

The background features a complex, abstract pattern of overlapping, semi-transparent geometric shapes in various colors including yellow, green, blue, purple, red, and pink. These shapes radiate from a central point, creating a starburst or sunburst effect against a black background.

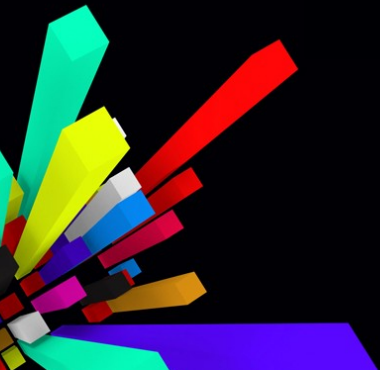
# Common Sense Performance Indicators

Nick Gerner  
June 24, 2010

# Goals

Common Sense in the Cloud  
same as outside the cloud

1. Tune performance
2. Investigate issues
3. Visualize architecture



# Nick Gerner

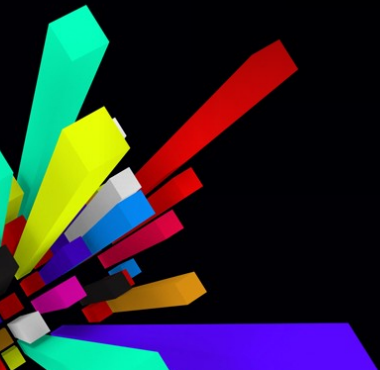
[www.nickgerner.com](http://www.nickgerner.com)  
@gerner

- Formerly senior engineer at SEOmoz
- Linkscape: index of the web for SEO
- Lead data services
- Developer
- Back-end ops guy



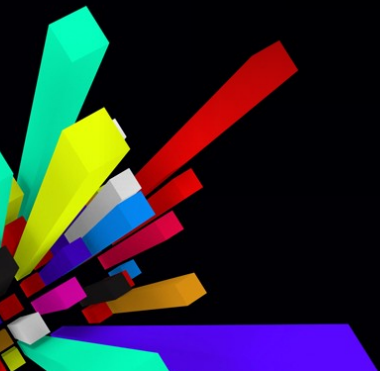
# SEOMoz

- Seattle-based Startup (~7 engineers)
- SEO Blog and Community
- Toolset and Platform  
OpenSiteExplorer.org
- 300TB/month processing pipeline
- 5 mil req/day API hits



# SEOmоз Engineering

- 50 < nodes < 500
- AWS based since 2008
  - EC2 – linux root access to bare VM
  - S3 – networked disk
  - EBS – local disk I/O
  - ELB – load balancing as a service



# SEOMoz Architecture Processing

The  
Web



Crawlers

Raw  
Storage

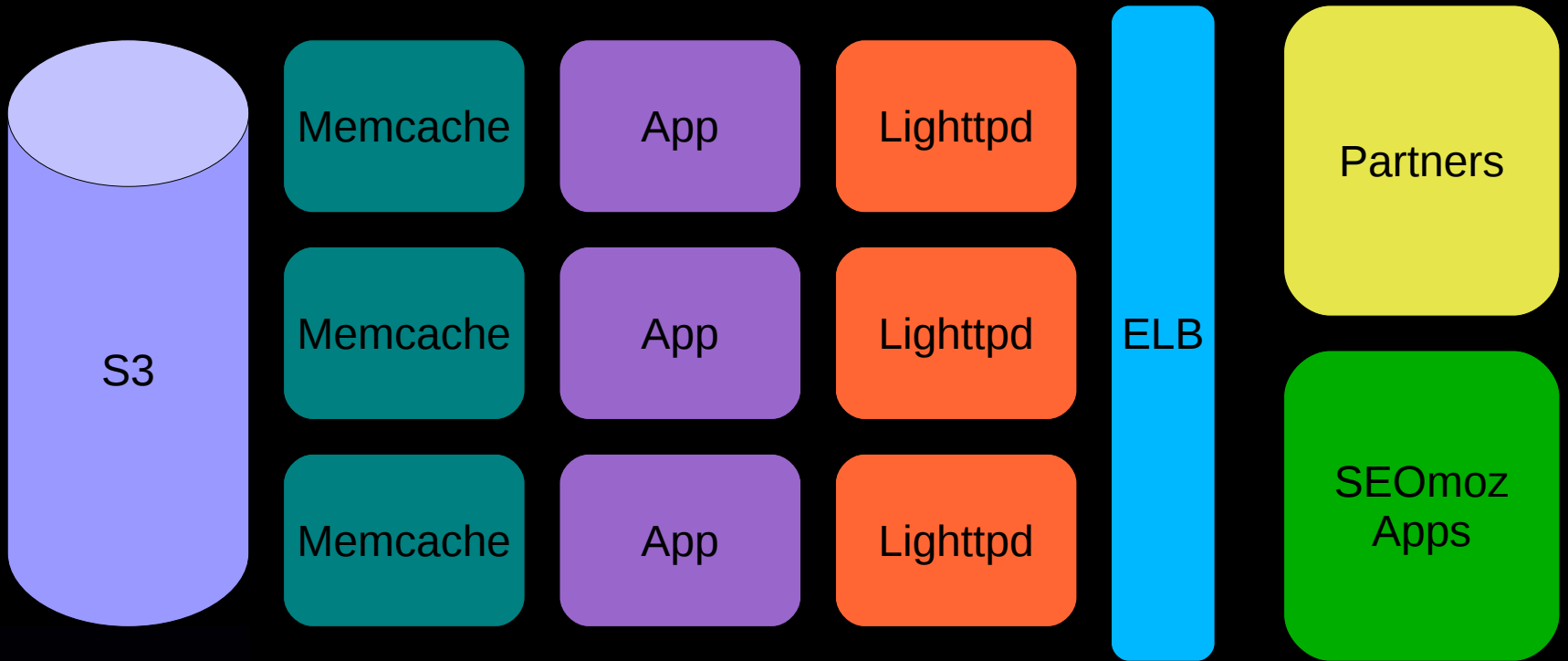
Process

Prepare

Data Pipeline



# SEOMoz Architecture API



# End-to-End Performance Indicators

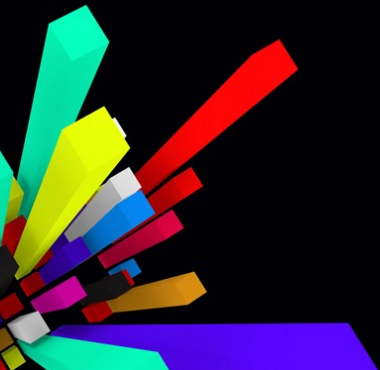
Latency

Conversion  
Rate

Time to  
On-load

DNS

Web  
Object  
Count



Great

...but not the focus of this talk

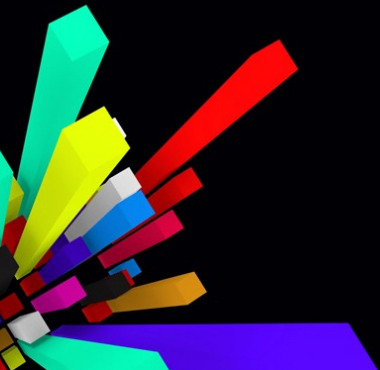
Latency

Conversion  
Rate

DNS

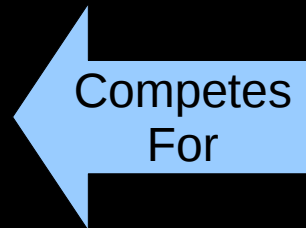
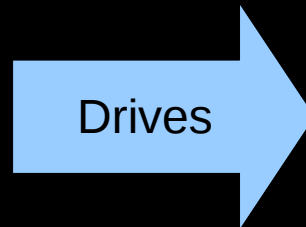
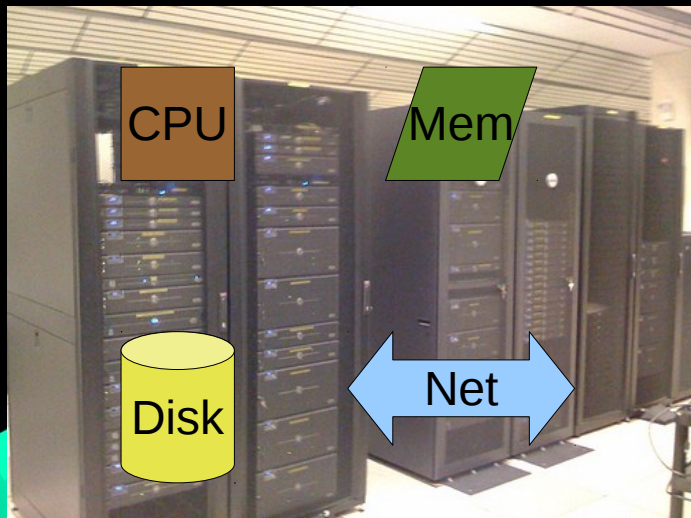
Time to  
On-load

Web  
Object  
Count

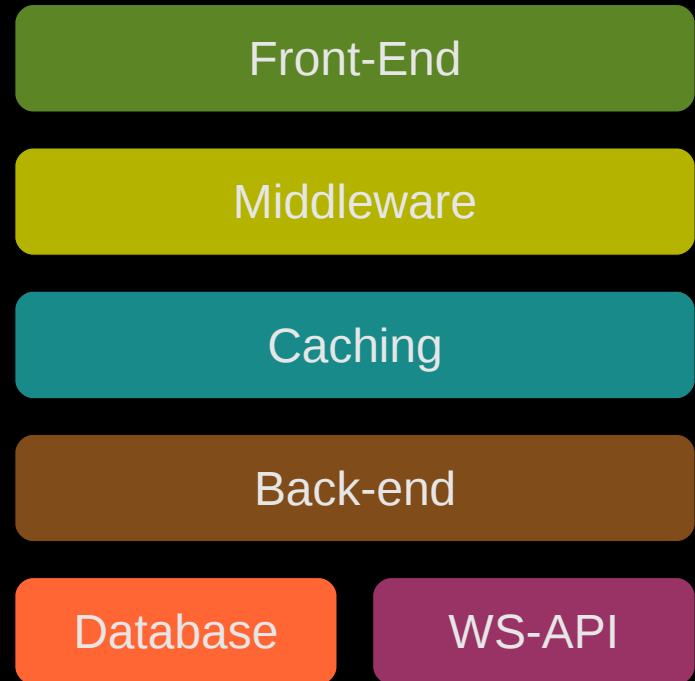


# Performance Indicators

## System Characteristics

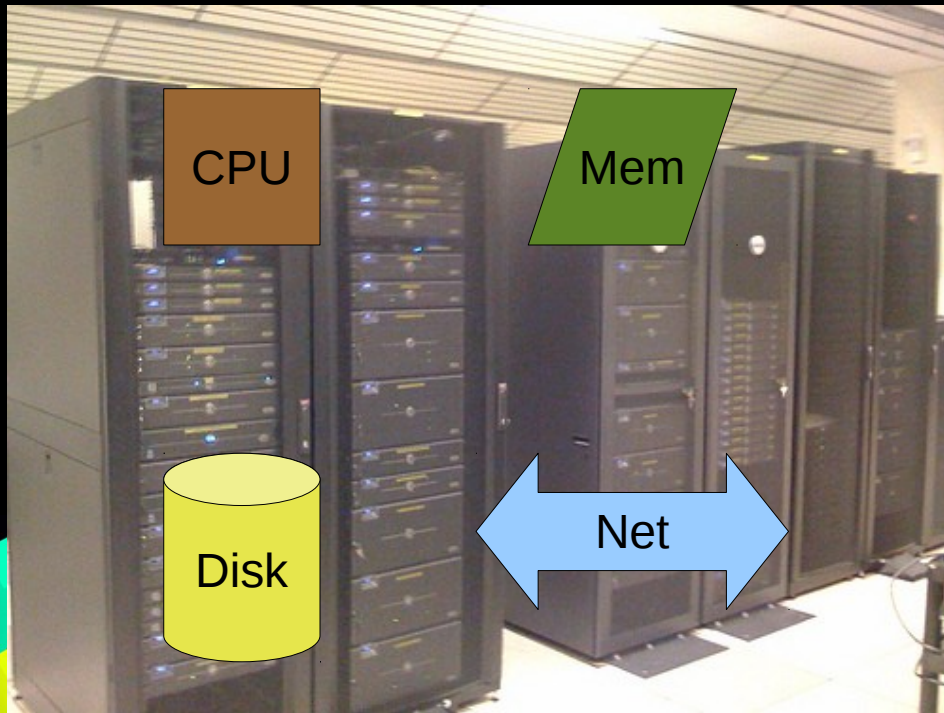


## App Stack

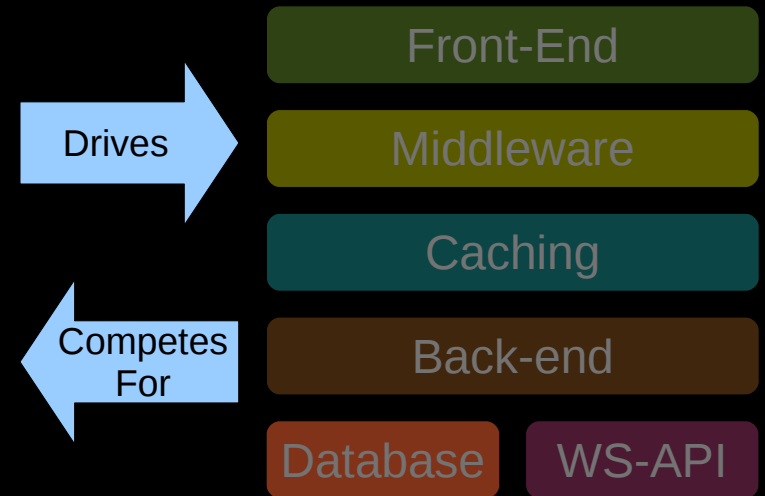


# Performance Indicators

## System Characteristics



## App Stack



PROC(5)

Linux Programmer's Manual

PROC(5)

**NAME**

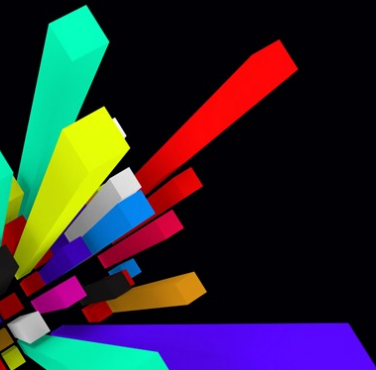
proc - process information pseudo-file system

**DESCRIPTION**

The proc file system is a pseudo-file system which is used as an interface to kernel data structures. It is commonly mounted at /proc. Most of it is read-only, but some files allow kernel variables to be changed.

# /proc

- System stats
- Per-process stats
- It all comes from here  
...but use tools to see it



# System Characteristics

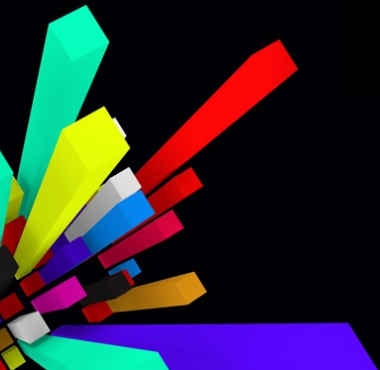
Load Average

CPU

Memory

Disk

Network

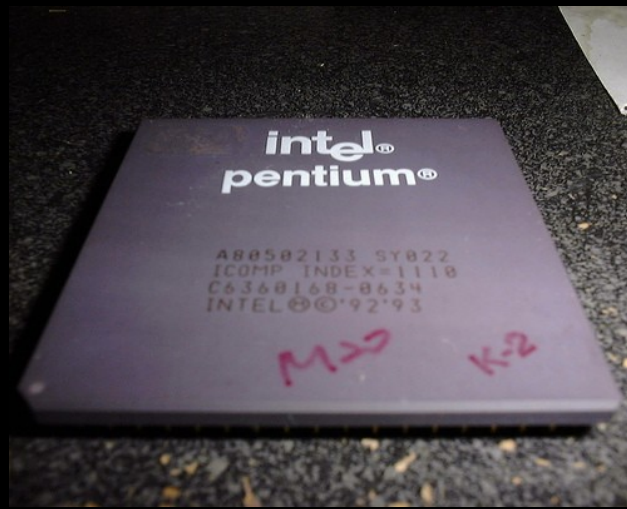


```
er, load average: 800.64, 794.71, 782.
eping, 0 stopped, 1 zombie
id, 11.1%wa, 2.3%hi, 2.3%si, 0.0%st
87760k free, 164k buffers
0k free, 3028k cached
```

%CPU	%MEM	TIME	COMMAND
23.4	1.1	497:50.09	mysqld
17.8	0.0	4:26.40	kswapd0
14.3	0.1	0:27.05	sshd
6.4	0.8	0:04.35	top

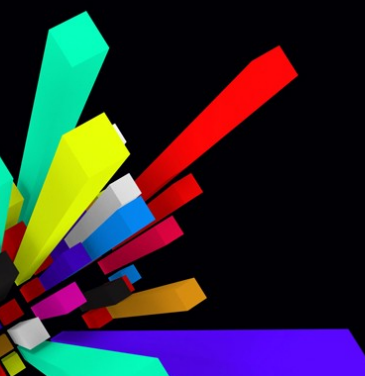
# Load Average

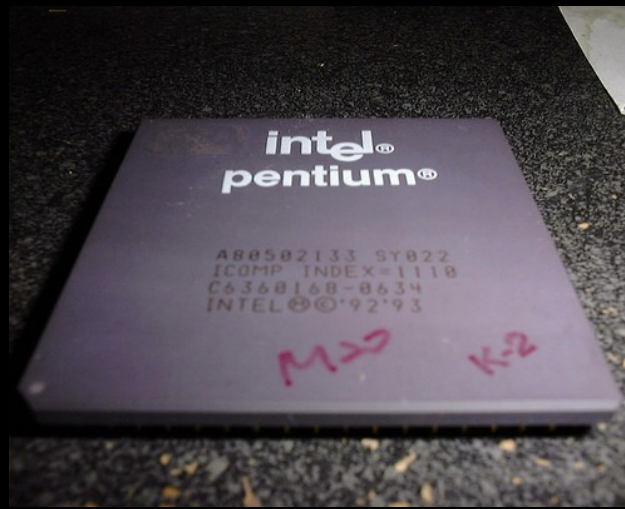
- Combines a few things
- Good place to start
- Explains nothing



# CPU

- Break out by process
- Break out user vs system
- User, System, I/O wait, Idle





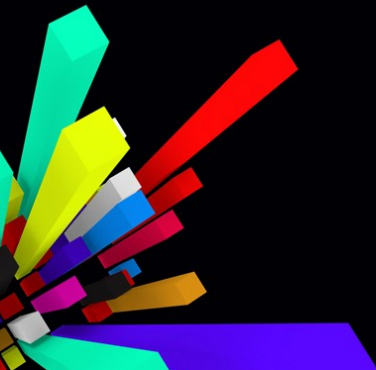
## Why watch it?

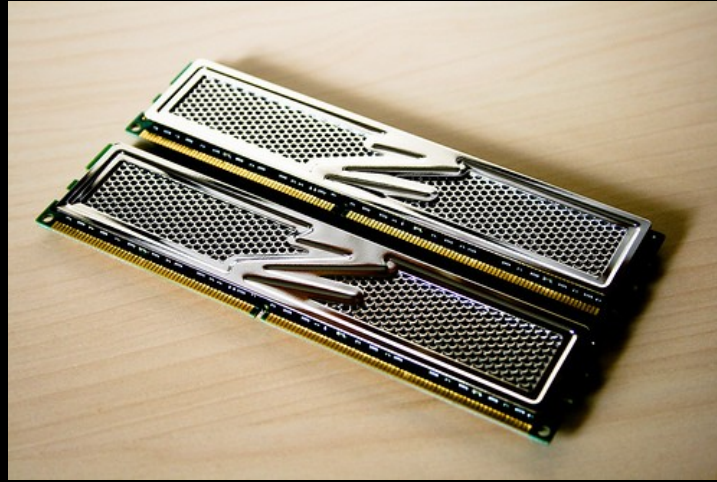
- Who's doing work
- Is CPU maxed?
- Blocked on I/O?
- Compare to Load Average



# Memory

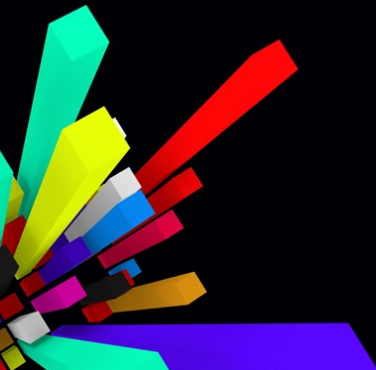
- Break out by Process
- Free, cached, used

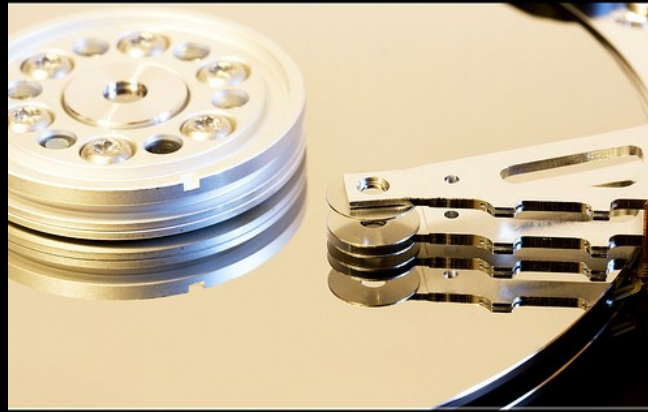




## Why watch it?

- Cached + Free = Available
- Do you have spare memory?
  - App uses
  - Memcache
  - DB cache

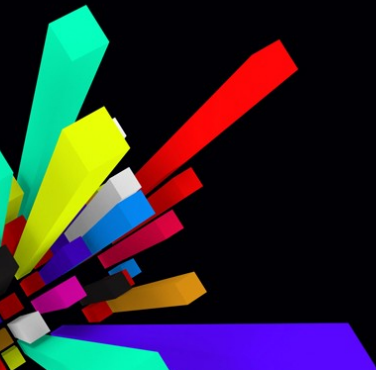


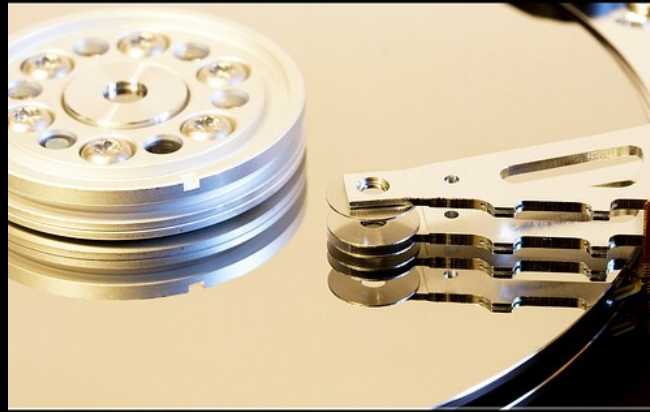


© 2008 Roberto F.

# Disk

- Read bytes/sec
- Write bytes/sec
- Disk utilization

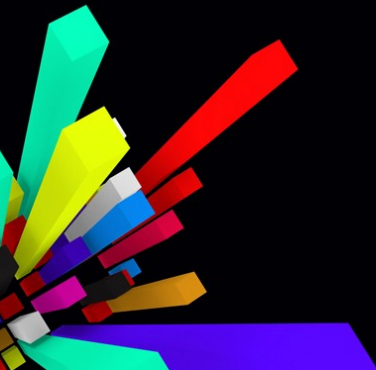


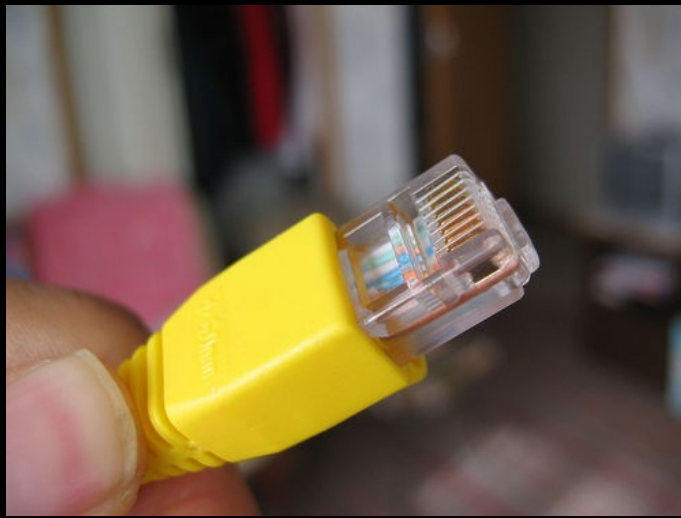


© 2008 Roberto F.

# Why watch it?

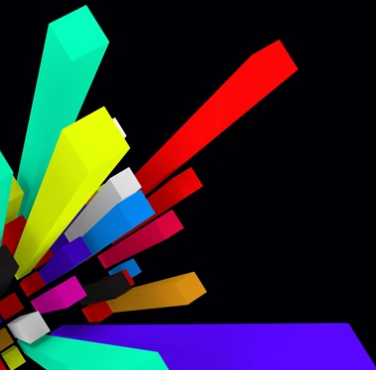
- Is disk busy?
- When?
- Who's using it?

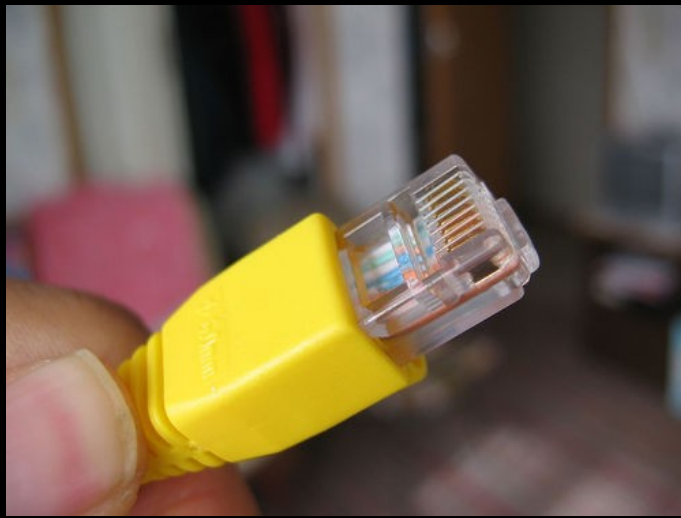




# Network

- Read bytes/sec
- Write bytes/sec
- Established connections





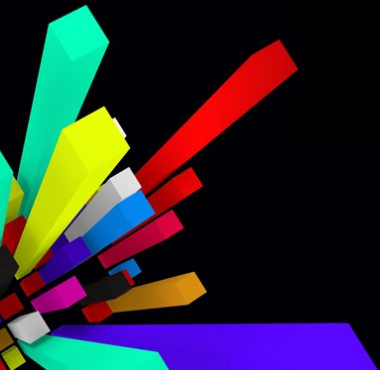
## Why watch it?

- Max connections  
(~1024 is magic)
- Bandwidth is \$\$\$
- When are you busy?
- SOA considerations

# √ Perf Monitoring Solution

FREE, in Apt

1. data collection (collectd)
2. data storage (rrdtool)
3. dashboard management (drraw)



# Perf Monitoring Architecture

Multiple Clusters

Multiple Applications

Nodes come up  
and go down



Cluster

Cluster

# Perf Monitoring Architecture



The diagram illustrates the Perf Monitoring Architecture. It features two light blue cloud shapes, each labeled 'Cluster'. A grey rectangular box on the right contains text describing the 'collectd agents' and their configuration. A thin grey line points from the top of the box to the right-hand cluster. In the bottom-left corner, there is a colorful, abstract graphic consisting of overlapping, semi-transparent shapes in shades of red, yellow, green, blue, and purple.

Cluster

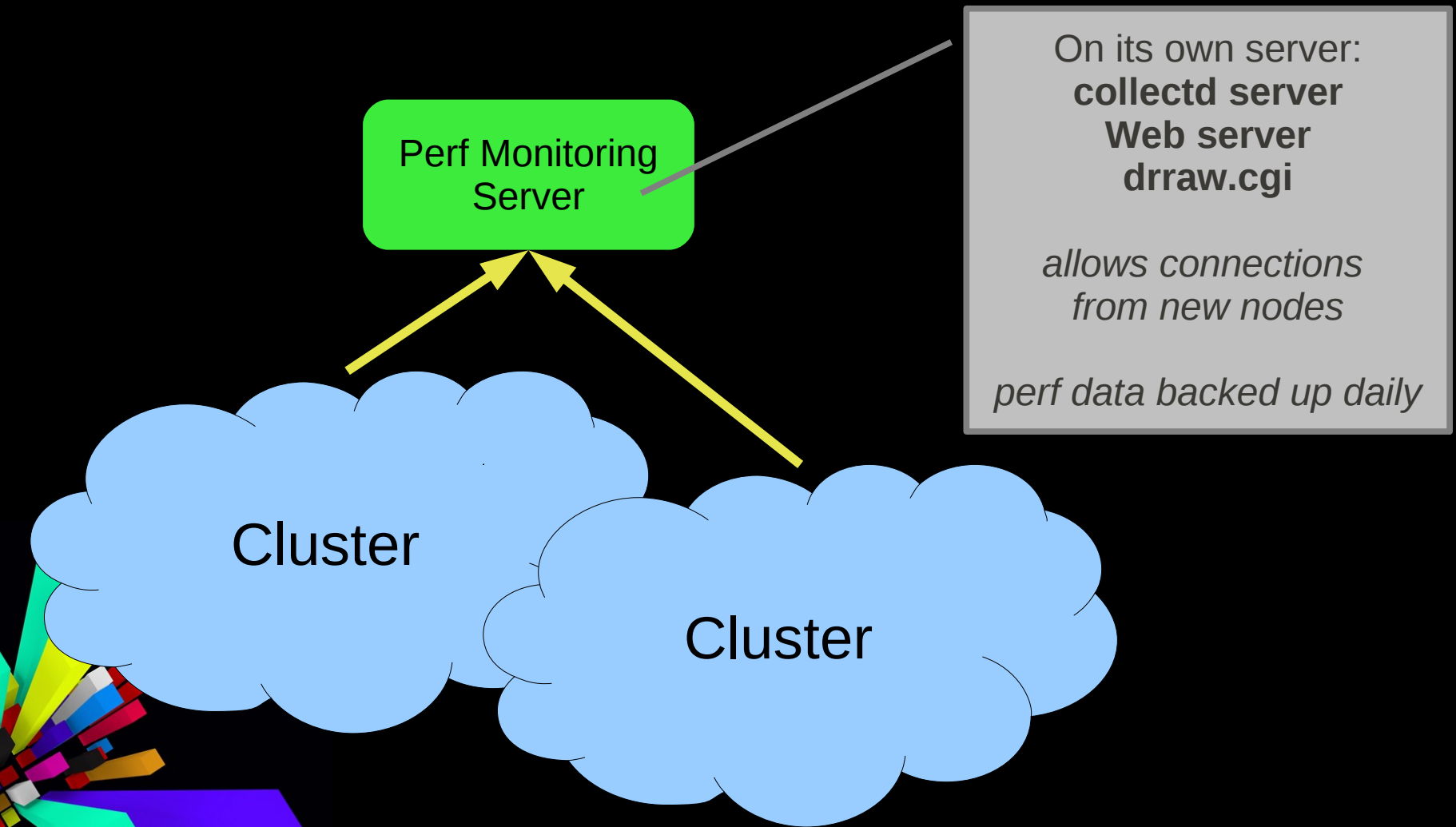
Cluster

**collectd agents**

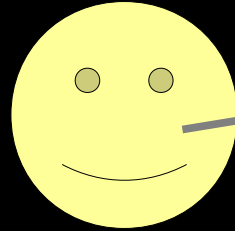
*new nodes get  
generic config*

*node names  
follow convention  
according to role*

# Perf Monitoring Architecture



# Perf Monitoring Architecture



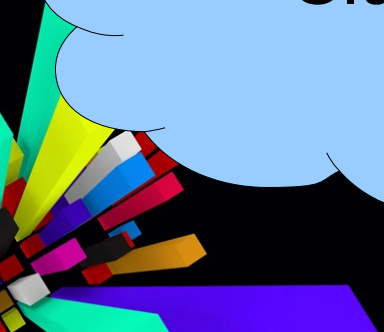
Happy Sysadmin

*Visibility into system  
history of performance*

Perf Monitoring  
Server

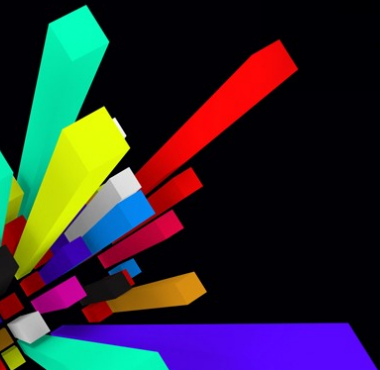
Cluster

Cluster

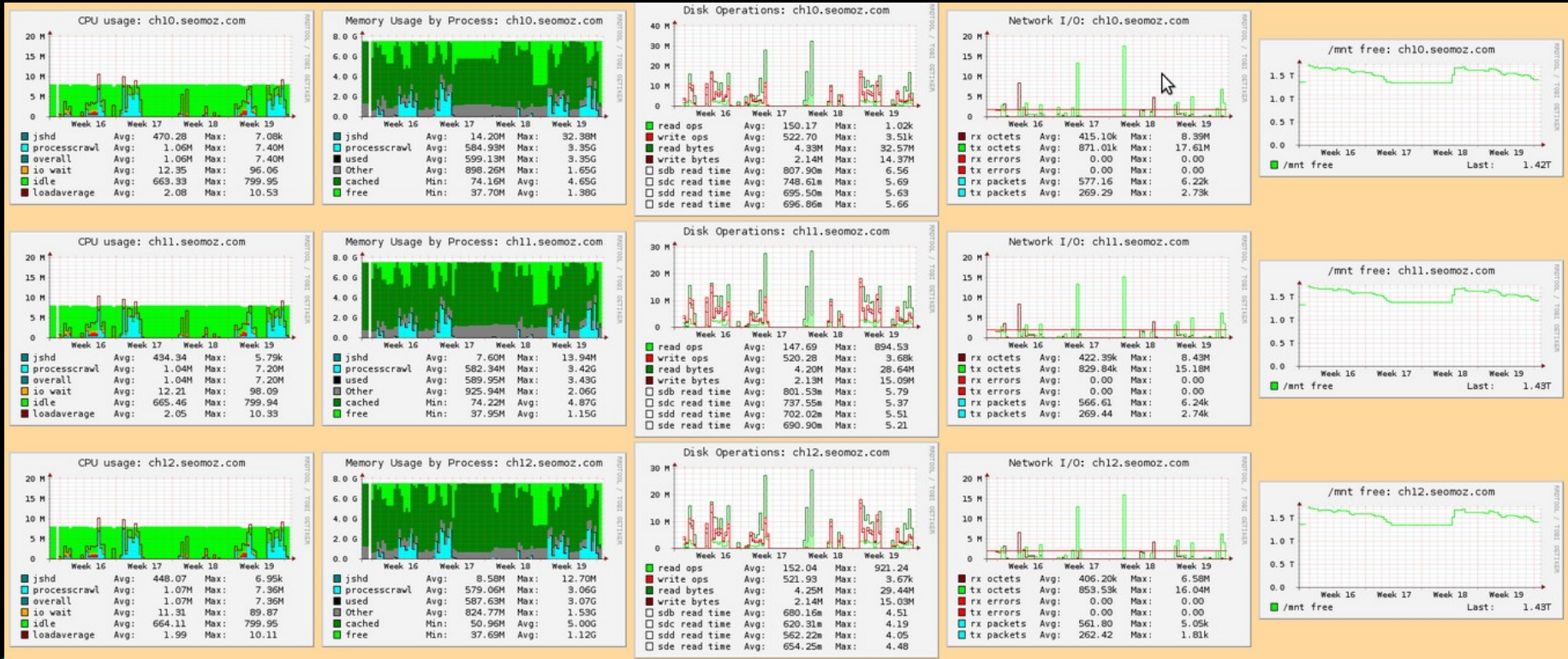


# Perf Dashboard Featurs

1. **Summarize** nodes/systems
2. Visualize data **over time**
3. **Stack** measurements
  - Per-process
  - Per-node
4. Handle **new nodes**

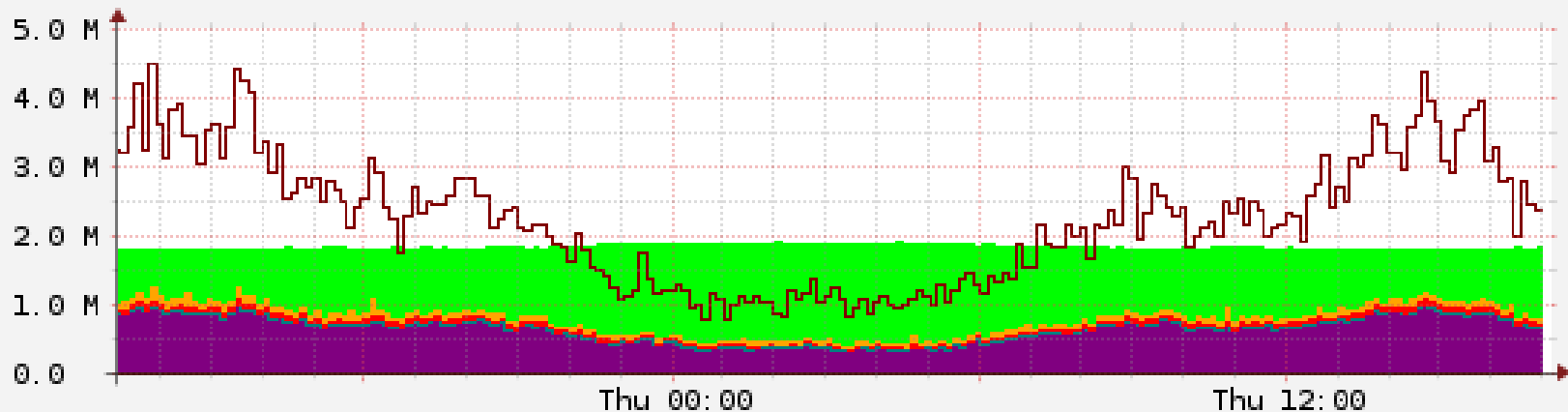


# Batch Mode Dashboard



# CPU

CPU usage: lsapi10.seomoz.com

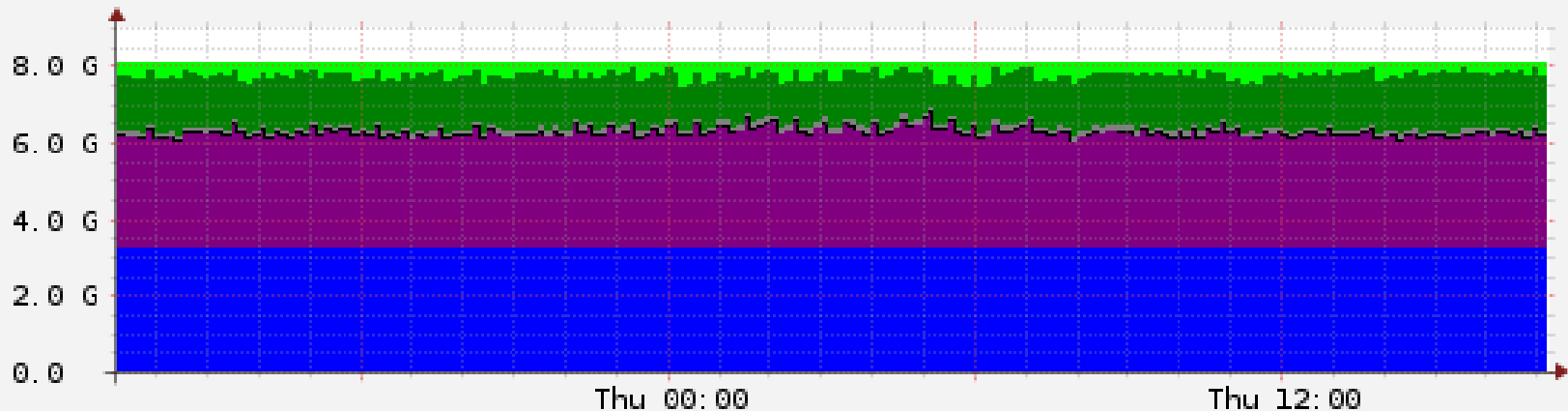


RRDTOOL / TOBI OETIKER

collectd	Avg:	1.02k	Max:	1.89k
lighttpd	Avg:	3.28k	Max:	5.82k
memcached	Avg:	1.24k	Max:	3.34k
processcrawl	Avg:	99.50m	Max:	20.00
requestserver.f	Avg:	640.55k	Max:	977.10k
overall	Avg:	646.09k	Max:	986.30k
io wait	Avg:	8.14	Max:	22.63
idle	Avg:	103.31	Max:	151.05
loadaverage	Avg:	2.25	Max:	4.51

# Memory

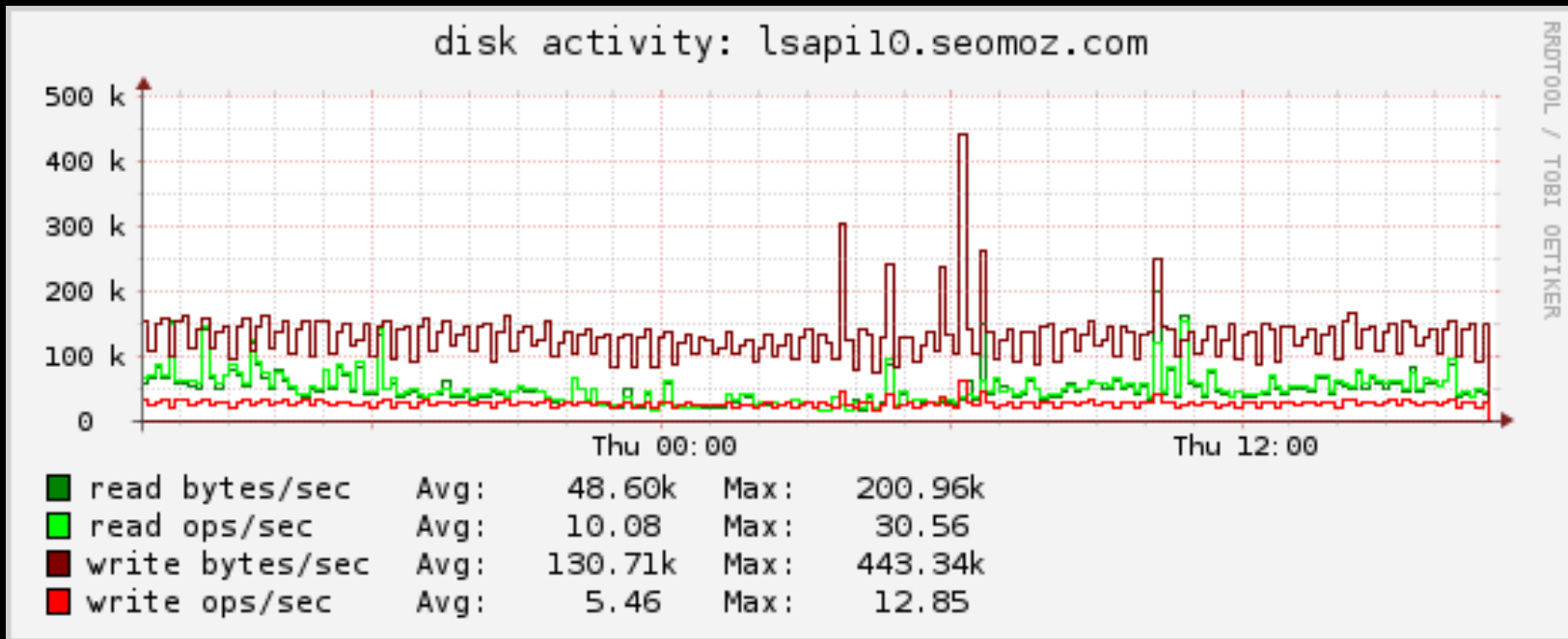
Memory Usage by Process: lsapi10.seomoz.com



RRDTOOL / TOBI OETIKER

collectd	Avg:	4.08M	Max:	4.09M
lighttpd	Avg:	8.55M	Max:	8.68M
memcached	Avg:	3.23G	Max:	3.23G
processcrawl	Avg:	1.66k	Max:	327.84k
requestserver.f	Avg:	3.06G	Max:	3.58G
used	Avg:	6.31G	Max:	6.82G
Other	Avg:	97.95M	Max:	133.86M
cached	Min:	1.00G	Avg:	1.35G
free	Min:	69.75M	Avg:	308.99M

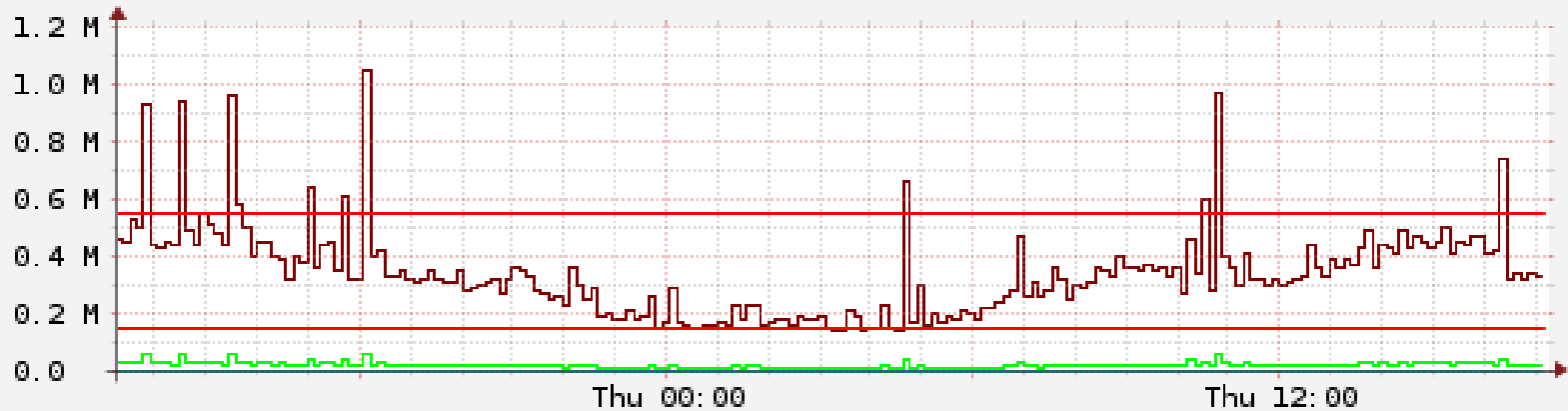
# Disk



# Network

Network I/O: lsapi10.seomoz.com

RRDTOOL / TOBI OETIKER



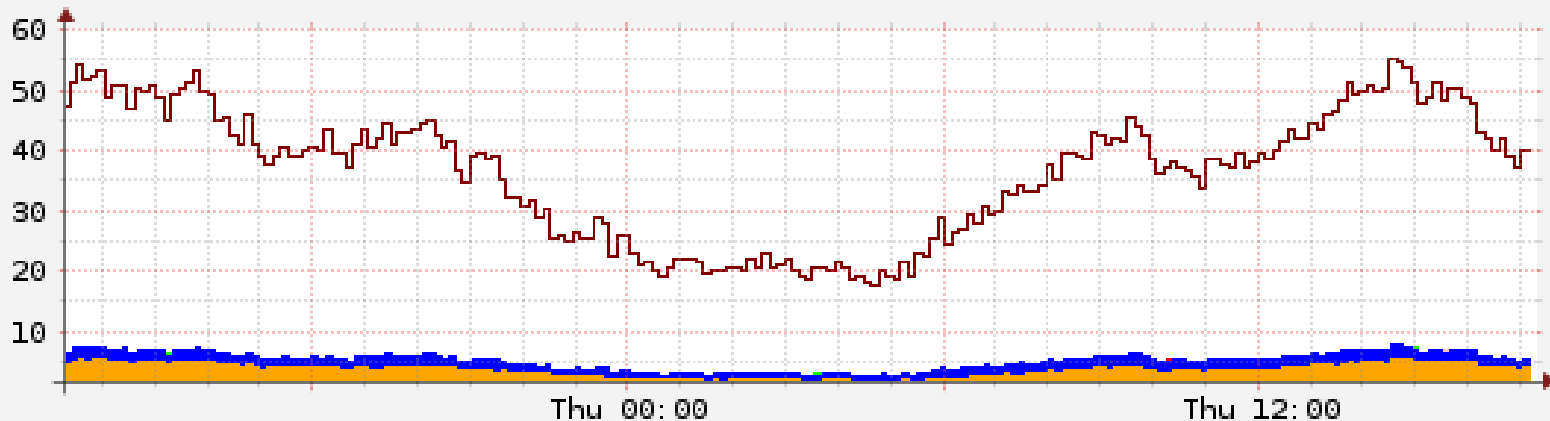
rx octets	Avg:	340.47k	Max:	1.05M
tx octets	Avg:	20.64k	Max:	58.70k
rx errors	Avg:	0.00	Max:	0.00
tx errors	Avg:	0.00	Max:	0.00
rx packets	Avg:	267.12	Max:	797.58
tx packets	Avg:	208.70	Max:	619.48

# Web Server Dashboard



# Web Requests

Requests Overall: lsapi10.seomoz.com



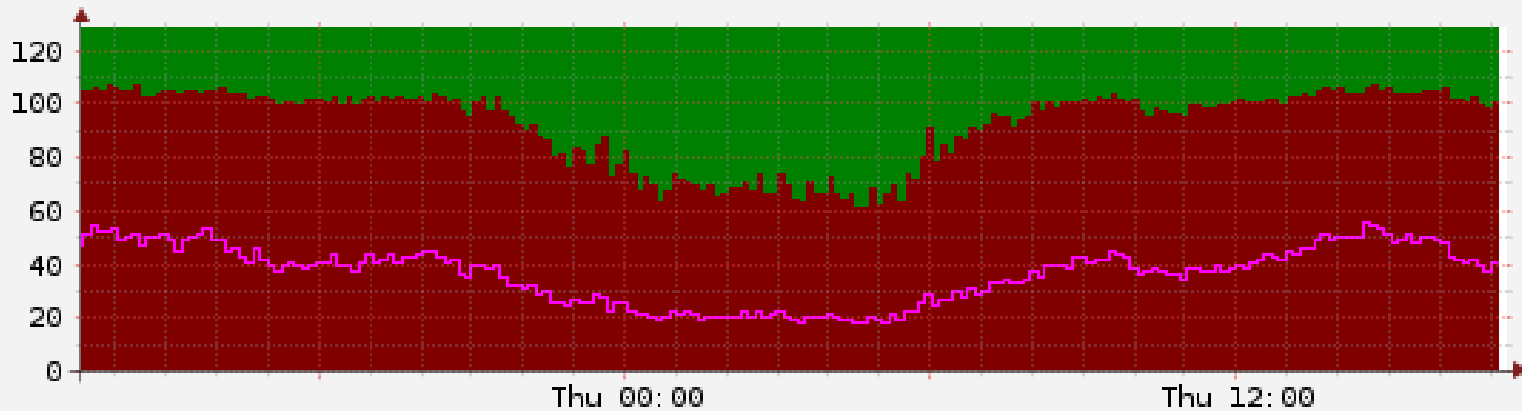
RAOTOOL / TOBI OETIKER

	Avg:	Max:	Last:
url-metrics	3.45	5.42	4.02
url-inlinks	388.06u	8.00m	0.00
pld-inlinks	19.90u	2.00m	0.00
fqdn-inlinks	69.65u	2.00m	0.00
toolbar	0.00	0.00	0.00
free-toolbar	69.65u	4.00m	0.00
mozrank	1.54	2.26	1.51
page-inlinks	69.65u	2.00m	0.00
302	0.00	0.00	0.00
500	119.40u	8.00m	0.00
503	89.55u	18.00m	0.00
all	36.40	54.96	40.11
<b>Total</b>	<b>3.66M</b>		

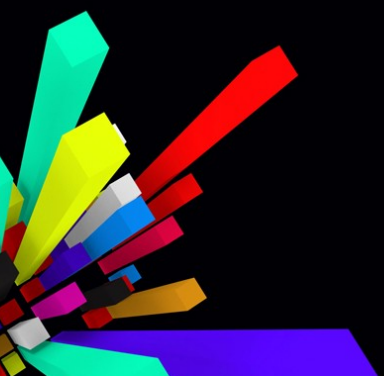
# mod\_status

Webserver Connections: lsapi10.seomoz.com

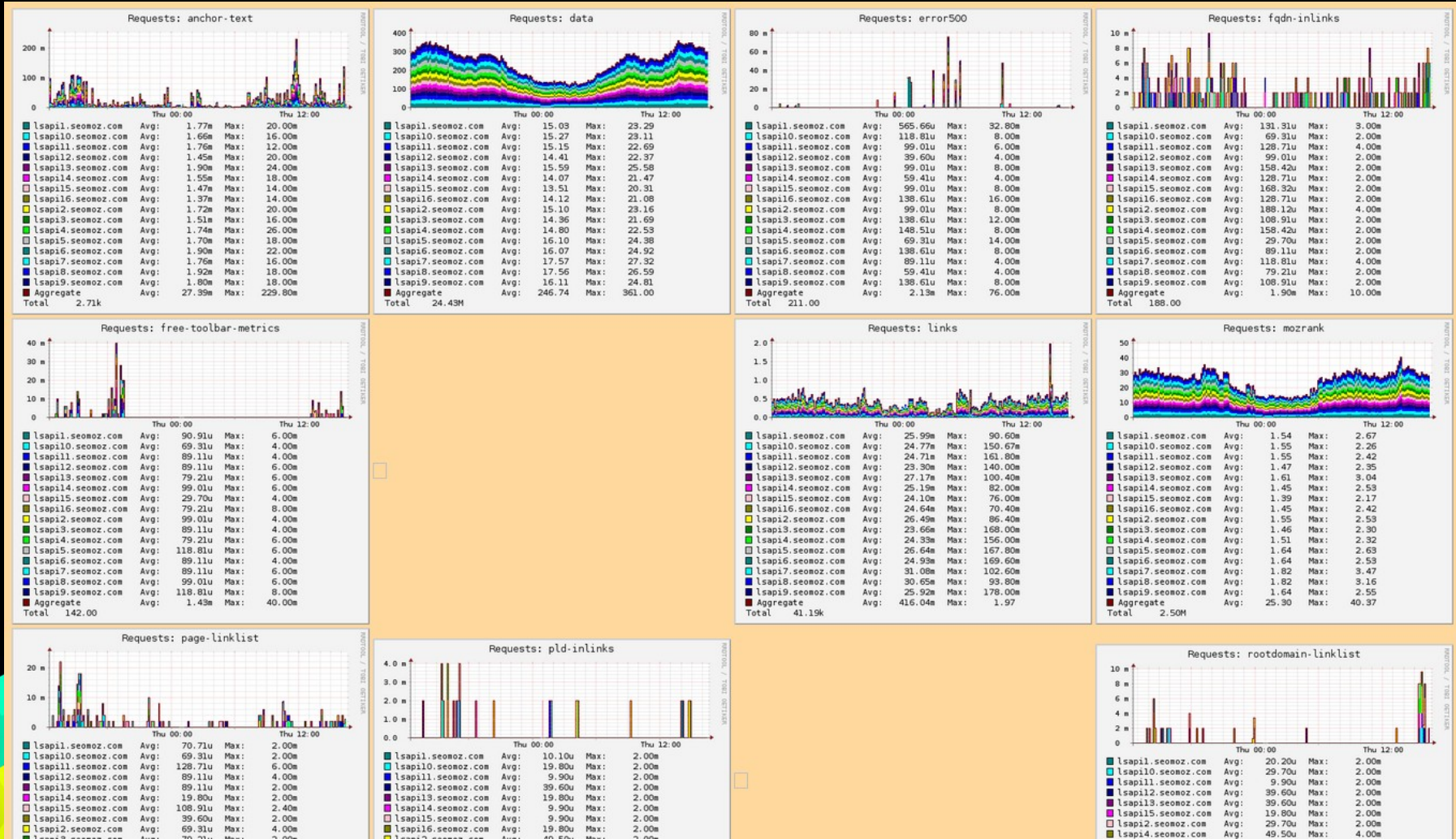
RRDTOOL / TOBI OETIKER



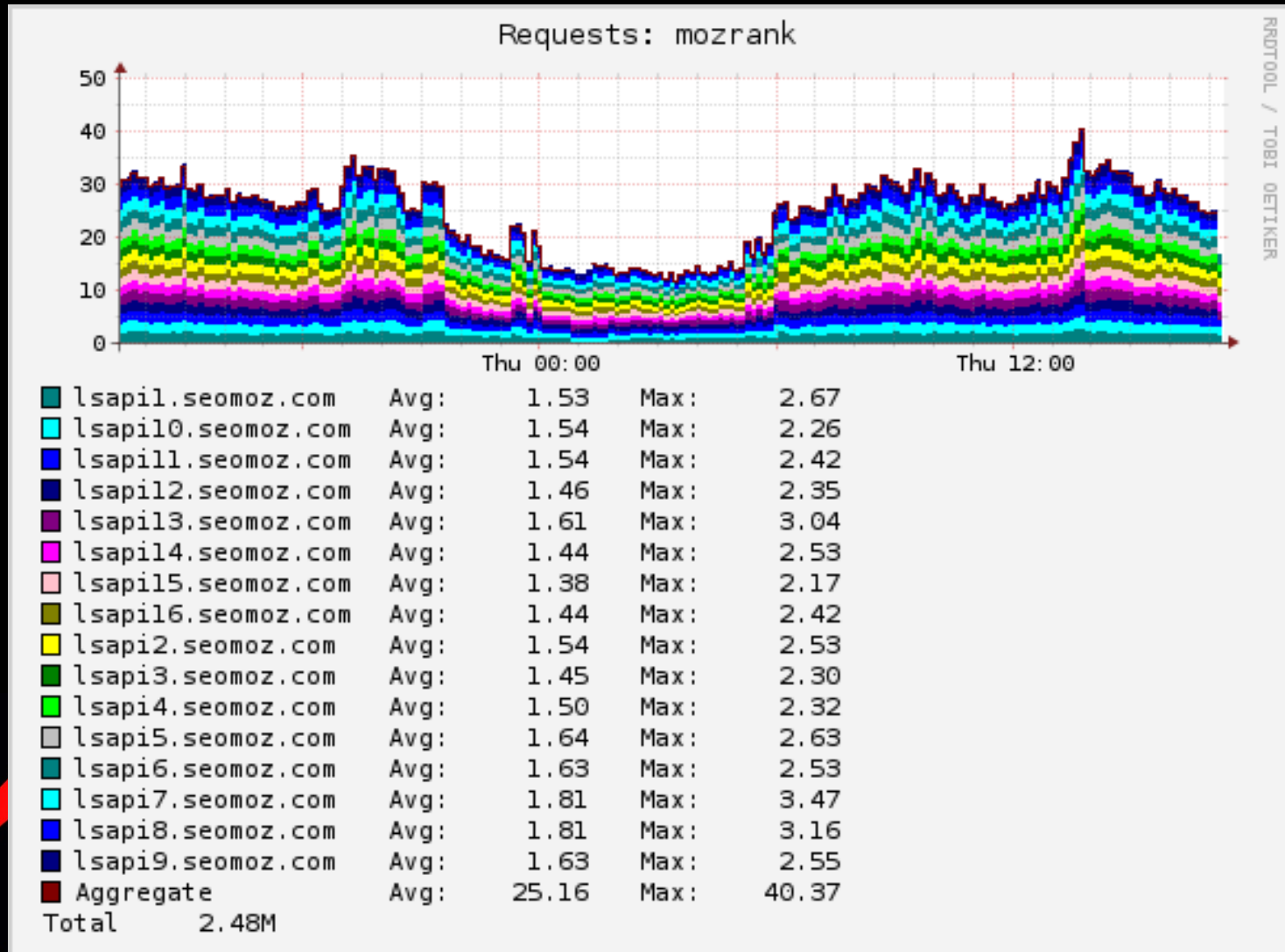
■ busy servers	Avg:	92.49	Max:	106.79
■ idle servers	Avg:	35.51	Max:	67.51
■ requests	Avg:	36.40	Max:	54.96



# System-Wide Dashboard

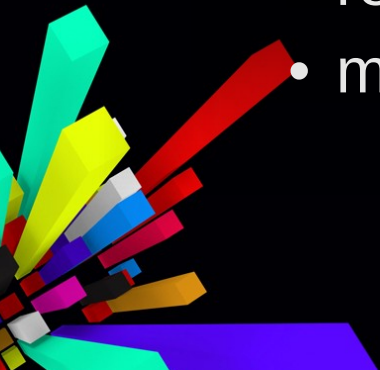


# Per-request



# Graph Summary

- cpu, mem, disk, net
- over time
- per node
- per process
- Through in relevant app measures
  - e.g. per request stats:
    - req/sec
    - median latency/req



# Ad-hoc Tools

- `$ dstat -cdnml`  
system characteristics
- `$ iotop`  
per-process disk I/O
- `$ iostat -x 3`  
detailed disk stats
- `$ netstat -tnp`  
fast, per-process TCP connection stats



# Resources

- Perf Testing: What, How, Why

<http://www.nickgerner.com/2010/02/performance-testing-what-andhow-why/>

- Perf Testing Case Study: OSE

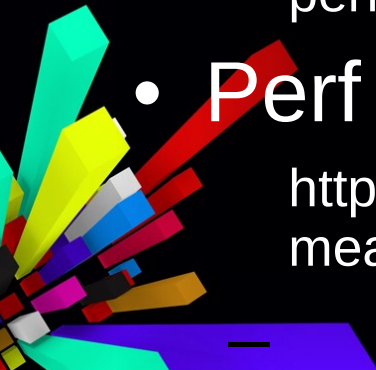
<http://www.nickgerner.com/2010/01/performance-testing-case-study-ose/>

- S3 Benchmarks

<http://twopieceset.blogspot.com/2009/06/s3-performance-benchmarks.html>

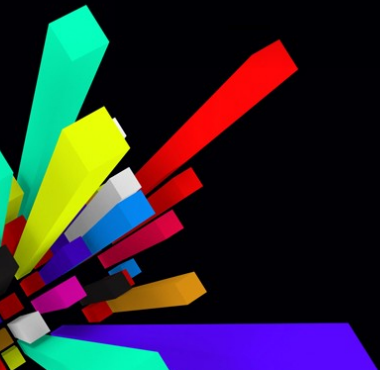
- Perf Measurement

<http://twopieceset.blogspot.com/2009/03/performance-measurement-for-small-and.html>

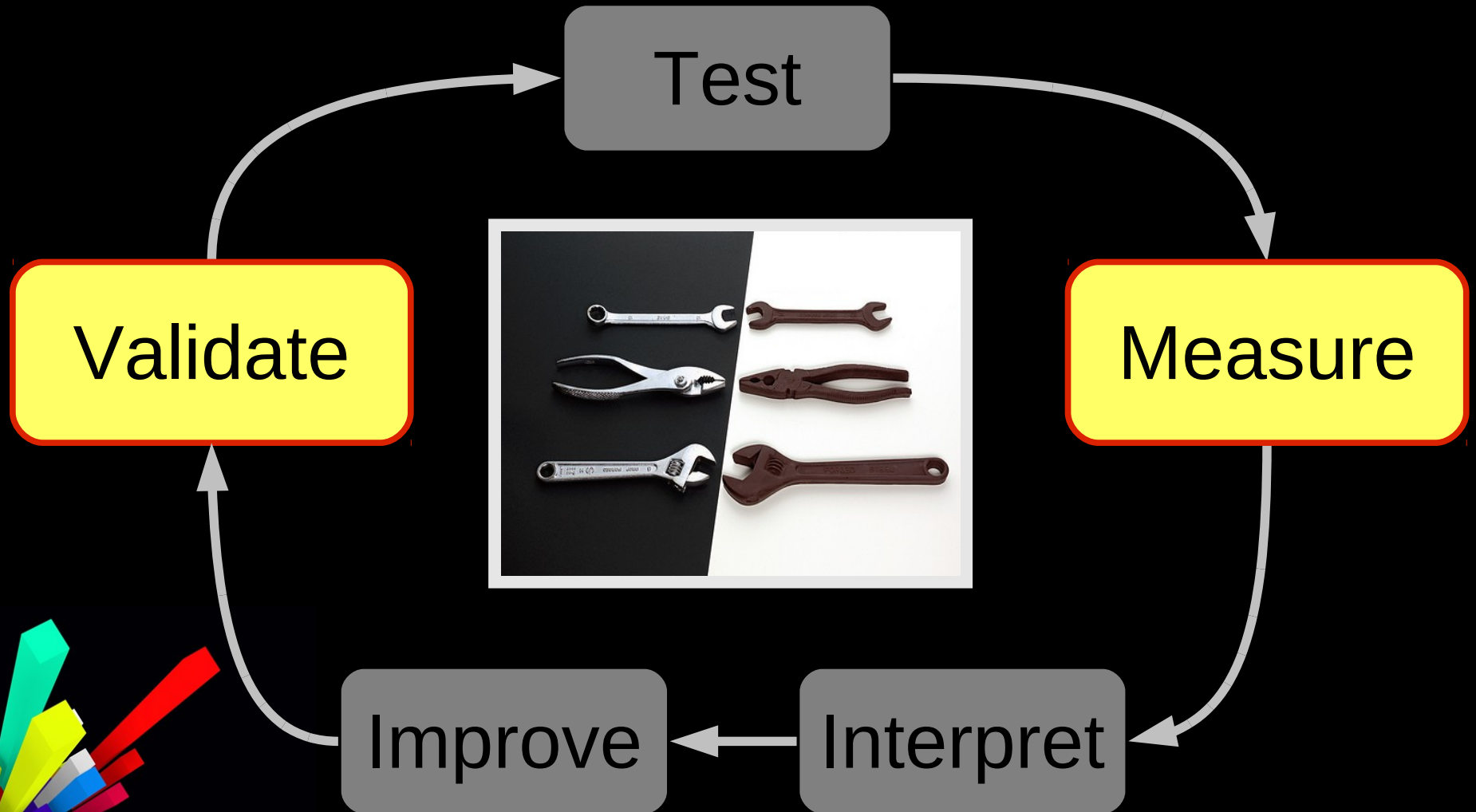


# More Resources

- <http://www.collectd.org>
- <http://oss.oetiker.ch/rrdtool/>
- <http://web.taranis.org/drraw/>
- <http://dag.wieers.com/home-made/dstat/>
  
- \$ man proc

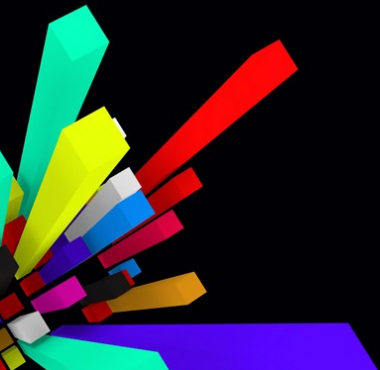
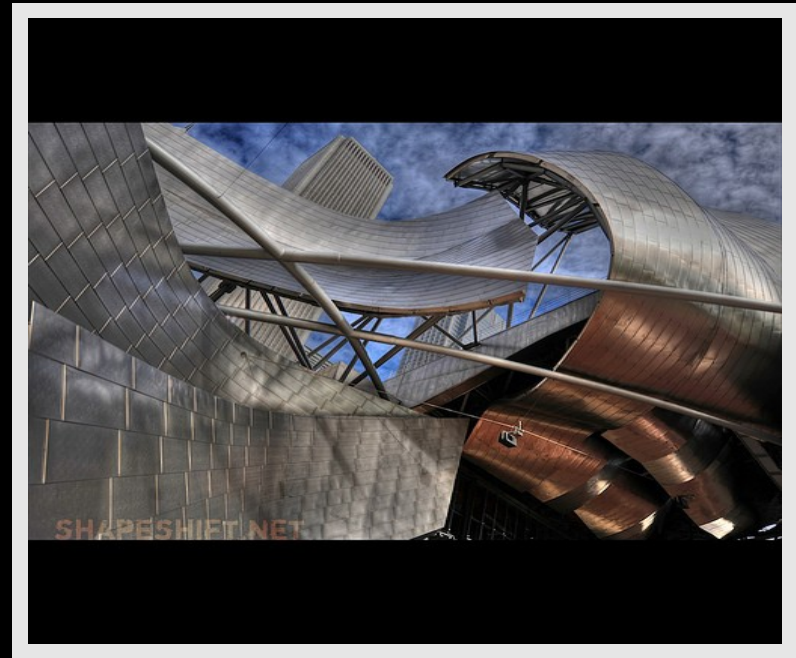


# Q: Why? A: Perf Tuning



# Q: Why? A: System Arch

- Better Devs/Ops
- Identify Bottlenecks
- Scaling Considerations



# Q: Why? A: Issue Investigation

- Machine Specific?
- System Wide?
- Which Component?
- Timeline?
- Cascading Failures?

