast, etc.

You can get the ‘present’ layer if you include ‘visage’ (http://auxesis.github.com/visage/)



Load 1 Min
Date: 25-Jan-2011 18:08:20
Value: 16.93

We use it for our (ve) stats feature
Runs on each physical server and collects data about all running virtual servers
Can send to multiple ‘collector’ servers that use RRD’s
No i/o load on any local boxes



x Collect

Transport

Process

x Store

~ Present

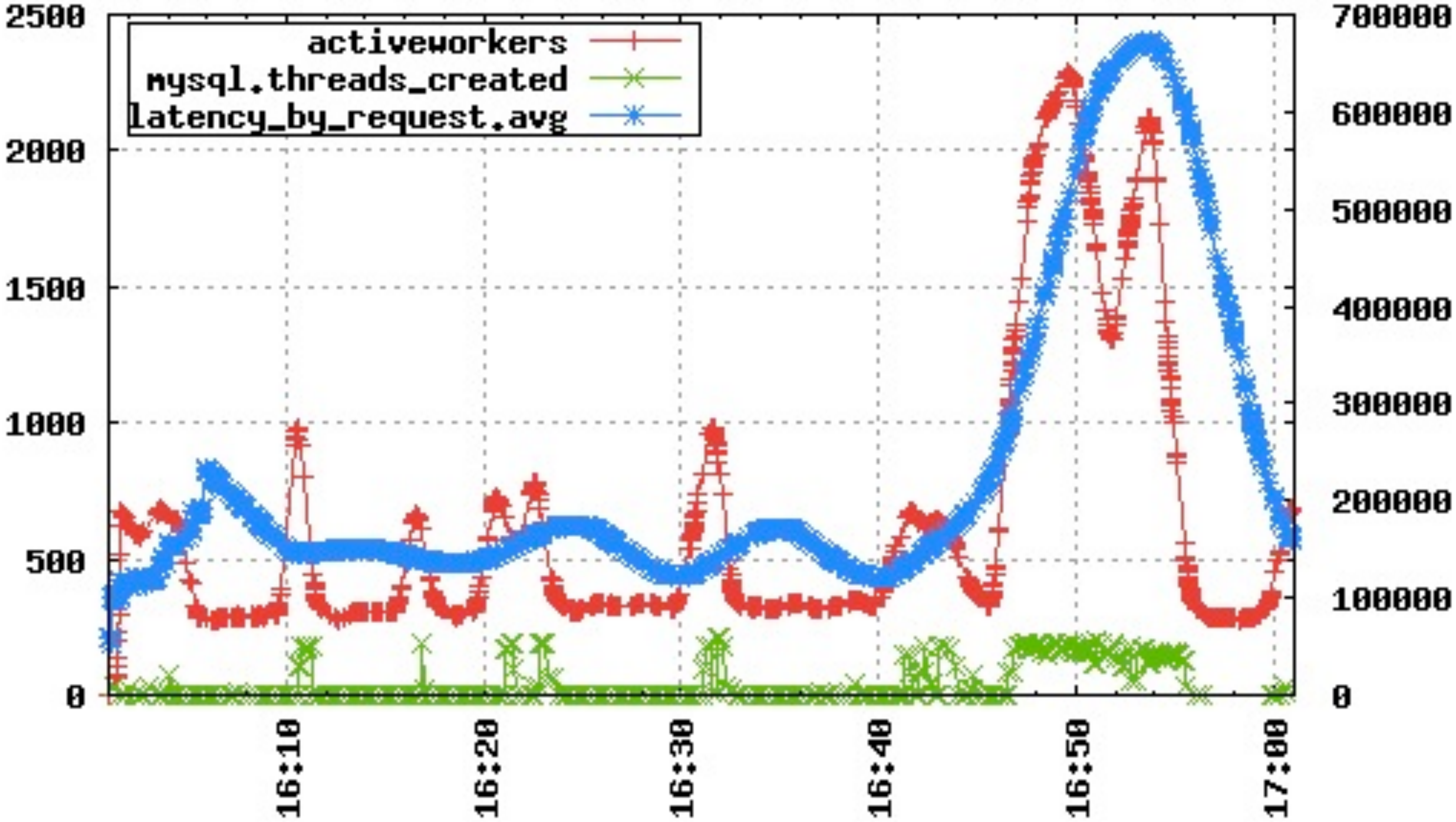
opentsdb

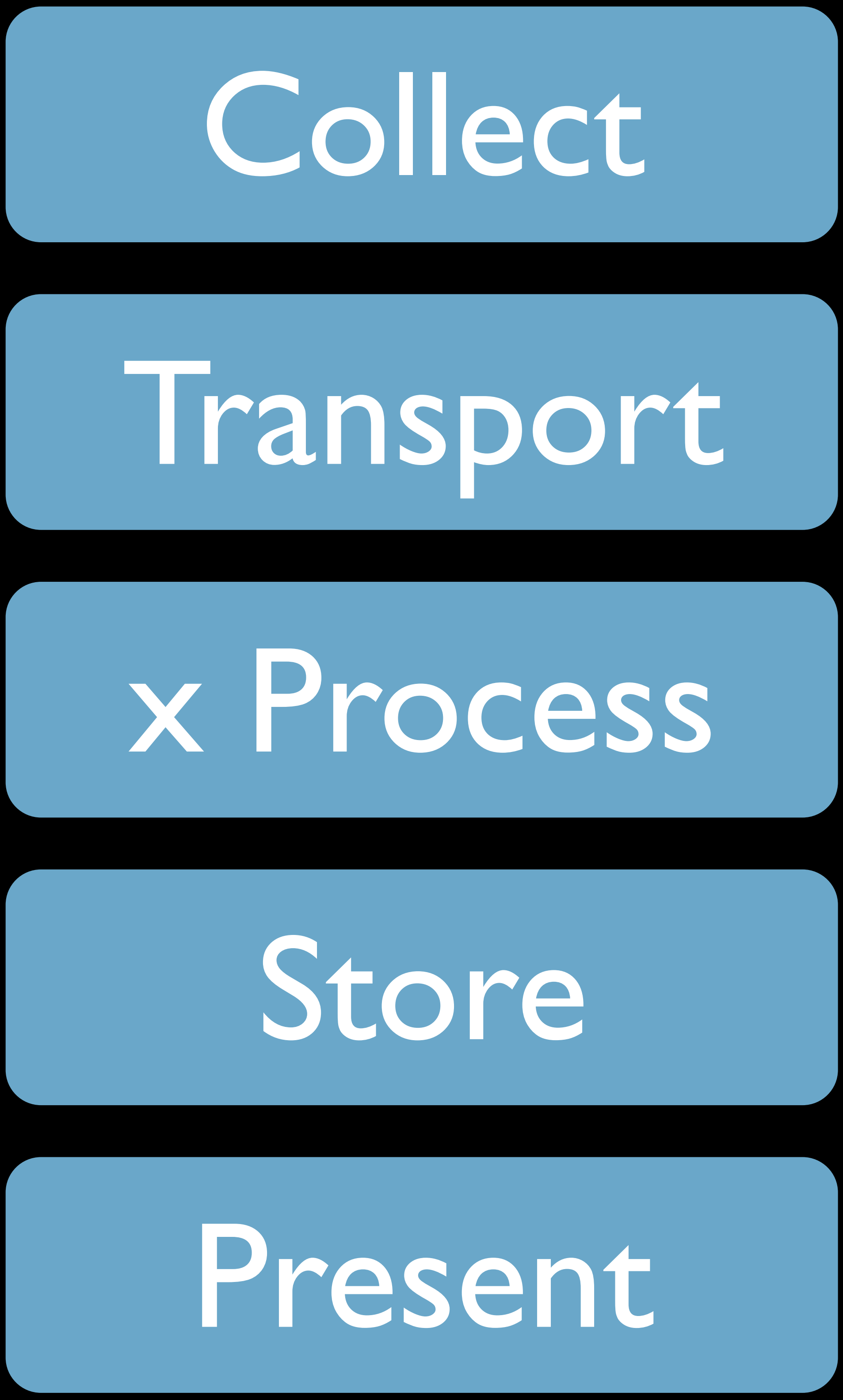
Very new project
Leverages the power of Hadoop’s HDFS
Gets scalable i/o power + elimination of SPOF’s

Submit data with names + tags

Presentation is still very limited but looks very powerful
Every data-gathering project I have ever been involved with has ended up i/o bound
Love the idea of being able to ‘throw hardware’ at it.

Includes the ‘tcollector’ project which fills a similar niche to collectd


activeworkers
mysql.threads_created
latency_by_request.avg
2500
2000
1500
1000
500
0
700000
600000
500000
400000
300000
200000
100000
0
16:10
16:20
16:30
16:40
16:50
17:00


Collect

Transport

x Process

Store

Present

esper

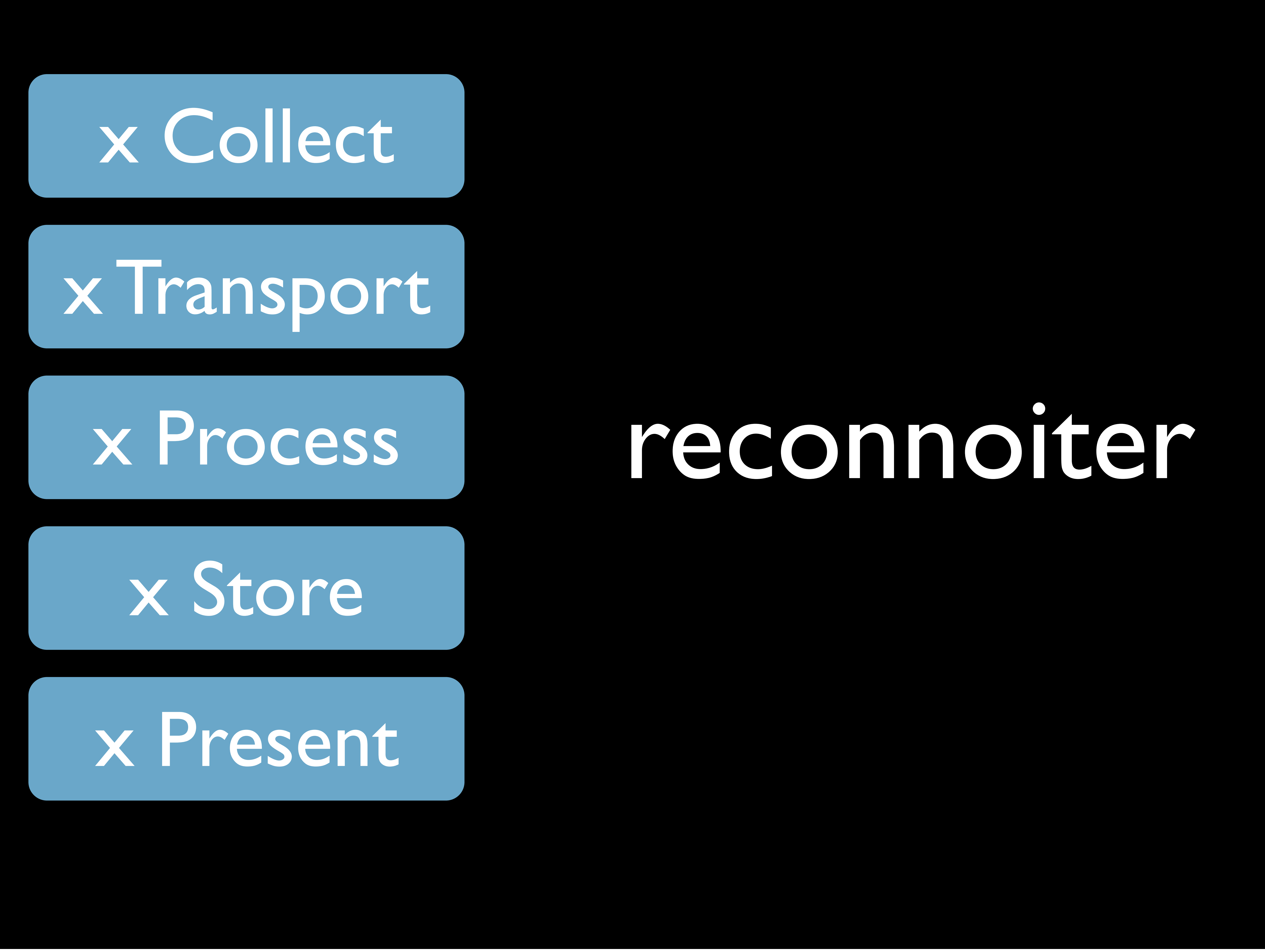
“Streaming Database”

Lets you write SQL-like queries against your data without having to store it all

* Sliding windows (30 second moving average of multidimensional data)

We actually do a lot of things you COULD do with esper via scripts that predate esper, so will mention those use cases when I talk about that pattern

It is java so you need to know a bit of it to make it work



x Collect

x Transport

x Process

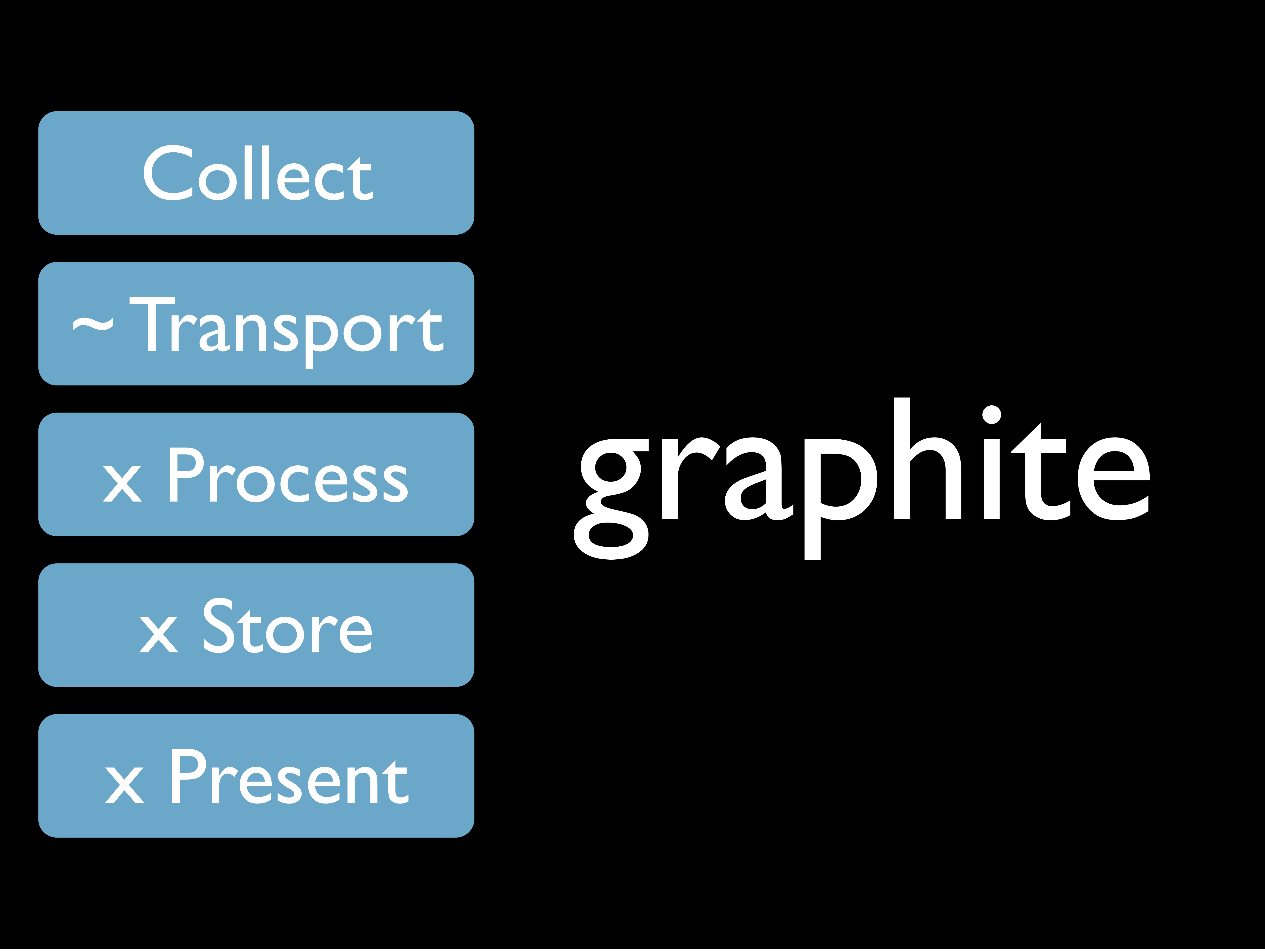
x Store

x Present

reconnoiter

https://labs.omniti.com/labs/reconnoiter
http://omniti.com/video/noit-oscon-demo

Still pretty hacktastic to get going
(Presumably?) the basis for the commercial Circonus SaaS monitoring solution
Watching, but have not used yet
Some really good ideas
- lua engine for health checks (go away nagios with your forking processes for every check!)
- Includes esper, but you need to code your own magic (log parsing/what 'queries' you want)
- Looks and feels a lot like networking gear (CLI configuration/usage)



Collect

~ Transport

x Process

x Store

x Present

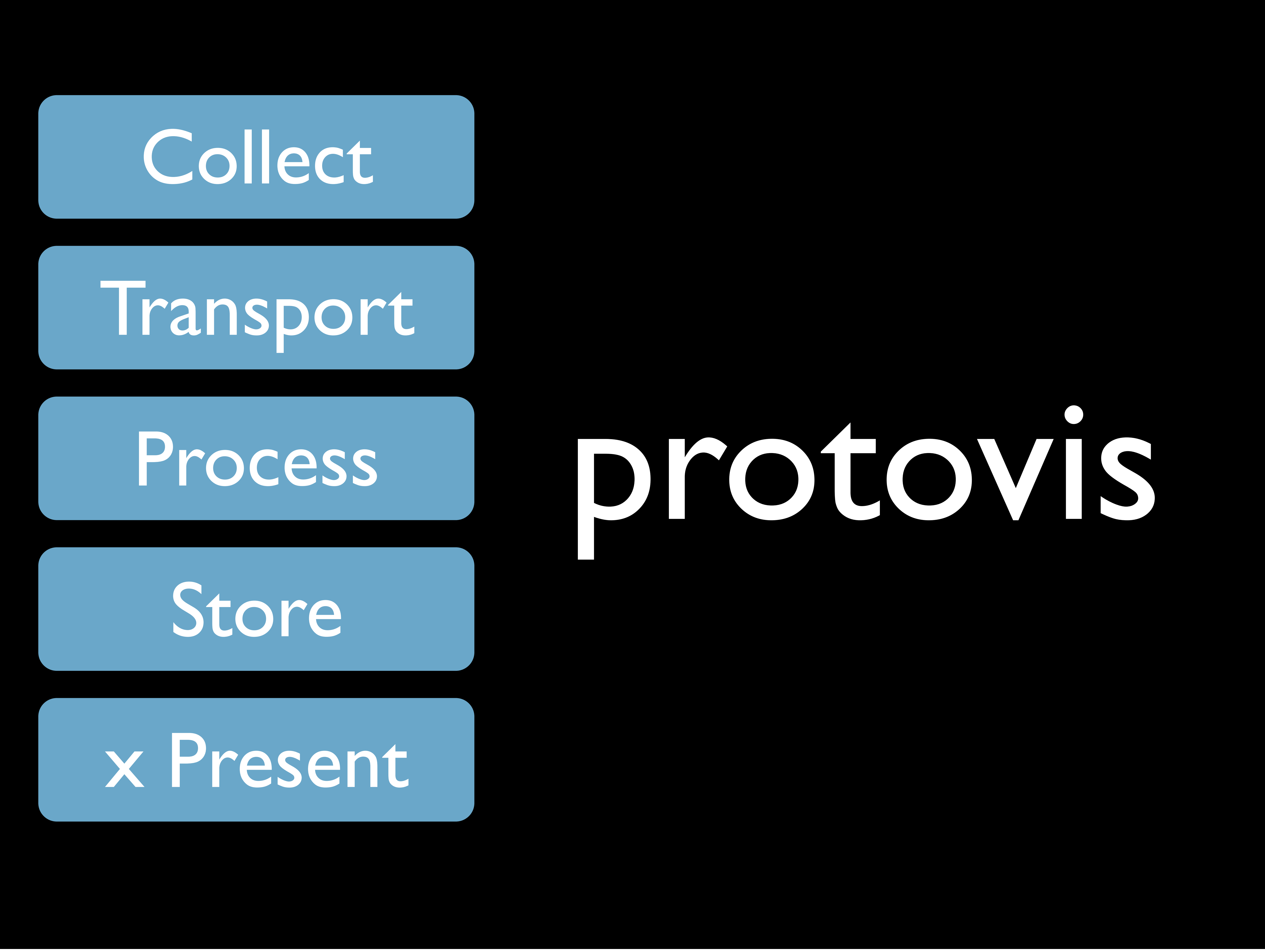
graphite

My favorite so far
Use super heavily @ (mt)
Will show many graphs from it

Allows you to shove data at it very simply.
(<ts> <some.name.with.dots> <value>)

That ends up creating a “folder structure” based on the dots
“Normal humans” can browse your data and learn a lot

Even though it doesn’t provide the Collection/Transport layers
There are lots of easy ways to get data to it because it’s so simple
collectd
Etsy’s statsd https://github.com/etsy/statsd/
Rocksteady (OSS, google via admob:
which adds the collect/transport layer, and again, integrates esper!)



Collect

Transport

Process

Store

x Present

protovis

Javascript toolkit from Stanford

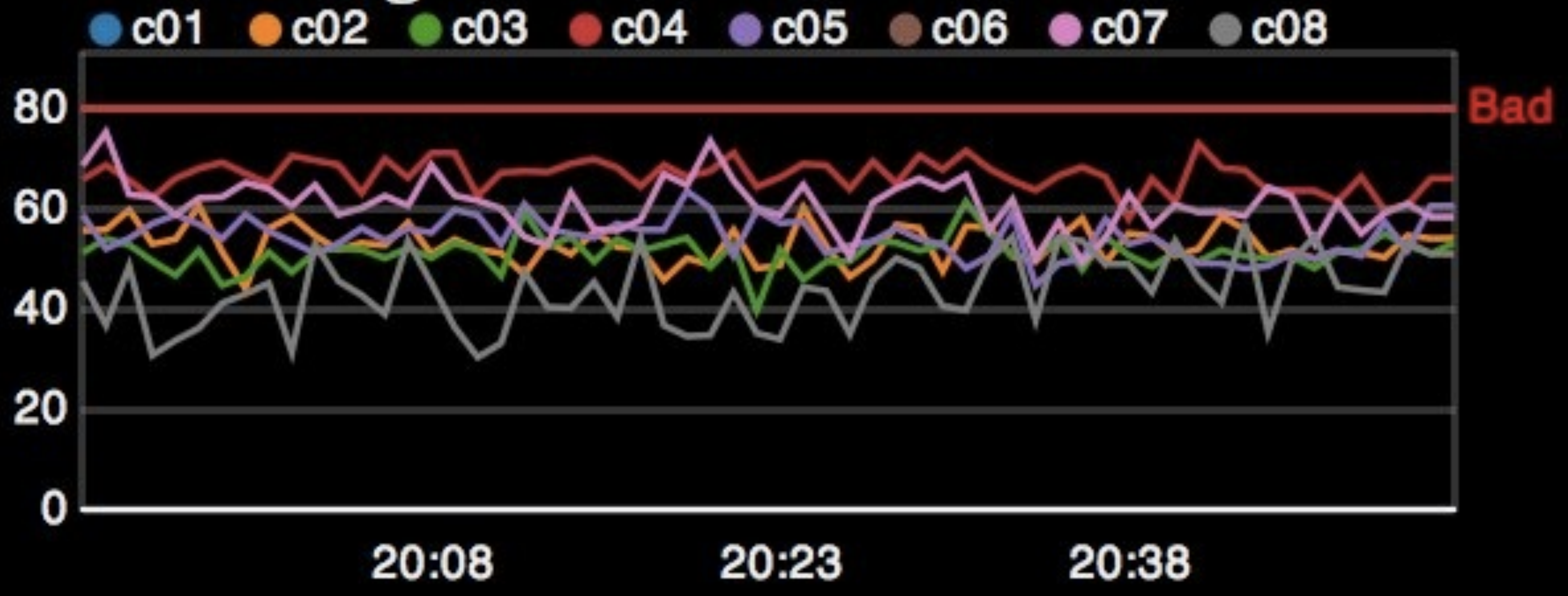
http://vis.stanford.edu/protovis/

JSON -> Interactive graphics

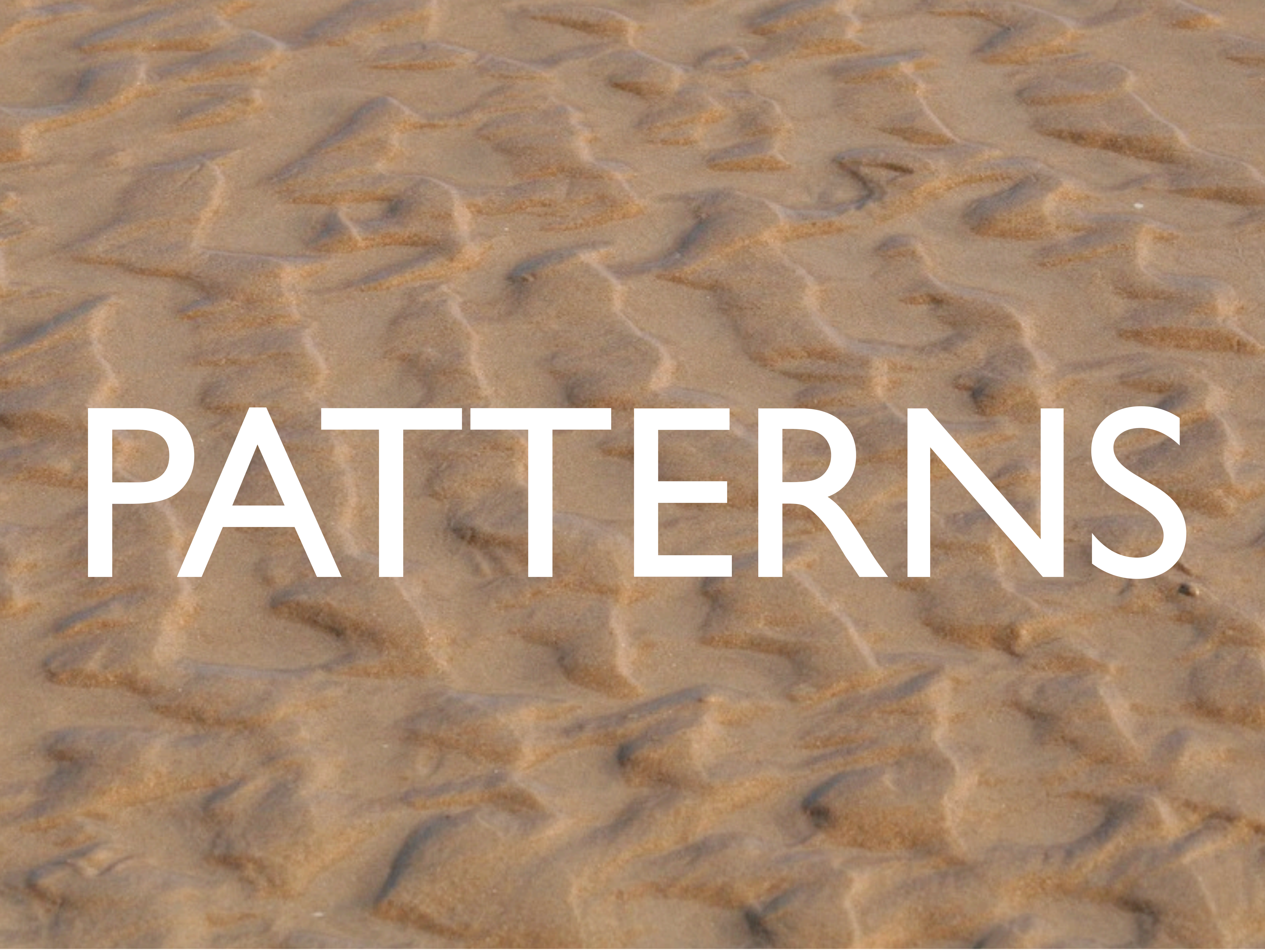
Lets you customize the display as much as you want

Easy to bridge to pretty much any other tool
Write a script that turns a query -> JSON and caches it, can do in any language

Web Node Avg CPU



PATTERNS



#1: Become the User

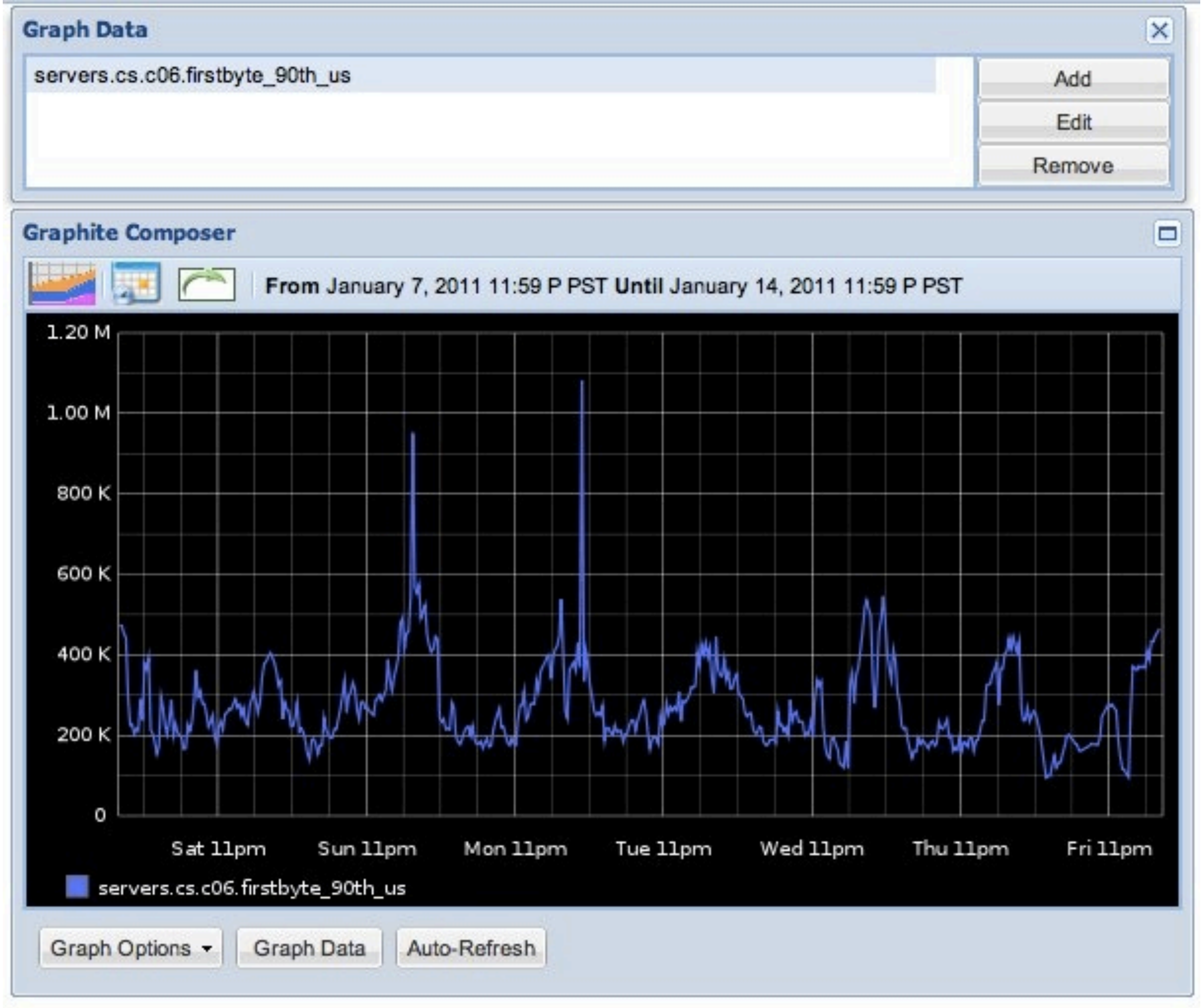
Or, “User” if your consumer is another service or system

Find out what matters most to the user
(Uptime? Responsiveness?)
Figure out a way to monitor that.

Web users don’t care about what the load is on your server, or your DB, or if the network is saturated, or any of that.

They care about how fast their page is loading. The rest is your problem. Measure it (real world) when you can!

Don’t time loading a test blog, it will be heisenbuggy
(better cache, no real traffic and interactions, probably sane plugins, etc)
Monitor how quick you are serving real ones!



servers.cs.c06.firstbyte_90th_us

From January 7, 2011 11:59 P PST **Until** January 14, 2011 11:59 P PST

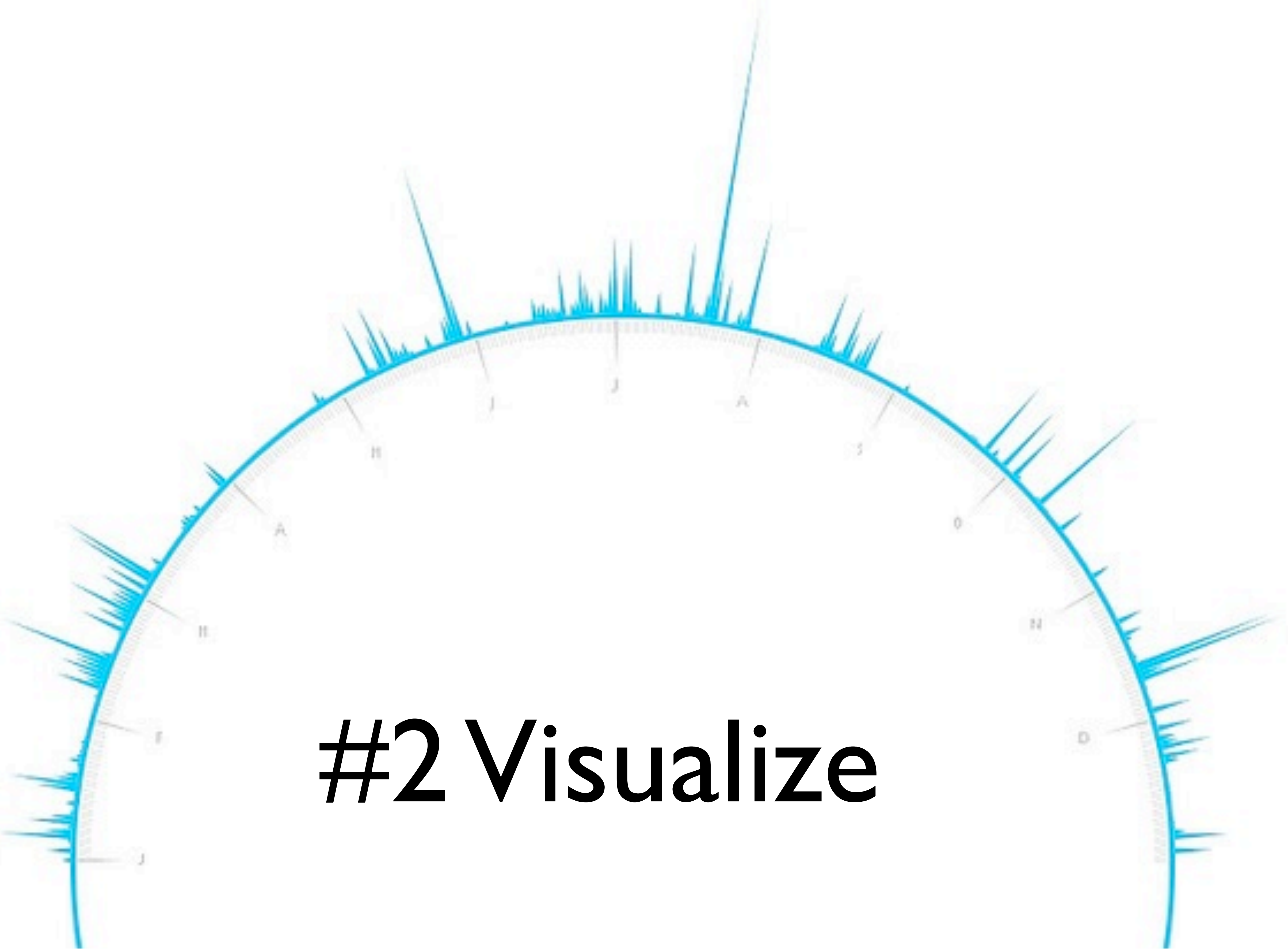
🟦 servers.cs.c06.firstbyte_90th_us

mod_log_firstbyte in apache is simple but useful
Adds a new counter that tells you (in microseconds) how long it takes to start serving the first byte of content.

Here’s the kind of data you can pull out of graphite very easily. 2nd week in January.
Serving 90% of our requests in between 200ms and 400ms, which is ok, but some peak load spikes Sunday and Monday that were high enough to be worrying.
(Thanks to graphite, I can quickly cross check that and discover what the bottleneck was!)

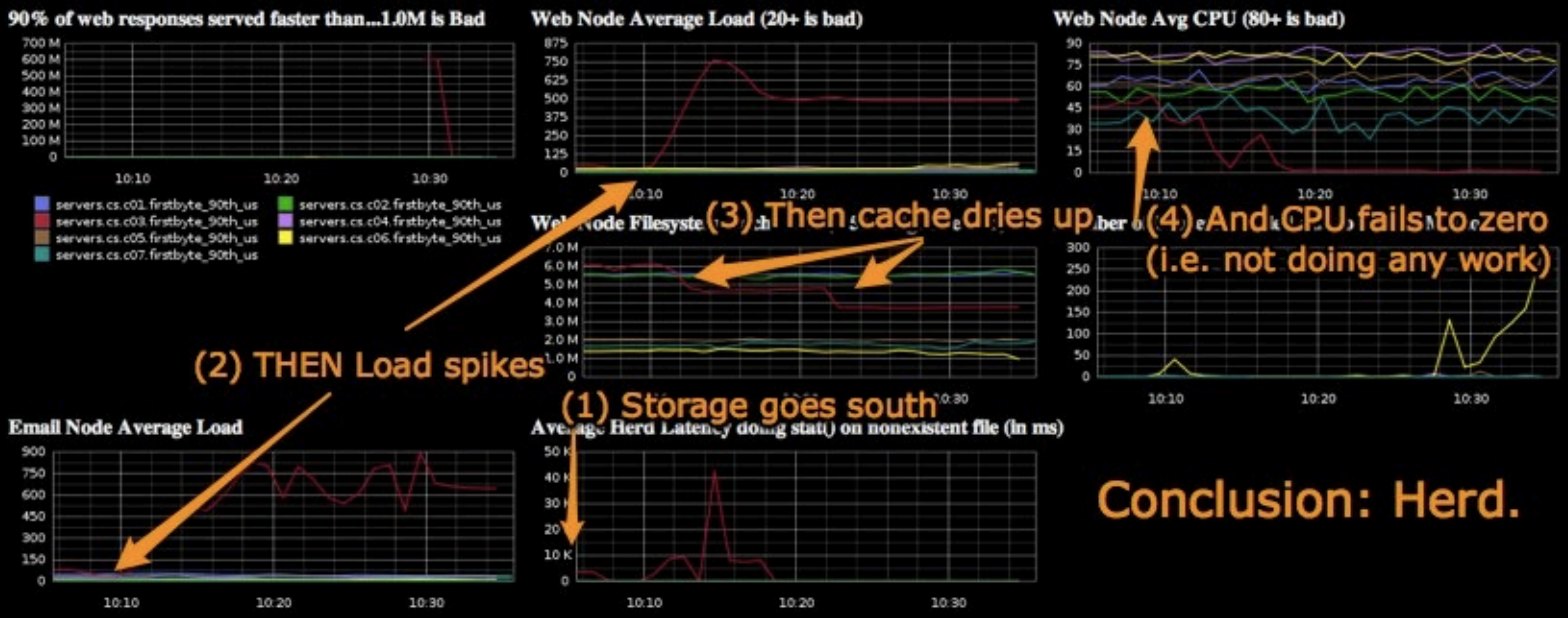
(Why 90th percentile? Web hosting is crazy, people attempt all kinds of things in their pages. The slowest loads on any system are EXTREMELY slow, generally due to pathological things.)

#2 Visualize



85% of our brain's neurons are about processing visual cues.
Presenting the data in a way we can unlock that horsepower on is key.

Graphite+PVIS let us consolidate what used to be 8 whole browser screens worth of stats into a small screen which shows the key indicators at a glance.



This is from an older (pre-protovis) version, but it’s massaged graphite data

The annotations on it are from Skitch (another good quick visualization aid)

It shows the fact that we can quickly see not only that a large cluster had an outage, but establish root cause “at a glance” by seeing what happened first

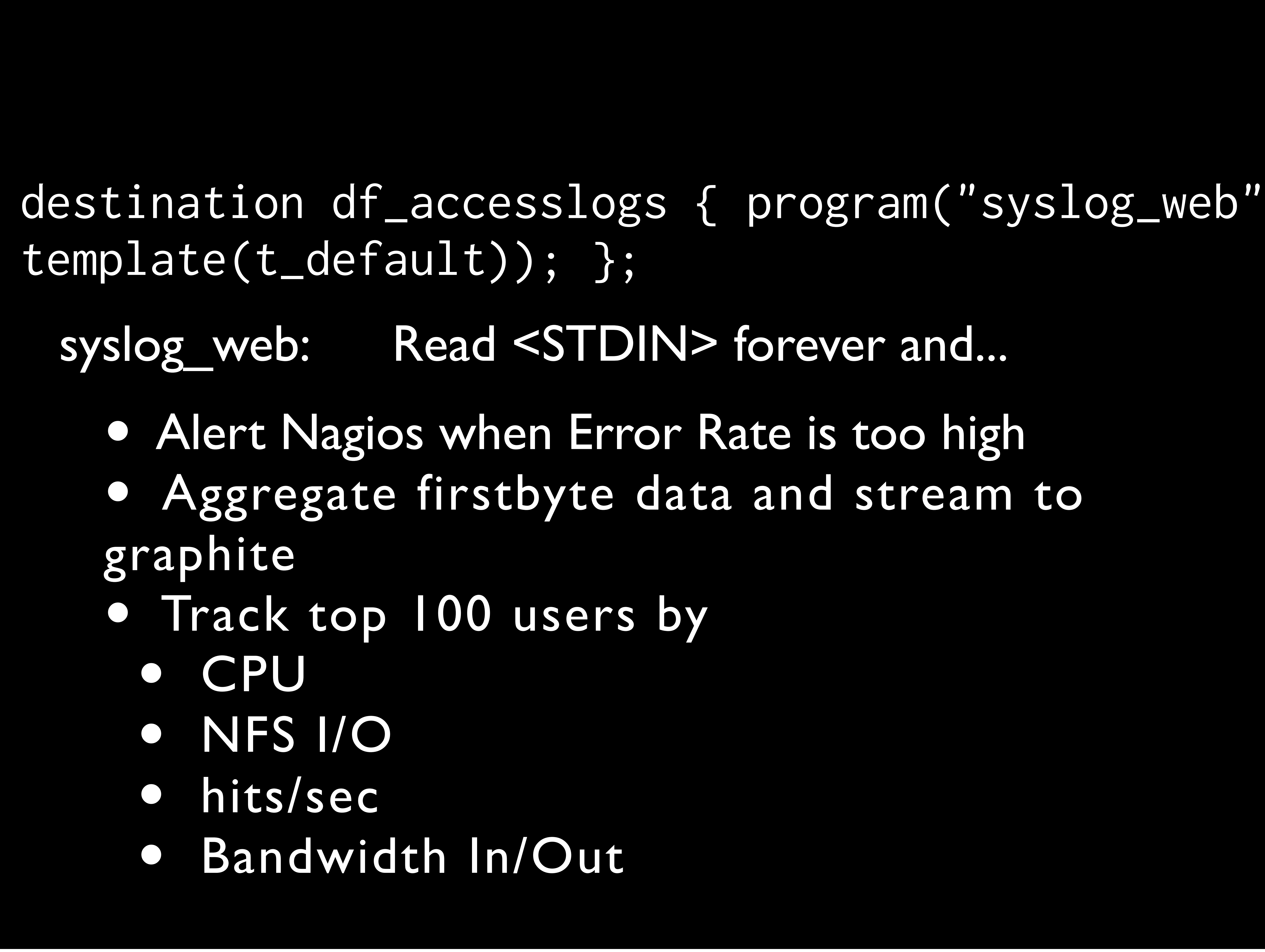
#3 Prune near the Edge

When you end up with TB and PBs of data, it gets harder and WAY more expensive to get a real-time handle on “what’s up.”

Using Esper (or other processing scripts) can extract what you need to know from your data in a hurry.

KB of ‘signal’ from PB of noise.

A little-known feature of syslog-ng (probably rsyslog?) is that you can stream logs through a program:



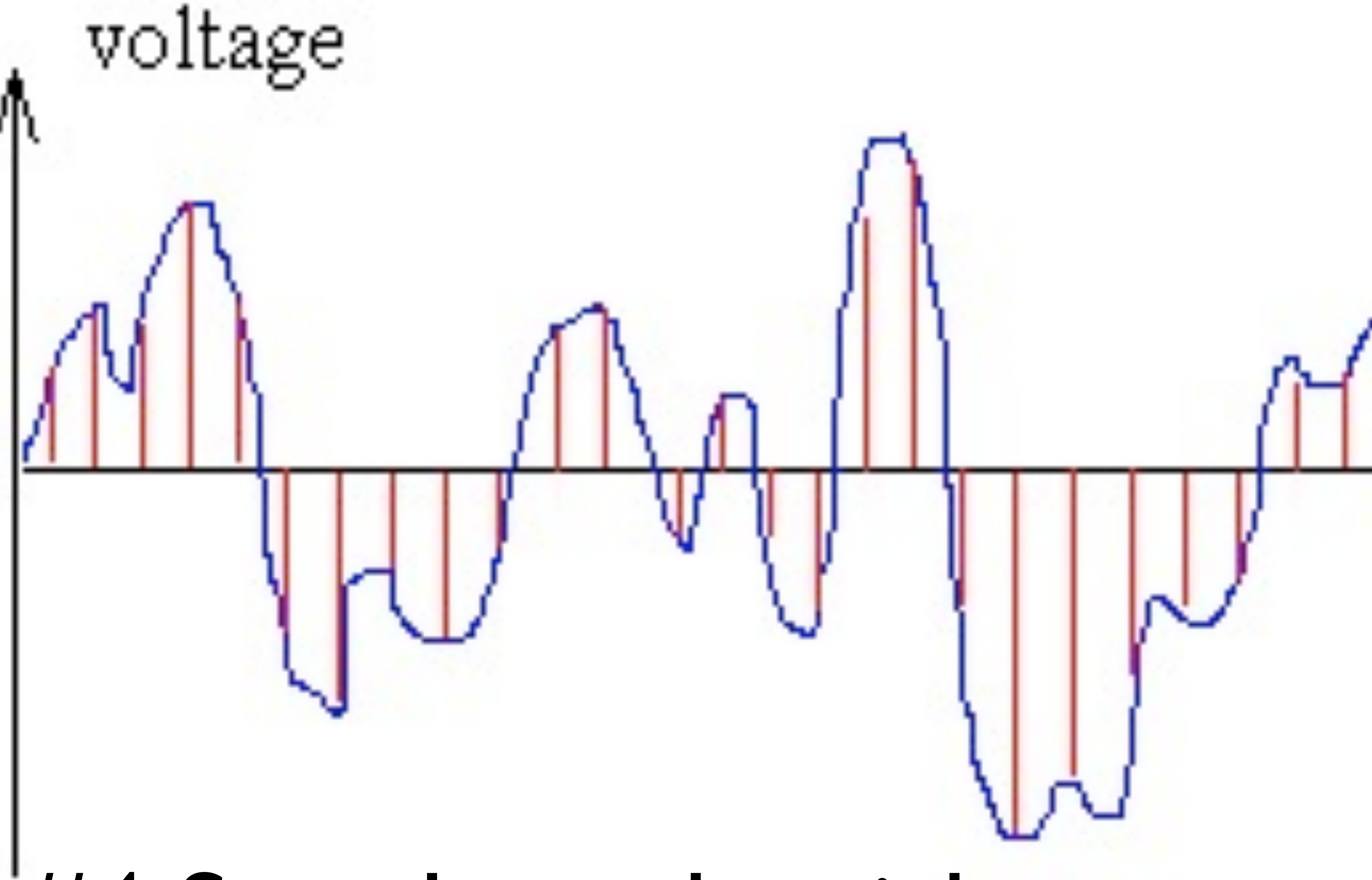
```
destination df_accesslogs { program("syslog_web"
template(t_default)); };
```

syslog_web: Read <STDIN> forever and...

- Alert Nagios when Error Rate is too high
- Aggregate firstbyte data and stream to graphite
- Track top 100 users by
  - CPU
  - NFS I/O
  - hits/sec
  - Bandwidth In/Out

You can do all of this with Esper as well
Would love to as it would be easier to add new features

Tracking about 5k lines/second, <2% CPU and 100MB of RAM



#4 Sample at the right rate

You don’t want to be tracking data that changes daily every minute.
(Lots of storage overhead, search overhead, etc with no value.)

You also NEED to be tracking it fast enough!
Had a problem with

* Polling counter data 1x/minute
* Average looked good, “not the problem”
* Turned out was 100% for 30 seconds, and then stalled (0) for 15 seconds...
* Needed to sample 1x/ second to see!

#5 Make it reliable

The paradox of getting more signal from your noise?

Turns out the signals are valuable, and you start to depend on them

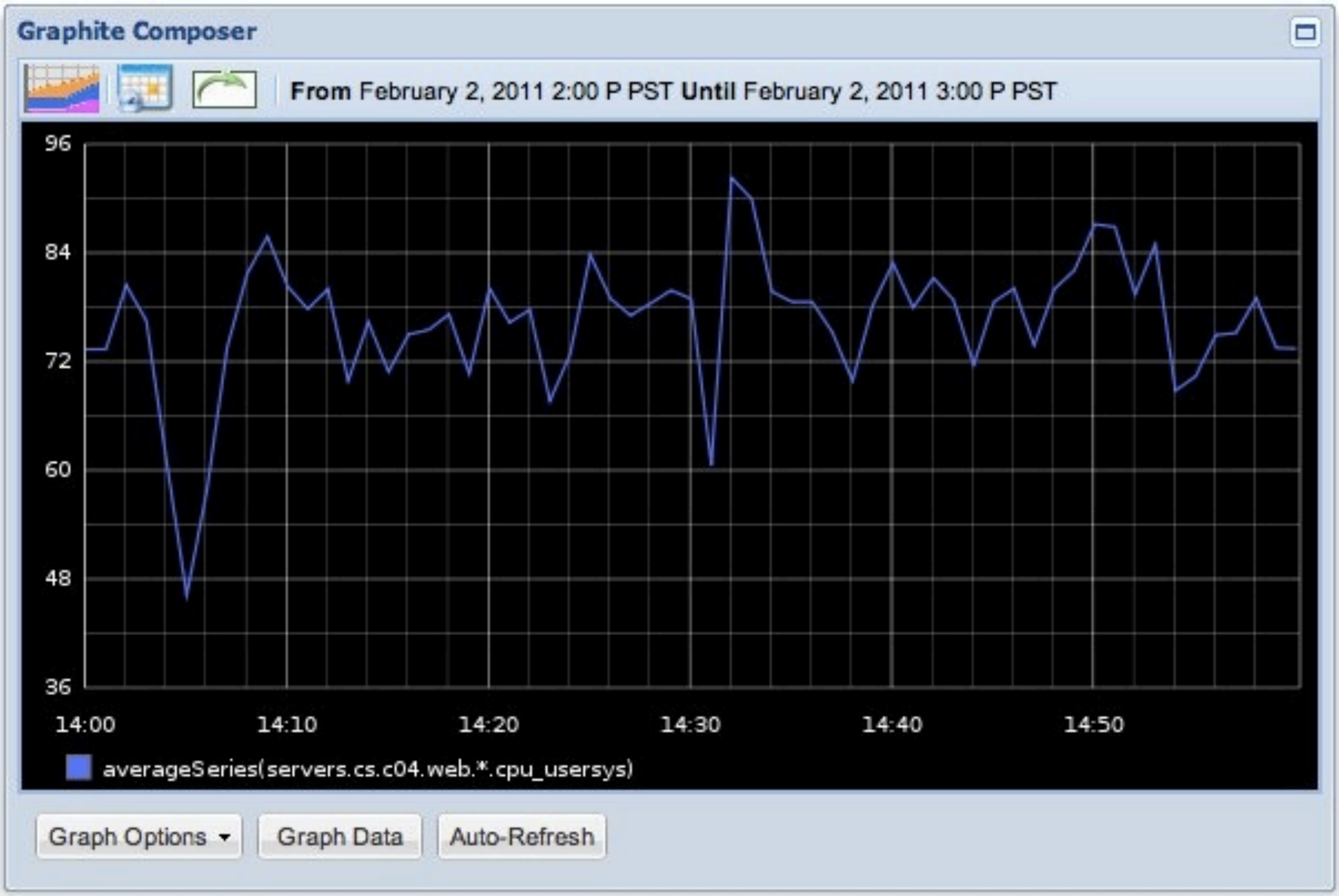
Also, nothing is more annoying than trying to do trending or post mortems and finding big holes in your data

This is one of the biggest compelling reasons for OpenTSDB to be so exciting!

#6 Love (and Fear) Aggregates

Aggregates (averages, maximums, top N’s, percentiles, etc) are one of the main ways you can simplify.

We really can’t grapple with 80 data points at once, (say a whole cluster) so you NEED something to simply that.

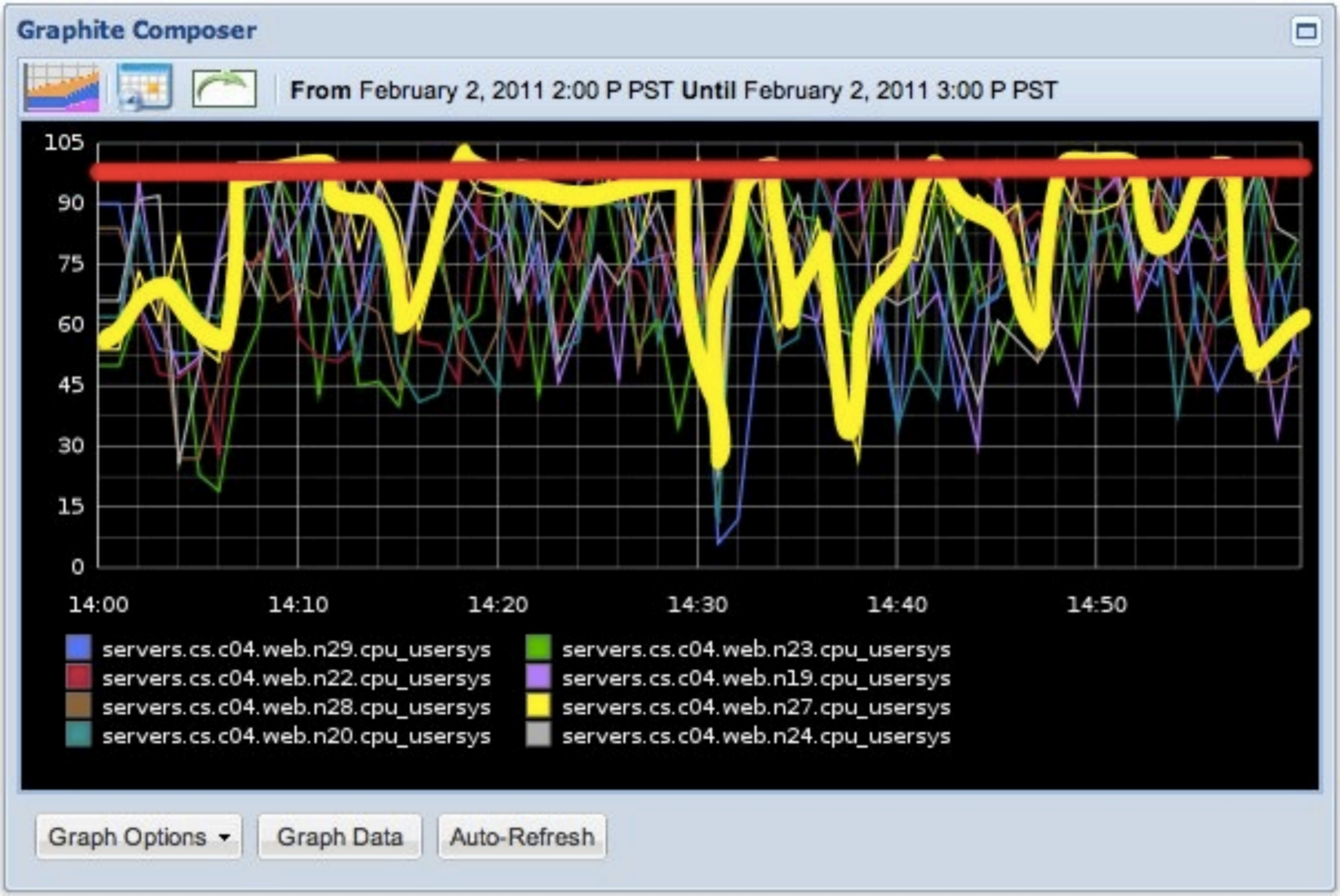


From February 2, 2011 2:00 P PST **Until** February 2, 2011 3:00 P PST

🟦 averageSeries(servers.cs.c04.web.*.cpu_usersys)

A Wednesday afternoon

Average of the CPU utilization (user+system time)



Graphite Composer
From February 2, 2011 2:00 P PST Until February 2, 2011 3:00 P PST
105
90
75
60
45
30
15
0
14:00
14:10
14:20
14:30
14:40
14:50
servers.cs.c04.web.n29.cpu_usersys
servers.cs.c04.web.n22.cpu_usersys
servers.cs.c04.web.n28.cpu_usersys
servers.cs.c04.web.n20.cpu_usersys
servers.cs.c04.web.n23.cpu_usersys
servers.cs.c04.web.n19.cpu_usersys
servers.cs.c04.web.n27.cpu_usersys
servers.cs.c04.web.n24.cpu_usersys
Graph Options
Graph Data
Auto-Refresh

But when we ‘unpack the average’ and look at some of the outlier hosts
(Thanks to graphite’s awesome “mostDeviant()” function)

Scribbled on it with skitch to see what was happening

CPU was actually “pegging” and “bouncing off”

Hitting 100%, then getting dropped from the load balancer, then bouncing back once it came back in.

So an Average of 80% (“bad, but not down yet”) is falsely assuring us things are not quite in the weeds

This is another reason the “get in your user’s shoes” -- this problem was discovered because the user performance metrics were worse than expected

#8 Learn Ad-hoc Skills

How do you know what is 'signal' in the first place?
What do you put in that gorgeous Protovis dashboard the whole team uses?
How do you handle it when you've missed something in your metrics system and you need to go back to the raw logs?

grep
awk
fex!: http://semicomplete.com/projects/fex/
bit.ly's 'data hacks': https://github.com/bitly/data_hacks
gnuplot
R

grep
awk
fex!: http://semicomplete.com/projects/fex/
bit.ly’s ‘data hacks’:

https://github.com/bitly/data_hacks

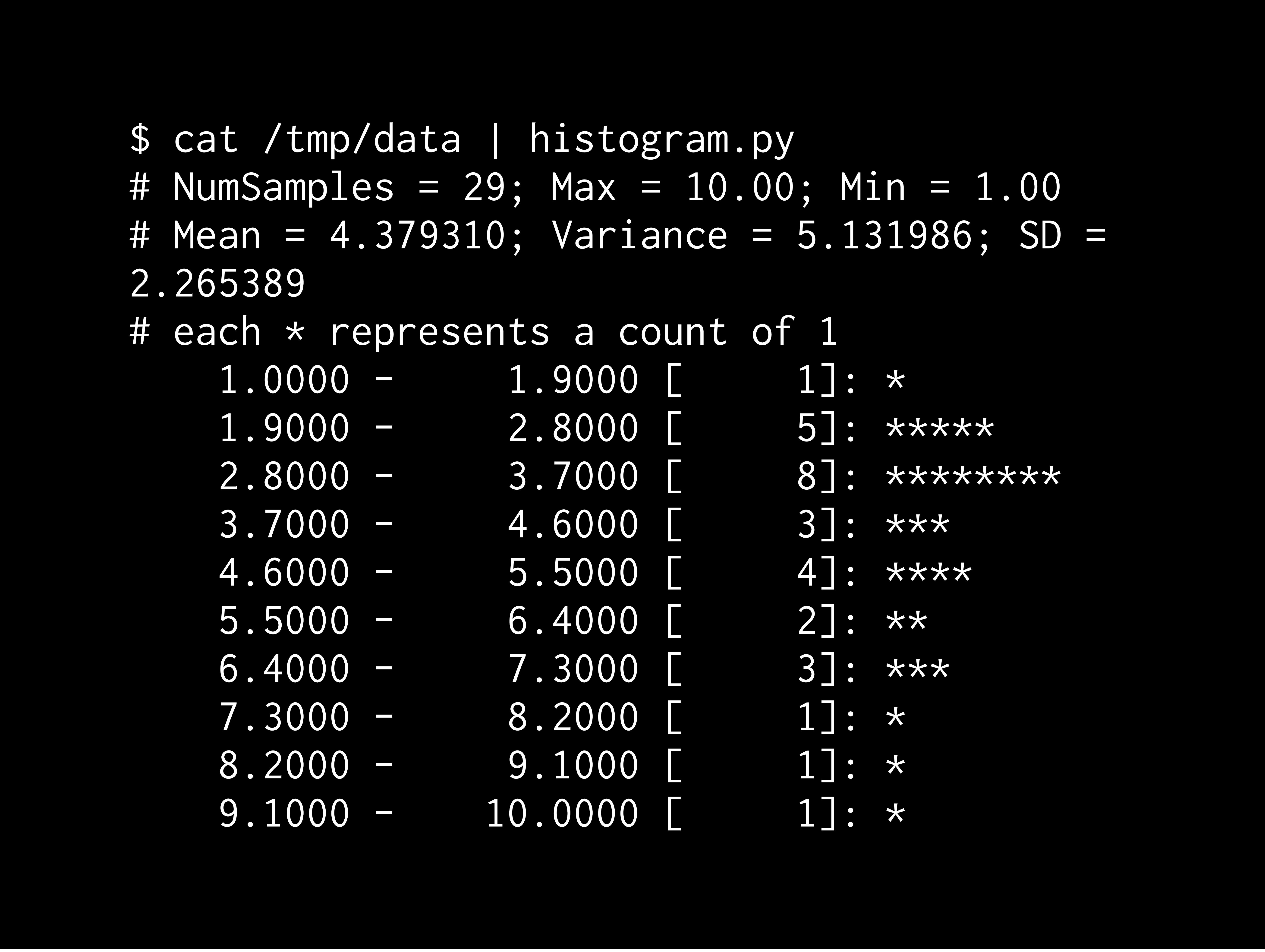
gnuplot
R

There are tons of command line and GUI tools you can use to learn a lot about your data in a hurry

Fex is a new development from the prolific/awesome Jordan Sissel
Check out his logstash project as well that I wanted to talk about today but cut for time
Really good for pulling fields out of files, regardless of delimiter crazyness

Data Hacks, some CLI tools from Bit.ly to quickly learn about your data
Percentiles, Histograms, Bar Charts, etc.

And the desktop powerhouses: gnuplot and R



```
$ cat /tmp/data | histogram.py
# NumSamples = 29; Max = 10.00; Min = 1.00
# Mean = 4.379310; Variance = 5.131986; SD =
2.265389
# each * represents a count of 1
    1.0000 -     1.9000 [     1]: *
    1.9000 -     2.8000 [     5]: *****
    2.8000 -     3.7000 [     8]: ********
    3.7000 -     4.6000 [     3]: ***
    4.6000 -     5.5000 [     4]: ****
    5.5000 -     6.4000 [     2]: **
    6.4000 -     7.3000 [     3]: ***
    7.3000 -     8.2000 [     1]: *
    8.2000 -     9.1000 [     1]: *
    9.1000 -    10.0000 [     1]: *
```

Quick Data Hacks output

An example of a recent R session

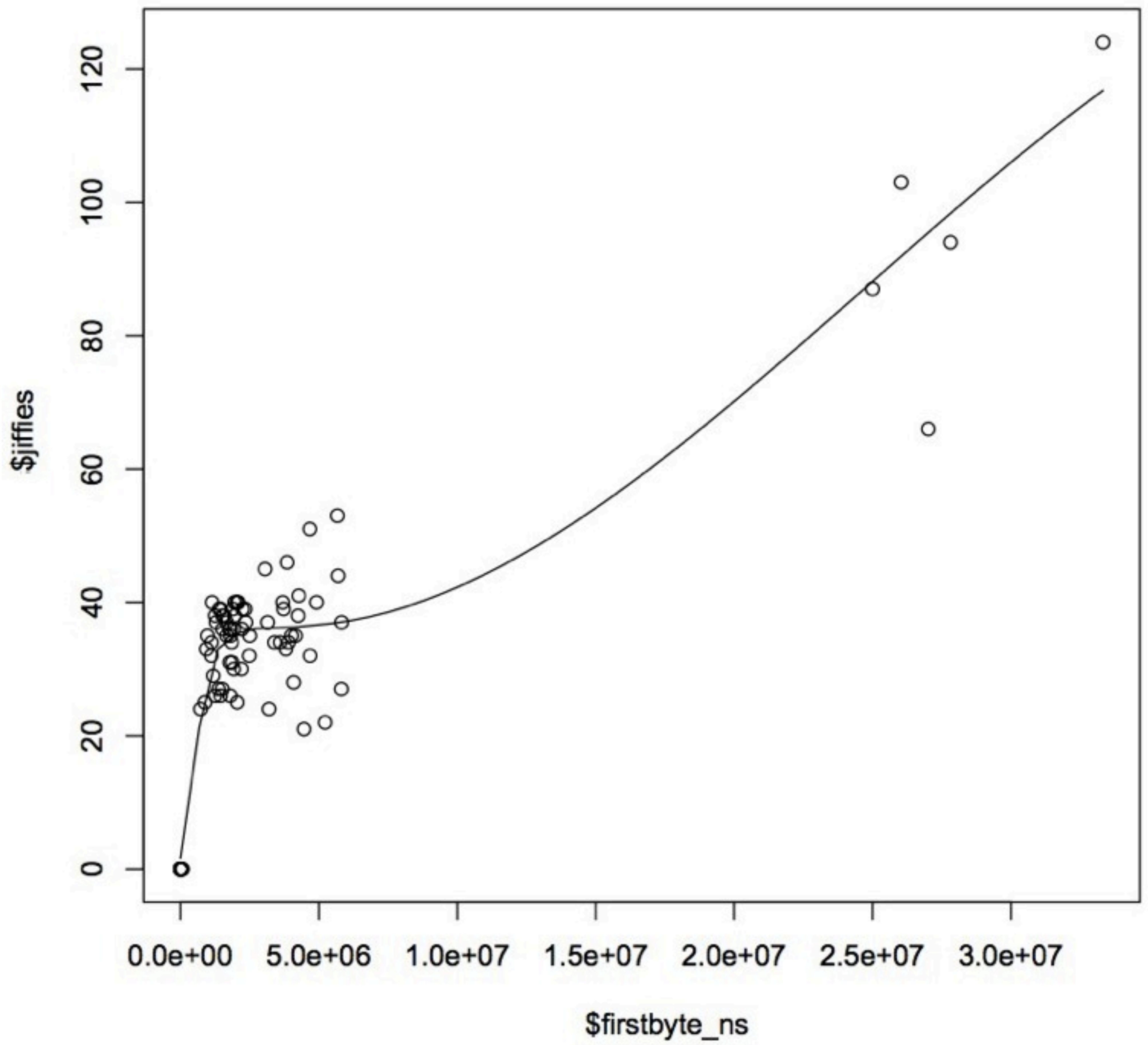
Trying to find out how much CPU time in jiffies typical sites use for dynamic contents

```
# @ the shell....

$ awk '{print $11,$14}' 10-access_log > /tmp/jiffies_v_time.txt
```

```
# In R.....

> nws <- read.table('jiffies_v_time.txt', na.strings = ("-"))
> colnames(nws) <- c("jiffies", "firstbyte_ns")
> scatter.smooth(x=nws$firstbyte_ns, y=nws$jiffies)
```



Amazing visual with a very advanced statistical model applied to the data for a curve fit

We can see there's a "sweet range" for this client
20-40 jiffies, takes .1->.7 seconds

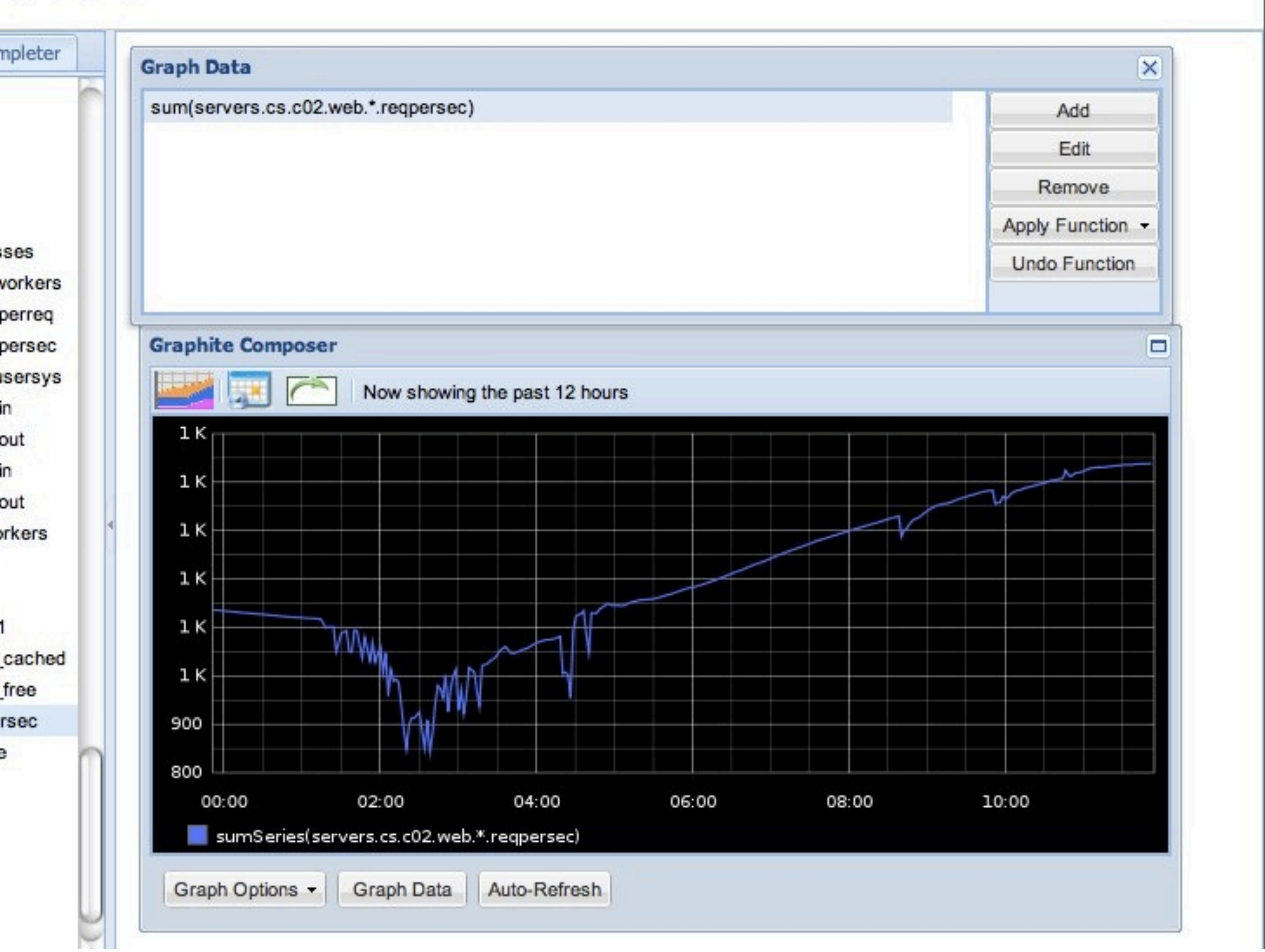
Some outliers take longer, but they are the ones that eat more CPU time

Great insight with 4 lines, and not anything we had been looking for from "pre-baked" analysis paths

#8 Sweat the Units

Units matter.

Firstbyte is stored in microseconds
Graphite doesn't know that (we could have scaled it on the way in)
So now 1 Mega-firstbyte == 1 second
Confusing



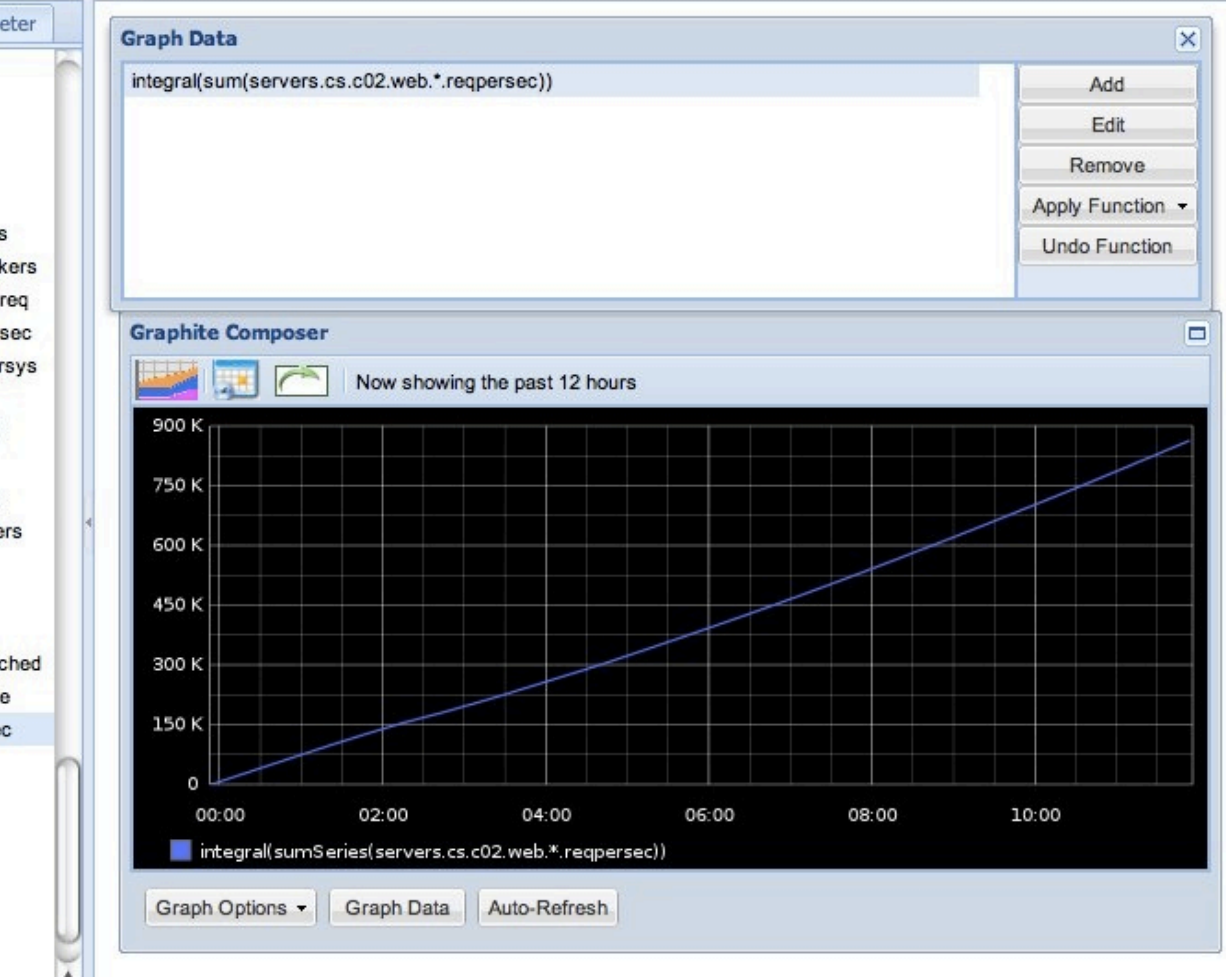
```
sum(servers.cs.c02.web.*.reqpersec)
```

Now showing the past 12 hours

▨ sumSeries(servers.cs.c02.web.*.reqpersec)

Graphite example, showing off more of the analysis power

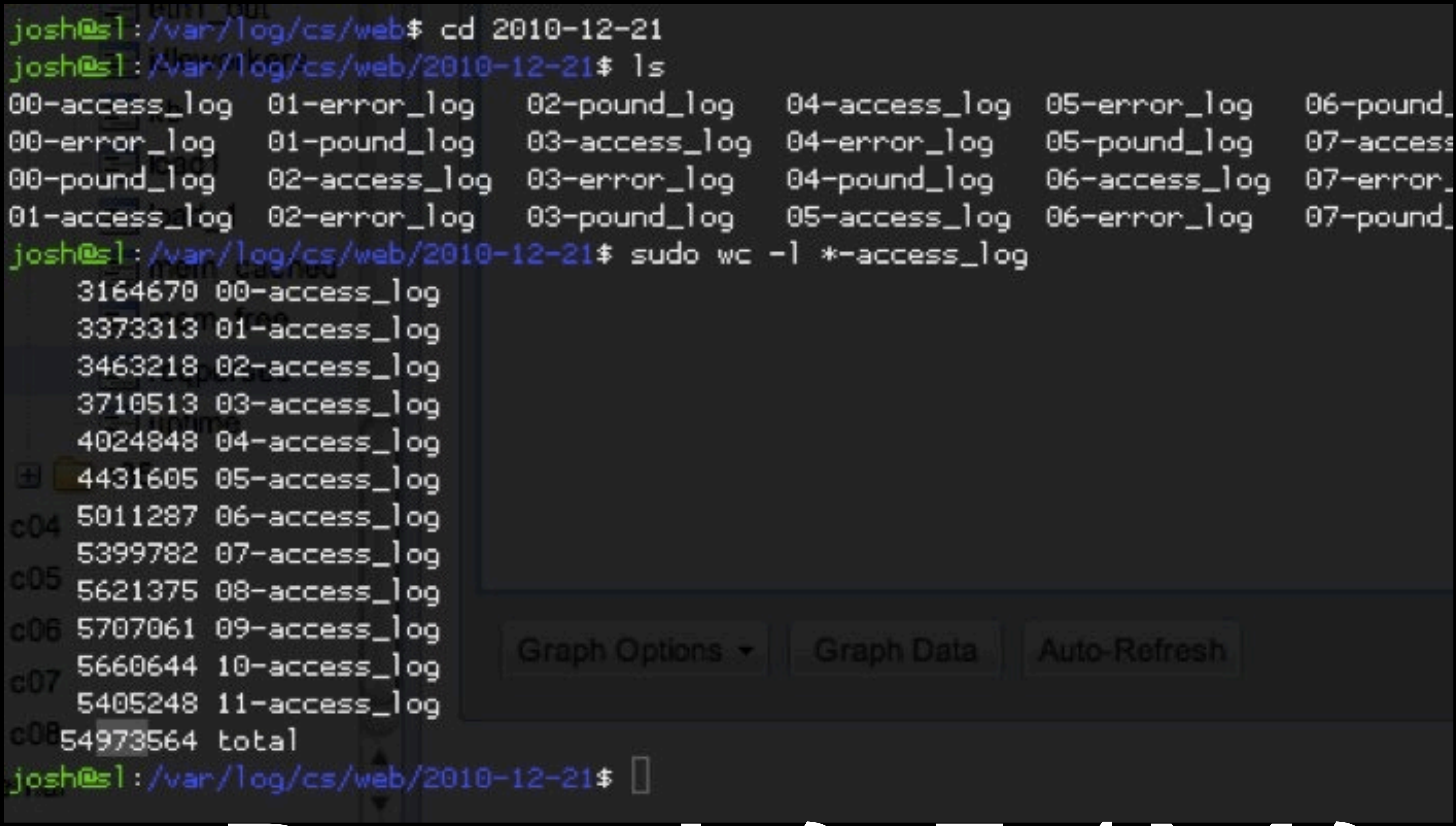
Sum all the web servers Requests/second
If we take the integral, we should be able to find “total requests for the day”



integral(sum(servers.cs.c02.web.*.reqpersec))

Now showing the past 12 hours

Cool, that works. Max out at 900k



```
josh@s1:/var/log/cs/web$ cd 2010-12-21
josh@s1:/var/log/cs/web/2010-12-21$ ls
00-access_log  01-error_log   02-pound_log   04-access_log  05-error_log   06-pound_
00-error_log   01-pound_log   03-access_log  04-error_log   05-pound_log   07-access
00-pound_log   02-access_log  03-error_log   04-pound_log   06-access_log  07-error_
01-access_log  02-error_log   03-pound_log   05-access_log  06-error_log   07-pound_
josh@s1:/var/log/cs/web/2010-12-21$ sudo wc -l *-access_log
   3164670 00-access_log
   3373313 01-access_log
   3463218 02-access_log
   3710513 03-access_log
   4024848 04-access_log
   4431605 05-access_log
   5011287 06-access_log
   5399782 07-access_log
   5621375 08-access_log
   5707061 09-access_log
   5660644 10-access_log
   5405248 11-access_log
  54973564 total
josh@s1:/var/log/cs/web/2010-12-21$ 
```

Buuuh? 54M?

How do we explain this?

Graphite stores data 1x/minute by default

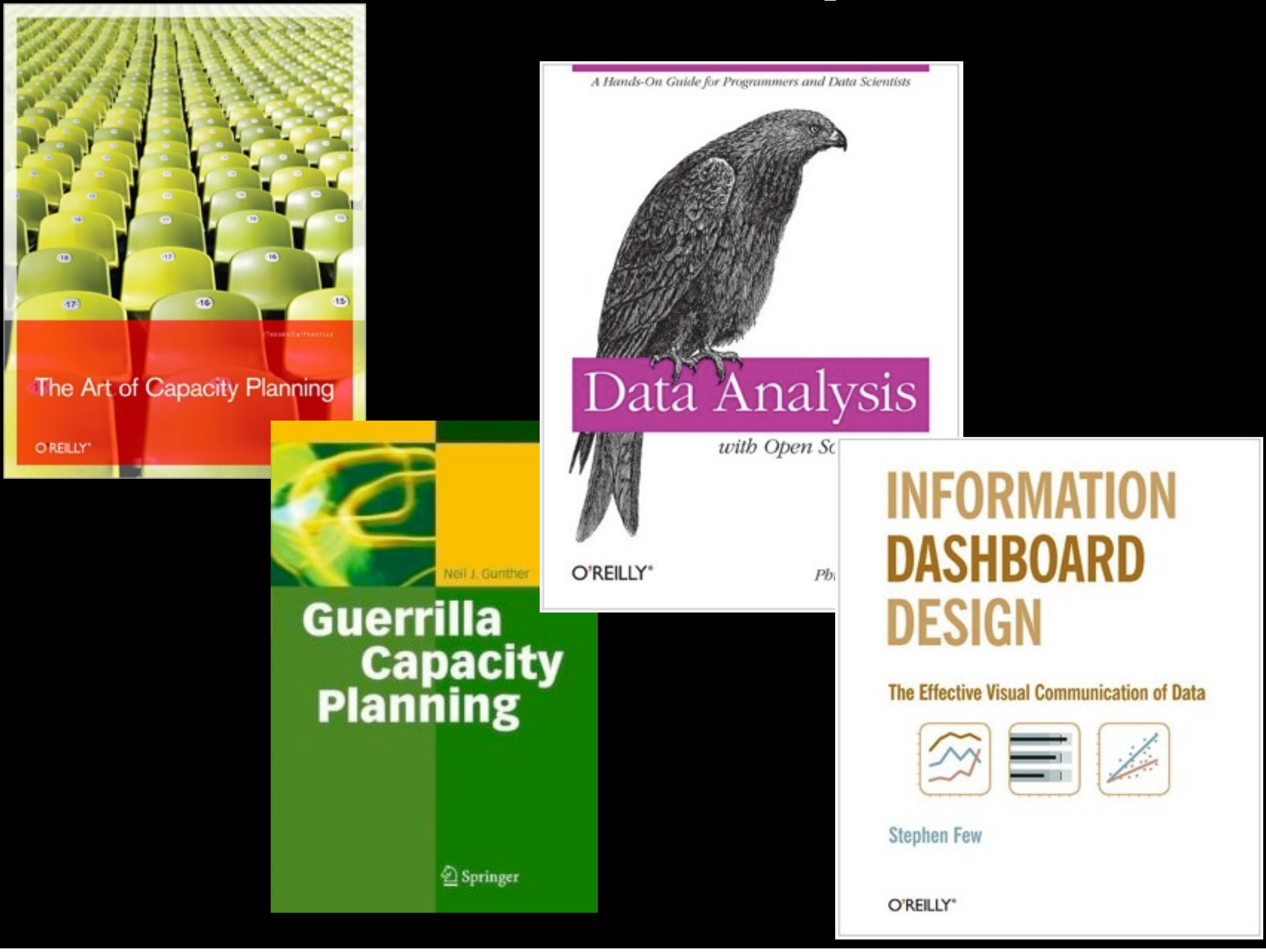
Every minute, store the live “hits per second” value

So when you integrate this, you’re storing “hits per second, once a minute”

Requests/sec
Stored every minute
890k requests/sec
* 60 = 53.4M requests

So if we multiply by 60, badda bing. The math works out.

Go Deeper


The Art of Capacity Planning
O'REILLY
Neil J. Gunther
Guerrilla Capacity Planning
Springer
A Hands-On Guide for Programmers and Data Scientists
Data Analysis
with Open So
O'REILLY
INFORMATION DASHBOARD DESIGN
The Effective Visual Communication of Data
Stephen Few
O'REILLY

Allspaw Capacity Planning
Gunther Capacity Planning
Data w/ Open Source Tools
Information Dashboard Design

R

Credits

"Become the User": http://www.flickr.com/photos/vramak/3567615703/
"Noise": http://www.flickr.com/photos/restlessglobetrotter/434218278/
"Signal": http://www.flickr.com/photos/altemark/304078711/
"Visualize": http://www.flickr.com/photos/yesyesnono/2514409253/
"Goals": http://www.flickr.com/photos/mad_african78/2741067789/
"Units": http://www.flickr.com/photos/lnx/14615953/
"Reliable": http://www.flickr.com/photos/alanenglish/2824228526/
"Dice": http://www.flickr.com/photos/darwinbell/440080655/
"Sample": http://www.flickr.com/photos/ethanhein/3027724070/
"Prune": http://www.flickr.com/photos/fui/870163461/
"Patterns": http://www.flickr.com/photos/foxypar4/422184320/
"Matrix": http://www.flickr.com/photos/trinity-of-one/20562069/
"Green DC": http://www.flickr.com/photos/traftery/4773457853/
"DoF Proliant" http://www.flickr.com/photos/schwenke/2421138425/
"Stopwatch": http://www.flickr.com/photos/purplemattfish/3020016417/
"Spotlight": http://www.flickr.com/photos/14171139@N08/4358123951/