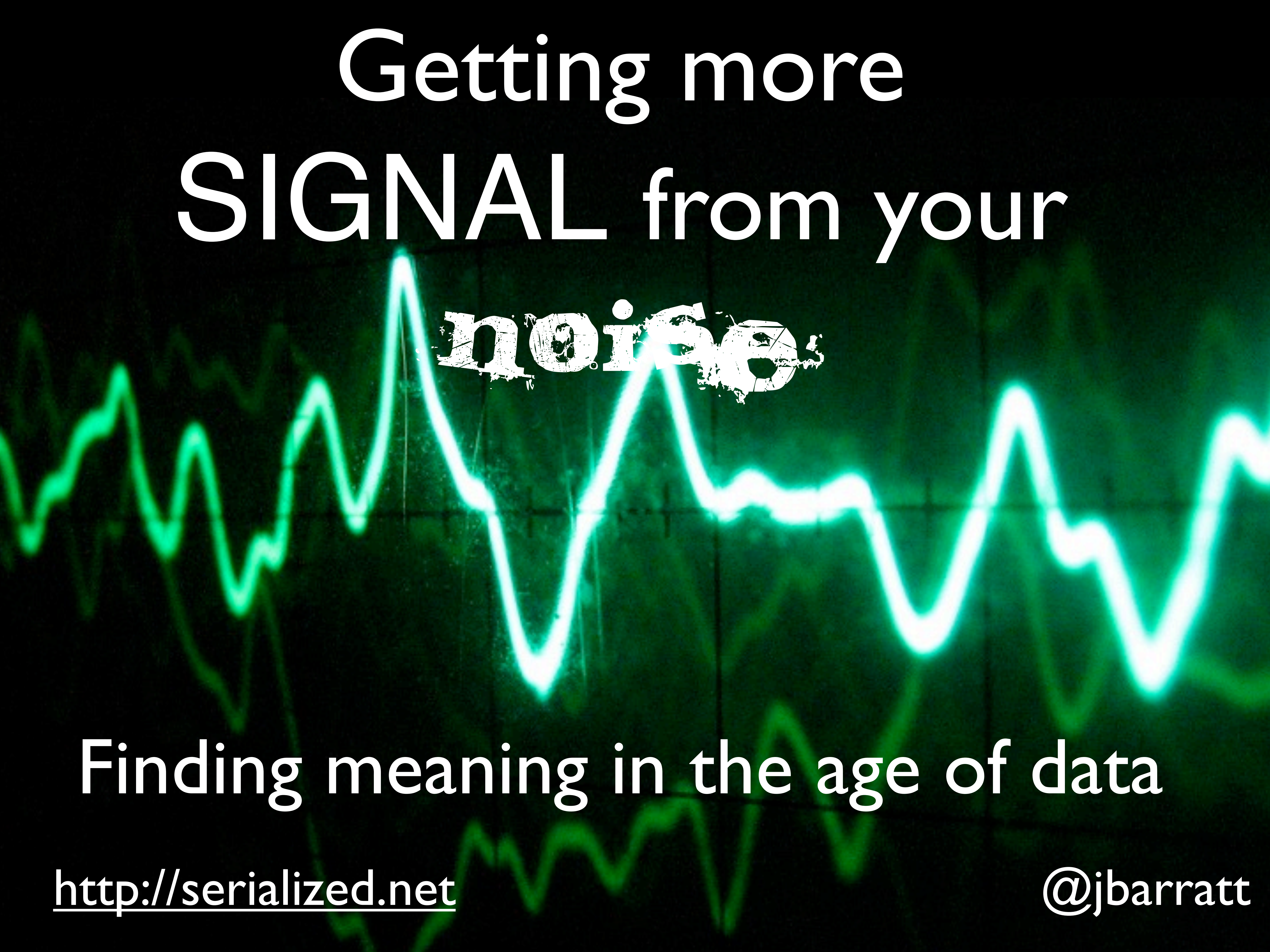
Getting more SIGNAL from your noise



Finding meaning in the age of data

http://serialized.net

@jbarratt

NOISE

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

60
5
10
15
20
25
30
35
40
45
50
55
3
6
9
12
18
21
24
27
30
HERWINS
SWISS
MADE

> “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

SIGNAL

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.

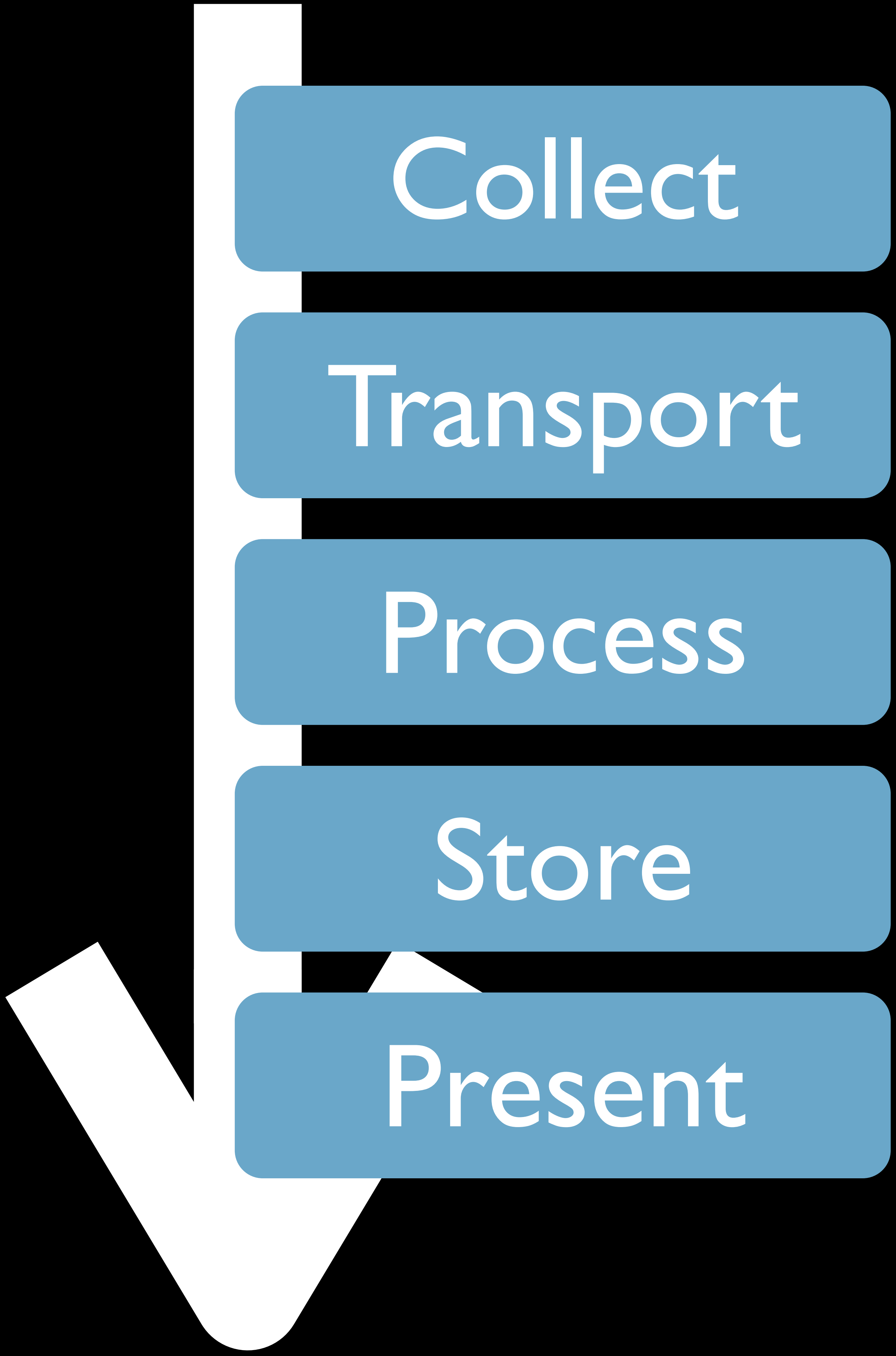
Benoit "tsuna" Sigoure
(OpenTSDB Creator)

Signal/Noise

+

Open Source

Goals



Collect

Transport

Process

Store

Present

Project Spotlight

!! Collect

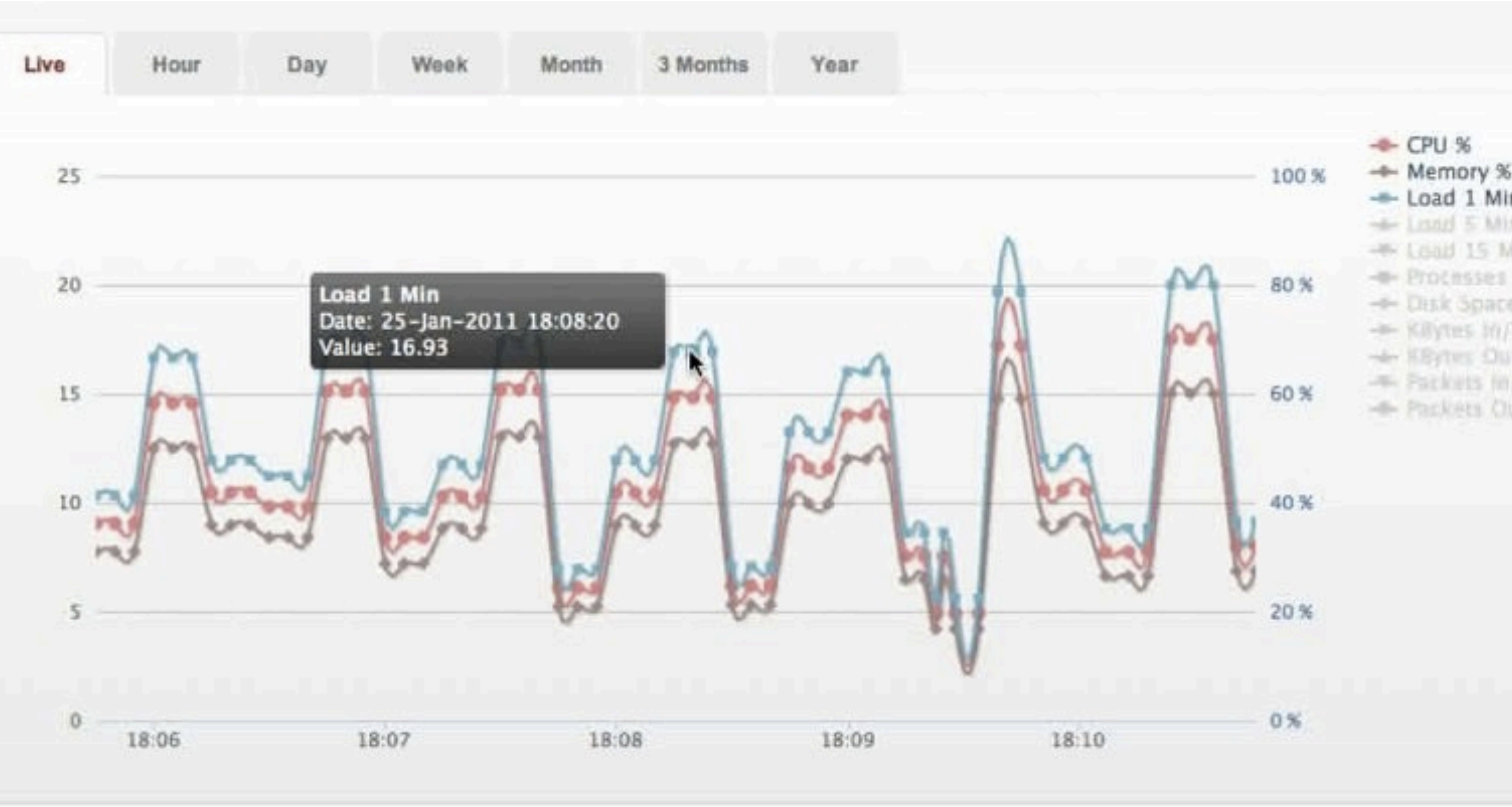
x Transport

~ Process

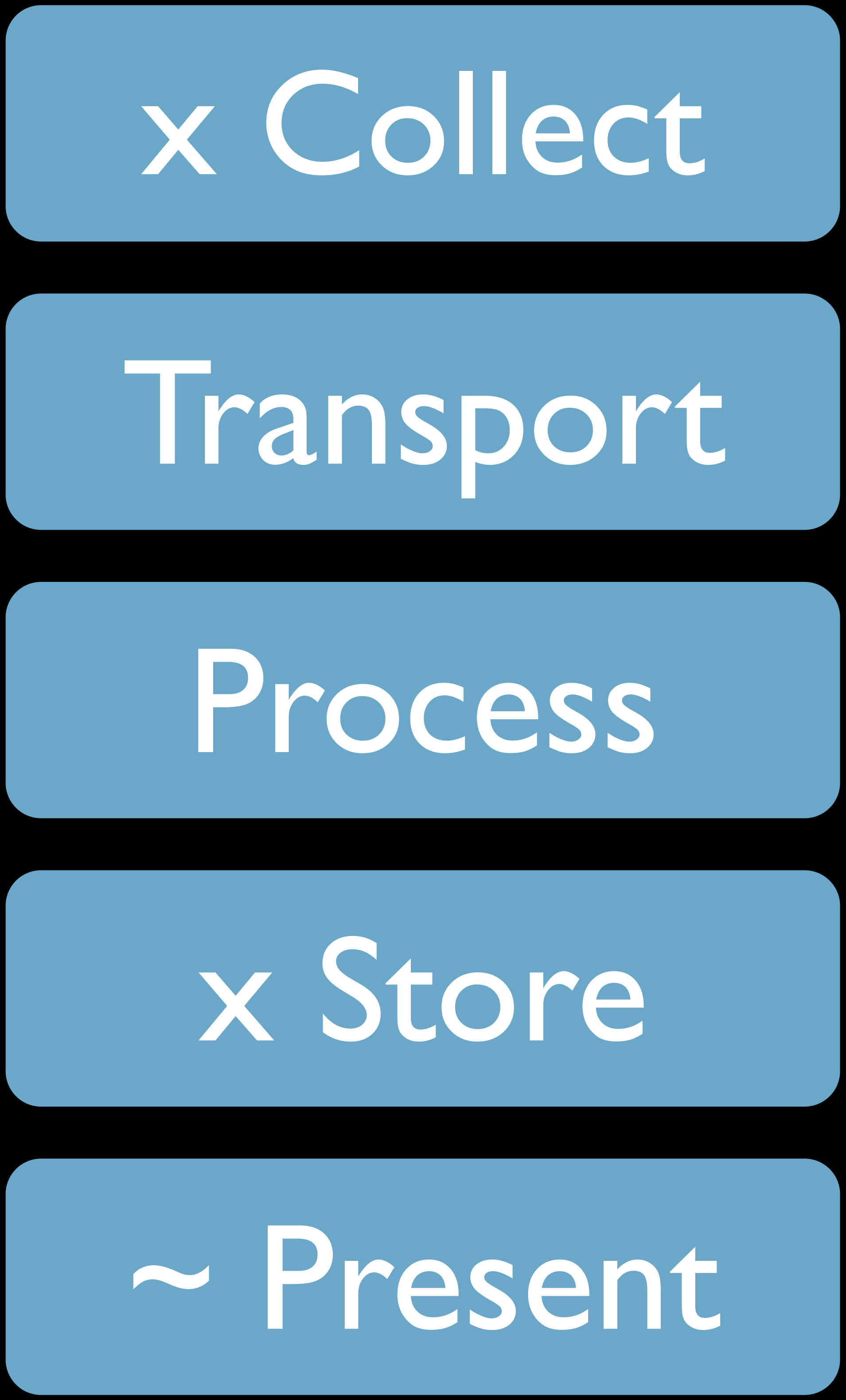
x Store

Present

collectd



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



x Collect

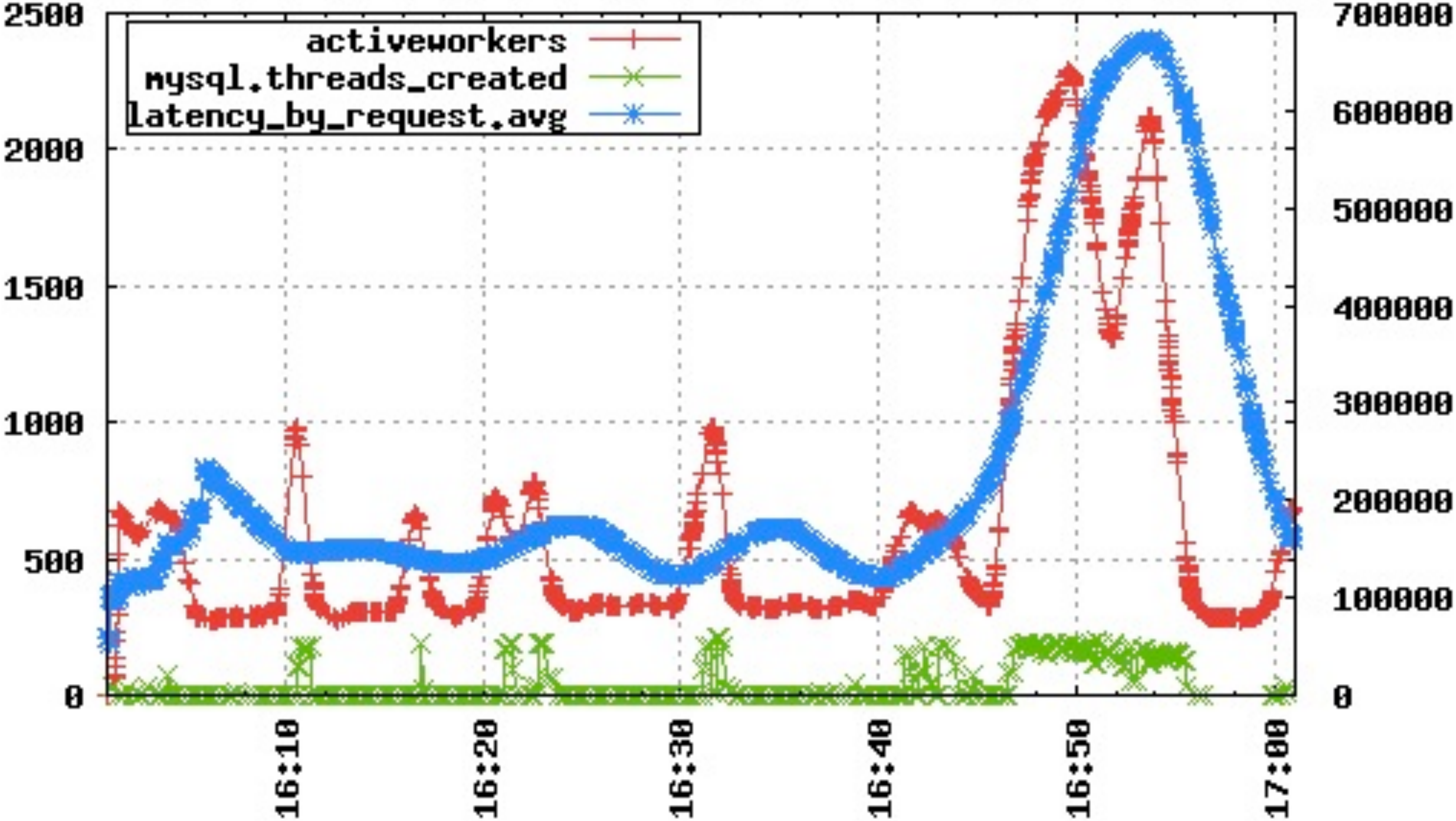
Transport

Process

x Store

~ Present

opentsdb



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

x Collect

x Transport

x Process

x Store

x Present

reconnoiter

Collect

~ Transport

x Process

x Store

x Present

graphite

Collect

Transport

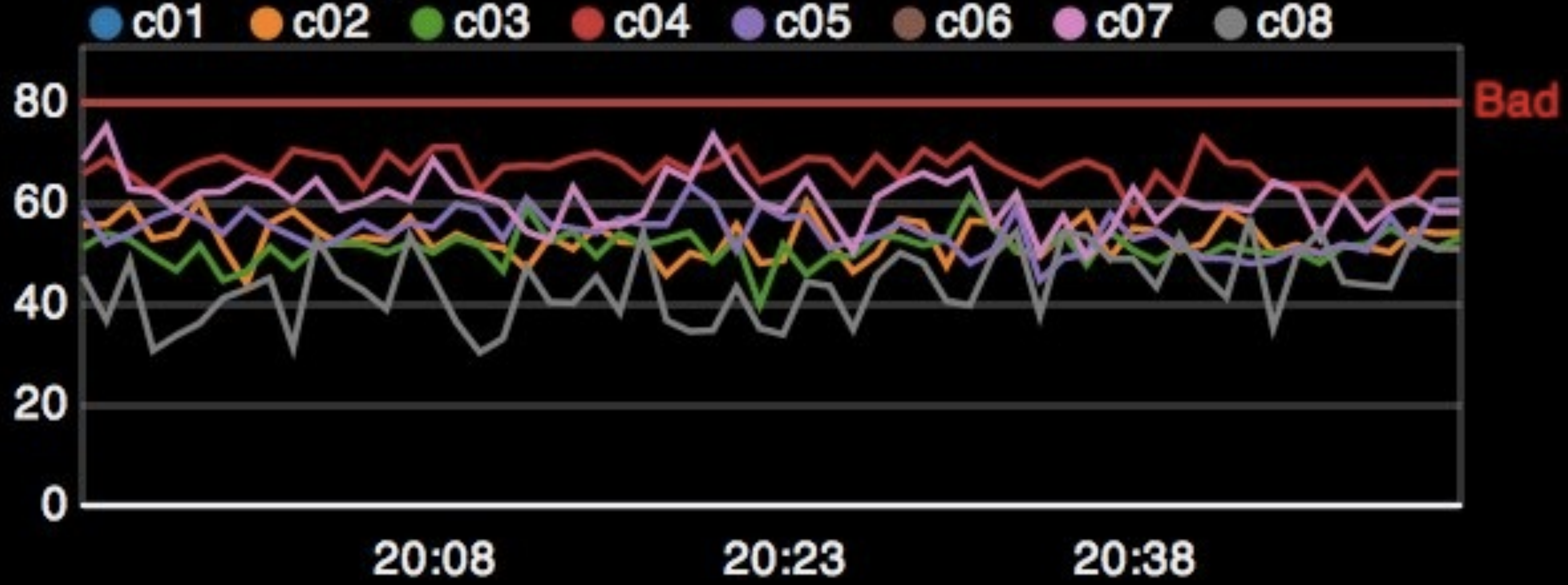
Process

Store

x Present

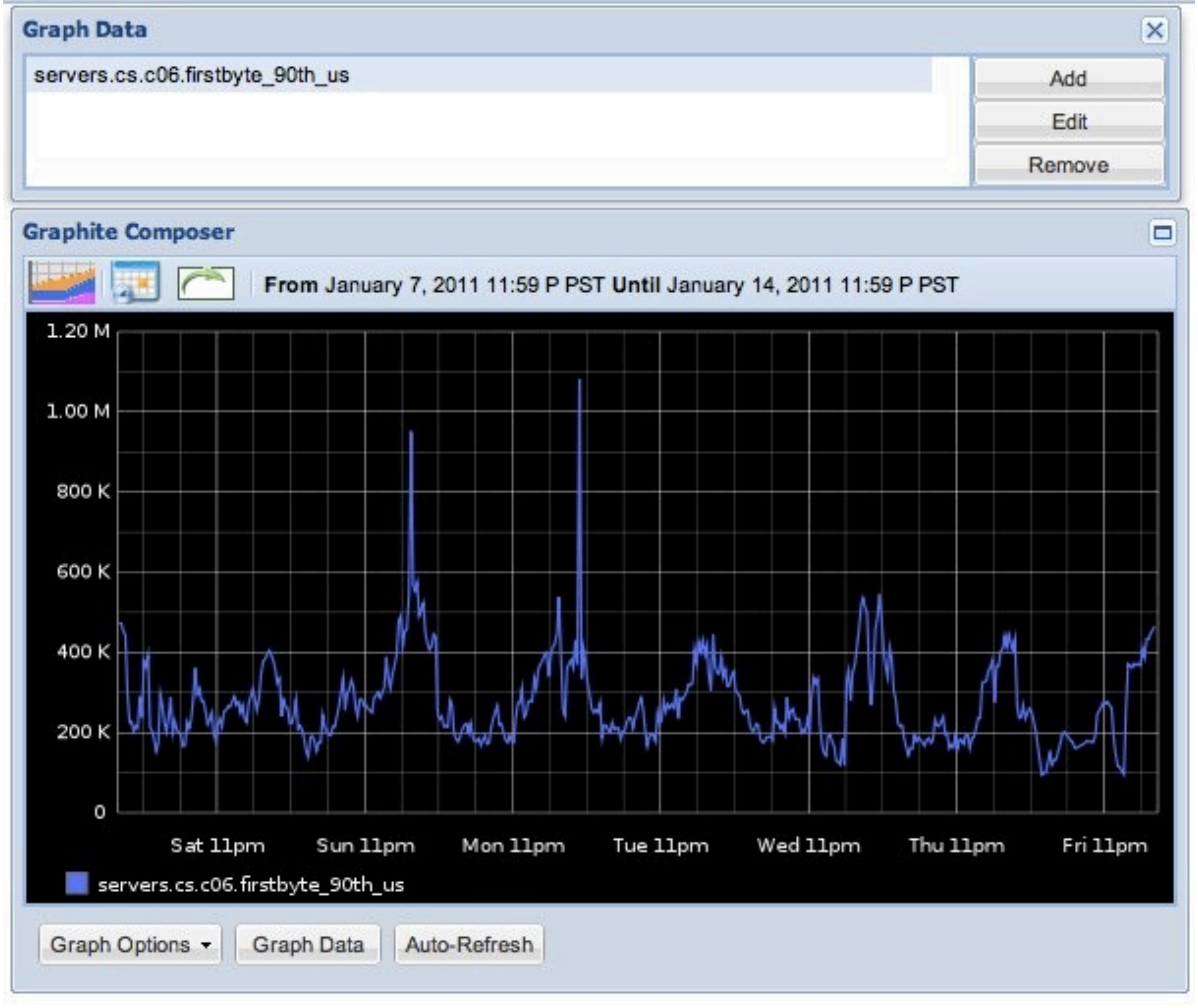
protovis

Web Node Avg CPU



PATTERNS

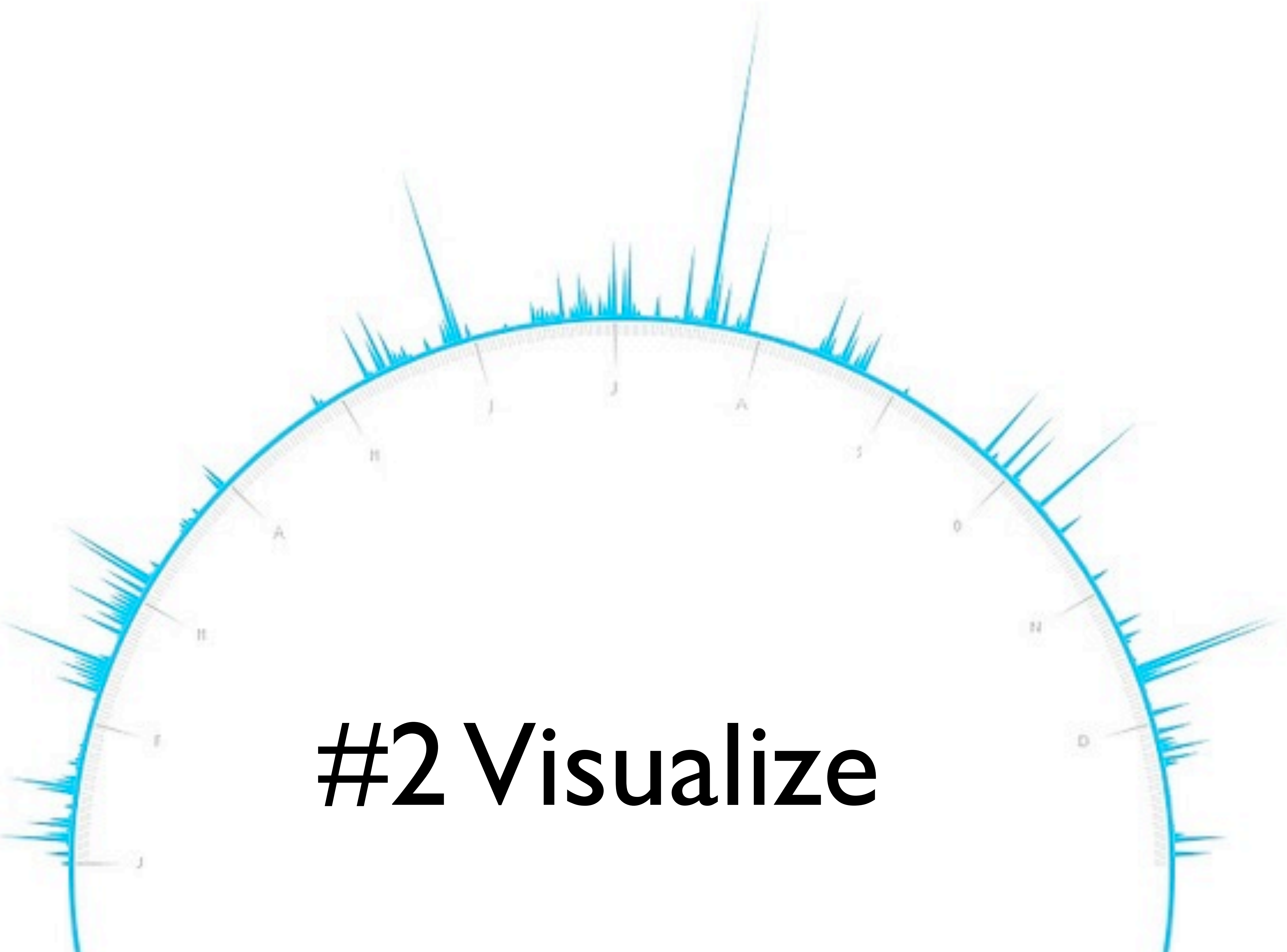
#1: Become the User



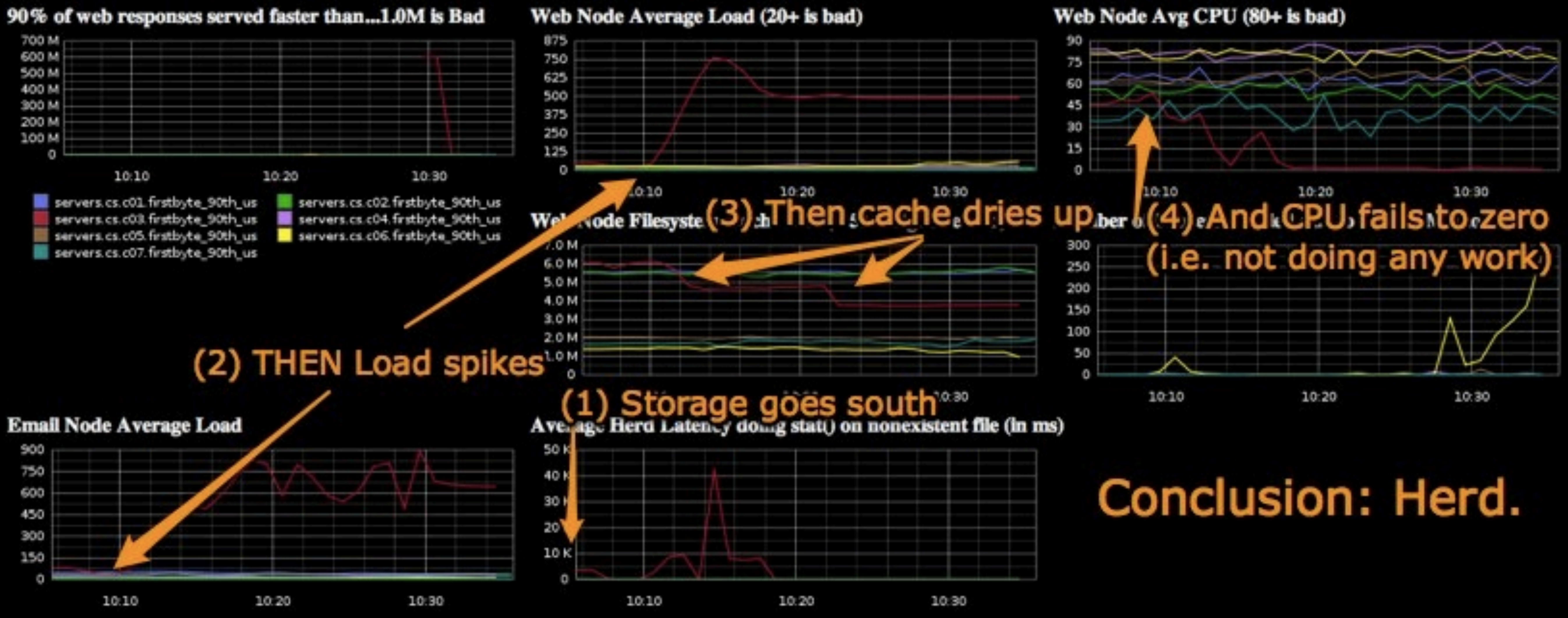
servers.cs.c06.firstbyte_90th_us

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

🟦 servers.cs.c06.firstbyte_90th_us



#2 Visualize


90% of web responses served faster than...1.0M is Bad
700 M
600 M
500 M
400 M
300 M
200 M
100 M
0
10:10
10:20
10:30
🟦 servers.cs.c01.firstbyte_90th_us
🟥 servers.cs.c03.firstbyte_90th_us
🟫 servers.cs.c05.firstbyte_90th_us
🟦 servers.cs.c07.firstbyte_90th_us
🟩 servers.cs.c02.firstbyte_90th_us
🟪 servers.cs.c04.firstbyte_90th_us
🟨 servers.cs.c06.firstbyte_90th_us
Web Node Average Load (20+ is bad)
875
750
625
500
375
250
125
0
10:10
10:20
10:30
Web Node Avg CPU (80+ is bad)
90
75
60
45
30
15
0
10:10
10:20
10:30
Web Node Filesyste
7.0 M
6.0 M
5.0 M
4.0 M
3.0 M
2.0 M
1.0 M
0
10:30
300
250
200
150
100
50
0
10:10
10:20
10:30
Email Node Average Load
900
750
600
450
300
150
0
10:10
10:20
10:30
Average Herd Latency doing stat() on nonexistent file (in ms)
50 K
40 K
30 K
20 K
10 K
0
10:10
10:20
10:30

(1) Storage goes south

(2) THEN Load spikes

(3) Then cache dries up

(4) And CPU fails to zero (i.e. not doing any work)

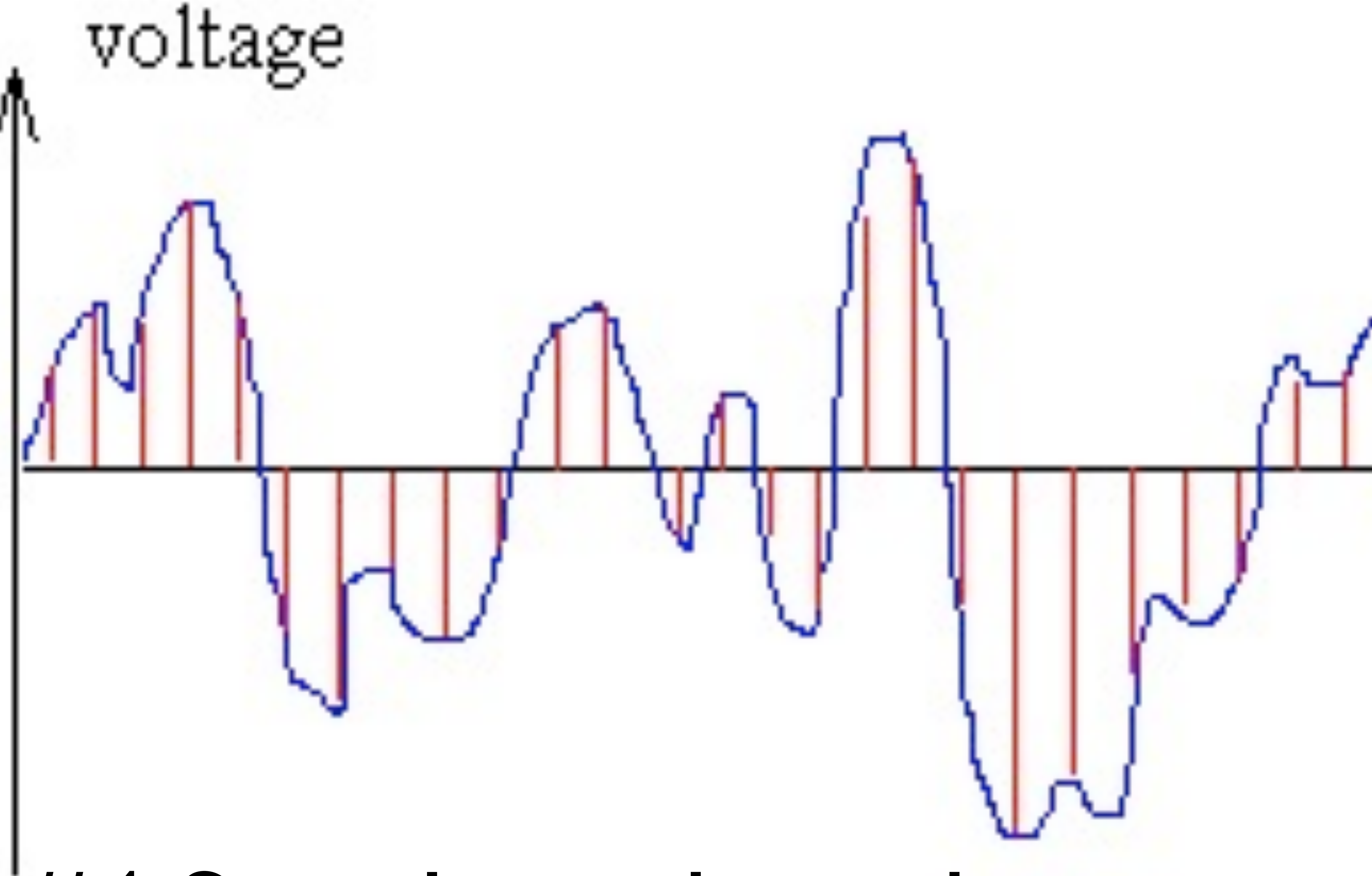
Conclusion: Herd.

#3 Prune near the Edge

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

syslog_web: Read <STDIN> forever and...

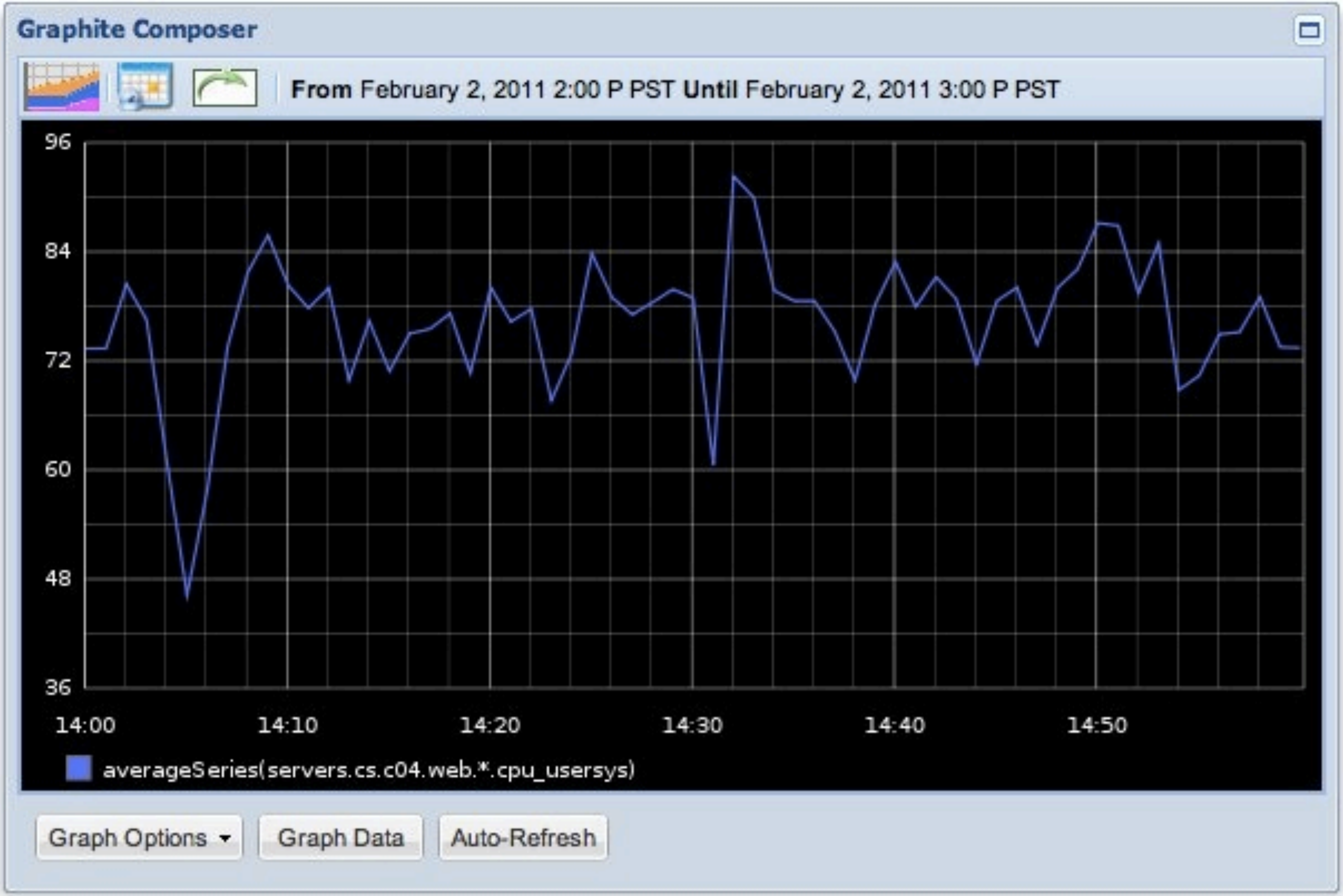
- Alert Nagios when Error Rate is too high
- Aggregate firstbyte data and stream to graphite
- Track top 100 users by
  - CPU
  - NFS I/O
  - hits/sec
  - Bandwidth In/Out



#4 Sample at the right rate

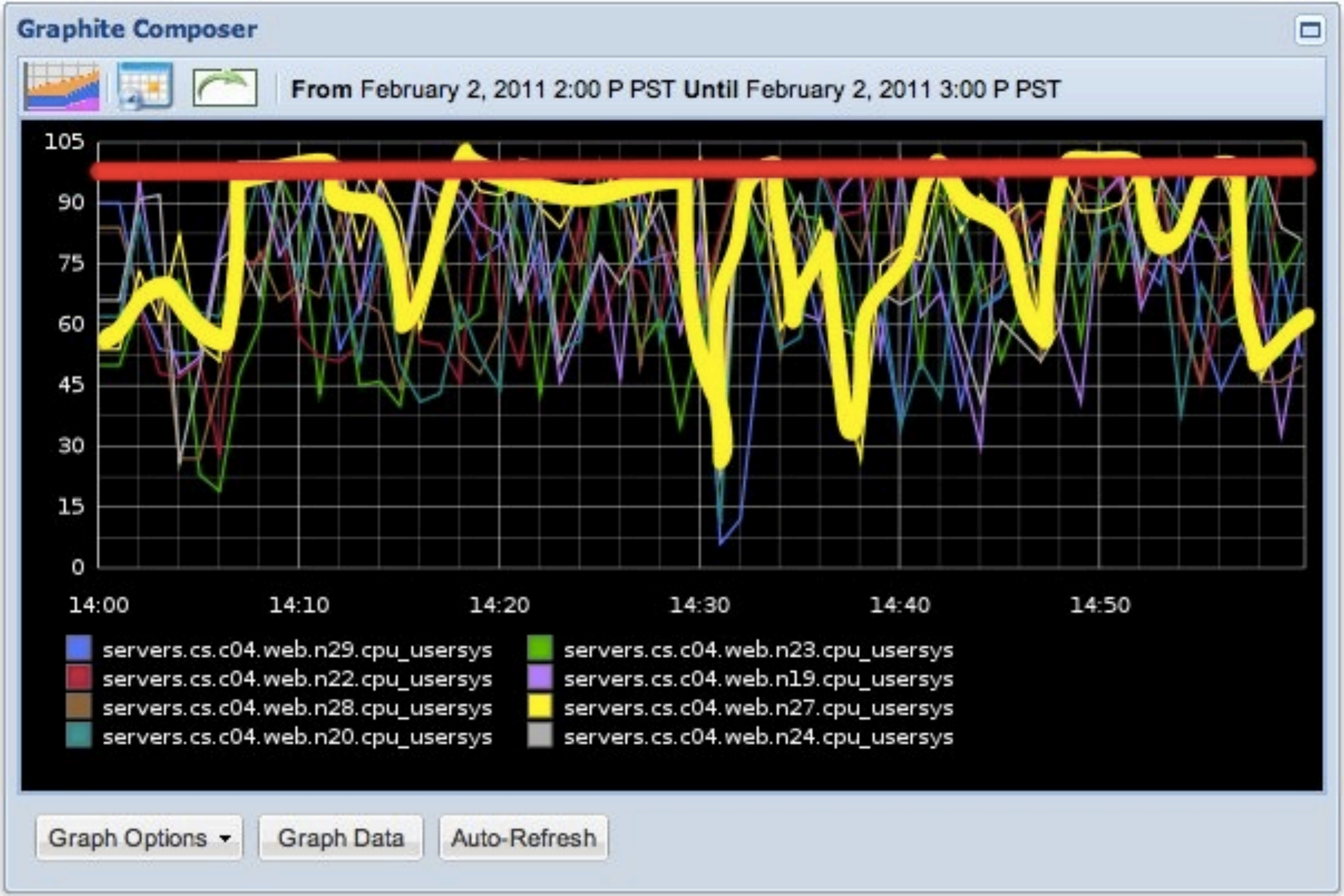
#5 Make it reliable

#6 Love (and Fear) Aggregates



Graphite Composer

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



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

#8 Learn Ad-hoc Skills

grep

awk

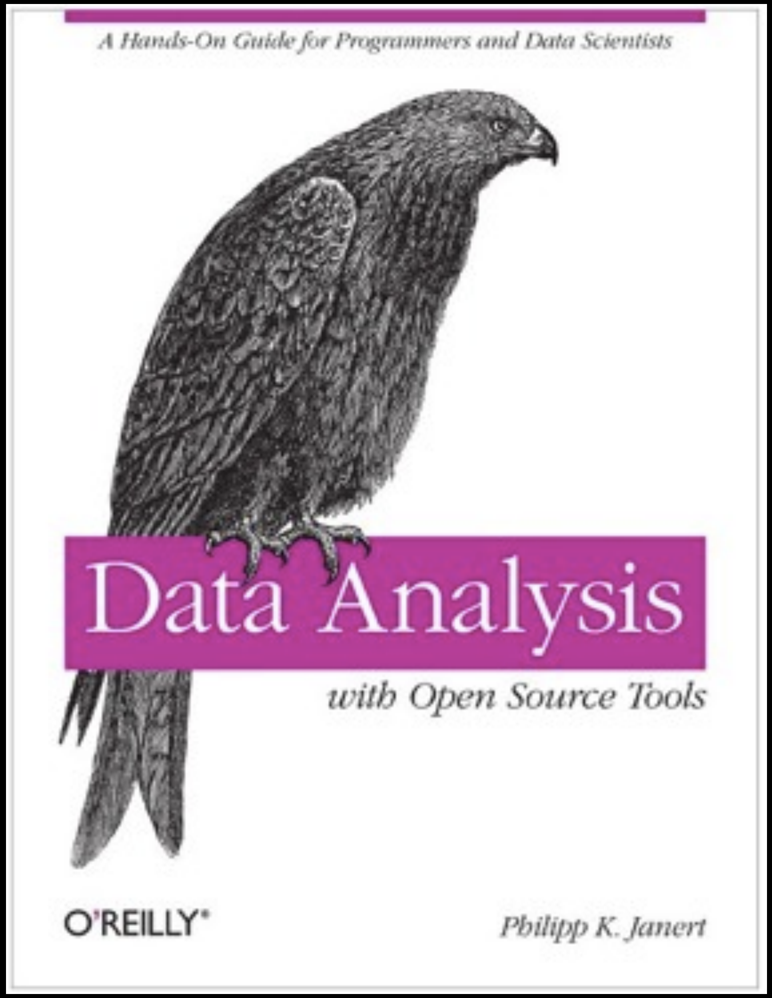
fex!: http://semicomplete.com/projects/fex/

bit.ly's 'data hacks':

https://github.com/bitly/data_hacks

gnuplot

R



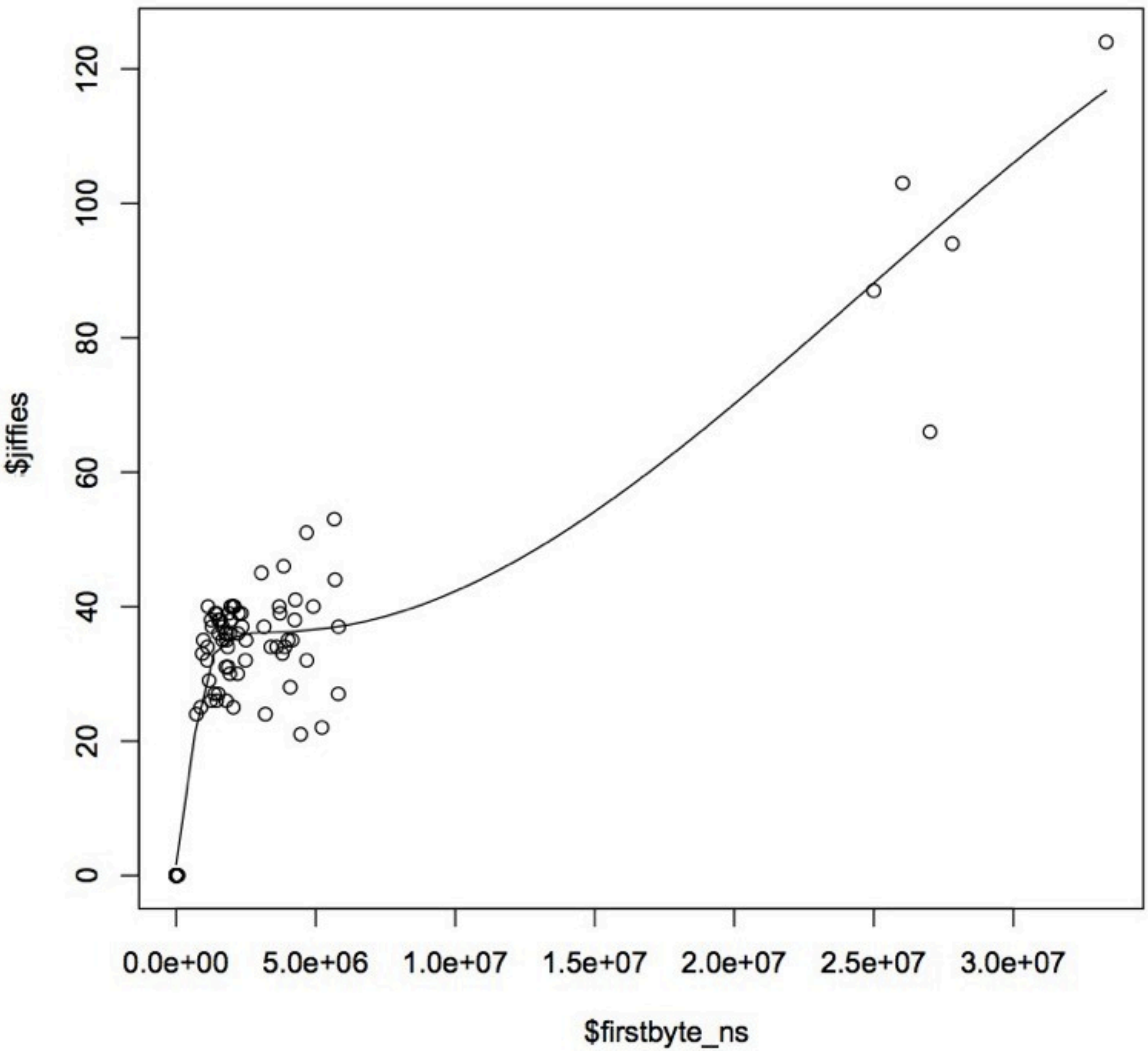
```
$ cat /tmp/data | histogram.py
# NumSamples = 29; Max = 10.00; Min = 1.00
# Mean = 4.379310; Variance = 5.131986; SD =
2.265389
# each * represents a count of 1
    1.0000 -     1.9000 [     1]: *
    1.9000 -     2.8000 [     5]: *****
    2.8000 -     3.7000 [     8]: ********
    3.7000 -    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

# @ the shell....

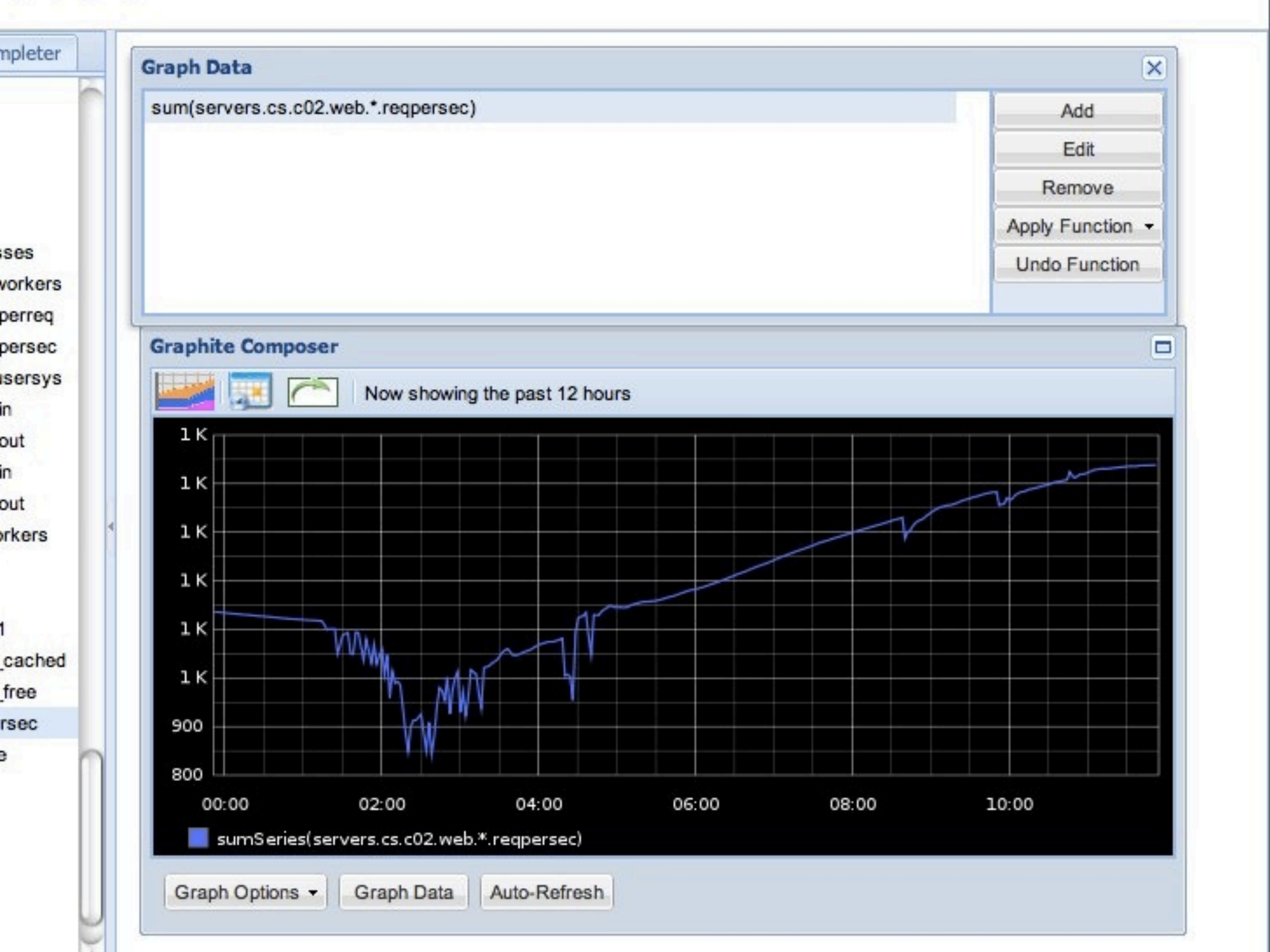
```
$ awk '{print $11,$14}' 10-access_log > /tmp/
jiffies_v_time.txt
```

# 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)
```


$jiffies
0
20
40
60
80
100
120
0.0e+00
5.0e+06
1.0e+07
1.5e+07
2.0e+07
2.5e+07
3.0e+07
$firstbyte_ns

#8 Sweat the Units



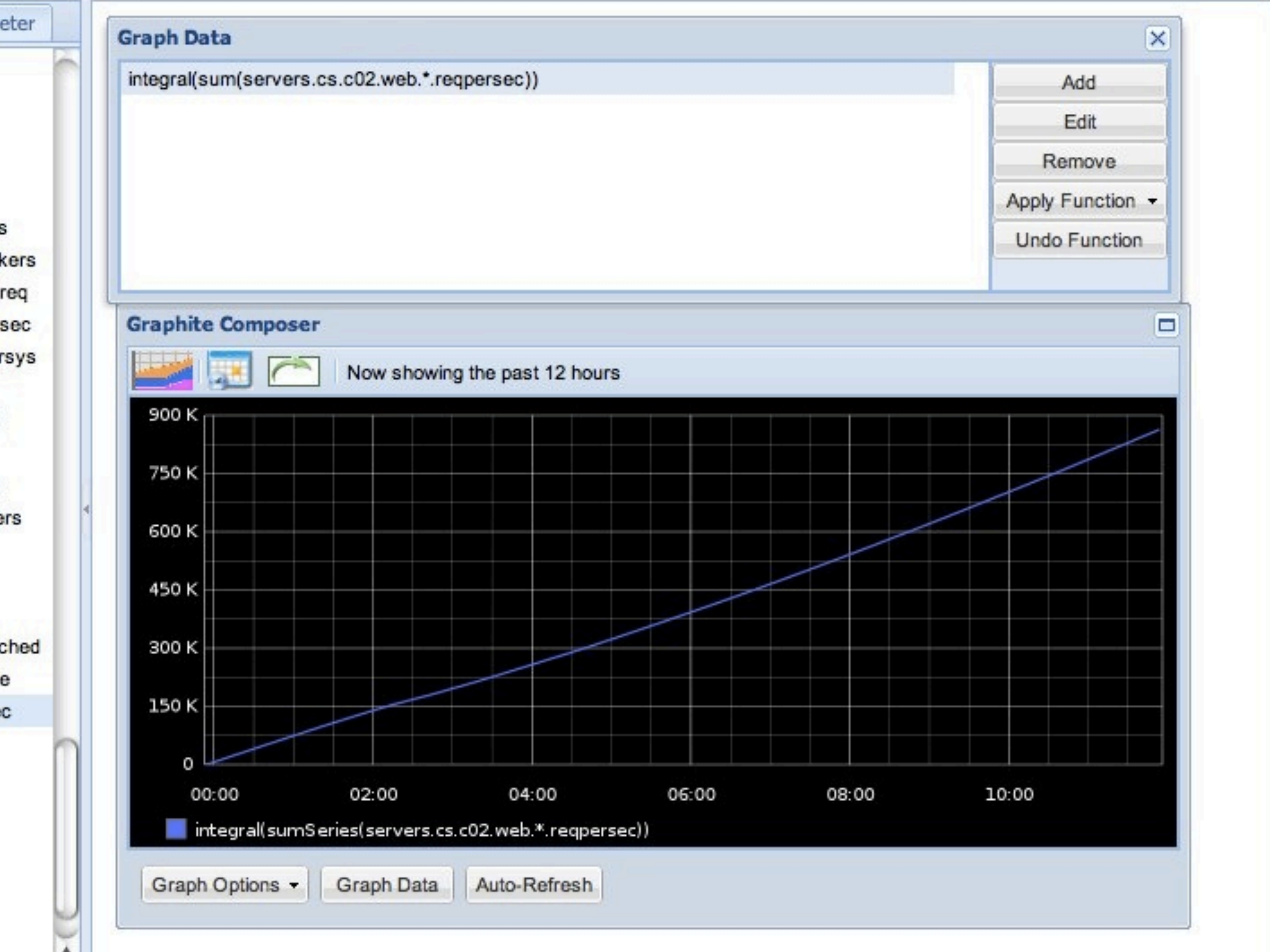
Graph Data

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

Graphite Composer

Now showing the past 12 hours

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

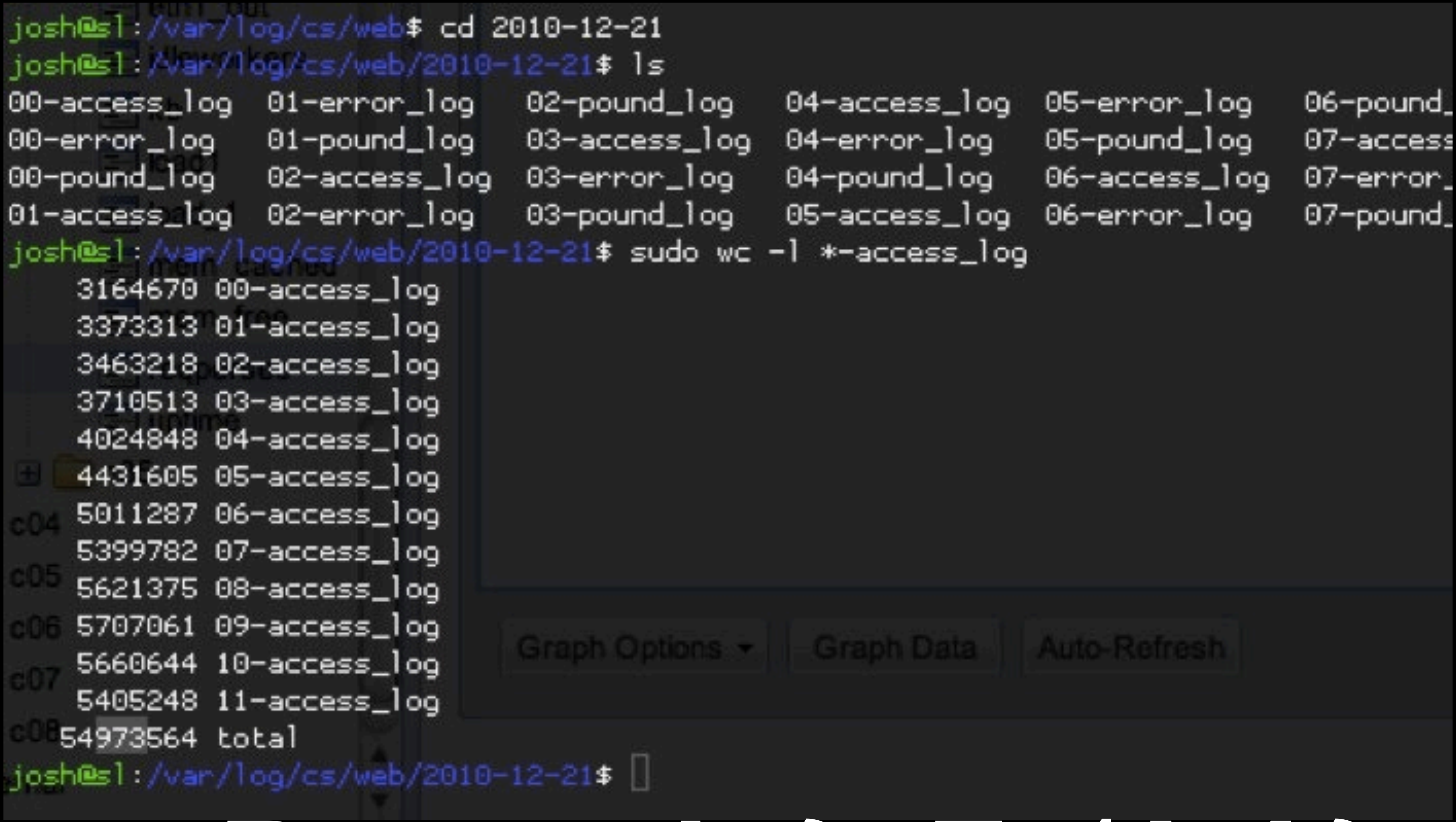


Graph Data

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

Graphite Composer

Now showing the past 12 hours



```
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?

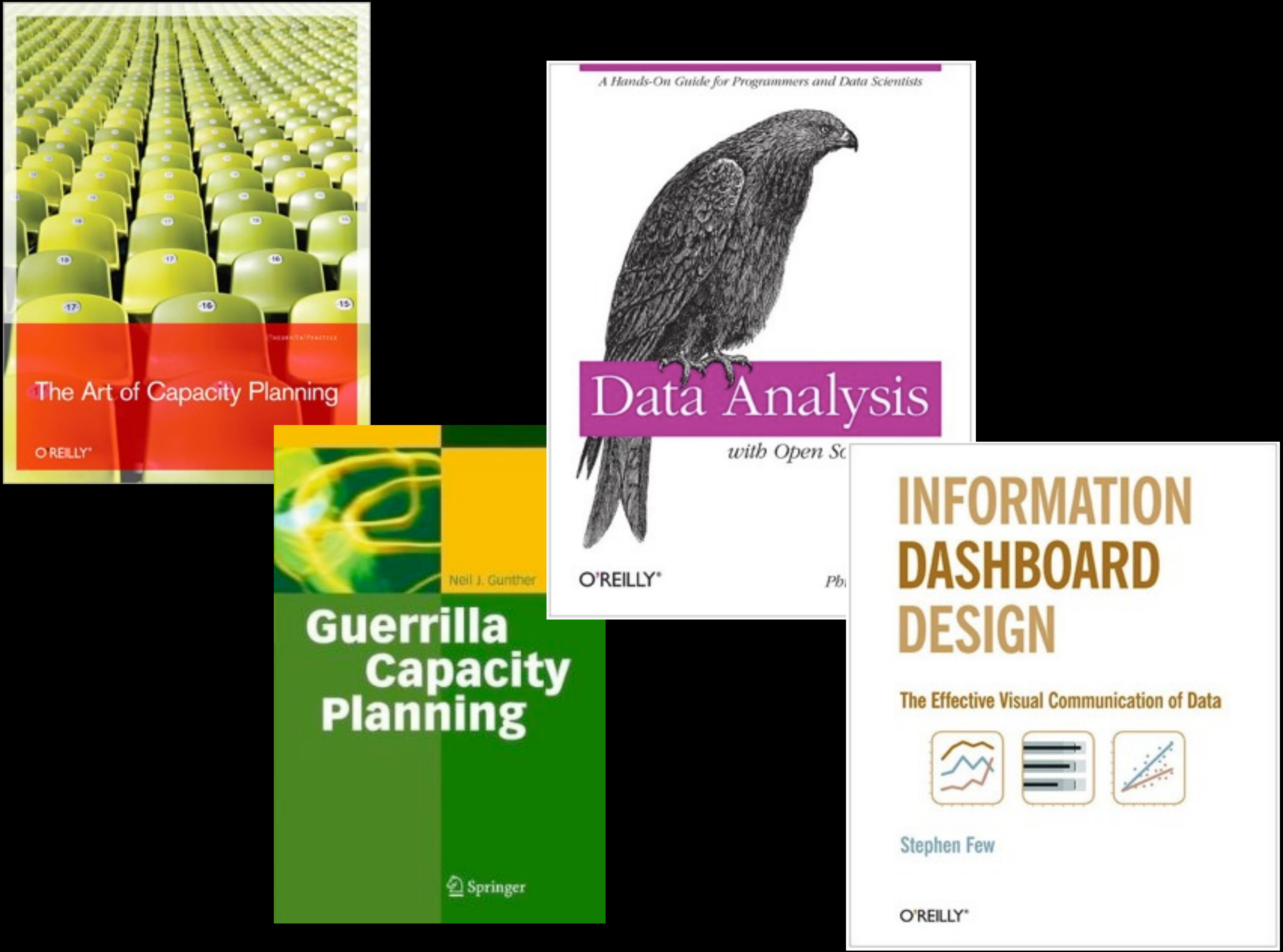
Requests/sec

Stored every minute

890k requests/sec

* 60 = 53.4M requests

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 Sc
O'REILLY
Ph
INFORMATION DASHBOARD DESIGN
The Effective Visual Communication of Data
Stephen Few
O'REILLY

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/