Debugging from the outside → in

Edd Steel

@eddsteel

SOA at HootSuite

- LAMP → SOA
- Lots of new Scala code
- Lots of new Akka code
- Production-ready means something
- (we're hiring!)

OH at the reactive meetup

What sucks about this stuff?

The tooling isn't there yet.

- IDE support
- profiling, debugging
- distributed systems are hard.

Does it work?

Does it work?

- Compiler
- Static Analysis
- Unit Testing
- Property Testing
- Integration Testing

OK, But does it work?

Test the whole system

- Load testing
- Profiling
- Typesafe Console
- Log analysis

Problem 1

tool's view != dev's view

- Actors and Futures not threads
  - `ActorSystem-akka.actor.worker-dispatcher-3`
- Message-passing components and anonymous functions
  - `Module$InnerModule$Foo$$anonfun$bar$1$$anonfun$apply$`
- Once our system is made to work with the tools, is it still the same system?

Problem 2

This isn't very agile.

Dev → Vagrant → Staging → Load Testing → Production

Problem 3

- SOA pushes system integration to deploy time.
- It's still your problem.

Plugging the gaps

We need a tool that will

- work in production, without impacting performance
- provide a central view of a distributed system
- Organise information flexibly
- Tell us, at a minimum
  - Did X happen and where?
  - How often?
  - How long did it take?
  - What was the impact?
  - `???`

MVP


MACGYVER
- THE ONLY TOOL YOU'LL EVER NEED
AS SEEN ON TV
MACGYVER
MULTITOOL
GLASSLITE
"A paperclip can be a wonderous thing. More times than I can remember one of these has gotten me out of a tight spot..."

Monitoring utilities?

- Requirements sound like characteristics of our monitoring utilities
- 3 flavours
  - Alerts
  - Graphs
  - Aggregated logs

Graphs

Killer App: tracking metrics

- Build dashboards that show key system metrics
  - execution time
  - request count
  - coffee supply
- See impact of code changes
- Monitor outages, early warnings

Graph stack

- statsd-client/ diamond
- statsd
- graphite
- graphite dashboards

Statsd/Graphite Features

- UDP and sampling from multiple sources
- counters, gauges, timing (mean/ 90th percentile/ 99th percentile)
- hierarchical data series
  - `{system}.{host}.{actor}.{function}.time`
- aggregation, combination, calculation

Log aggregators

Killer app - logging exceptions

- Post mortems
- User tracing across systems
- rare, obscure bugs
- edge cases and exotic browser/OS/device combinations
- generate test data

Log stack

- udp-logger
- logstash
- elastic search
- kibana
- hadoop

Elastic Search features

- UDP support from multiple sources
- schemaless, structured messages + search
- map/reduce batch jobs

So, monitoring utilities?

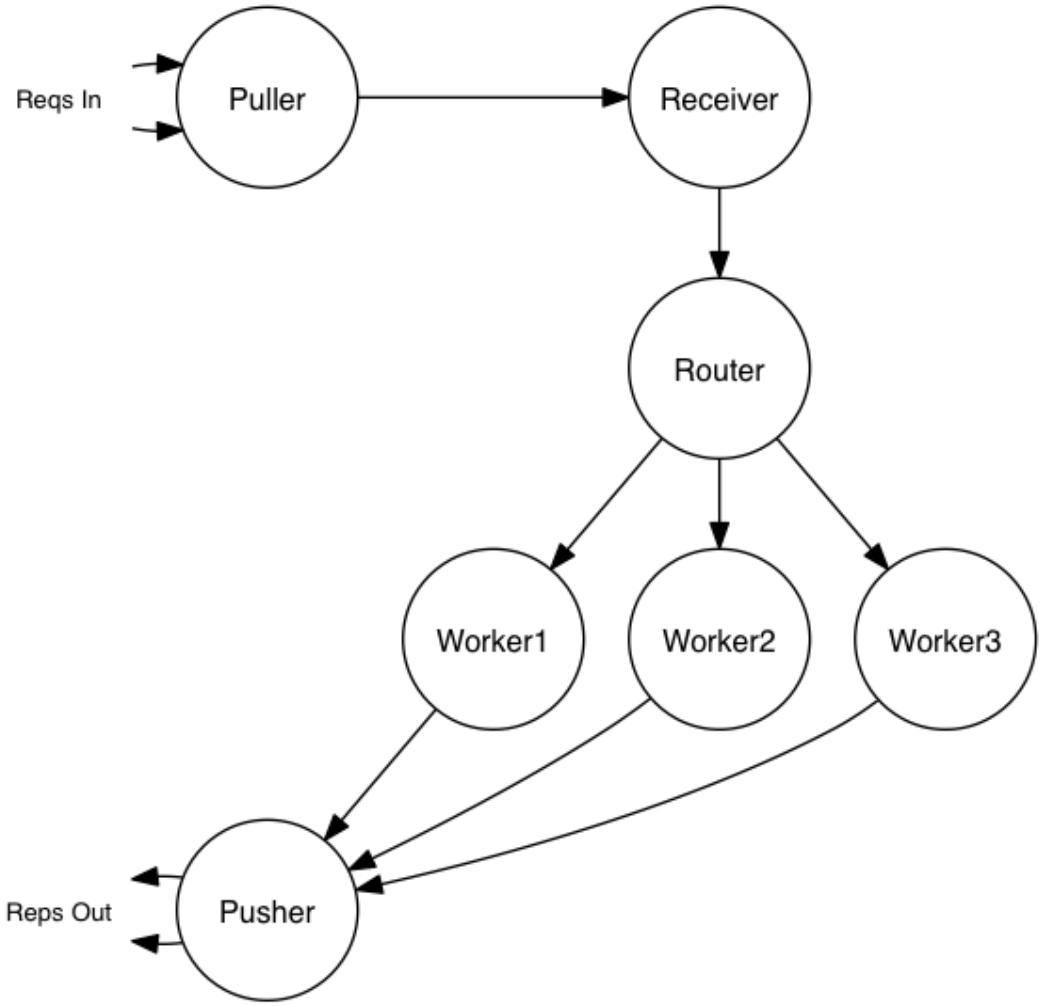
- both work in production, without impacting performance
- provide a central view of a distributed system
- organise information flexibly

- Graphing X shows
  - if it happened
  - where (if that's in the key)
  - how long it took
  - how the system looked at the same time.

- Logging X shows
  - if it happened
  - where (if it's part of the message)
  - the context
  - trends around X (kibana or hadoop)

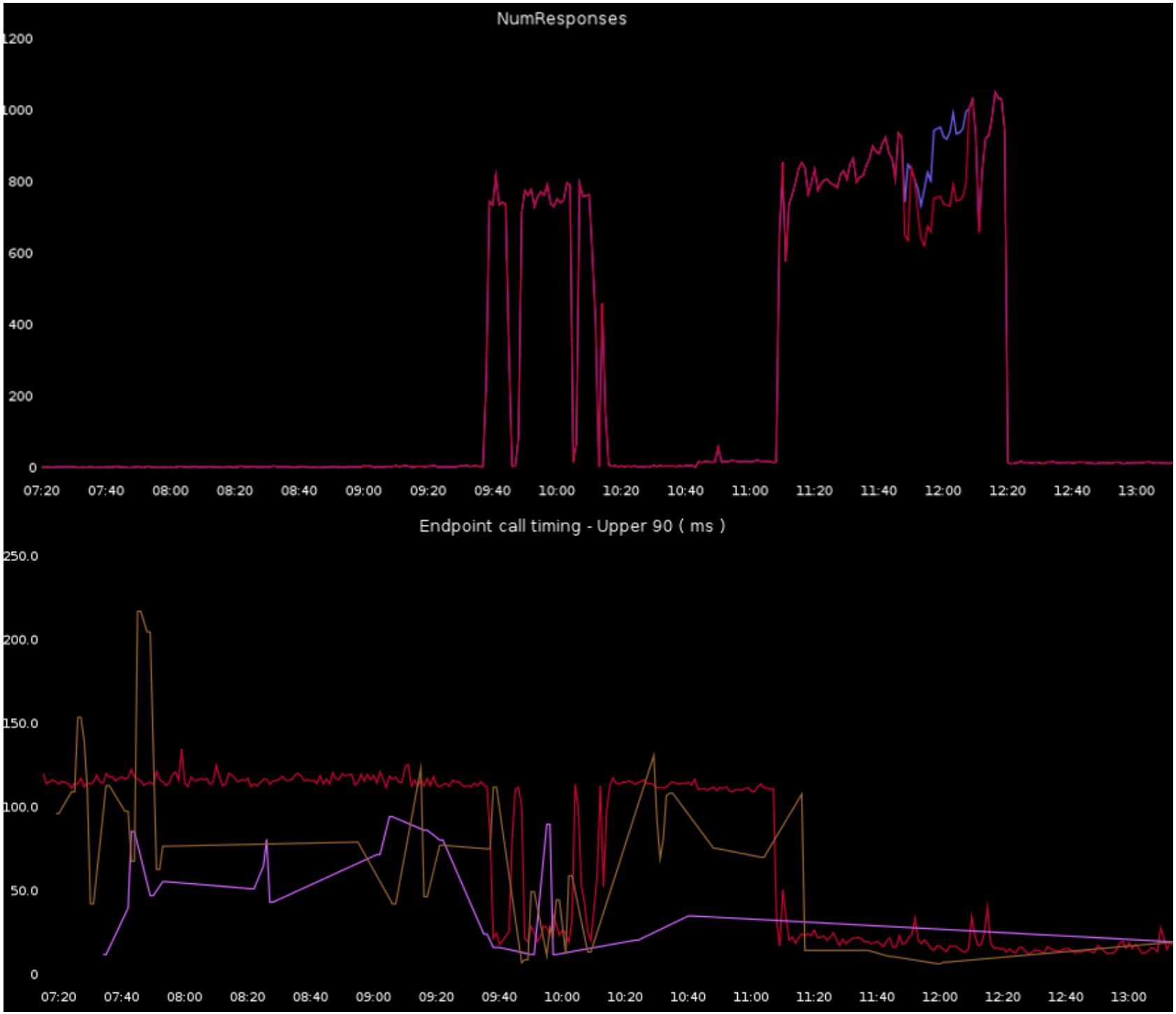
Graph Example I

Our System

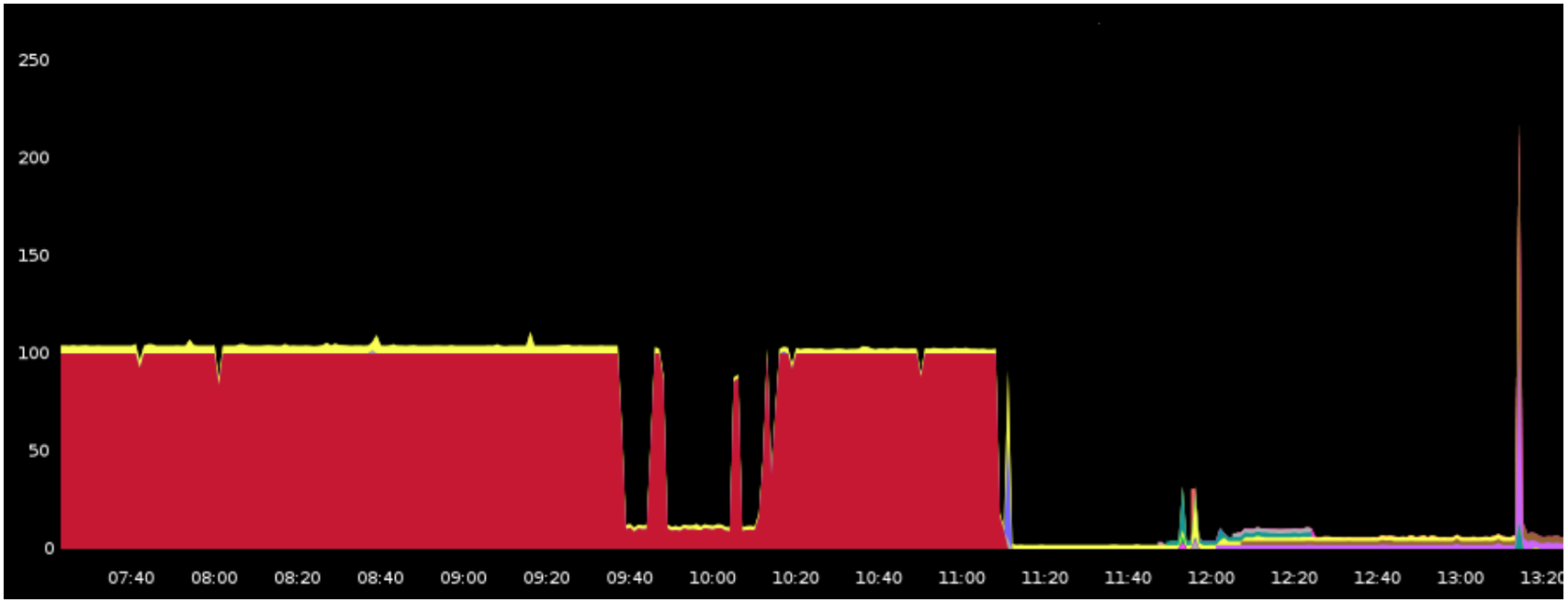

Reqs In
Puller
Receiver
Router
Worker1
Worker2
Worker3
Reps Out
Pusher

The Metric: Execution time

- System was underperforming with low load
- Performance improved when we increased load
- Testing showed no issue with receiving across two sockets
- Requests In matched Responses Out

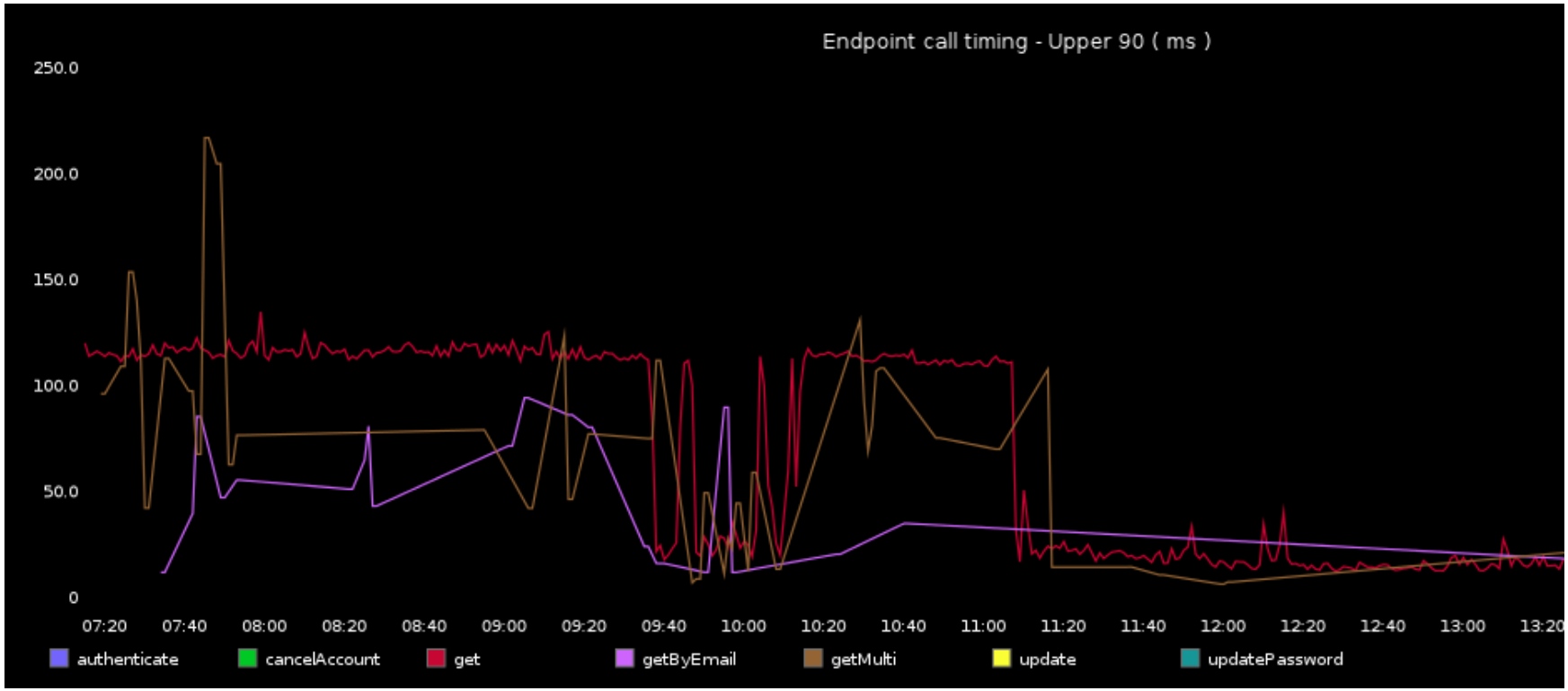

NumResponses
1200
1000
800
600
400
200
0
07:20
07:40
08:00
08:20
08:40
09:00
09:20
09:40
10:00
10:20
10:40
11:00
11:20
11:40
12:00
12:20
12:40
13:00
Endpoint call timing - Upper 90 (ms)
250.0
200.0
150.0
100.0
50.0
0
07:20
07:40
08:00
08:20
08:40
09:00
09:20
09:40
10:00
10:20
10:40
11:00
11:20
11:40
12:00
12:20
12:40
13:00

A Clue



puller to receiver slice is large

A Clue



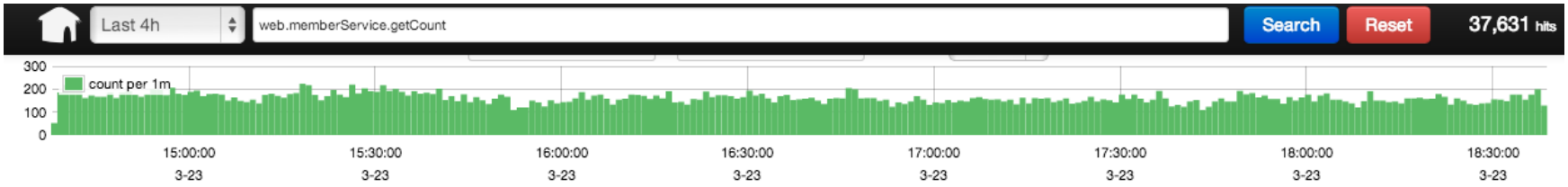
minimum time, at low load was about the same as socket timeout

The Hypothesis

- waiting for both sockets to have requests, instead of reading and processing off whichever had work.
- big improvement in clarity
  - ~~some part of the system is causing delays~~
  - there's a bug in our polling code

Log Example I

service client logs how many service calls are made in a web request

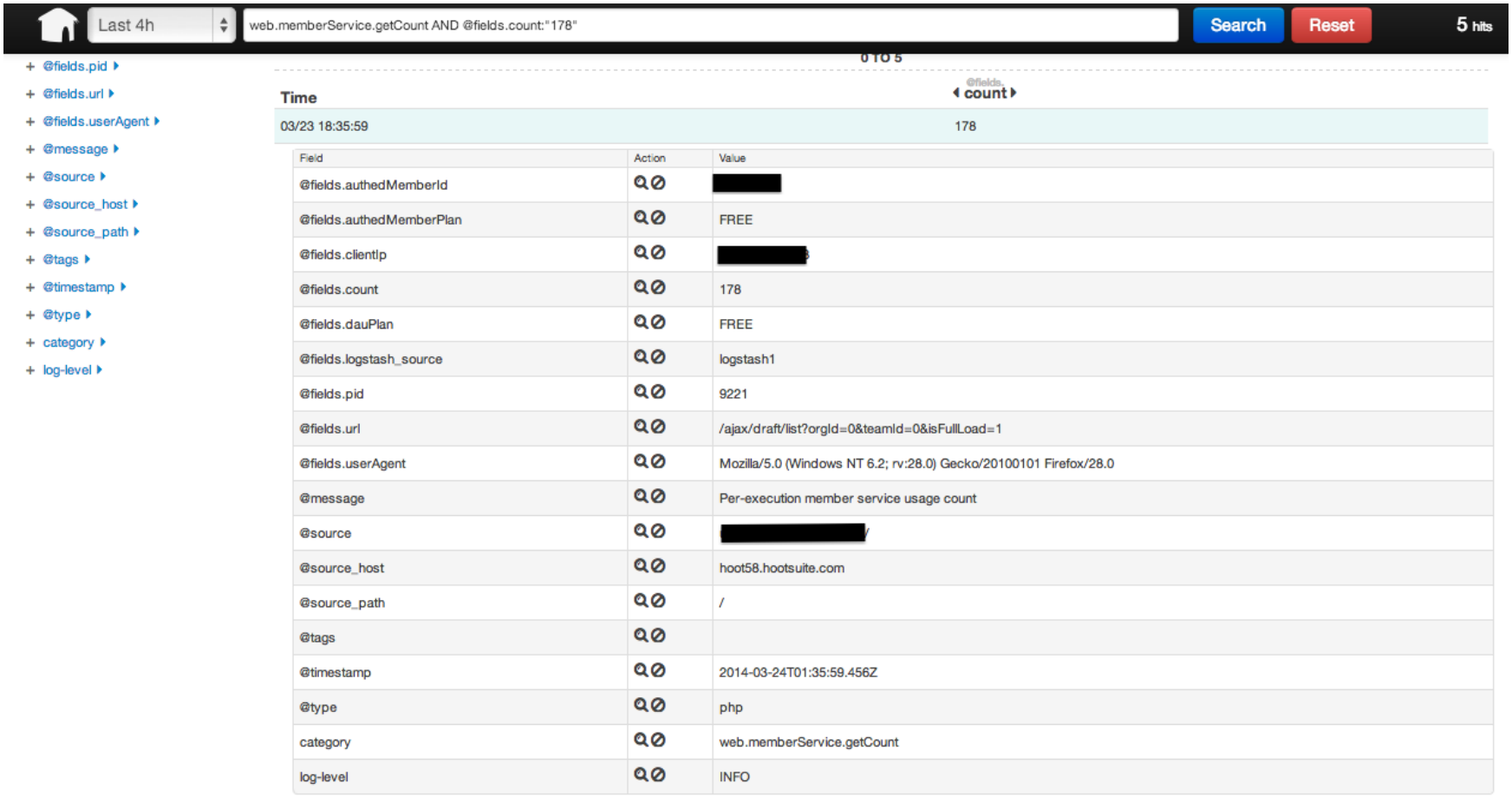


Quick analysis of @fields.count field(s)

back to logs

This analysis is based on the **2000 most recent** events for your query in your selected timeframe.

Rank	@fields. count	Count	Percent	Action
1	3	1292	64.6%	
2	4	414	20.7%	
3	5	93	4.65%	
4	6	70	3.5%	
5	7	55	2.75%	
6	10	10	0.5%	
7	32	8	0.4%	
8	11	7	0.35%	
9	12	6	0.3%	
10	16	6	0.3%	
11	13	5	0.25%	
12	8	4	0.2%	
13	38	3	0.15%	
14	40	3	0.15%	
15	33	3	0.15%	
16	20	3	0.15%	
17	39	2	0.1%	
18	77	2	0.1%	
19	9	2	0.1%	
20	37	2	0.1%	
21	178	2	0.1%	
22	96	1	0.05%	


Last 4h
web.memberService.getCount AND @fields.count:"178"
Search
Reset
5 hits
+ @fields.pid
+ @fields.url
+ @fields.userAgent
+ @message
+ @source
+ @source_host
+ @source_path
+ @tags
+ @timestamp
+ @type
+ category
+ log-level
0 TO 5
Time
@fields.count
03/23 18:35:59
178
Field
Action
Value
@fields.authedMemberId
@fields.authedMemberPlan
FREE
@fields.clientIp
@fields.count
178
@fields.dauPlan
FREE
@fields.logstash_source
logstash1
@fields.pid
9221
@fields.url
/ajax/draft/list?orgId=0&teamId=0&isFullLoad=1
@fields.userAgent
Mozilla/5.0 (Windows NT 6.2; rv:28.0) Gecko/20100101 Firefox/28.0
@message
Per-execution member service usage count
@source
@source_host
hoot58.hootsuite.com
@source_path
/
@tags
@timestamp
2014-03-24T01:35:59.456Z
@type
php
category
web.memberService.getCount
log-level
INFO

A process emerges

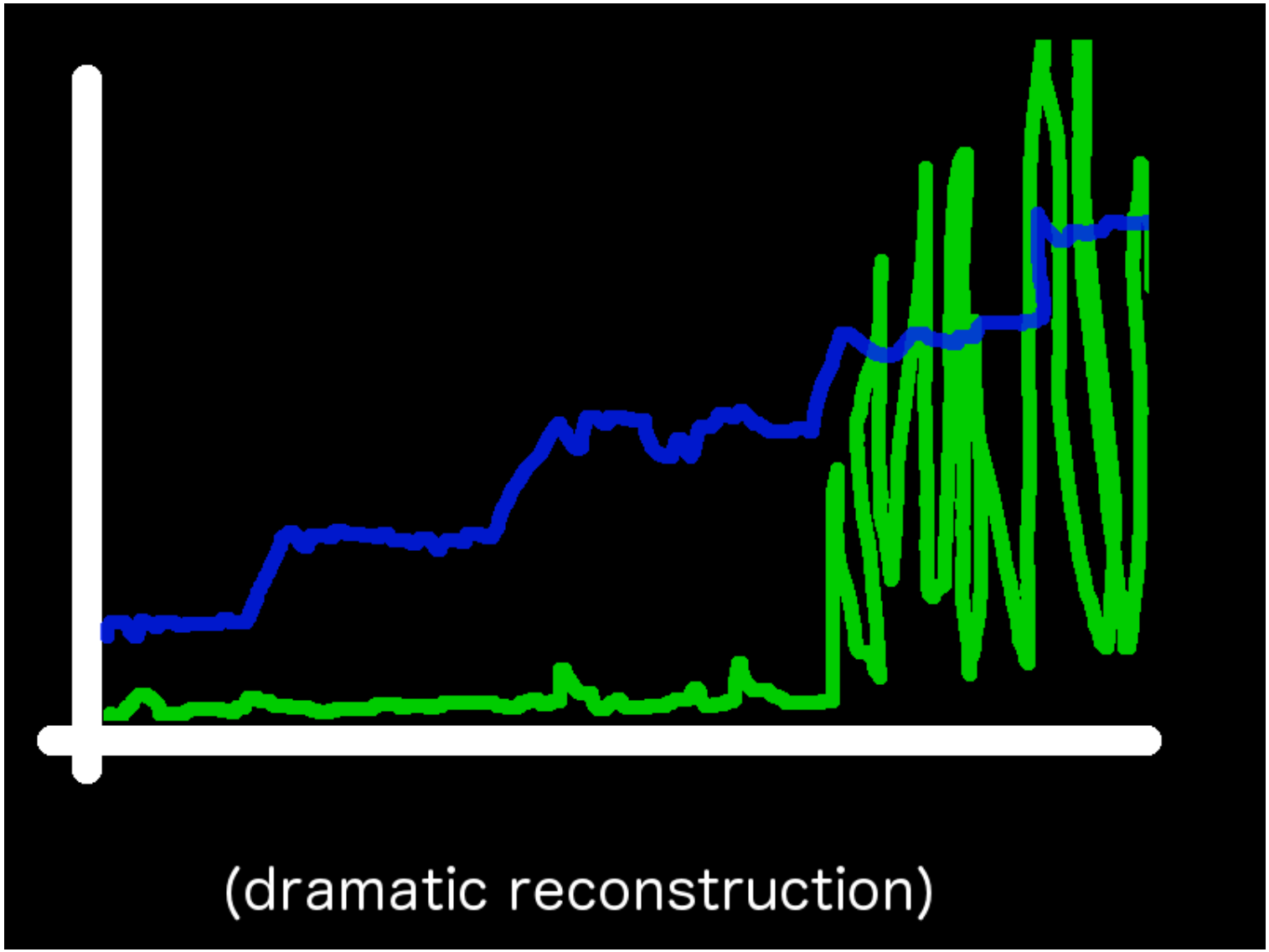
Our "process"

- Identify a metric (or add one)
- Find a clue
- Form a hypothesis
- Fix and watch the metric change (else repeat)

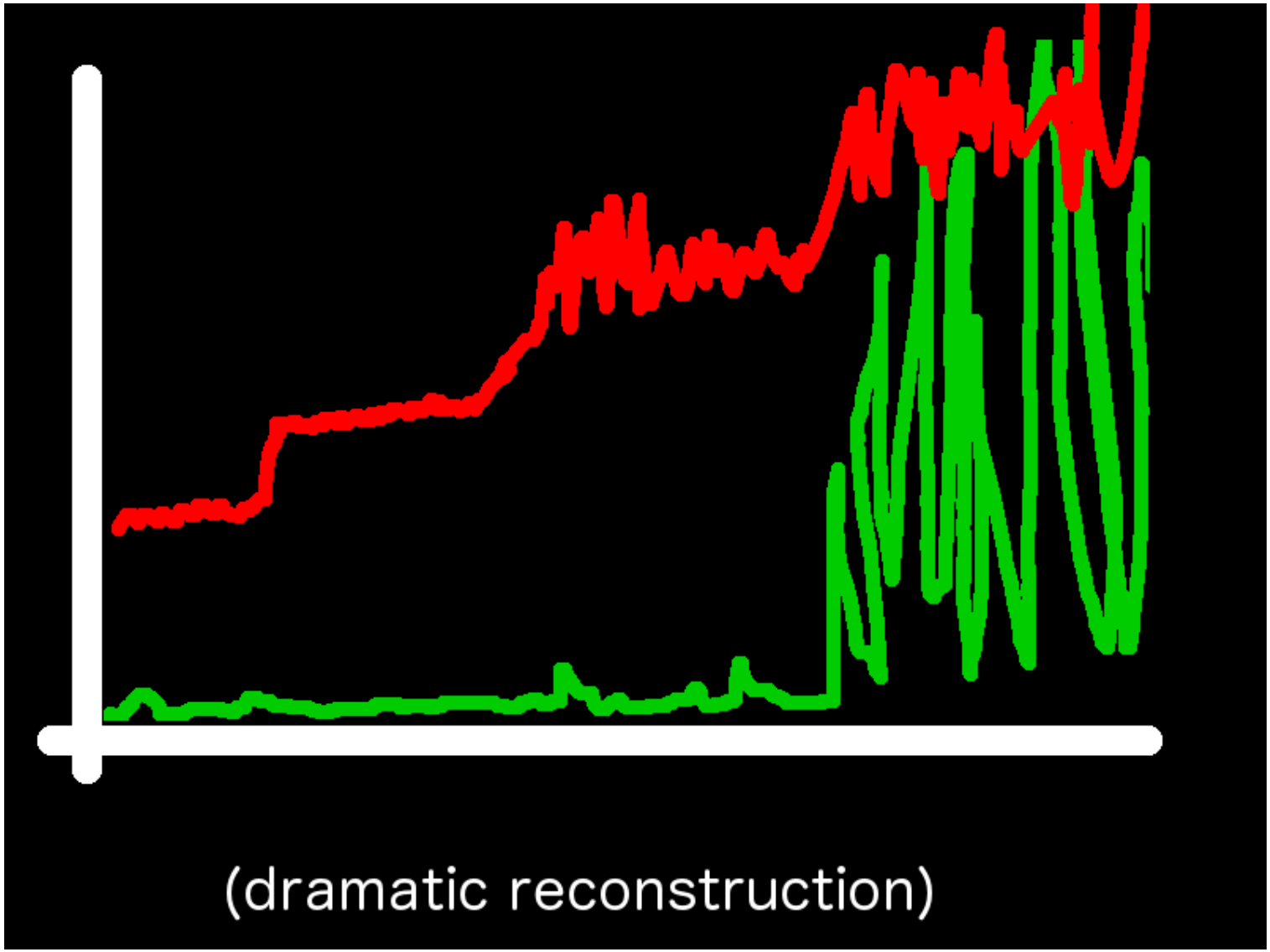
Graph Example II

Example 1: The Metric

- request timeouts increased when load increased
- sudden change
- cause unclear

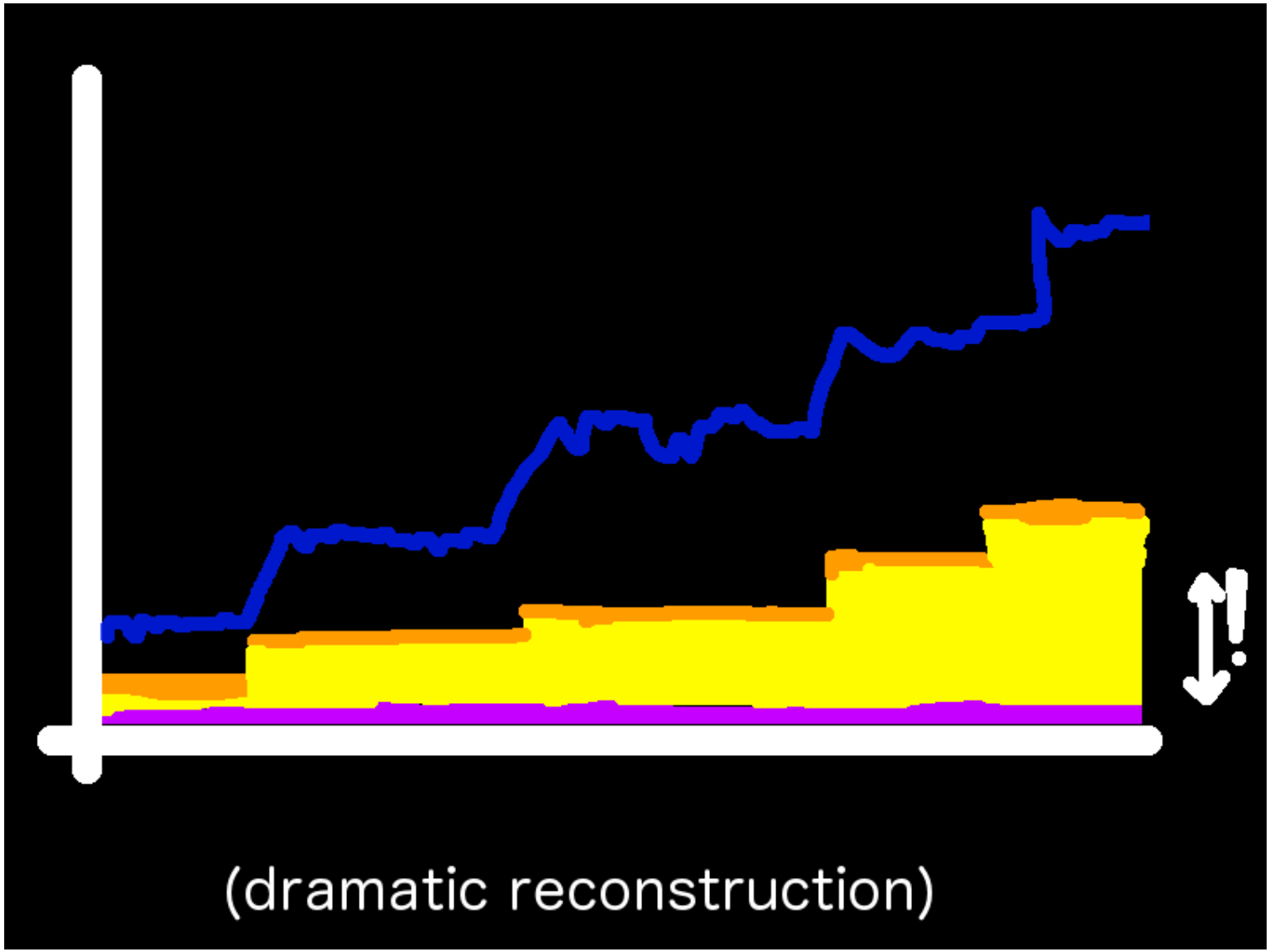

(dramatic reconstruction)

A Clue



size of request batches correlated, increased

A Clue



problem when upper time b/w puller and worker == request timeout

Hypothesis

- bug: no cap on request batch size
- first requests in batch timed out before the batch was sent.

Log Example II

- DB replication during rollout
- PHP and scala systems, old and new DBs, tungsten replicator to sync them
- Soft-launch
  - percentage use new then old (and skip replication)
  - rest use just old (and replicate)

03/25 04:12:50	web http request: ajax: /ajax/analytics/save
03/25 04:12:11	web http request: ajax: /ajax/analytics/save
03/25 04:05:47	Per-execution member service usage count
03/25 04:05:47	ADD_TO_SKIP_LIST table:hootsuite.member id:[redacted]
03/25 04:05:47	SKIP sql_cmd:UPDATE tablehootsuite.member details: - ROW# = 0 -- DROPPED - (35: aria_status_code) = 0 -- - COL(2: email) = [redacted] - COL(5: is_email_confirmed) = 1 - COL(7: full_name) = [redacted] - COL(14: modified_date) = 2014-03-25 04:05:47 - KEY(1: member_id) = [redacted] - KEY(2: email) = [redacted] - KEY(5: is_email_confirmed) = 0 - KEY(7: full_name) = [redacted] - KEY(14: modified_date) = 2014-03-25 03:22:51
03/25 04:05:47	Debug: SDK client: update called successfully
03/25 04:05:47	web http request: ajax: /ajax/settings/save-account
03/25 04:05:32	Debug: SDK client: exception occurred in client::update
03/25 04:05:32	Debug: SDK client: update called successfully
03/25 04:05:32	web http request: ajax: /ajax/settings/save-account
03/25 03:56:19	web http request: ajax: /ajax/stream/save-box
03/25 03:56:15	web http request: ajax: /ajax/stream/save-box
03/25 03:56:05	web http request: ajax: /ajax/stream/save-box
03/25 03:55:40	web http request: ajax: /ajax/stream/save-box

Replicator, skipping replication.

Scala receiving request

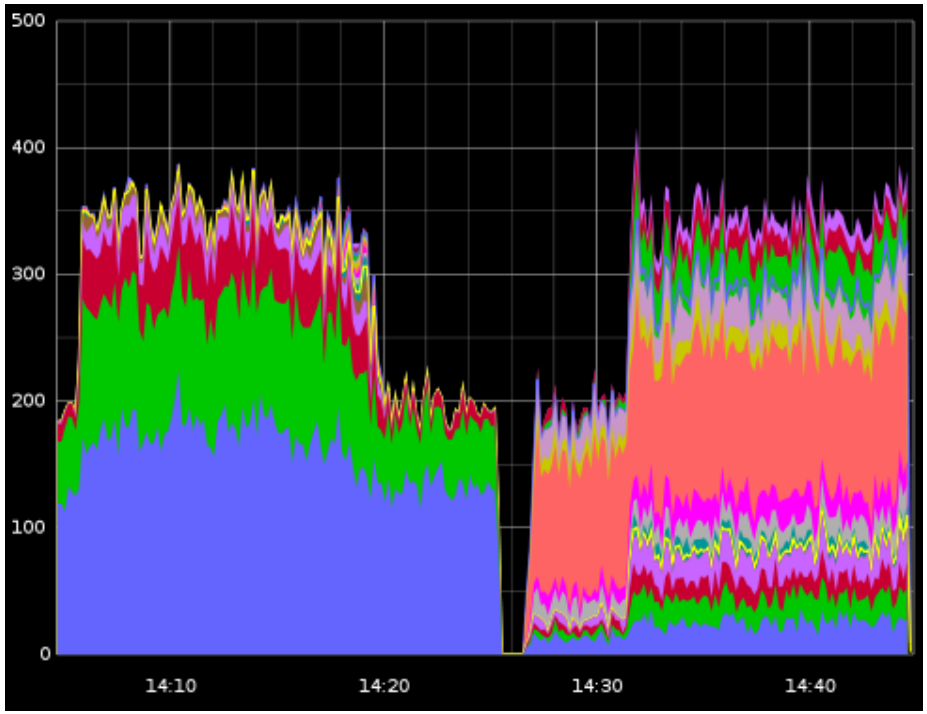
PHP receiving request

User adding streams then updating profile

Progress of an update through old and new systems

The kinds of questions we've been asking and answering

- what's my workload distribution like?
- what's the best number of workers for this traffic level?
- which part of our pipeline is the bottleneck?
- did my config change have the desired effect?


500
400
300
200
100
0
14:10
14:20
14:30
14:40

- are my actors keeping up with their work?
- are those lost messages due to a bug or poor performance?
- are my assumptions correct? Does *x* affect *y*?
- did the last two weeks of work actually make things better?

Not to mention

- Is twitter down?
- Did someone redeploy?
- Is instance 2 amazony today?

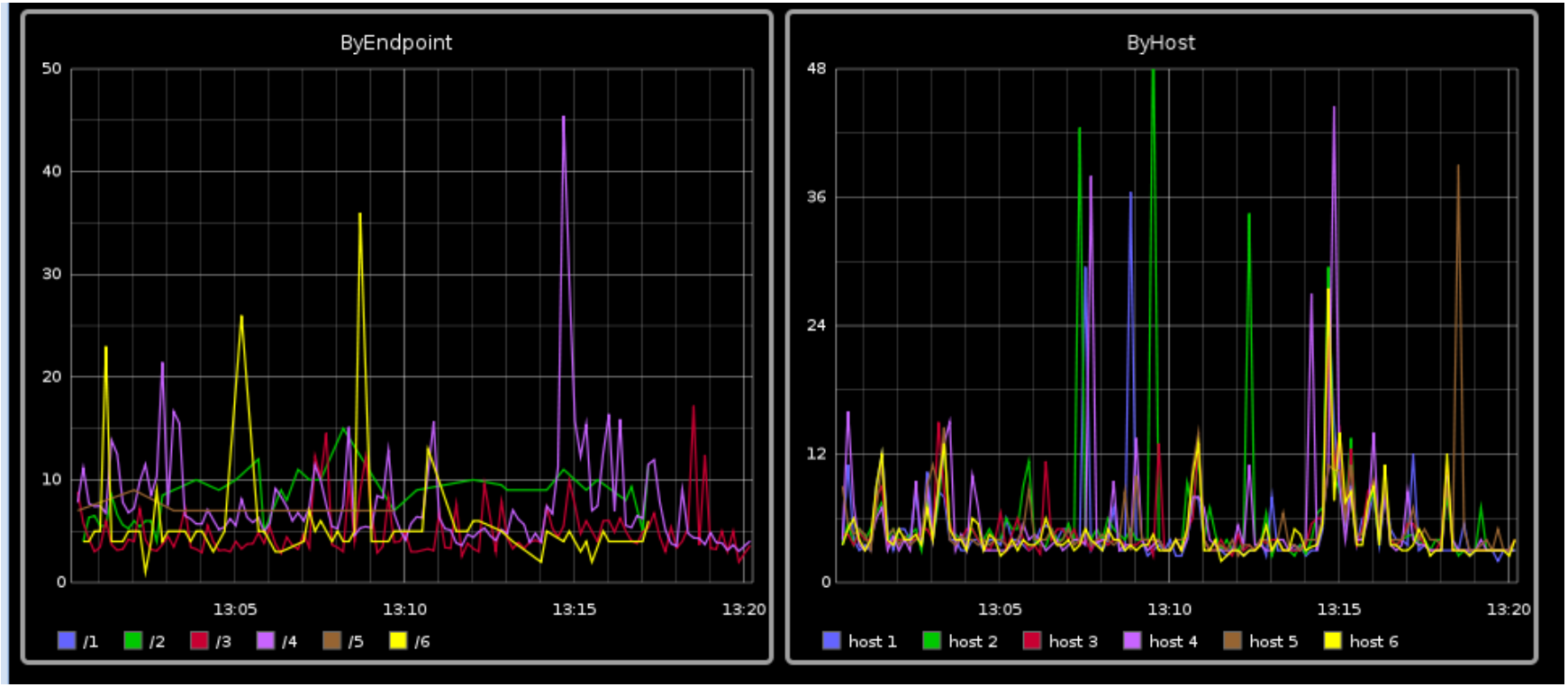
Guidelines

The regular rules apply

- don't fix what you haven't measured
- don't prematurely optimise
- test your assumptions
- remove noise

Organising metrics

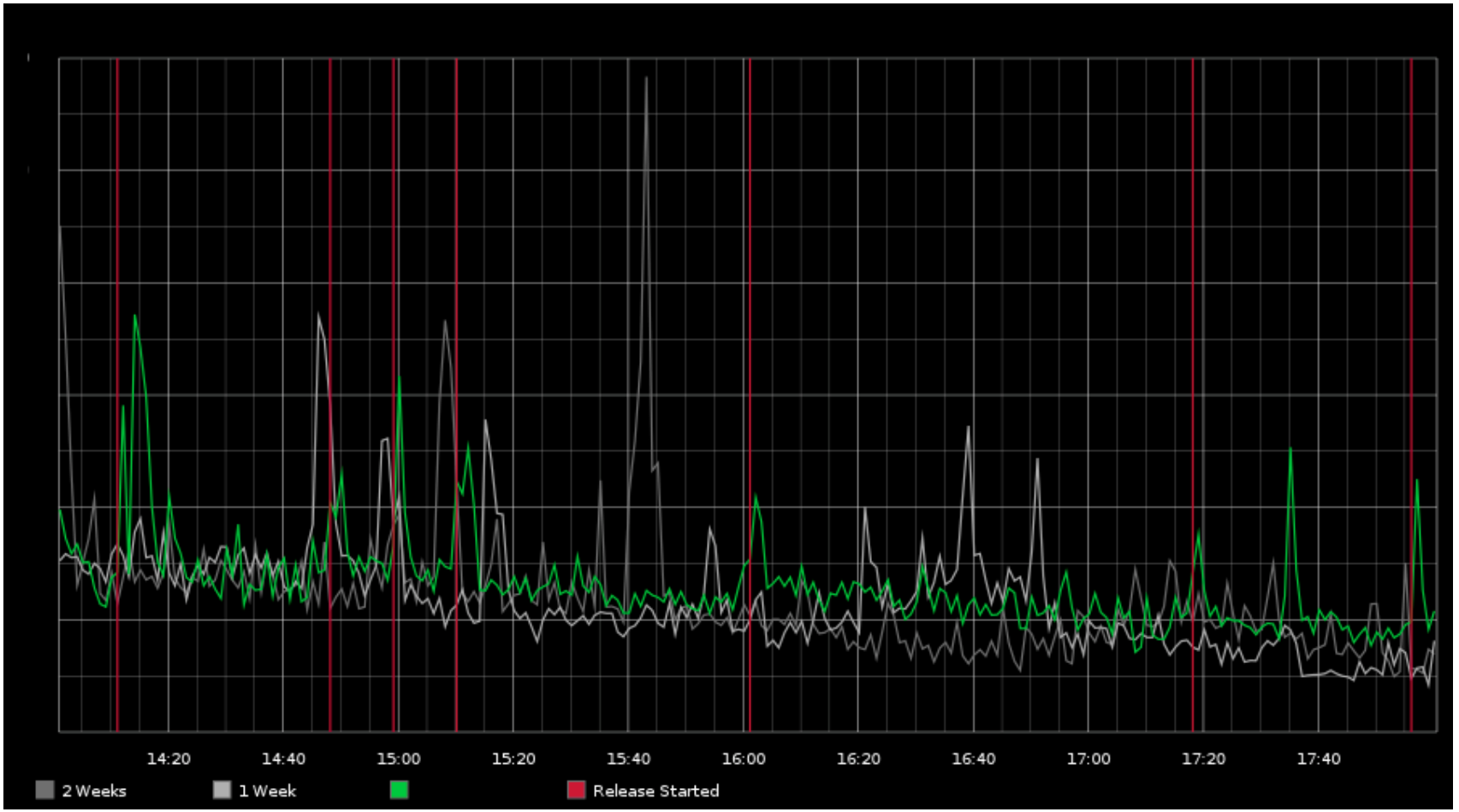
- record fine-grained, then aggregate



- `requests.*.endpoint1` / `requests.host1.*`

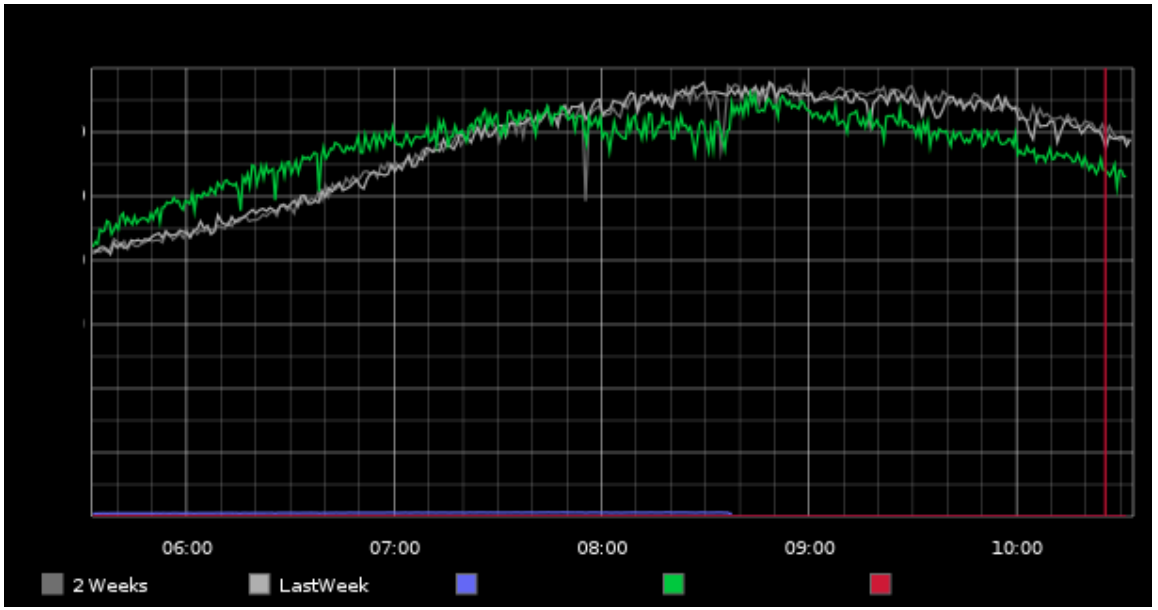
Misc.

- sample if necessary.
- system stats are useful
- combine graphs to demonstrate correlation
- graph significant external events
- graph history


14:20
14:40
15:00
15:20
15:40
16:00
16:20
16:40
17:00
17:20
17:40
2 Weeks
1 Week
Release Started

What sucks about this stuff?

- It's a paperclip
- The graphite UI can suck
- UDP can suck



udp-logger is open source (Apache)

- **https://github.com/hootsuite/udp-logger**
- Back end for log4j and slf4j
- Built in DNS SRV record support for discovery of logstash
- You'll still need to format your messages usefully.
- typesafe config

statsd-client is open source (Apache)

- **https://github.com/hootsuite/statsd-client**
- Wrapper for etsy's statsd client
- More idiomatic/ lower-profile
- typesafe config

```
val callParent = monsters exists { m =>
   timed(s"monster-check.${checker}.$m") {
     checkUnderTheBed(m)
   }
}
```

Thanks!

- Edd Steel
- `@eddsteel`
- `code.hootsuite.com`
- `github.com/hootsuite`