A New Age in Alerting with Bosun

The First Alerting IDE

http://bosun.org

What is Bosun?

- A new Open Source Monitoring system that includes an expression language, notification templates, and a testing interface

- It is written in Go and Angular and uses OpenTSDB as its time series database

- The project includes an agent called scollector that gathers data on Linux, Windows, and can poll VSphere and SNMP.

Who am I?

Kyle Brandt

- Director of Site Reliability at Stack Exchange (keep Stack Overflow and Co. online)

- Co-author of Bosun (with Matt Jibson)

- Sometimes Blogger: http://blog.serverfault.com

- @KyleMBrandt

Okay, Let’s Talk Alerting

Alerting is a *hard* problem

because…

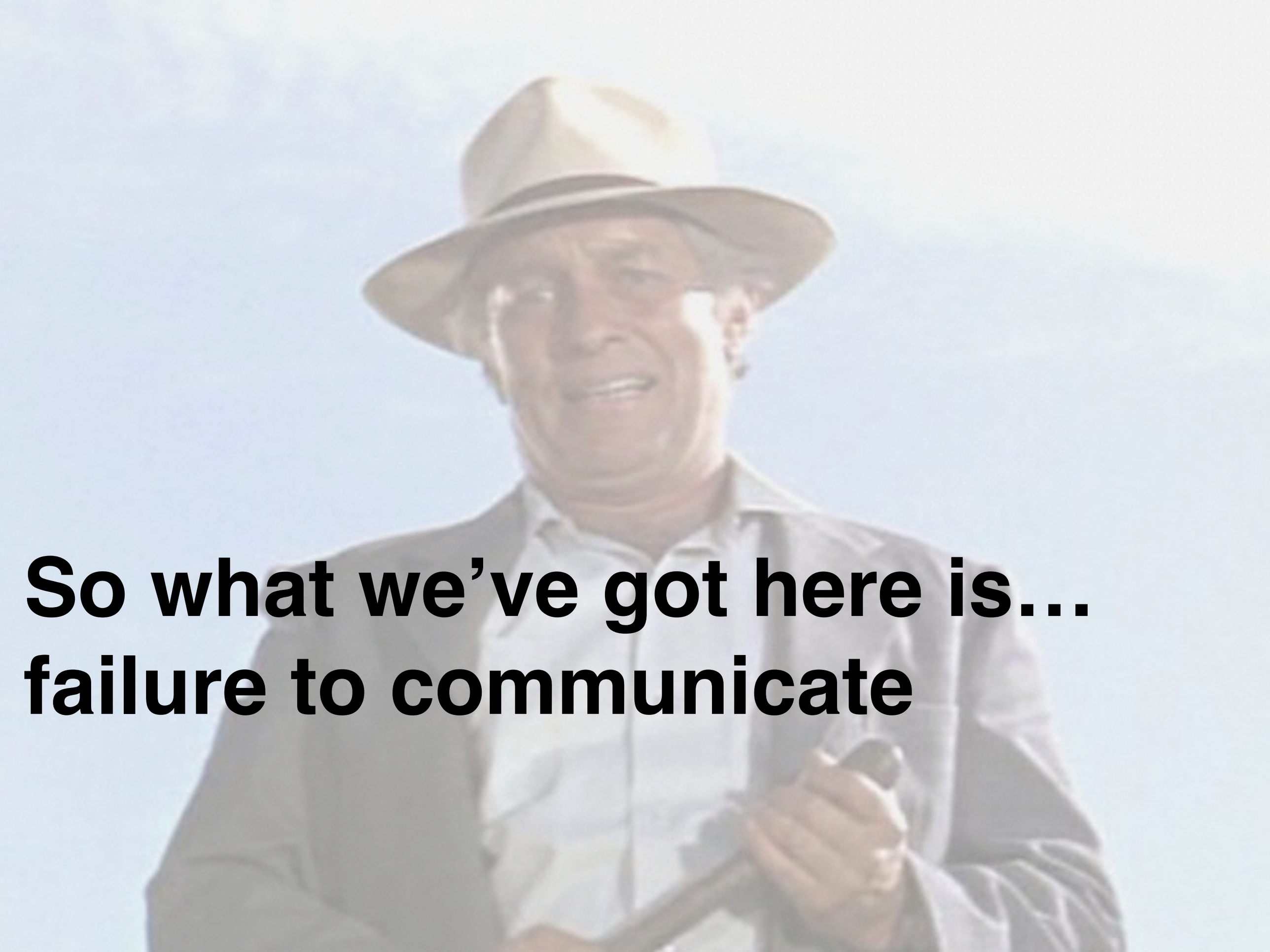
Excellence in Alerting Means Owning Attention

The **two scarce elements** of our economy are ***trust and attention*... Attention** is scarce because it **doesn't scale**. We can't do more than one thing at a time, and the number of organizations and ideas that are competing for our attention grows daily.

—Seth Godin

Too much email from coworkers, alerting systems, vendors, conferences and spam.

All competing for you attention.



So what we've got here is… failure to communicate

How do we Own Attention with Alerts?

1. Have a **good signal to noise ratio** (actionable vs un-actionable alerts)

2. Provide **informative notifications**

3. **Notify the correct people in time** for them to do something about it

We lose attention

Network Performance . (53)	Hardware Component Critical: OR-FS01 :: Battery 0 DELL - OR-	3:21 pm
Network Performance . (15)	ALRM: Disk Usage: OR-SQL02 - E:\ Label:Data be55e420 is 1.3	3:20 pm
Network Performance Moni.	REC: OR-APACHE01-Apache-httpd-Up - NetPerMon Event Log: (	3:02 pm
Network Performance . (12)	REC: NY-LSELASTIC06-Linux CPU Monitoring Perl-Run queue-	2:44 pm
Network Performance M. (7)	ALRM: NY-LSELASTIC06-Linux CPU Monitoring Perl-Run queu	2:42 pm
Network Performance Moni.	ALRM: OR-APACHE01-Apache-httpd-Warning - Component http	2:42 pm
Network Performance Moni.	Memory on NYHQ-DC02 is currently 1.9 GB - Memory on NYHQ-	2:09 pm
Network Performance M. (6)	ALRM: Gossip Redis-Redis Replication-Check Replication-Dow	2:07 pm
Network Performance Moni.	REC: NYOnly_AG-Exceptions-Number of Exceptions in the Pas	2:07 pm
Network Performance Moni.	Number of Exceptions in the Past 10 Minutes on Application Exc	2:06 pm
Network Performance Moni.	ALRM: NYOnly_AG-Exceptions-Number of Exceptions in the Pa	2:03 pm

by spamming people

Network Performance . (10)	Hardware Component Critical: NY-DEVSQL01 :: PS2 Status - NY	11:23 am
Network Performance M. (5)	ALRM: Disk Usage: NYHQ-DC01 - C:\ Label: 345d2535 is 38.1 G	10:36 am
Network Performance Moni.	REC: NY-LSELASTIC02-Linux CPU Monitoring Perl-Run queue-	10:32 am
Network Performance Moni.	ALRM: NY-LSELASTIC02-Linux CPU Monitoring Perl-Run queu	10:22 am
Network Performance Moni.	ALRM: Disk Usage: DEN-FS01 - E:\ Label:DATA f2454e8d is 227	9:36 am
Network Performance Moni.	REC: NY-LSELASTIC01-Linux CPU Monitoring Perl-Run queue-	8:31 am
Network Performance Moni.	ALRM: NY-LSELASTIC01-Linux CPU Monitoring Perl-Run queu	8:27 am
Network Performance . (11)	ALRM: Disk Usage: NY-SQL02 - D:\ Label:Data 94f13420 is 2.0 T	8:15 am
Network Performance M. (5)	ALRM: NY-LSELASTIC06-Linux CPU Monitoring Perl-Run queu	7:51 am
Network Performance Moni.	ALRM: NY-INTSQL02-Windows Server 2003-2012 Services and	7:07 am
Network Performance . (11)	ALRM: Analytics Redis-Redis Replication-Check Replication-Dc	6:57 am

Inbox Not Zero

with uninformative alerts

[sysadmin-team] ALRM: Analytics Redis-Redis Replication-Check Replication-Down Alerts x

Network Performance Monitor

Component Check Replication on Application Redis Replication on Node Analytics Redis is Down

Details:
http://NY-ORION02:80/Orion/View.aspx?NetObject=AM:5076

Acknowledge:
http://NY-ORION02:80/Orion/Netperfmon/AckAlert.aspx?AlertDefID=72d04b8f-c833-4d59-a4f7-905f0645c7d6:5076:APM%3a+Component&viaEmail=true

ALRM: Node OR-APACHE01 is Down. - APACHE01 is Down. Node Details: http://NY-(1:13 am
ALRM: Node OR-VM03 is Down. - VM03 is Down. Node Details: http://NY-ORION02:8(1:13 am
ALRM: Node OR-APACHE02 is Down. - APACHE02 is Down. Node Details: http://NY-(1:13 am
ALRM: Node OR-WEB04 is Down. - WEB04 is Down. Node Details: http://NY-ORION0: 1:13 am
ALRM: Node OR-MYSQL01 is Down. - MYSQL01 is Down. Node Details: http://NY-OR 1:13 am
ALRM: Node OR-WEB01 is Down. - WEB01 is Down. Node Details: http:// 1:13 am
ALRM: Node OR-VM01 is Down. - VM01 is Down. Node Details: http://NY-ORION02:8(1:13 am
ALRM: CoreML Redis-Redis Replication-Check Replication-Down - Redis is Down D 1:13 am
ALRM: Careers Redis-Redis Replication-Check Replication-Down - Redis is Down D 1:13 am
ALRM: Core Redis-Redis Replication-Check Replication-Down - Redis is Down Deta 1:13 am
ALRM: Node or-puppet02.ds.stackexchange.com is Down. - com is Down. Node Det 1:13 am
ALRM: Node OR-DEVSQL01 is Down. - DEVSQL01 is Down. Node Details: http://NY-(
ALRM: N
ALRM: N
ALRM: N
ALRM: N
ALRM: N
ALRM: Node or-git01 is Down. - git01 is Down. Node Details: http://NY-ORION02:80/O 1:13 am
ALRM: Node OR-EDGE01 (PEAK) is Down. -) is Down. Node Details: http://NY-ORIOI 1:13 am
ALRM: Node OR-NEXSCAN01 is Down. - NEXSCAN01 is Down. Node Details: http://N 1:13 am
ALRM: Node OR-FS01 is Down. - FS01 is Down. Node Details: http://NY-ORION02:80/ 1:13 am
ALRM: Node OR-VMWEB02 is Down. - VMWEB02 is Down. Node Details: http://NY-OI 1:13 am
ALRM: Node OR-MAIL01 is Down. - MAIL01 is Down. Node Details: http://NY-ORION0 1:13 am
ALRM: Node OR-EDGE02 (PEAK) is Down. -) is Down. Node Details: http://NY-ORIOI 1:13 am
ALRM: Node OR-SERVICE03 is Down. - SERVICE03 is Down. Node Details: http://NY-
ALRM: Node OR-REDIS01 is Down. - REDIS01 is Down. Node Details: http://NY-ORIO
ALRM: Node OR-SQL02 is Down. - SQL02 is Down. Node Details: http://NY-ORION02:
ALRM: Node OR-PDU04 is Down. - PDU04 is Down. Node Details: http://NY-ORION02

DOWN

DOWN

DOWN

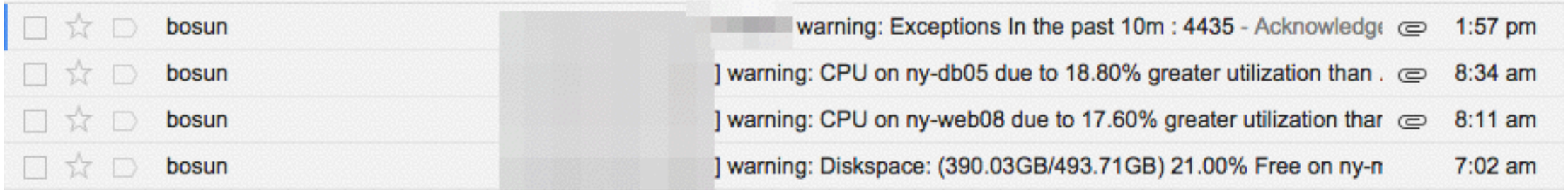
DOWN

DOWN

that are too late.

However…

It doesn't have to be spammy.



[redacted] critical: Redis has 3 of the expected 2 slaves for Analytics (Careers) [:6382] TSAF x

[redacted]

They can be informative.

Acknowledge alert

View the Rule + Template in the Bosun's Rule Page

Notes: This alert determines if replication is working by counting the number of slaves. It counts the number of slaves by finding out the master_link_status and summing that status (1 for working) to get the expected number of slaves. So redis instance is either not a slave, or the slave is not syncing properly then master_link_status will not be 1. Some of redis instances expect there to be 2 masters and 2 slaves, others expect 1 master and 3 slaves. We also trigger if there are more than 1 connected slaves on any single instance since we expect replication to always take place in series (not a star topology)

Redis has 3 of the expected 2 slaves for Analytics (Careers) [:6382]

View Redis Information in Opserver

Slave Status

Server	Is Slave
ny-redis01	slave
ny-redis02	master
or-redis01	slave
or-redis02	slave

Master Sync In Progress Status

Server	Sync In Progress?
ny-redis01	no
or-redis01	no
or-redis02	no

Connected Slaves

Server	Connected Slaves
ny-redis01	1
ny-redis02	1
or-redis01	1
or-redis02	0

Subject

warning: Diskspace: (390.03GB/493.71GB) 21.00% Free on ny-ml02:D (Est. 6.66 days remain)

Body

.... 21.00% Free on ny-ml02:D (Est. 6.66 Days Remain)

Acknowledge alert

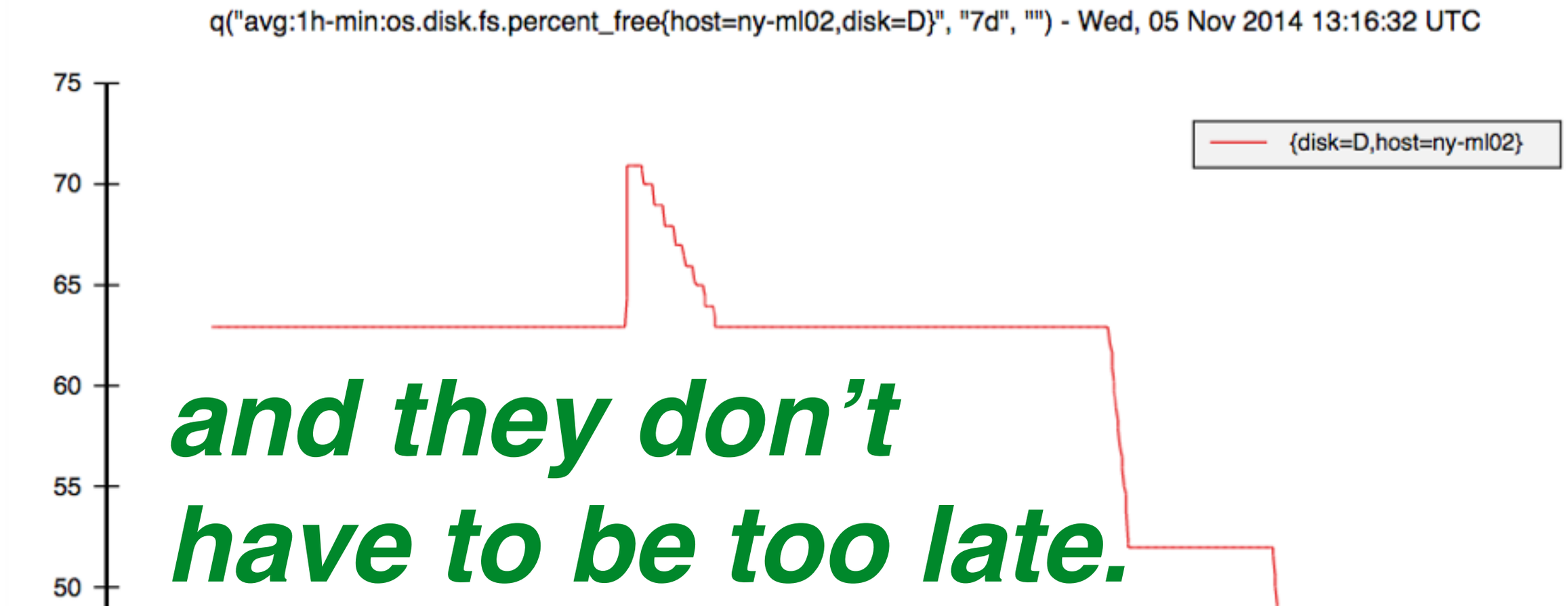
View the Rule + Template in the Bosun's Rule Page

Notes: This alert triggers when there are issues detected in disk capacity. Two methods are used. The first is a traditional percentage based threshold. This alert also uses a linear regression to attempt to predict the amount of time until we run out of space. Forecasting is a hard problem, in particular to apply generically so there is a lot of room for improvement here. But this is a start

View Host ny-ml02 in Opserver

Host: ny-ml02
Disk: D

Percent Free: 21.00%
Used: 390.03GB
Total: 493.71GB
Est. 6.66 days remain until 0% free space



Why do we have these problems?

Why is it spammy?

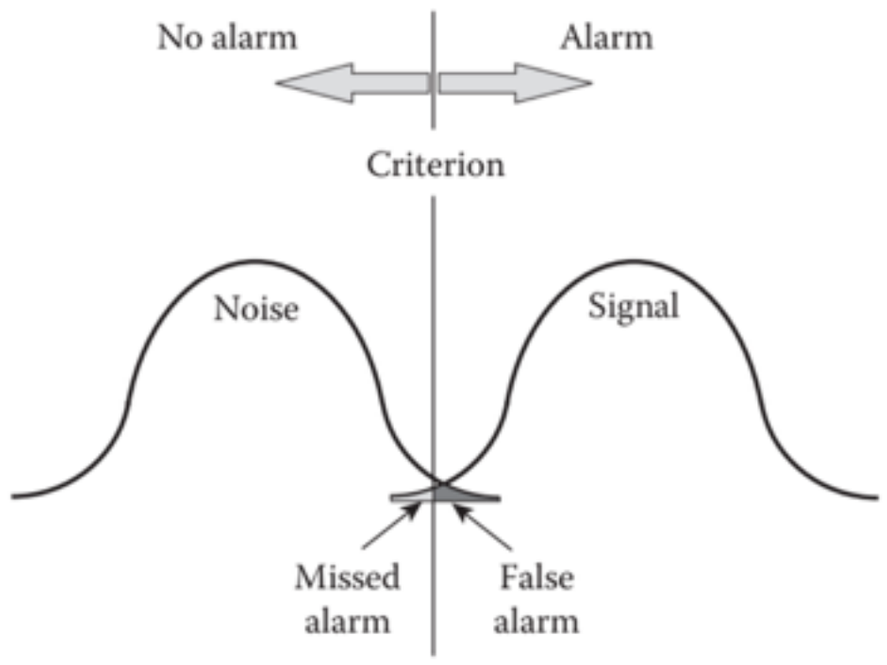
- **The ability to tune alerts in existing systems is highly limited**: Most of what we can tune is just recent duration and thresholds
- **The development cycle for tuning alerts is too slow** and includes too much friction: Deploy a change, wait and see if it triggers when it should - can take days or weeks

Why are Alerts Uninformative?

- Access to data in notifications is limited
- Ability to manipulate the way the data displayed is limited

Why are they too late?

- In order to make alerts less noisy, we make them less sensitive, and by the time we get the alert is too late



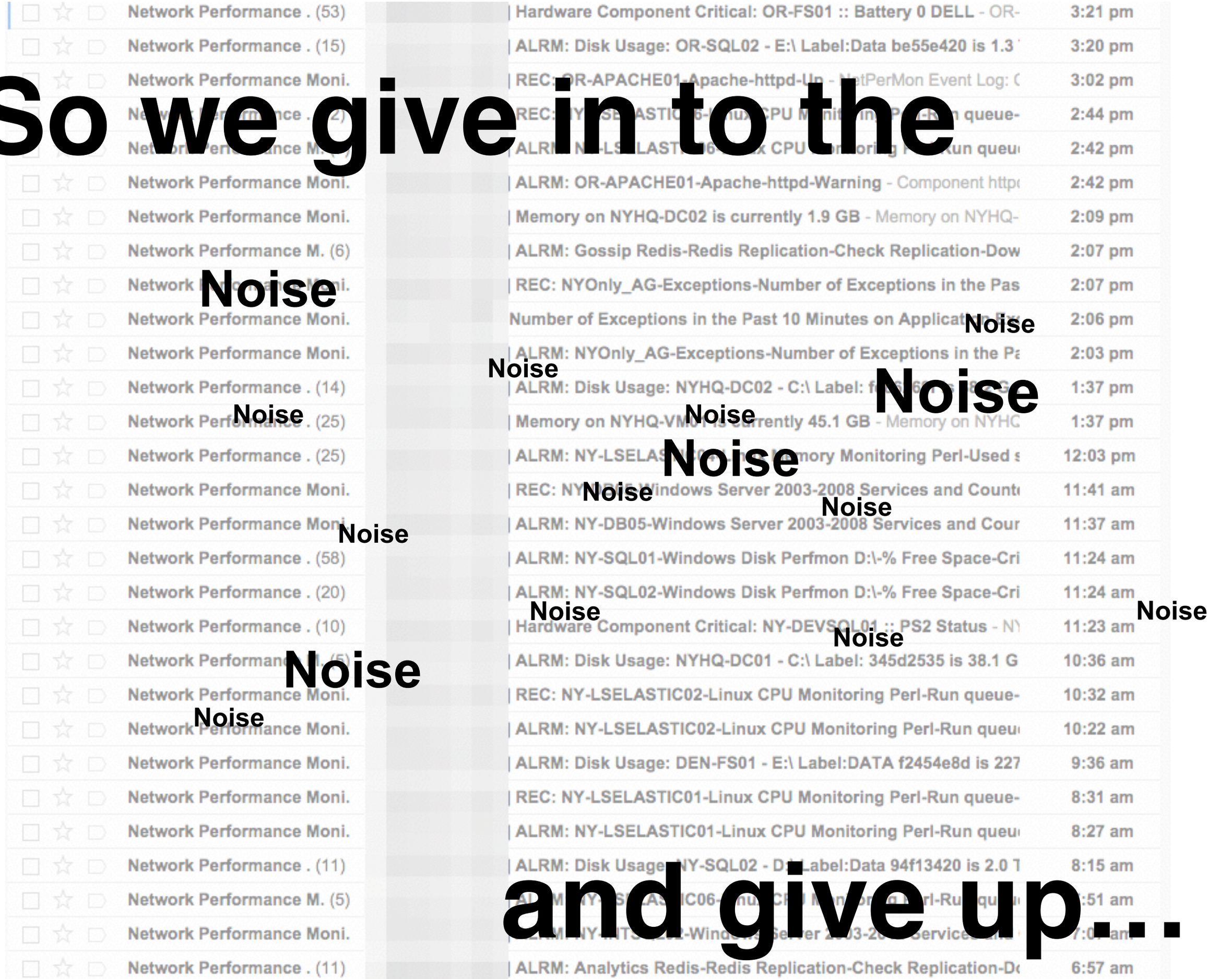
- Forecasting generally isn't a feature (and it can be).

John Allspaw

Lastly: **Too Much Maintenance.**

Easy things are hard, hard things are easy

- New hosts require configuration
- Have to re-collect the same data differently to change alert behavior
- Slow alert tuning cycle - need to wait for alert to trigger to see if it works.


So we give in to the
Noise
Noise
Noise
Noise
Noise
Noise
Noise
Noise
Noise
Noise
Noise
Noise
Noise
Noise
Noise
Noise
Noise
and give up...
Network Performance . (53)
Hardware Component Critical: OR-FS01 :: Battery 0 DELL - OR-
3:21 pm
Network Performance . (15)
ALRM: Disk Usage: OR-SQL02 - E:\ Label:Data be55e420 is 1.3
3:20 pm
Network Performance Moni.
3:02 pm
Network Performance Moni.
Memory on NYHQ-DC02 is currently 1.9 GB - Memory on NYHQ-
2:09 pm
Network Performance M. (6)
ALRM: Gossip Redis-Redis Replication-Check Replication-Dow
2:07 pm
REC: NYOnly_AG-Exceptions-Number of Exceptions in the Pas
2:07 pm
Network Performance Moni.
2:03 pm
Network Performance . (14)
1:37 pm
Network Performance . (25)
1:37 pm
Network Performance . (25)
12:03 pm
Network Performance Moni.
11:41 am
Network Performance Moni.
ALRM: NY-DB05-Windows Server 2003-2008 Services and Cour
11:37 am
Network Performance . (58)
ALRM: NY-SQL01-Windows Disk Perfmon D:\-% Free Space-Cri
11:24 am
Network Performance . (20)
ALRM: NY-SQL02-Windows Disk Perfmon D:\-% Free Space-Cri
11:24 am
Network Performance . (10)
11:23 am
Network Performance Moni.
ALRM: Disk Usage: NYHQ-DC01 - C:\ Label: 345d2535 is 38.1 G
10:36 am
Network Performance Moni.
REC: NY-LSELASTIC02-Linux CPU Monitoring Perl-Run queue-
10:32 am
Network Performance Moni.
ALRM: NY-LSELASTIC02-Linux CPU Monitoring Perl-Run queu
10:22 am
Network Performance Moni.
ALRM: Disk Usage: DEN-FS01 - E:\ Label:DATA f2454e8d is 227
9:36 am
Network Performance Moni.
REC: NY-LSELASTIC01-Linux CPU Monitoring Perl-Run queue-
8:31 am
Network Performance Moni.
ALRM: NY-LSELASTIC01-Linux CPU Monitoring Perl-Run queu
8:27 am
Network Performance . (11)
8:15 am
Network Performance M. (5)
Network Performance Moni.
Network Performance . (11)
ALRM: Analytics Redis-Redis Replication-Check Replication-D
6:57 am

Alerting is a hard problem.

That can be solved, *if*, we respect it.

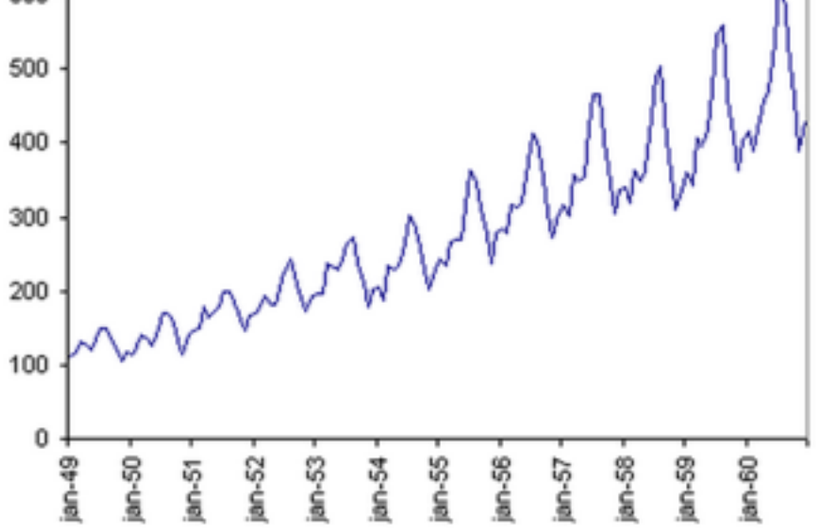
So what do we need?

1. **Data**

2. **Expressive evaluation** of the data to create an alert condition

3. The ability to compose **informative notifications** with that data

4. **Fast iteration** - being able to test alert and notification changes



Data

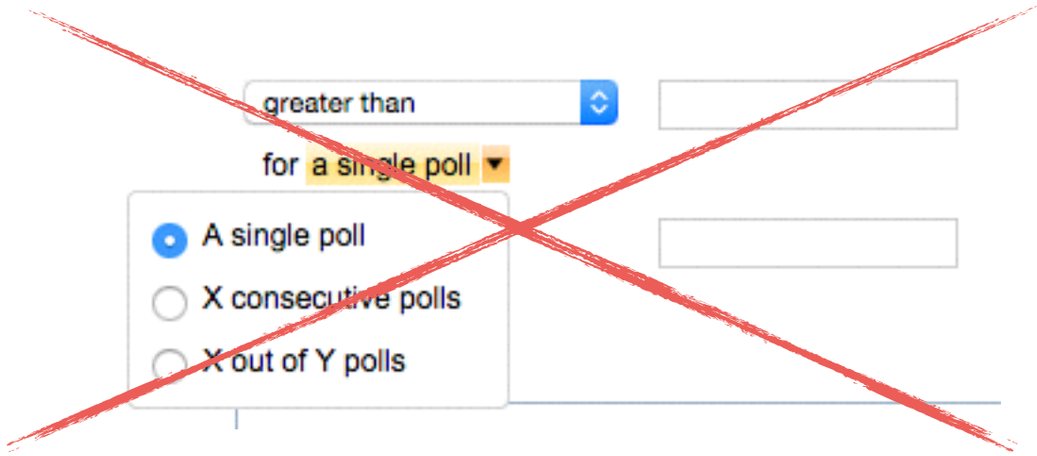
The alerting data must be a *complete time-series*, not just the last few values

Because that is the system's *history*

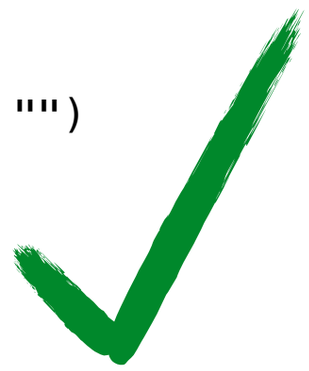
History provides *context*.

Context allows for more accurate trigger conditions, and more informative alerts

Expressive Formulas




```
alert lb.ip_count_changed {
    macro = host_based
    template = generic
    $q = diff("sum:linux.net.ip_count{version=4,host=*-lb*}", "5m", "")
    crit = $q
    critNotification = default
}
```



Expressive Notifications

```
define command{
    command_name    notify-host-by-email
    command_line    /usr/bin/printf "%b" "***** Nagios *****\n\nNotification Type:
$NOTIFICATIONTYPE$\nHost: $HOSTNAME$\nState: $HOSTSTATE$\nAddress: $HOSTADDRESS$\nInfo:
$HOSTOUTPUT$\n\nDate/Time: $LONGDATETIME$\n" | /bin/mailx -s "** $NOTIFICATIONTYPE$ Host
Alert: $HOSTNAME$ is $HOSTSTATE$ **" $CONTACTEMAIL$
}
```

Well… okayish, but:

Developminstrators use real templates

Just need to know:

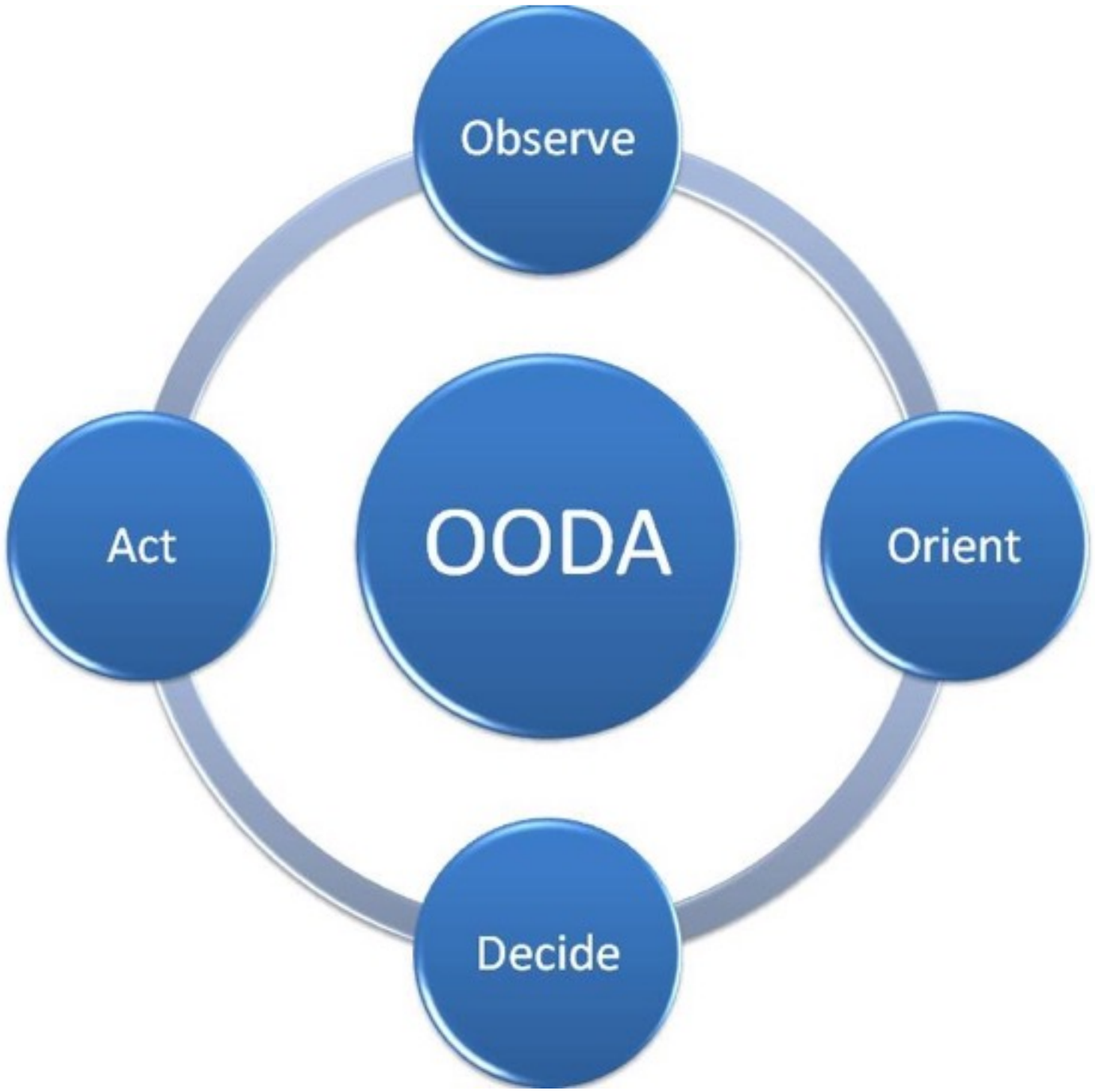
- HTML
- A templating language (like we use in Config Management)

```
template redis.replication {
    body = `{{template "header" .}}
    <p>Redis has {{.Alert.Vars.q | .Eval}} of the expected {{.Lookup "redis" "slave_count"}}
slaves for {{.Lookup "redis" "name"}} [:{{.Group.port}}]
    <p><a href="https://status.stackexchange.com/redis">View Redis Information in Opserver</
a>
     <h2>Slave Status</h2>
    <table>
    <tr><th>Server</th><th>Is Slave</th></tr>
    {{range $r := .EvalAll .Alert.Vars.is_slave}}
       {{ if $r.Group.Subset $.Group}}
        <tr>
             <td>{{$r.Group.host}}</td>
             <td>{{if eq $r.Value 0.0}} master {{else}} slave {{end}}</td>
       {{end}}
    {{end}}
    </table>
```

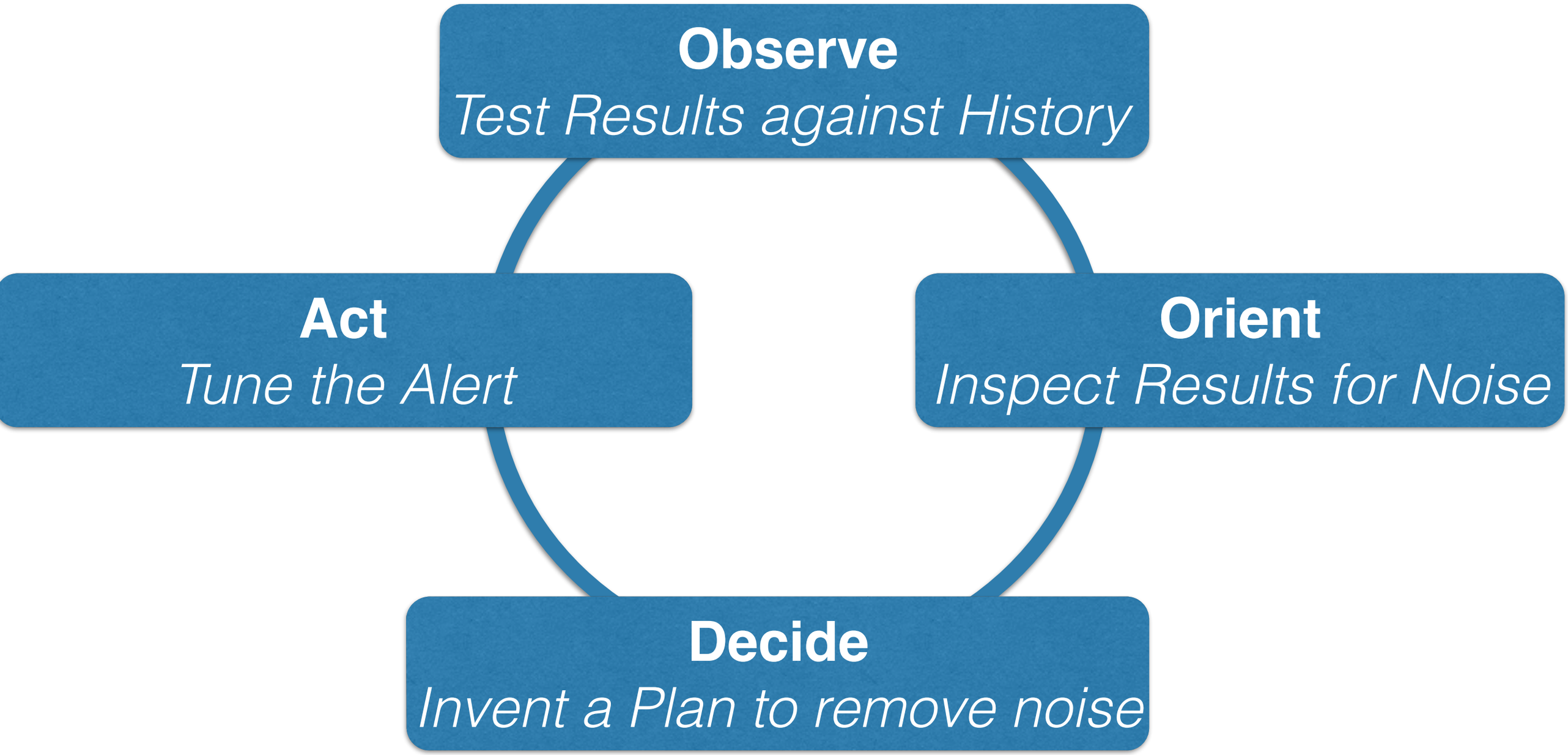
Testing

Testing and the OODA Loop

Faster Iteration



OODA Applied: Less Alert Noise



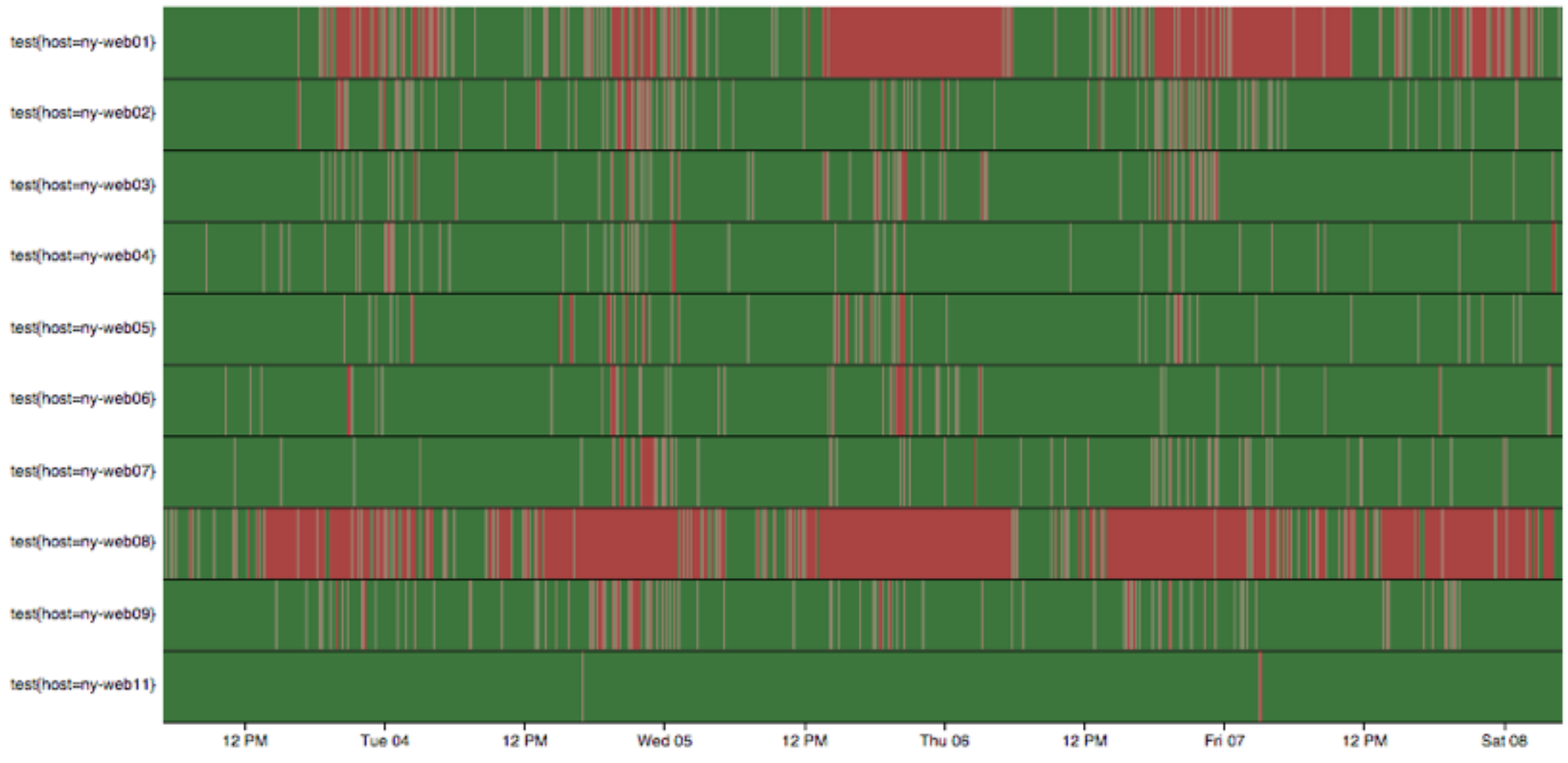
Test Alerts Against History

From 2014-11-03 HH:MM **To** 2014-11-08 HH:MM **Intervals** 1440 **Step Duration (m)** 5 Test

Email **Template Group**

Shift-enter to test. If neither From nor To are present, will run for now. If one present, at that timestamp. If both present, that timespan. Times default to midnight if not set.

Results Template Timeline



2014/11/08-18:27:48
Alert

test{host=ny-web01} 235 events

Tune and Retest

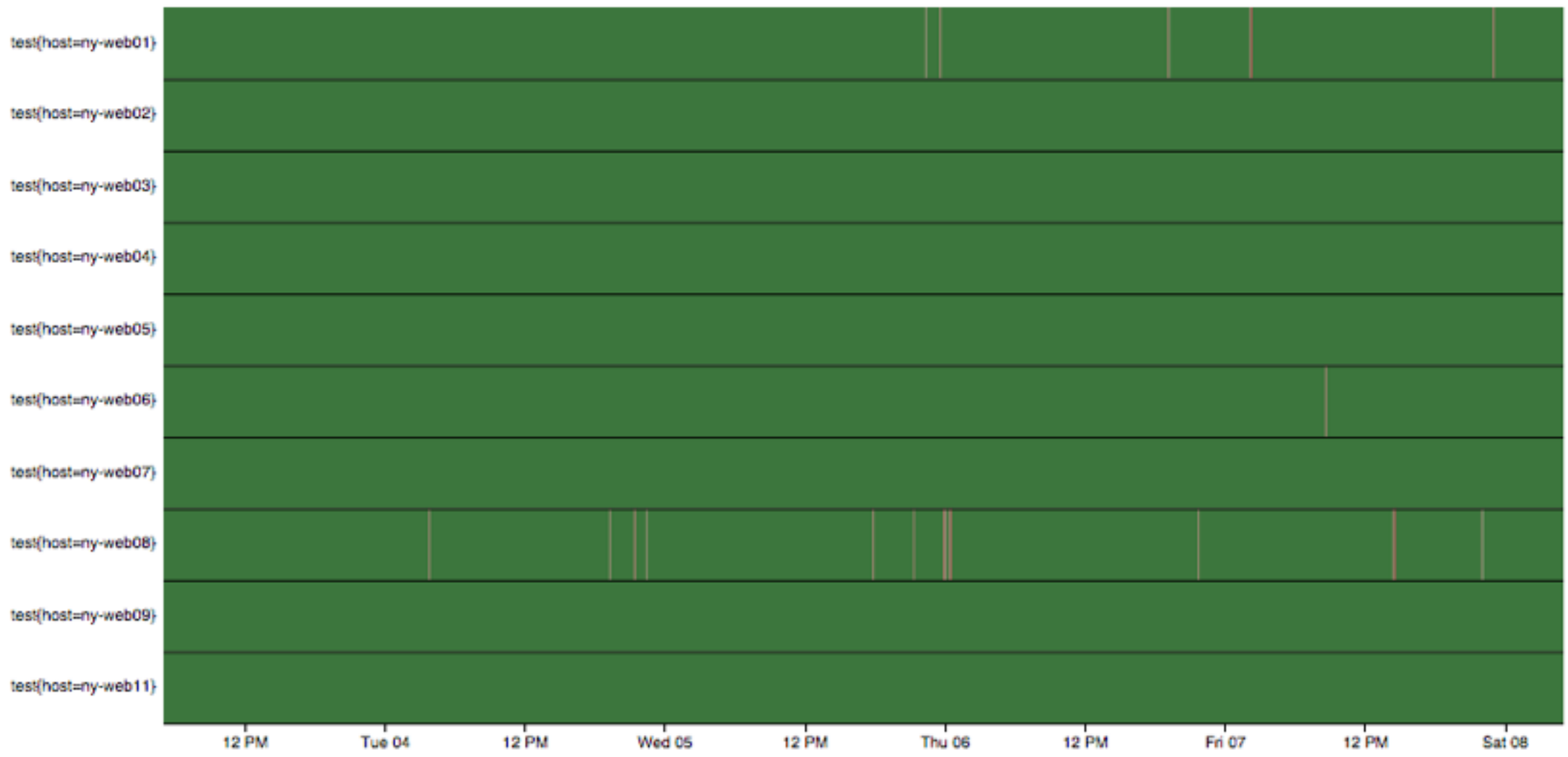
Less Red is Good

From 2014-11-03 HH:MM **To** 2014-11-08 HH:MM **Intervals** 1440 **Step Duration (m)** 5 **Test**

Email **Template Group**

Shift-enter to test. If neither From nor To are present, will run for now. If one present, at that timestamp. If both present, that timespan. Times default to midnight if not set.

Results Template Timeline



2014/11/07-12:28:13
test{host=ny-web01}

test{host=ny-web01} 11 events

Testing Means

- Design Alerts to be more accurate before they go into production
- More time spent fixing the hard things, less time on easy things
- Less friction in tuning alerts

When things are easy, they get done.

Things you can do once you have:

1. Data
2. Expression Language
3. Notification Templates
4. Testing

- Combine Metrics: i.e ratio of one thing to another
- Make timespan a facet of tuning
- Thresholds based on history (Anomalous data)
- Alert at Various Scopes: How should components in your environment be grouped? By Host, subsystem, cluster, a combination of those things
- Use statistical reduction functions like: Min, Percentile, Median, Deviations, etc
- Relative Thresholds: For example, how does one item in a cluster compare to the others
- Boolean Conditions: Don't alert when other things are true or not true
- And more … but this font is getting pretty small…. so let's look at some examples …

Formulas that Combine Metrics

- We can use current and limit metrics to create a percentage, and then alert on that.
- Because of this, we don't need to change the data that is collected, we can just use what HAProxy gives us by default

```
alert haproxy_session_limit {
    ….
    $current_sessions = max(q("sum:haproxy.frontend.scur{host=*,pxname=*,tier=*}",
"5m", ""))
    $session_limit = max(q("sum:haproxy.frontend.slim{host=*,pxname=*,tier=*}", "5m",
""))
    $q = ($current_sessions / $session_limit) * 100
    warn = $q > 80
    crit = $q > 95
}
```

Timespan as a facet of tuning

- Alert if puppet has been consistently disabled for the past 24 hours

```
alert puppet.left.disabled {
    macro = host_based
    template = generic
    $notes = More often than not, if puppet has been consistently disabled for more
than 24 hours some forgot to re-enable it
    $oquery = "avg:24h-min:puppet.disabled{host=*}"
    $q = min(q($oquery, "24h", ""))
    warn = $q > 0
}
```

Boolean Example

- Don’t alert on linux swapping if there happens to be a high exim mail queue - Doesn’t require an additional alert

```
alert linux.swapping {
    macro = host_based
    template = generic
    $notes = This alert does not trigger for our mail servers with the mail queue is
high
    #NV makes it so that if mailq doesn't exist for the Host, the NaN that gets
returned gets replaced with a 1 (True)
    $mail_q = nv(max(q("sum:exim.mailq_count{host=*}", "2h", "") > 5000), 1)
    $metric = "sum:rate{counter,,1}:linux.mem.pswp{host=*,direction=in}"
    $q = (median(q($metric, "2h", "")) > 1) && ! $mail_q
    warn = $q
    squelch = host=ny-devsearch*|ny-git01
}
```

Another Boolean Example

- Alert if a bond has a bad slave, *or* there is only 1 slave

```
alert linux.bonding {
    template = linux.bonding
    macro = host_based
    $notes = This alert triggers when a bond only has a single interface, or the
status of a slave in the bond is not up
    $slave_status = max(q("sum:linux.net.bond.slave.is_up{bond=*,host=*,slave=*}",
"5m", ""))
    $slave_status_by_bond = sum(t($slave_status, "host,bond"))
    $slave_count = max(q("sum:linux.net.bond.slave.count{bond=*,host=*}", "5m", ""))
    $no_good = $slave_status_by_bond < $slave_count || $slave_count < 2
    #Make it by host, so we only get one alert per host
    $by_host = max(t($no_good, "host"))
    warn = $by_host
}
```

What about Alerting on Anomalies

- Alerts based on anomaly detection (deviation from history) can be effective when selectively applied by a skilled operator

- Sometimes, it is the only practical option

Anomalous Changes

We track performance per Web Route, there are thousands of them so setting thresholds for each route is not feasible

Subject

warning: Median Response Time Change of 19.62 ms (Current: 25.60 ms Past: 5.98 ms) on Ad/Impression

Body

Acknowledge alert

View the Rule + Template in the Bosun's Rule Page

Notes: Response time is based on HAProxy's Tr Value. This is the web server response time (time elapsed b
server and the moment it send its complete response header

Route: Ad/Impression
Past Median: 25.60 ms
Current Median: 5.98 ms
Difference: 19.62 ms
Route Hits: 10515343.73 hits
Total Hits: 149003023.13 hits
Route Hit Percentage of Total: 7.06%

Slower Web Route

If the current median is greater than the past + 2 StdDevs, and this route makes up more than 1% of our web hits

```
alert slower.route.performance {
    $notes = Response time is based on HAProxy's Tr Value. This is the web server
response time (time elapsed between the moment the TCP connection was established to
the web server and the moment it send its complete response header
    $duration = "1d"
    $route=*
    $metric = "sum:10m-avg:haproxy.logs.route_tr_median{route=$route}"
    $route_hit_metric = "sum:10m-avg:rate{counter,,
1}:haproxy.logs.hits_by_route{route=$route}"
    $total_hit_metric = "sum:10m-avg:rate{counter,,1}:haproxy.logs.hits_by_route"
    $route_hits = change($route_hit_metric, $duration, "")
    $total_hits = change($total_hit_metric, $duration, "")
    $hit_percent = $route_hits / $total_hits * 100
    $current_hitcount = len(q($metric, $duration, ""))
    $period = "7d"
    $lookback = 4
    $history = band($metric, $duration, $period, $lookback)
    $past_dev = dev($history)
    $past_median = percentile($history, .5)
    $current_median = percentile(q($metric, $duration, ""), .5)
    $diff = $current_median - $past_median
    warn = $current_median > ($past_median + $past_dev*2) && abs($diff) > 10 &&
$hit_percent > 1
}
```

Controlling Scope

Scope impact the number of notifications per event

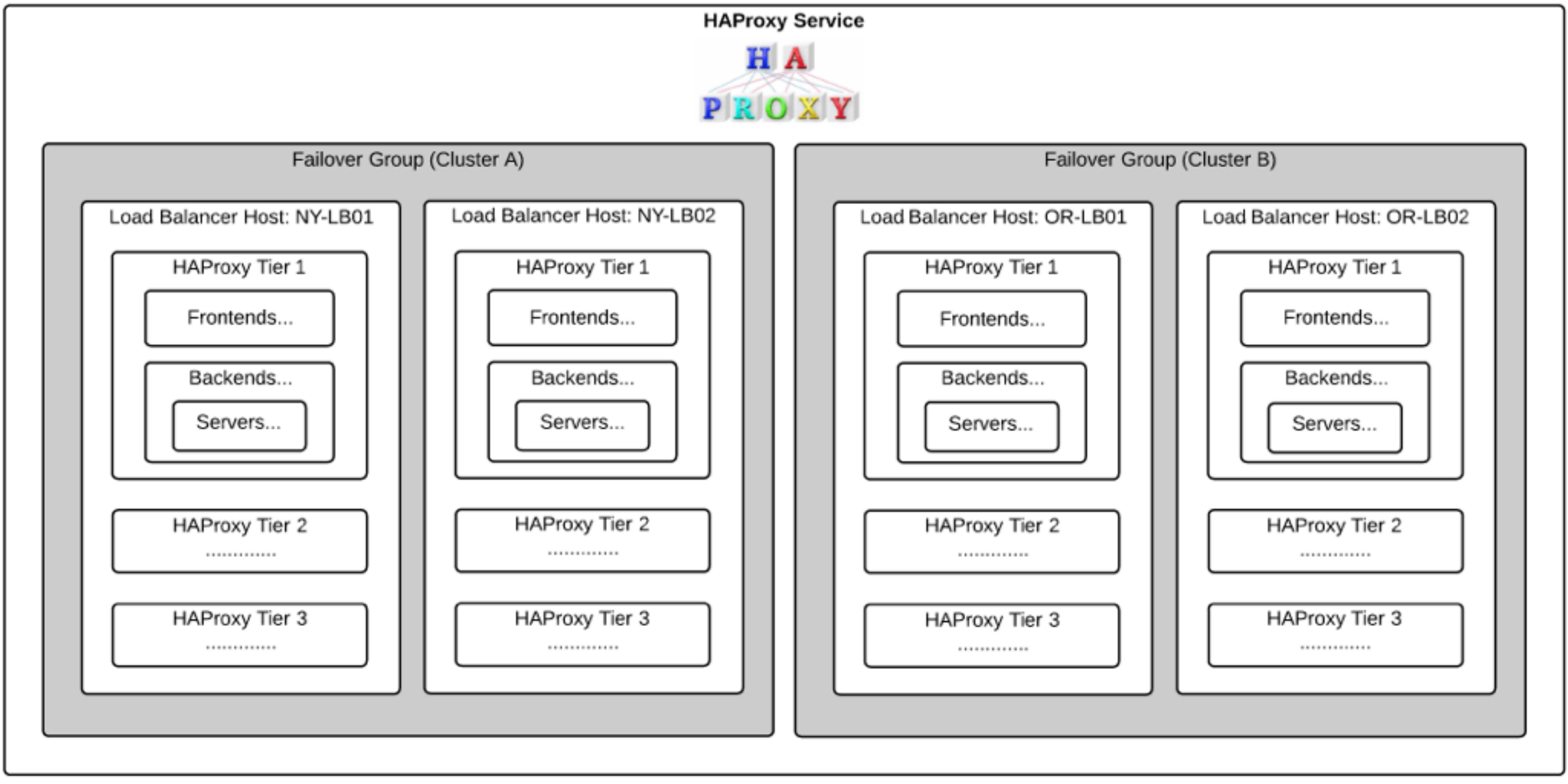
Scope

- Most monitoring systems alert (instantiate) on a metric + host
- You can also think of it as GROUP BY. So normal monitoring systems force you to group by object and metric, where object is generally a host, or a single instance of a thing on a host

Fixing Scope: Make Things Orthogonal

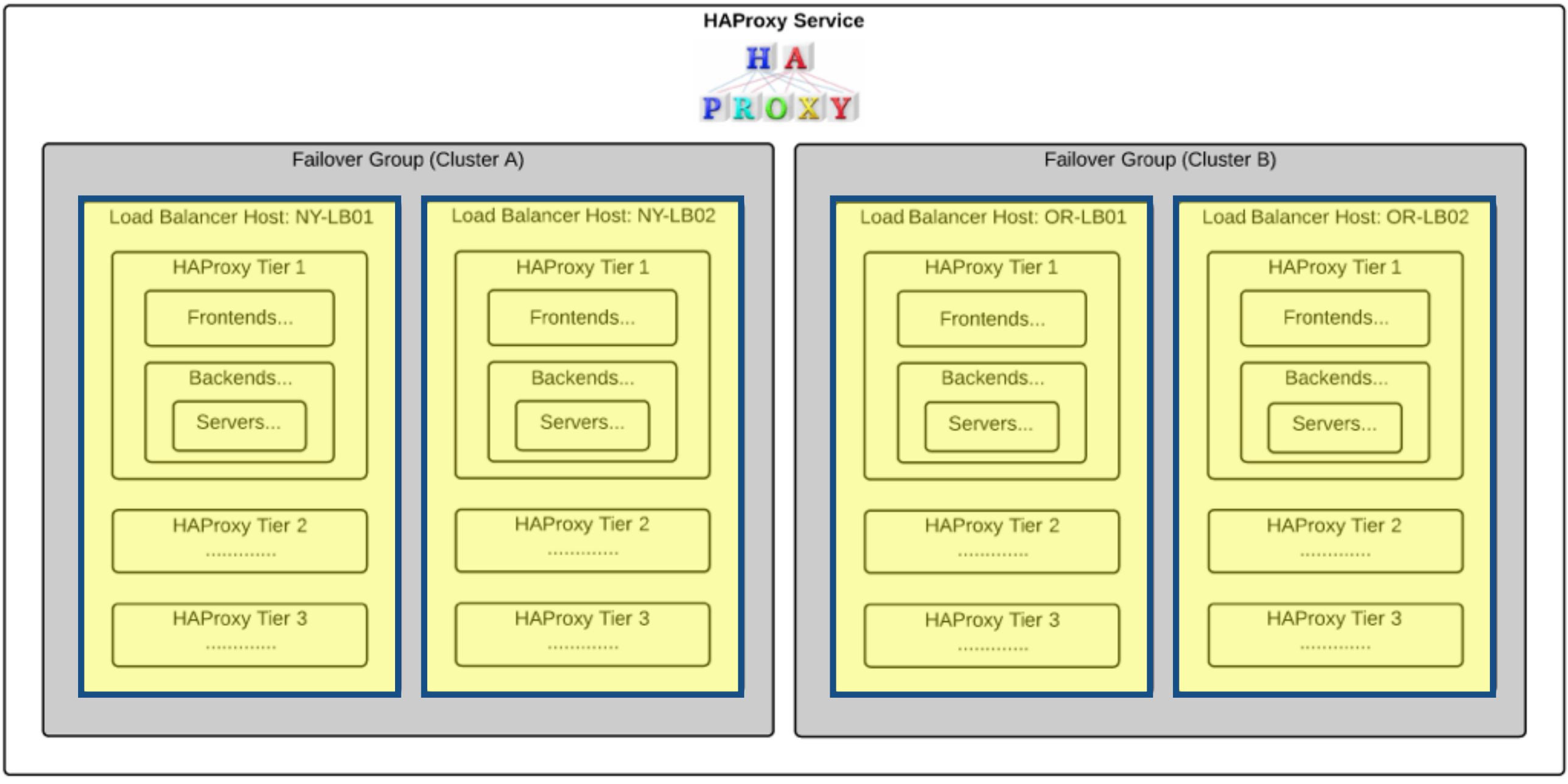
- Free Metrics from Objects: i.e. CPU Utilization is a metric, CPU Core or Host is an object
- Then free objects and/or metrics from alerting scope

Exploring Scope: Stack Exchange's HAProxy Setup



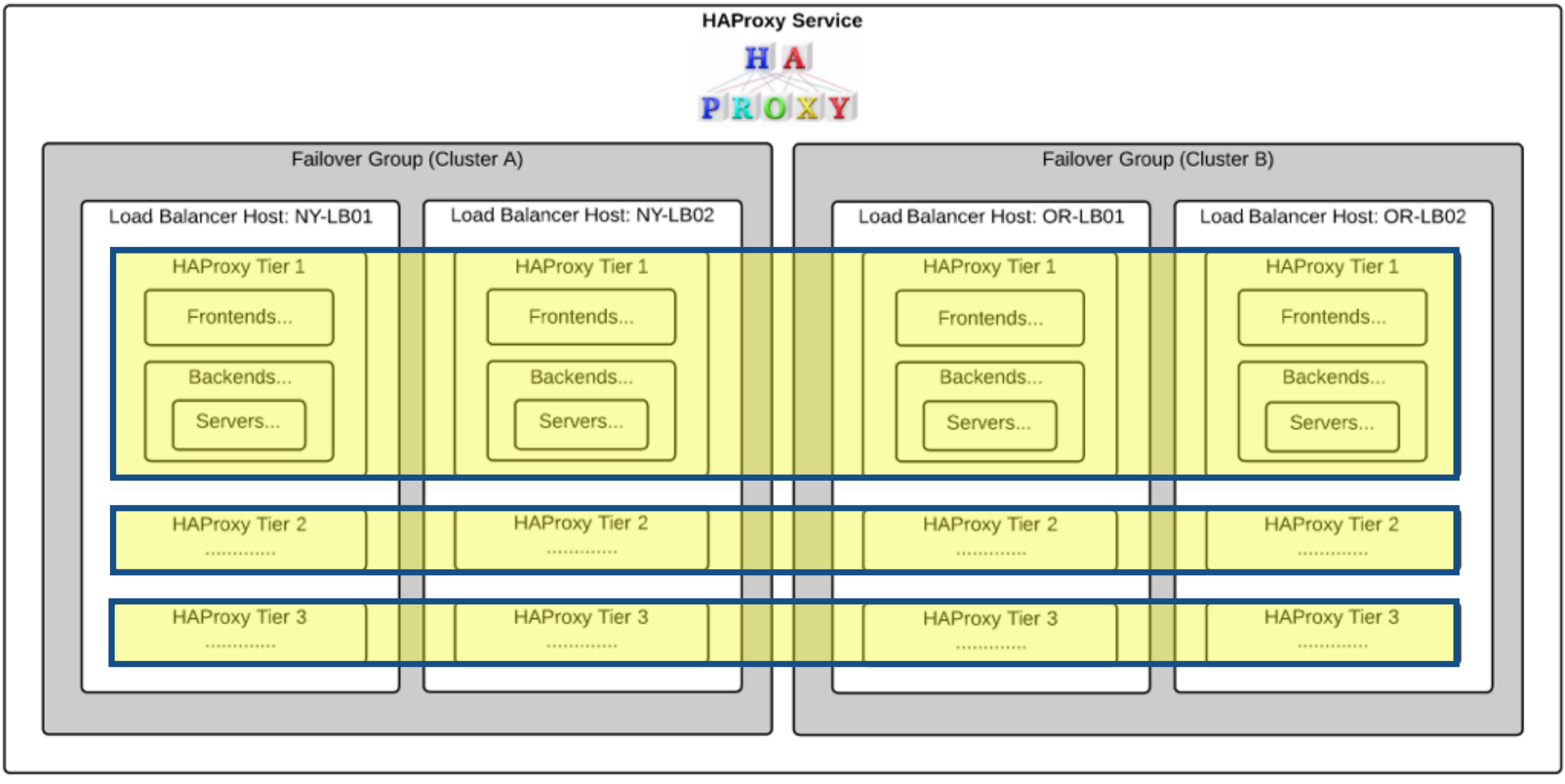
Host Based

4 Possible Alerts



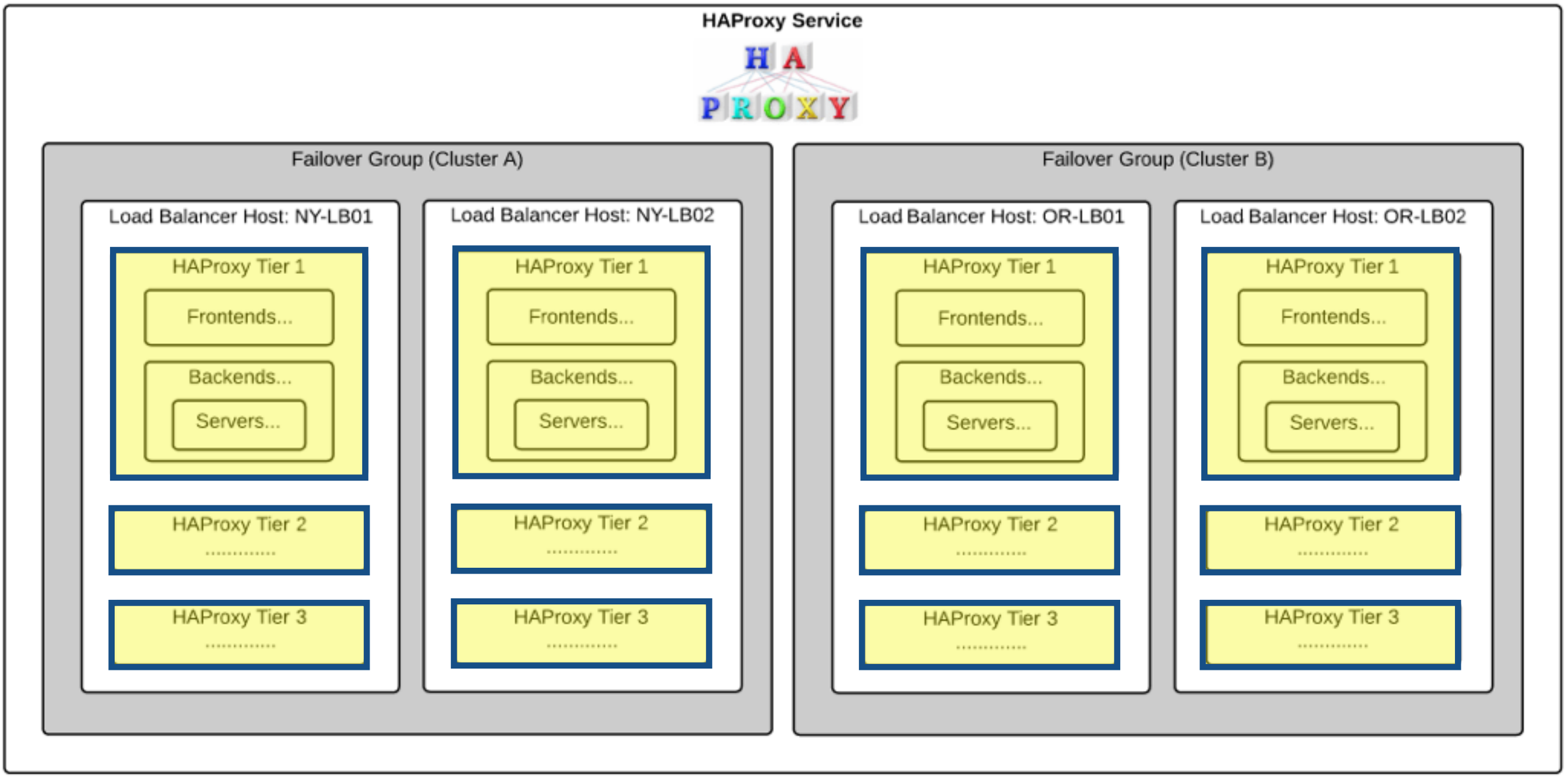
Tier Based

3 Possible Alerts



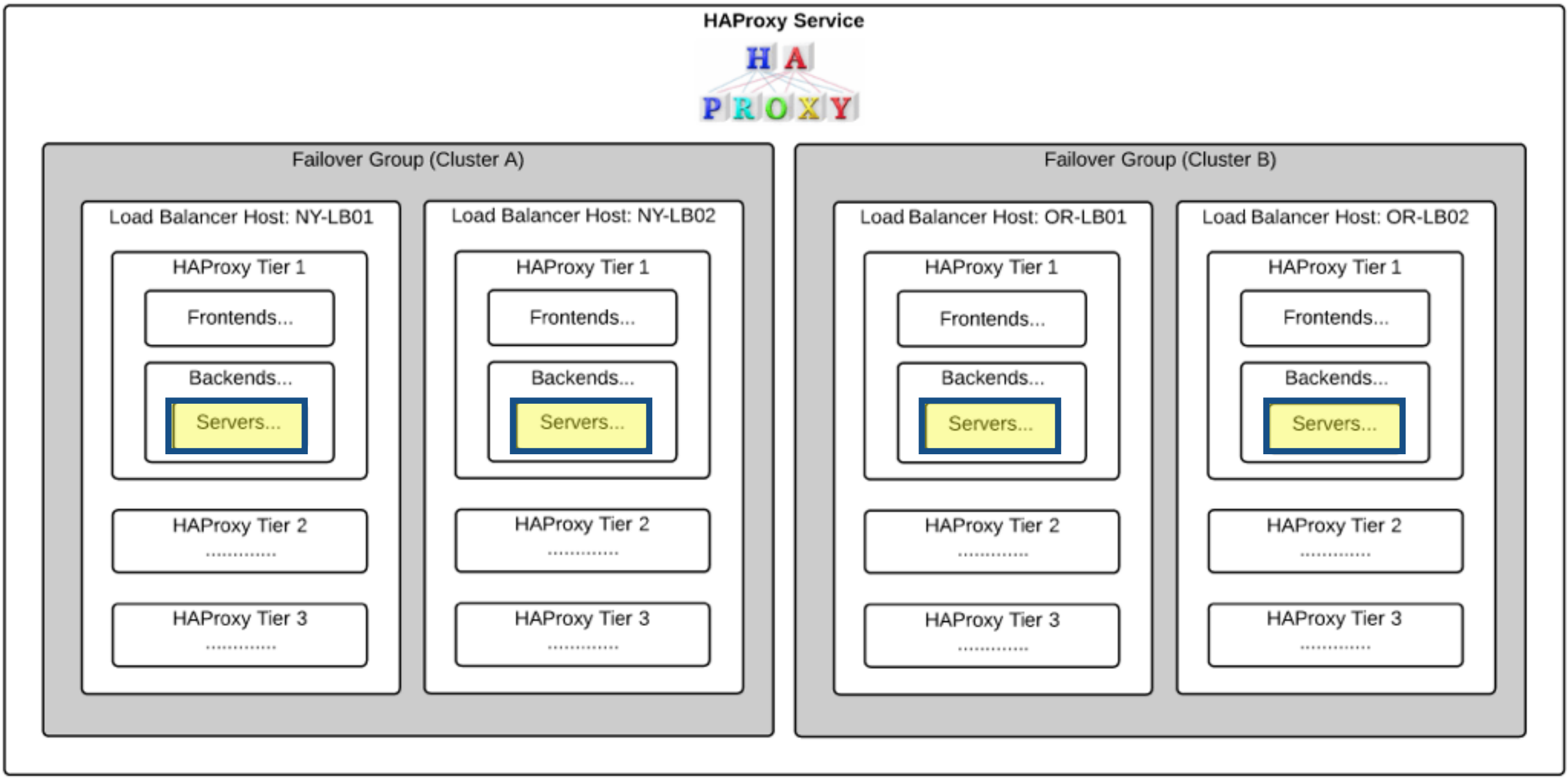
Tier+Host

12 Possible Alerts



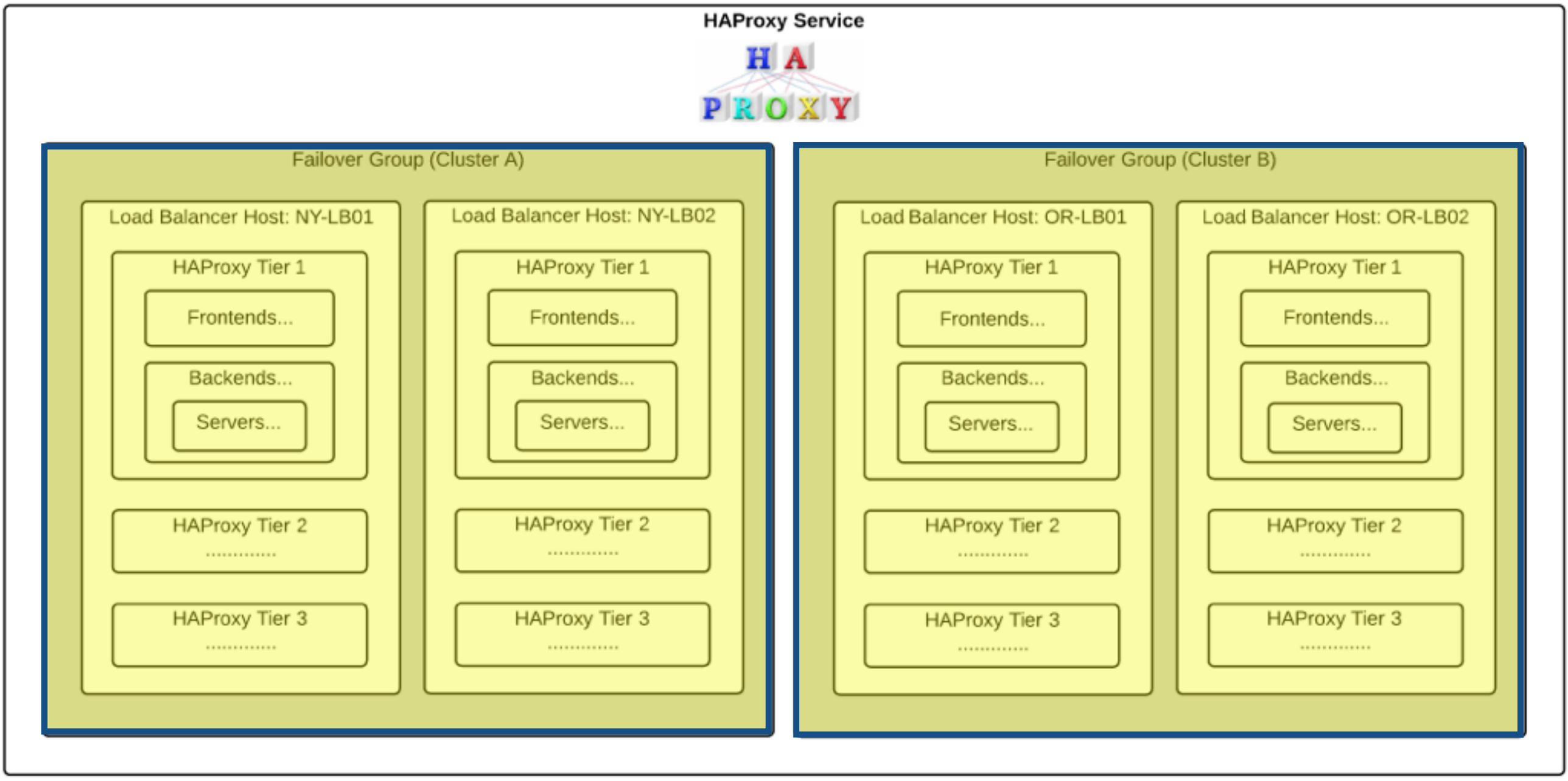
Per Server

N Servers * N Backends * * N Hosts * N Tiers = **Crapload of Alerts**



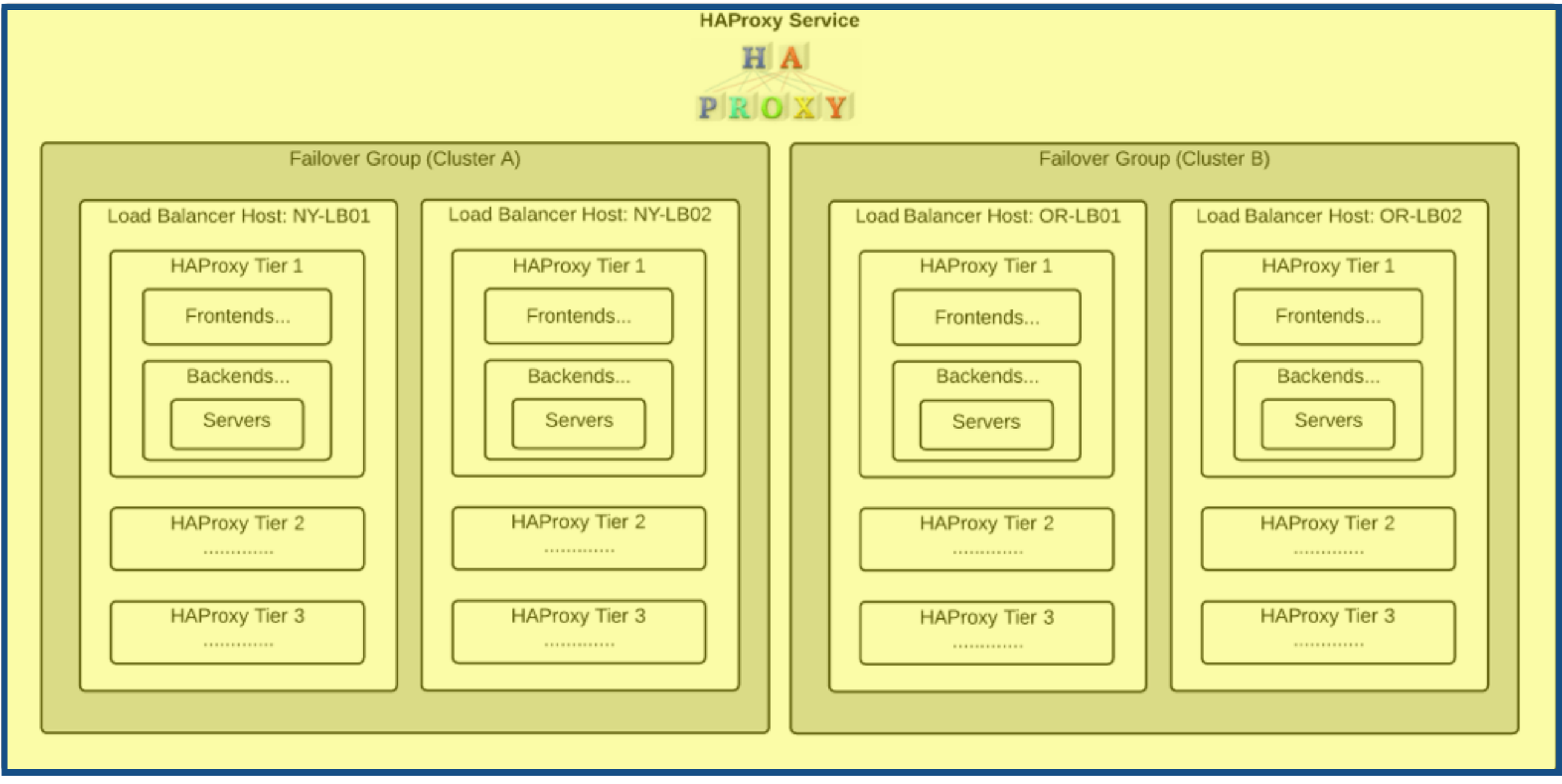
Cluster

2 Possible Alerts



Whole Service

1 Possible Alert



Scope Example: Service Level for HAProxy

Subject

warning: At least one backend has 1 servers down

Body

Acknowledge alert

View the Rule + Template in the Bosun's Rule Page

Notes: This alert triggers when any server (as haproxy defines a server) has been down for more than 2 of the last 5 minutes and that server is not in maintenance. The notification is scoped to our entire haproxy service. The consequence of that is that if 1 server goes down somewhere after one is down we won't get another notification. However, there will be another critical notification if there are more than 50% of servers down on any unique Host,Backend,Server combination. Because of this, it is important to be disciplined in handling this alert.

View HAProxy info in Opserver

Down HAProxy Servers By Host, Backend, and Server

Host	Backend	Server	Seconds Down in the Past "5m"
or-lb01	be_blog_se	or-apache02	300
or-lb01	be_wordpress	or-apache02	300
or-lb02	be_blog_se	or-apache02	300
or-lb02	be_wordpress	or-apache02	300

One Email instead of 4
(The issue is one server is down)

Percentage of Down Servers By Host and Backend

Host	Backend	Percent of Down Servers
or-lb02	be_wordpress	50
or-lb01	be_wordpress	50
or-lb02	be_blog_se	50
or-lb01	be_blog_se	50

Scope in a Nutshell

- Broader Scope Means:
  - Less Notifications, but more information must be included the notification
  - There is no “correct” universal scope, the operator knows best
- Accurate Scope means less alert noise

Overall System status is Bad

Acknowledge alert

View the Rule + Template in the Bosun's Rule Page

Notes: This alert triggers on omreport's "system" status, which *should* be a rollup of the entire system state. So it is possible to see an "Overall System status" of bad even though the breakdowns below are all "Ok". If this is the case, look directly in OMSA using the link below

View Host ny-devsql01 in Opserver

OMSA Login for host ny-devsql01

General Host Info

Service Tag:
Model: PowerEdge R620
OS: Microsoft Windows [Version 6.2.9200]

Power Supplies

Power Supply Id	Status
1	Bad
0	Ok

Batteries

Battery Id	Status
0	

Controllers

Controller Id	Status
0	Ok

Enclosures

Enclosure Id	Status
0_1	Ok

Physical Disks

Physical Disk Id	Status
0_1_0	Ok
0_1_1	Ok

Another Scope Example Hardware

One alert per host, not per broken component

So the point of these examples?

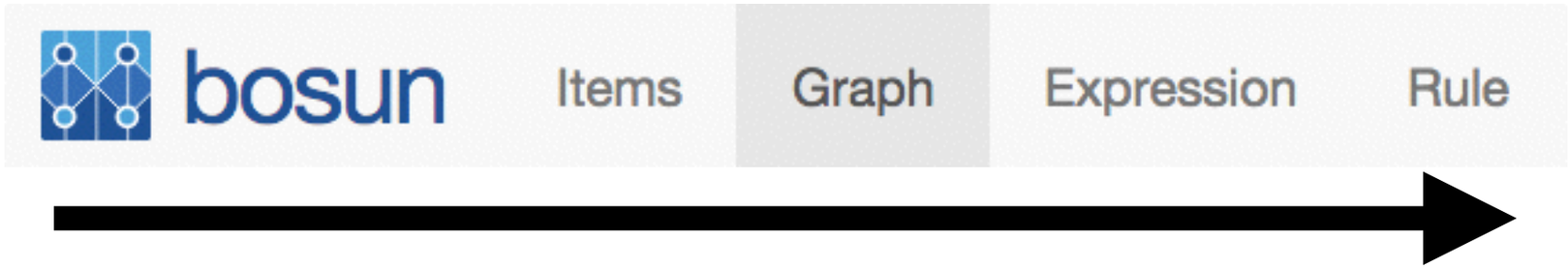
You're Creativity should be the limiting factor

Not the monitoring system

A Closer Look at Bosun

Alerting Workflow

Why we call it an IDE



1. Graph: Generate Expression
2. Expression: Reduce to Single Number
3. Build Rule and Notification: Use Variables, etc
4. Test Rule and Notification: See timeline over history, notification preview
5. Commit

bosun | Items | Graph | Expression | Rule | Test Config | Silence | Submit Data | Short Link | Help

1w-ago | End | Query | Switch Time | Auto Downsample | Auto Refresh | embed | image

y min | y max

Query 1 +

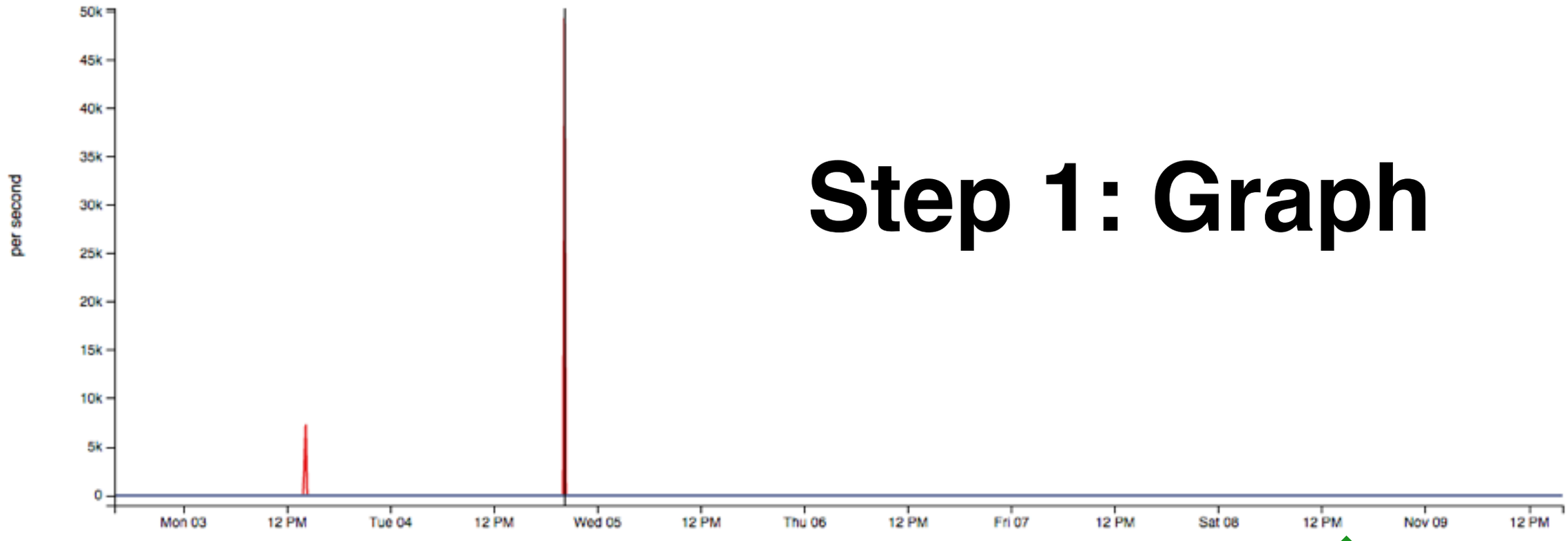
Metric os.net.dropped | **Series Type** auto | **host** "lb"

Aggregator sum | **direction**

Downsample max | **iface**

Window 600s | **iname**

type



Step 1: Graph

Time: 2014/11/04-20:06:29 (4d 19:46:49 ago)
os.net.dropped{host=ny-lb05}: 49.36369k
os.net.dropped{host=or-intlb01,iface=eth0}: 0

Queries

q("sum:600s-max:rate{counter,,1}:os.net.dropped{host="lb"}", "1w", "") | Expression | Rule

Step 2: Expression

- *Reduce a* Series to Single Number: An Alert is Non-Zero

q("sum:rate{counter,,1}:os.net.dropped{host=*lb*}", "5m", "")

q("sum:rate{counter,,1}:os.net.dropped{host=*lb*}", "5m", "")

Date yyyy-mm-dd Time HH:MM Test Rule Image

Results Graph

Queries

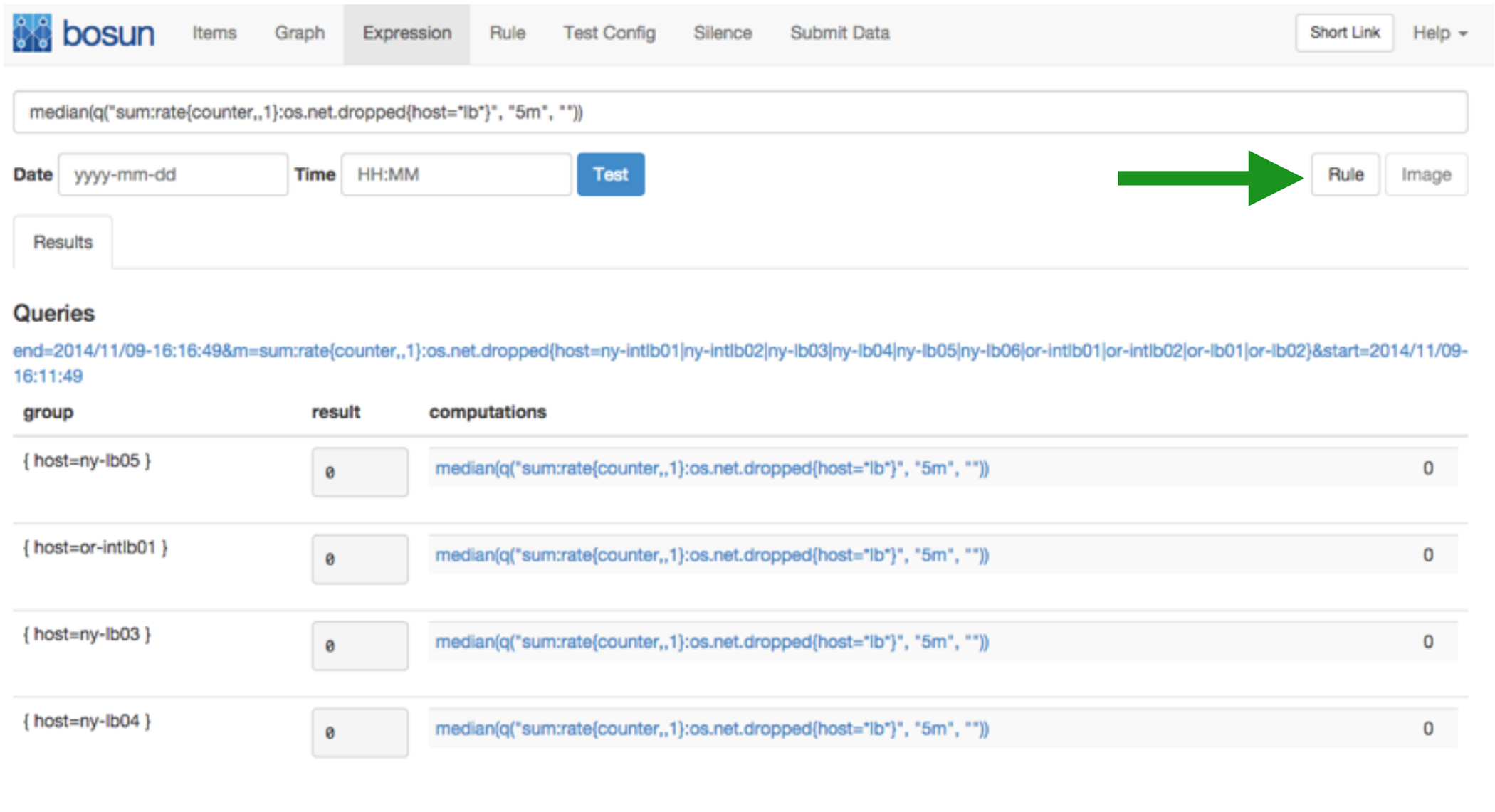
end=2014/11/09-15:58:12&m=sum:rate{counter,,1}:os.net.dropped{host=ny-intlb01|ny-intlb02|ny-lb03|ny-lb04|ny-lb05|ny-lb06|or-intlb01|or-intlb02|or-lb01|or-lb02}&start=2014/11/09-15:53:12

group	result	computations
{ host=or-lb02 }	{ "1415548406": 0, "1415548421": 0, "1415548436": 0, "1415548451": 0, "1415548466": 0, "1415548481": 0, "1415548496": 0, "1415548511": 0, "1415548526": 0, "1415548541": 0, "1415548556": 0, "1415548571": 0, "1415548586": 0, "1415548601": 0, "1415548616": 0, "1415548631": 0, "1415548646": 0, "1415548661": 0, "1415548676": 0, "1415548691": 0 }	
{ host=or-lb01 }	show	

Step 2: Expression

- *Reduce to single number with reduction function like: median() a* Series to Single Number: An Alert is Non-Zero

median(q(“sum:rate{counter,,1}:os.net.dropped{host=*lb*}", "5m", “”))



Step 3: Rule
(Trigger Condition)

- Test against history and see how it performs over history.

- Try different thresholds or reduction functions (i.e. sum(), avg(), percentile(…, .95) and see how they perform

- Use variables (Actually just dumb string replacement) to make the alert more readable and to add data to reference in the template

Step 3: Rule

(Notification Template)

- Add Notes
- Add Graphs
- Possibly Change Scope
- Include other queries that provide more information and context. Display those as nice HTML tables to Graphs

Alert

Load Alert Definition

```
alert test {
        template = test
        $notes = Alert on dropped packets....
        $time = 5m
        $dropped = q("sum:rate{counter,,1}:os.net.dropped{host="|b"}", "$time", "")
        $dropped_by_interface_dir =
change("sum:rate{counter,,1}:os.net.dropped{host="|b",face=",direction="}",
"$time", "")
        $graph_by_interface = q("sum:1m-
max:rate{counter,,1}:os.net.dropped{host=",face="}", "30m", "")
        $median_dropped = median($dropped)
        $total_dropped = change("sum:rate{counter,,1}:os.net.dropped{host="|b"}",
"$time", "")
        $max_dropped = max($dropped)
        warn = $max_dropped
}
```

Alert Def

Template

Load Template Definition

```
template test {
    body = `
    <p>Notes: {{.Alert.Vars.notes}}
    <h2>Stats for the Last {{.Alert.Vars.time}}</h2>
    Median: {{.Eval .Alert.Vars.median_dropped | printf "%.2f"}}
    <br>Total: {{.Eval .Alert.Vars.total_dropped | printf "%.2f"}}
    <br>Max: {{.Eval .Alert.Vars.max_dropped | printf "%.2f"}}
    <h2>By Interface and TX/RX</h2>
    <table>
    <tr><th>Interface</th><th>Direction</th><th>Total Packets</th></tr>
    {{ range $r := .EvalAll .Alert.Vars.dropped_by_interface_dir}}
      {{ if $r.Group.Subset $.Group }}
        <tr>
          <td>{{$r.Group.iface}}</td>
          <td>{{$r.Group.direction}}</td>
          <td>{{$r.Value}}</td>
        </tr>
      {{end}}
    {{end}}
    </table>
    <h2>By Interface</h2>
    {{.Graph .Alert.Vars.graph_by_interface}}
    `
    subject = {{.Last.Status}}: Dropped Packets: {{.Eval .Alert.Vars.median_dropped |
printf "%.2f"}}{{if .Alert.Vars.unit_string}}{{.Alert.Vars.unit_string}}{{end}} on
{{.Group.host}}
}
```

Template Def

From 2014-11-03 14:13 **To** yyyy-mm-dd HH:MM **Intervals** 144 **Step Duration (m)** 4 **Test**

Email **Template Group** host=ny-lb05

Shift-enter to test. If neither From nor To are present, will run for now. If one present, at that timestamp. If both present, that timespan. Times default to midnight if not set.

Results | Template | Timeline

Subject

warning: Dropped Packets: 19776.87 on ny-lb05

Body

Notes: Alert on dropped packets....

Stats for the Last 5m

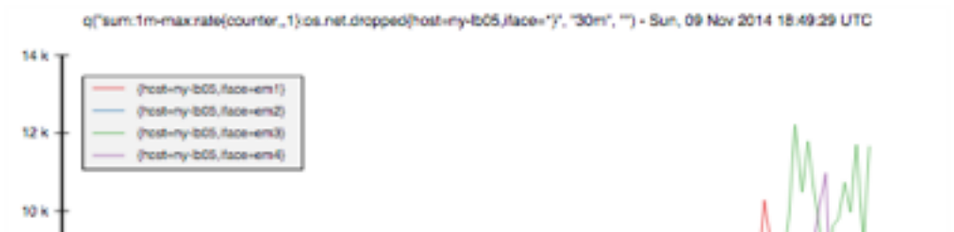
Median: 19776.87
Total: 5459587.00
Max: 24675.93

By Interface and TX/RX

Interface	Direction	Total Packets
em1	in	1220040.00
em1	out	0.00
em2	in	159344.00
em2	out	0.00
em3	in	2357636.00
em3	out	0.00
em4	in	1722567.00
em4	out	0.00

Email Preview

By Interface



```
alert test {
  template = test
  $notes = Alert on dropped packets....
  $time = 5m

  $dropped = q("sum:rate{counter,,1}:os.net.dropped{host=*lb*}", “$time")

  $dropped_by_interface_dir = change("sum:rate{counter,,
1}:os.net.dropped{host=*lb*,iface=*,direction=*}", "$time", “")

  $graph_by_interface = q("sum:1m-max:rate{counter,,1}:os.net.dropped{host=*,iface=*}",
"30m", “")

  $median_dropped = median($dropped)

  $total_dropped = change("sum:rate{counter,,1}:os.net.dropped{host=*lb*}", "$time", “")

  $max_dropped = max($dropped)
  warn = $max_dropped
}
```

```
template test {
    body = `
    <p>Notes: {{.Alert.Vars.notes}}
    <h2>Stats for the Last {{.Alert.Vars.time}}</h2>
    Median: {{.Eval .Alert.Vars.median_dropped | printf "%.2f"}}
    <br>Total: {{.Eval .Alert.Vars.total_dropped | printf "%.2f"}}
    <br>Max: {{.Eval .Alert.Vars.max_dropped | printf "%.2f"}}
    <h2>By Interface and TX/RX</h2>
    <table>
    <tr><th>Interface</th><th>Direction</th><th>Total Packets</th></tr>
    {{ range $r := .EvalAll .Alert.Vars.dropped_by_interface_dir}}
        {{ if $r.Group.Subset $.Group }}
            <tr>
                <td>{{$r.Group.iface}}</td>
                <td>{{$r.Group.direction}}</td>
                <td>{{$r.Value | printf "%.2f"}}</td>
            </tr>
        {{end}}
    {{end}}
    </table>
    {{.Graph .Alert.Vars.graph_by_interface}}
`

    subject = {{.Last.Status}}: Dropped Packets:
{{.Eval .Alert.Vars.median_dropped | printf "%.2f"}}
{{if .Alert.Vars.unit_string}}{{.Alert.Vars.unit_string}}{{end}} on
{{.Group.host}}
}
```

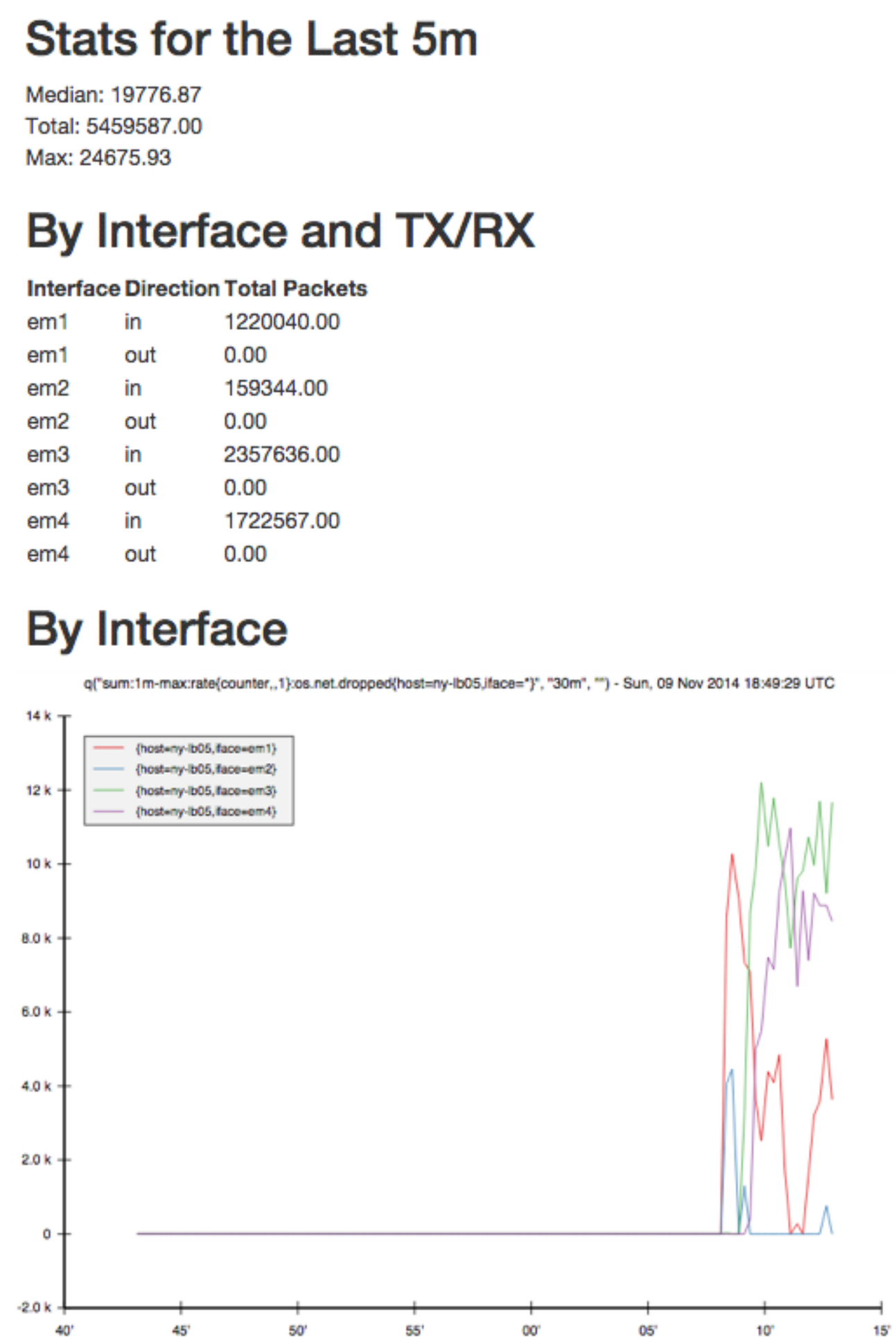
Stats for the Last 5m

Median: 19776.87
Total: 5459587.00
Max: 24675.93

By Interface and TX/RX

Interface	Direction	Total Packets
em1	in	1220040.00
em1	out	0.00
em2	in	159344.00
em2	out	0.00
em3	in	2357636.00
em3	out	0.00
em4	in	1722567.00
em4	out	0.00

By Interface



Bosun’s Architecture

scollector agent

Built-in Collectors

External Collectors (Scripts)

Built-in Pollers

Data Queue

Instrumented Applications

Data Sources

JSON HTTP

JSON HTTP

Bosun

HTTP JSON REST API

Web Interface

Config

Scheduler

OpenTSDB

scollector

- Lots of Built-in Collectors
- Auto discovers applications like Redis, IIS, SQL Server, MySQL, etc
- Windows and Linux are both first class systems: (Go deep into WMI to and /proc to get you raw counters) …
- Queues data when Bosun can’t be reached
- Counters are good: Don’t lose information between sends or run the risk of aliasing

scollector con't

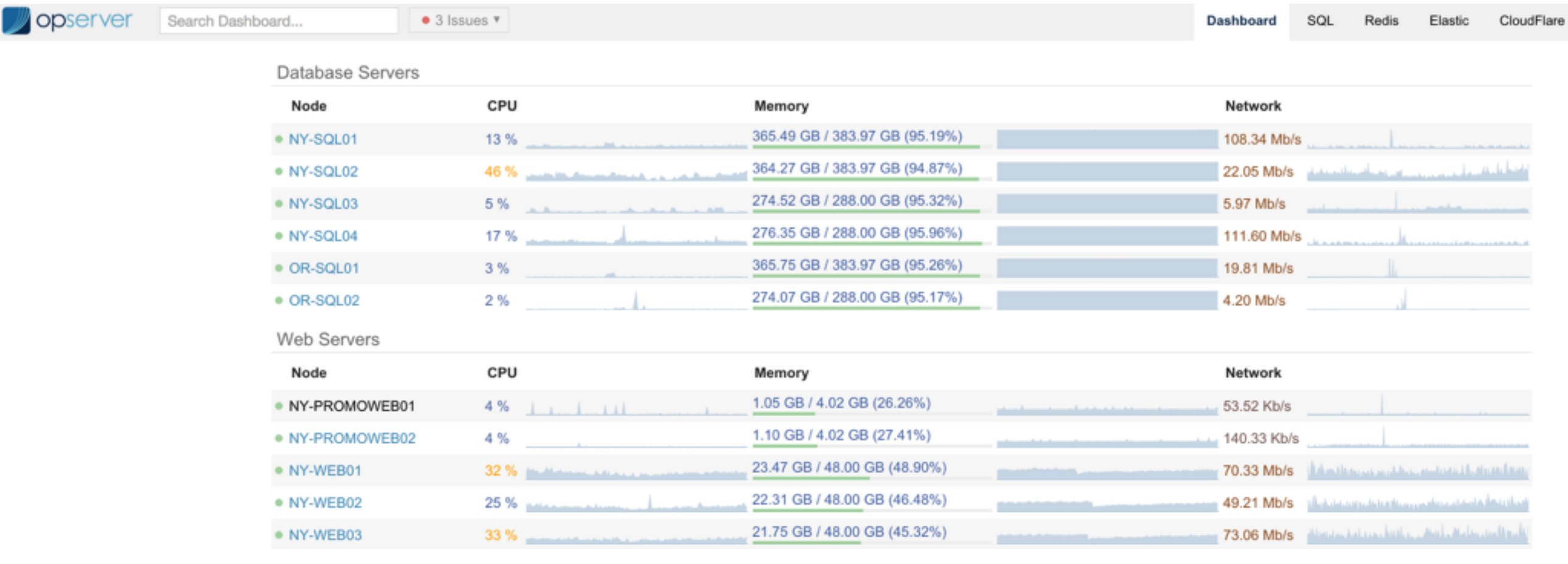
- Sends Metadata: Units, Description, type of Metric (Counter vs Gauge)
- Can run external collectors (scripts)
- No dependencies on Libraries or runtime: Just a Compiled Binary
- Collects every 15 seconds to help have enough data to detect anomalies sooner

Our Time series are stored using OpenTSDB

- Extremely storage efficient: Never have to roll up data
- Seems to be fast
- Helps with aggregation when data is designed well
- Scales using HBase

Integrates with Opserver

- Opserver is another monitoring open source project form Stack Exchange
- Opserver is a very refined dashboard, and also has very impressive SQL server views (Execution Plans, Top Queries)
- Also HAProxy Administration, Redis, and elastic views



What have I learned about Alerting Best Practices?

Not Enough, need your help

But a few things are…

Best Practices

- One alert per object: i.e. Don't have a forecast *and a* threshold alert for disk. Combine the logic of both so you only get one notification

- Discipline to tune alerts and silence things during maintenance is still required

- Try to see the alert how the receiver will see it in order to provide context (This is hard, include notes, units, etc)

I WANT YOU

- Your creativity in designing alerts
- Finding out where bugs are and what needs documenting
- Bosun Contributors: Know someone who likes Go?
- Scollector Contributors: More data, tuning, etc
- To come work at Stack Exchange!

Go Try It

- Check out our Getting Started Guide and Examples on http://bosun.org

- Get the Docker Image: `docker run -d -p 4242:4242 -p 8070:8070 stackexchange/bosun`

- Install scollector to get metrics from Windows and Linux

- Tell us what is broken, make creative alerts!

Recommend Reading

- http://www.kitchensoap.com/2013/07/22/owning-attention-considerations-for-alert-design/

- Monitoring Chapters of The Practice of Cloud System Administration: Designing and Operating Large Distributed Systems, Volume 2