Scaling at Showyou

Operations

September 26, 2011

I’m Kyle Kingsbury

Handle	aphyr
Code	http://github.com/aphyr
Email	kyle@remixation.com
Focus	Backend, API, ops

What the hell is Showyou?


iPad
9:41 AM
All
Featured
Friends
Search
Boing Boing
99 videos
Kaylee Fly Fishing 2011
flyguides — 17 hrs ago
The Daily Show
126 videos
WebODF screencast
hackernewsbot — yesterday
da da da da da HEY!
Phil Kufak — 12 hrs ago
L.A. River Fishing compresse
Moldychum — yesterday
FINAL FANTASY
arstechnica — 2 days ago
2011 Oregon Football Fall C
SUQ MADIQ
Reddit TV
554 videos - updated 10 hrs a
Grids
Watch Later
19
Notifications
Profile
3G
9:41 AM
20
SFO to Paris in Two Minutes
ekai
yesterday
ekai
yesterday
This is what 2,459 photos shot from the window of a SFO to Paris bound flight looks like. By the amazingly talented Boltron.
markhall
yesterday
@ekai Awesome, this.
Feed
Networks
Popular
More

Nontrivial complexity

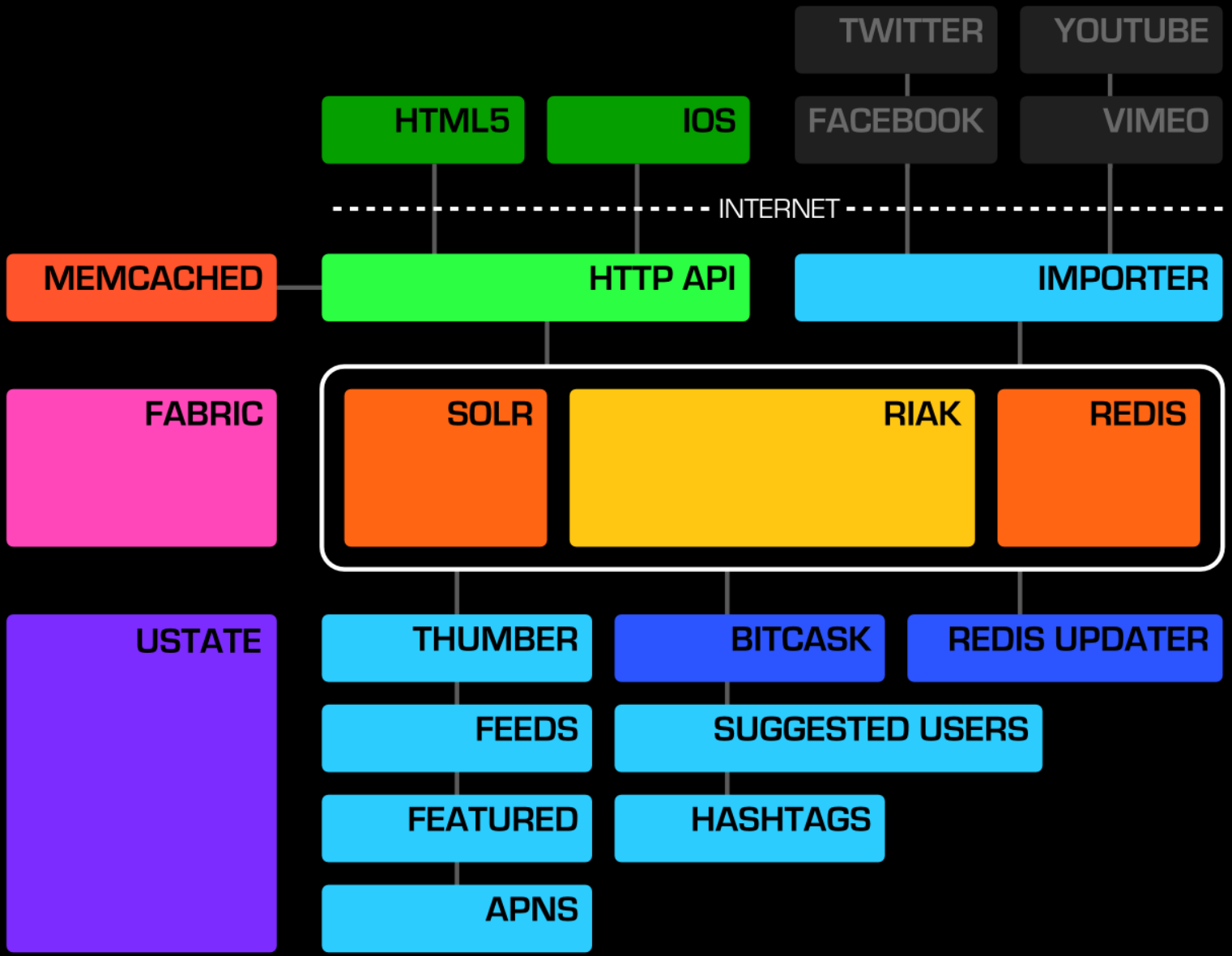
Challenges

- Scanning social networks
- Feeds
- Search
- Trends
- Responsive client experience

Challenges

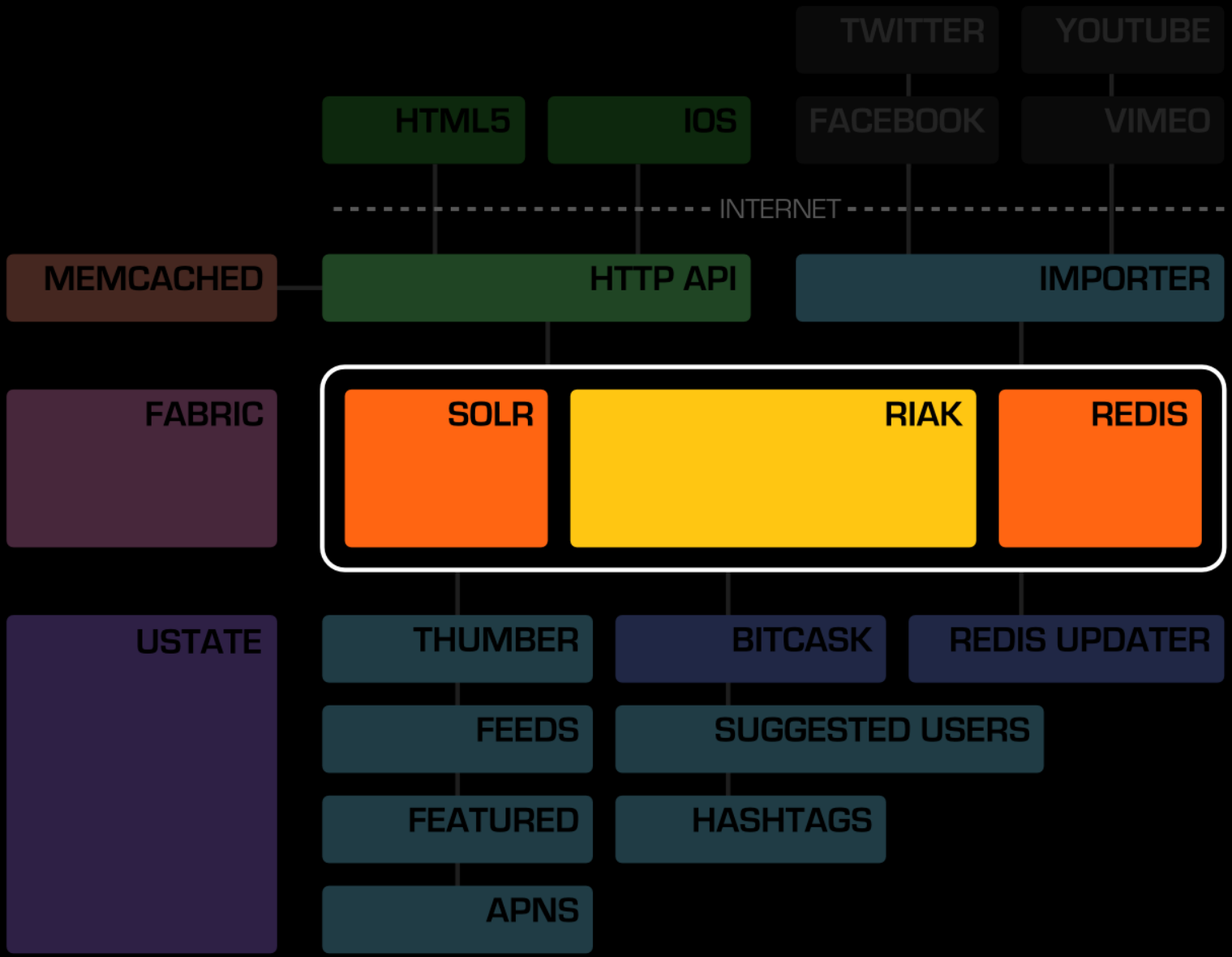
- Scanning social networks
- Feeds
- Search
- Trends
- Responsive client experience
- *Everything fails all the time*

SHOWYOU STACK



Storage

SHOWYOU STACK



We left MySQL

We left MySQL

- Changing the schema requires downtime
- Crashes
- Master-slave lag
- Slow restarts
- Node replacements difficult
- Fully normalized queries complex, slow

MySQL *does* scale

But there are tradeoffs

Riak

- Key/value store
- Homogenous
- Scales linearly with nodes
- Excellent durability/recoverability
- Eventually consistent

We use Riak as our *durable* datastore

- Users, feeds, videos, etc
- Highly denormalized
- Limited MR queries (feeds, etc)
  - Latency-bounded MR jobs are Erlang
  - Hot-deployable
- Extensive use of conflict resolution
  - Made possible by Risky

Riak at Showyou

- 51 million keys (153 M replicated)
- 100 GB of data (300 GB replicated)
- 260 gets/sec (baseline)
- 75 puts/sec (baseline)
- Capable of over 3000 ops/sec

SSDs are amazing

WD 7200RPM

- 100 ops/sec
- 95%: 100-300ms

Micron RealSSD P300

- 1000+ ops/sec
- 95%: 3-5ms

When Riak fails,

- Another node takes up the slack
- Clients connected to that node reconnect to others
- Typically no service interruption
  - However, latencies may rise
  - Especially for MR jobs

Riak has downsides

- Difficult to debug
- Membership changes are dangerous
- Significantly slower than MySQL
- (Bitcask) All keys must fit in memory
- Mapreduce is only appropriate for known keys
- List-keys can take down your cluster

Long story short: it's *only* a KV store

+Redis

We use Redis for fast, temporary state

- List of users
- List of videos
- Counters
- Queues

Incredibly fast, excellent primitives

When Redis fails,

- Daemons using those indexes pause
- Frontend service continues
- Bitcask scanners and incremental updaters repair any lost data

When Redis fails,

- Daemons using those indexes pause
- Frontend service continues
- Bitcask scanners and incremental updaters repair any lost data

Eventually consistent.

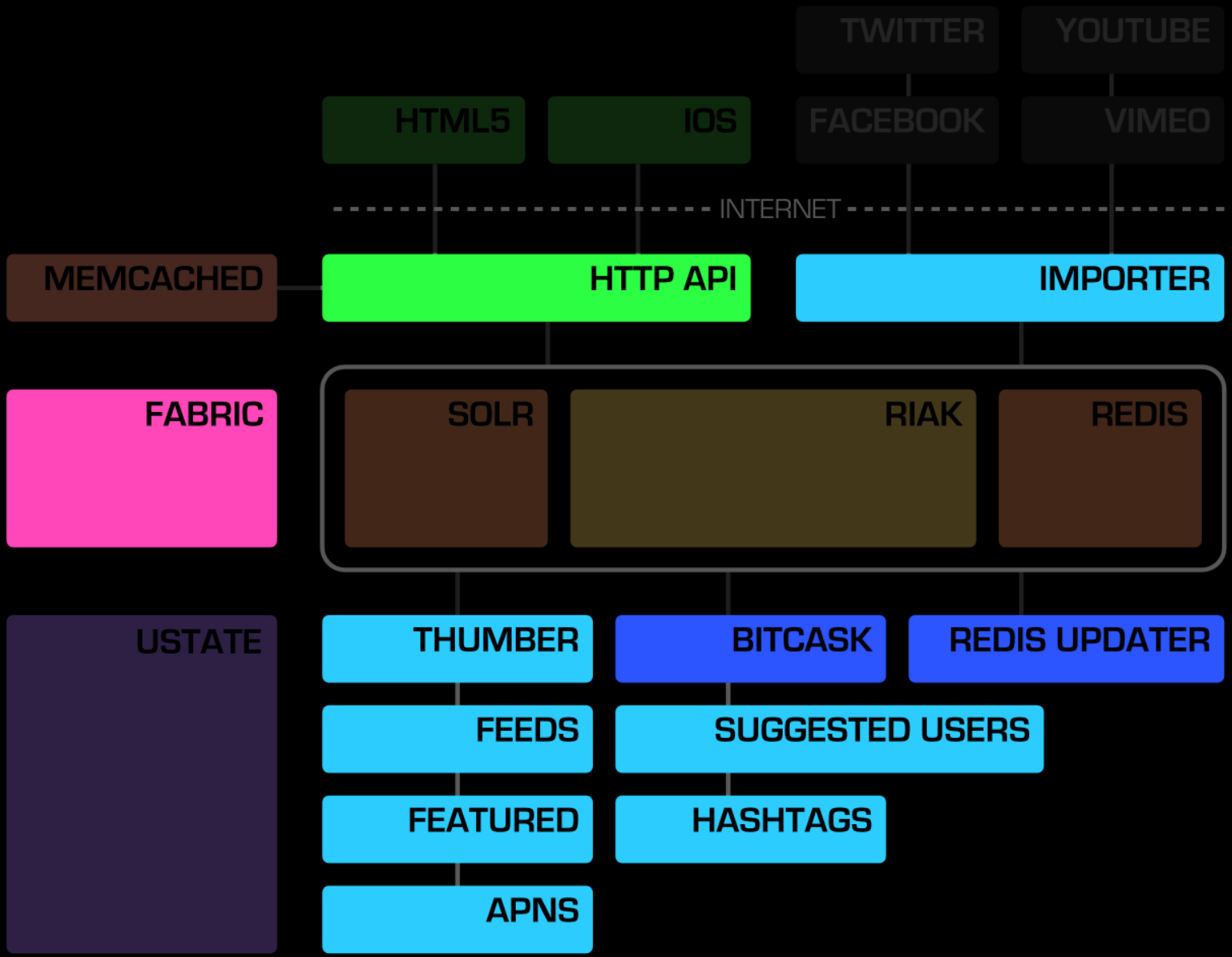
We also use SOLR extensively

- Supplements Riak
- Complex indices
- Full-text search
- Analytics

More on that later. . .

Processing

SHOWYOU STACK



Do one thing well

Lots of small processes handling well-defined tasks

- Easier to debug
- Easier to test
- Simplifies parallelism
- Simplifies error handling
- Less likely to cause total system failure

Minimize Shared State

- Vector clocks for concurrent modification
- Queues for message passing
- Riak for durable storage
- Redis for fast synchronous state

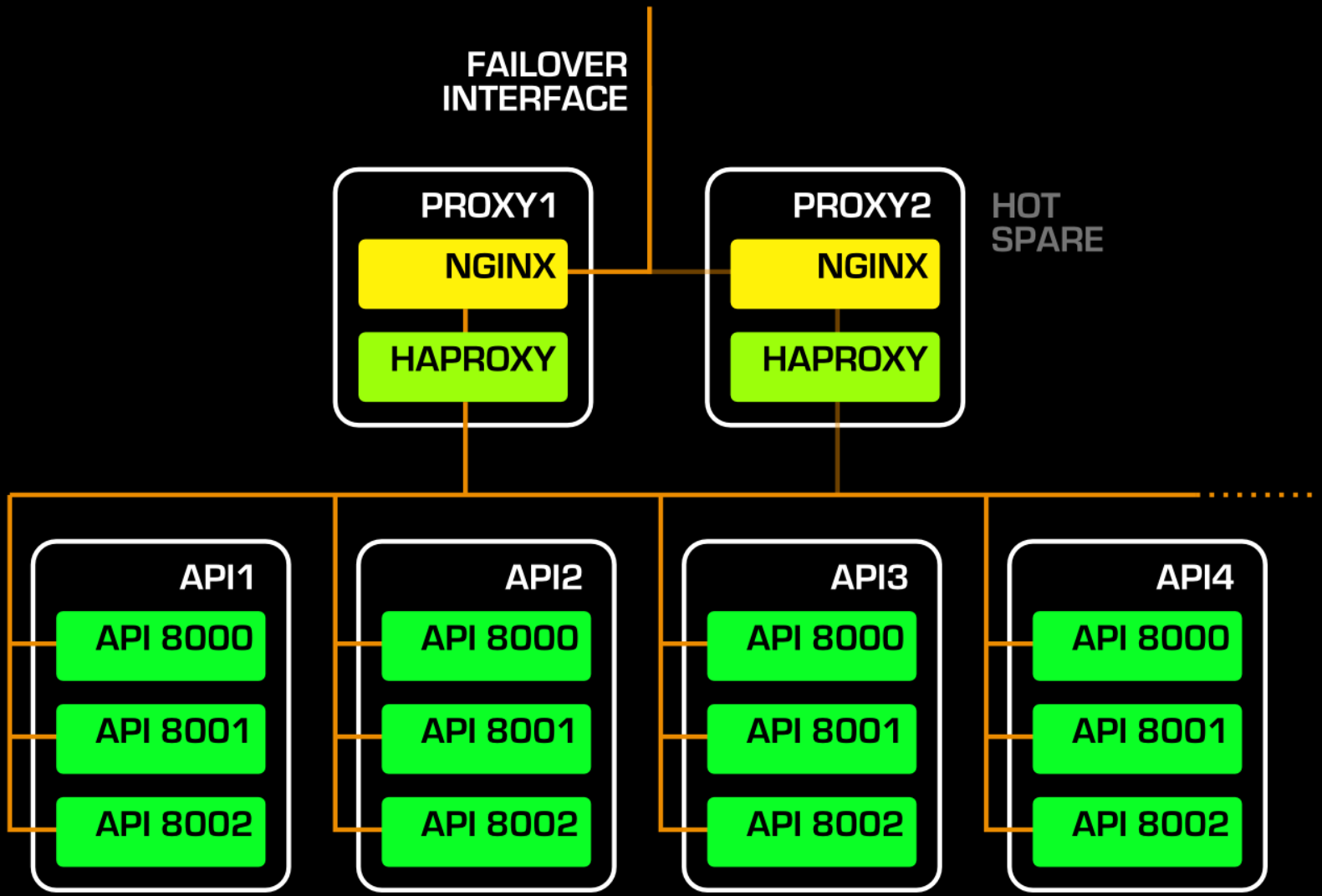
Crash by Default

- Someone else will take your work
- Repair constantly
- Assume everybody is out to kill you

Distribute

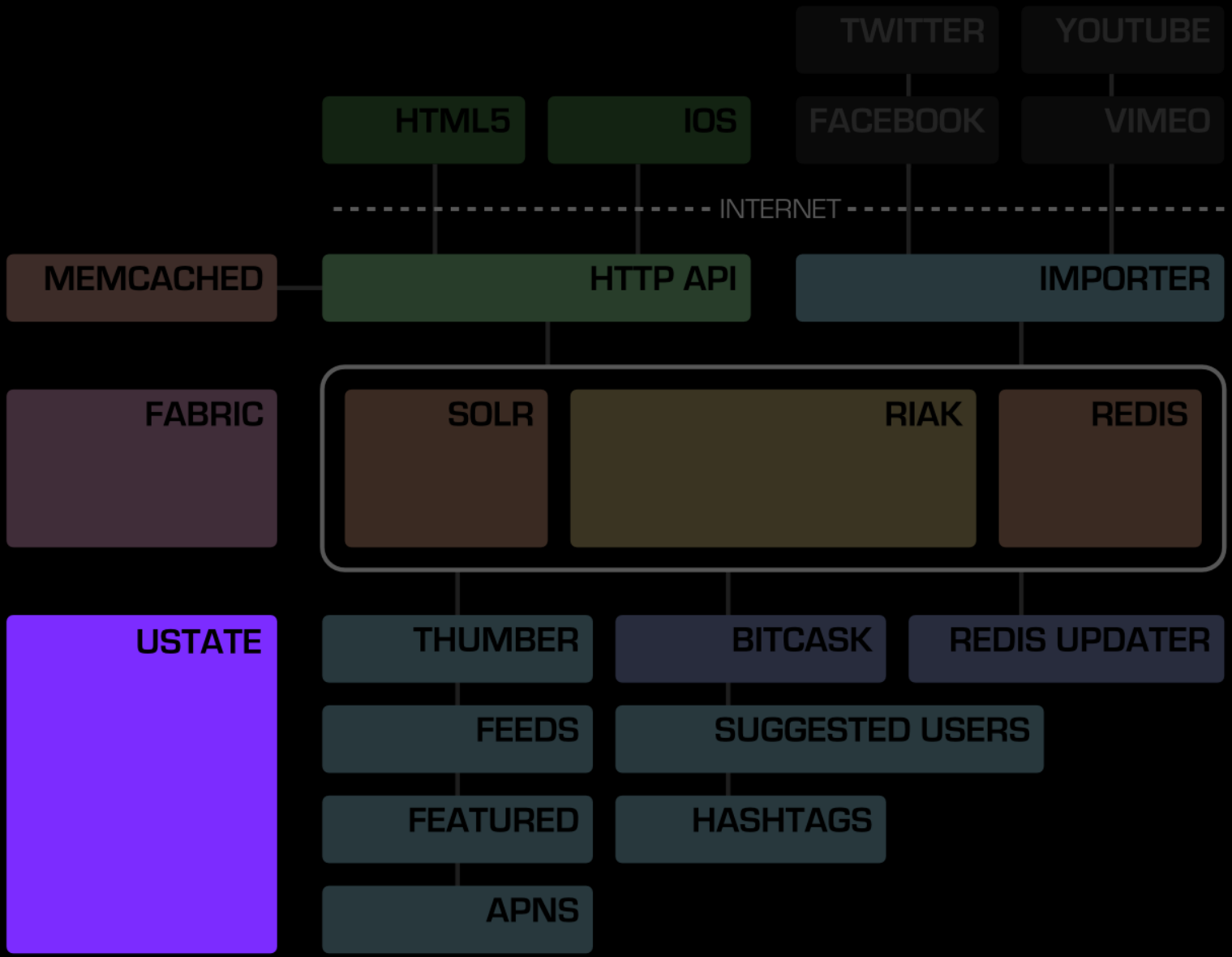
- Multiple threads, processes, hosts
- Failover IPs with Heartbeat
- Rolling restarts mean frequent deploys and nobody notices
- Losing a node is no big deal
- Scaling out is easy

SHOWYOU HTTP



Monitoring

SHOWYOU STACK



UState: A state aggregator

Receive states over protobufs

Host	backend1.showyou.com
Service	feed merger rate
Time	unix epoch seconds
State	ok
Metric	12.5
Description	12.5 feed items/sec

Query states

- state = "warning" or state = "critical"
- service =~ "api %" and host != null

- Combine states together (sum, average, . . .)
- Send email on changes
- Forward to another UState server
- Forward to Graphite
- Dashboard

Dashboard

(as of 01:52:02)

Status: Winning.

Health

	cpu	disk /	disk /boot	disk /disk1	disk /disk2	disk /ssd1	disk /ssd2	disk /ssd3	load	memory
api1.tx	0.07	0.13	0.67	0.01	0.54				0.12	0.29
api2.tx	0.07	0.14	0.67	0.01	0.62				0.13	0.38
api3.tx	0.07	0.15	0.67	0.01	0.61				0.14	0.41
api4.tx	0.11	0.14	0.67	0.01	0.61				0.12	0.41
api5.tx	0.11	0.14	0.67	0.01	0.60				0.12	0.32
api7.tx	0.07	0.14	0.67	0.01	0.67				0.13	0.37
be1.tx	0.01	0.03	0.67						0.01	0.57
be2.tx	0.01	0.02	0.67						0.02	0.26
hub3.tx	0.01	0.01	0.43						0.01	0.26
importer.tx	0.06	0.17	0.45		0.01				0.11	0.44
solr2.tx	0.01	0.07	0.67			0.02	0.13	0.33	0.01	0.84

Tablet

Queue

alerts	0.00
apns	0.00
tablet_batched_user_import	3.00
tablet_featured	0.00
tablet_feed	0.00
tablet_thumbnails	0.00
tablet_user_import	0.00
tablet_user_import_new	0.00

Api

50	95	99	rate
21.28	132.08	132.08	3.15

Apns relay

be1.tx	0.00
be2.tx	0.00

Redis

	blocked_clients	connected_clients	db0 keys	memory	user updater
be1.tx	15.00	60.00	14.00	2904.25	0.13
be2.tx	0.00	1.00		16.98	0.13

Bitcask scanner

api1.tx	254.36
api2.tx	253.42
api3.tx	250.23
api4.tx	253.14
api5.tx	254.71
api7.tx	258.13

Tablet suggested users updater

be1.tx	1.18

Tablet thumbnailer

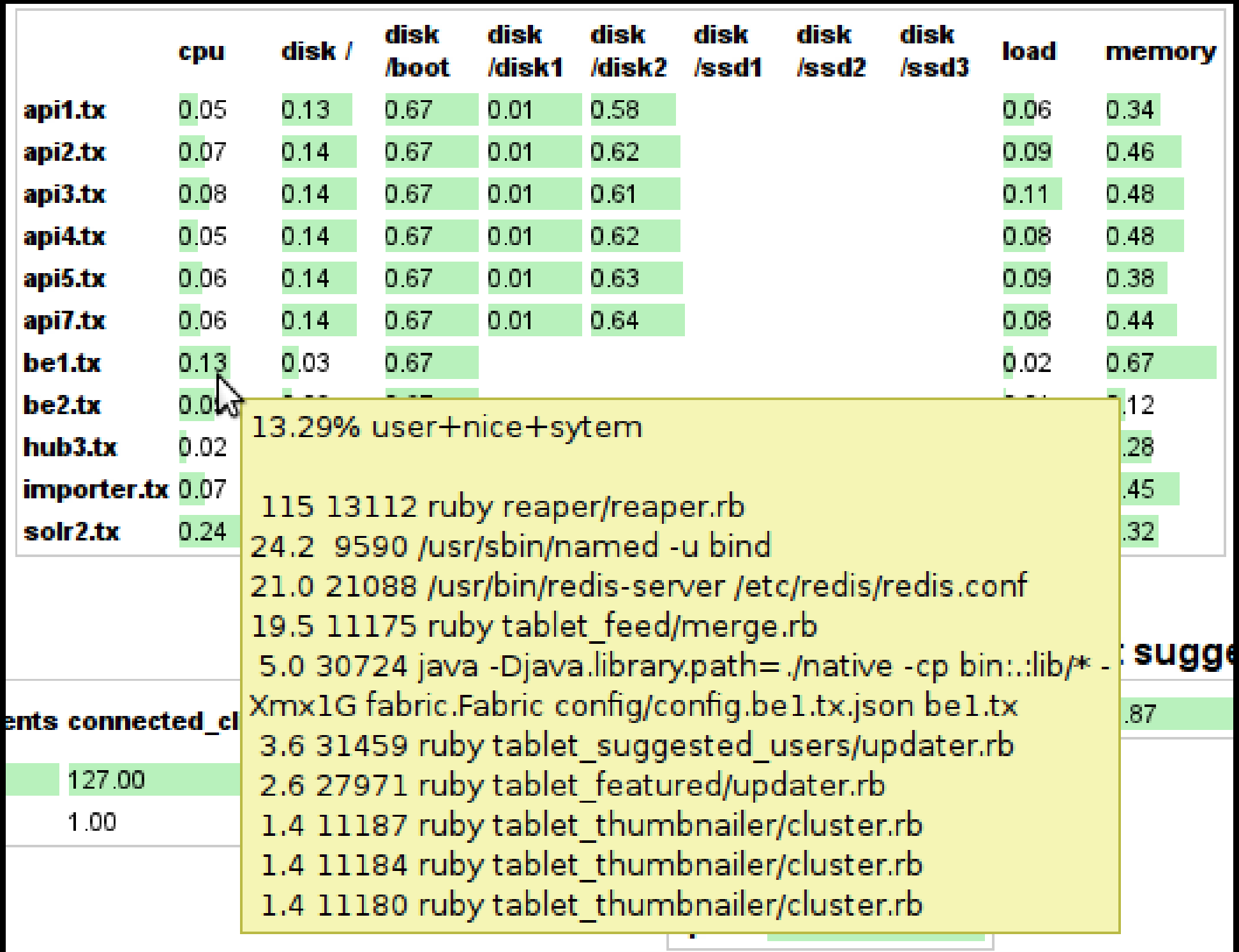
	0	1	2	3	4	5
	0.34					
be1.tx	0.02	0.02	0.01	0.07	0.02	0.03
be2.tx	0.01	0.04	0.04	0.04	0.03	0.02

Tablet feed merger

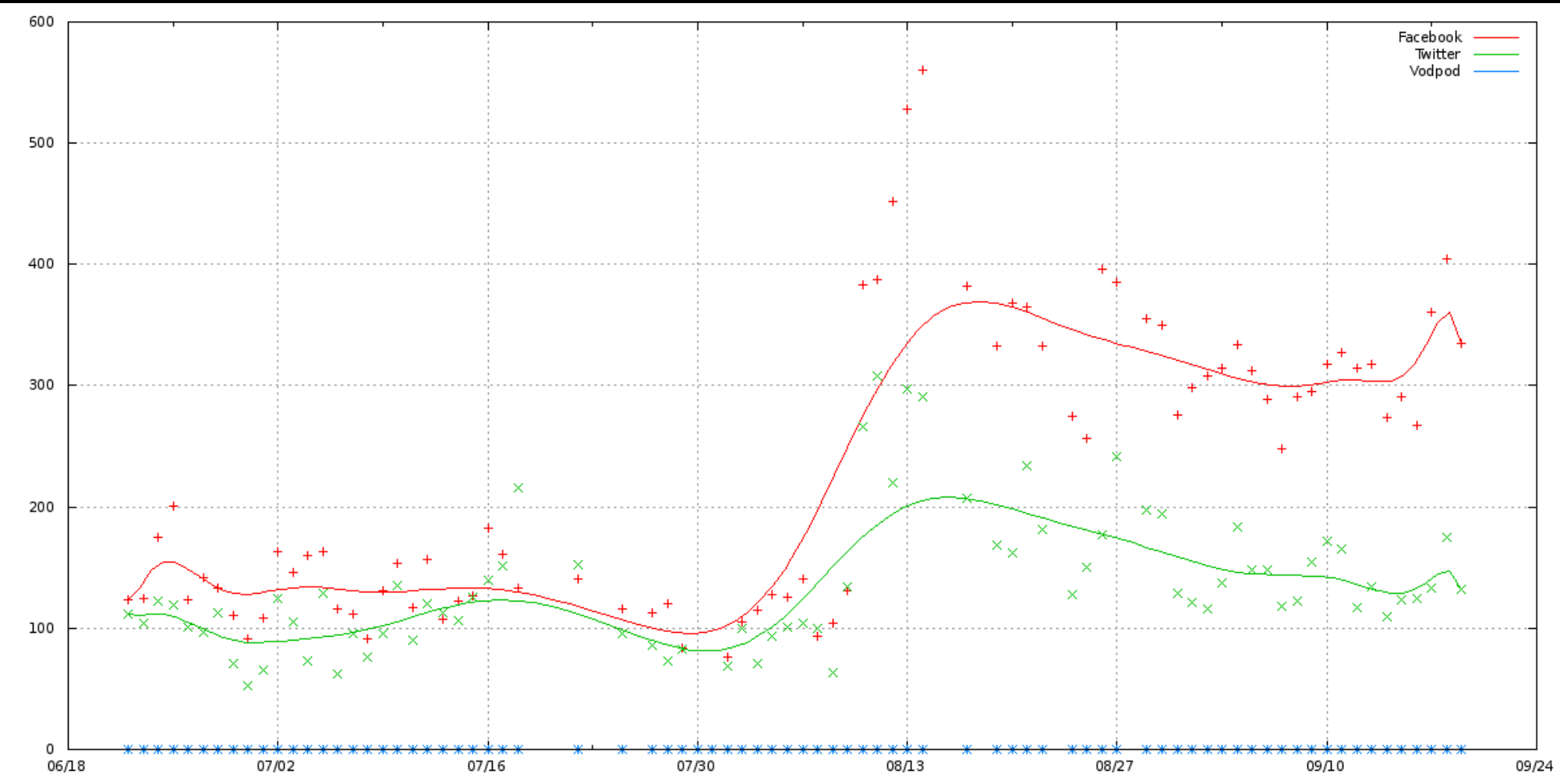
	50	95	99
	3.04		
be1.tx	1.44	0.05	0.87	2.26
be2.tx	1.60	0.05	0.24	2.08

Riak

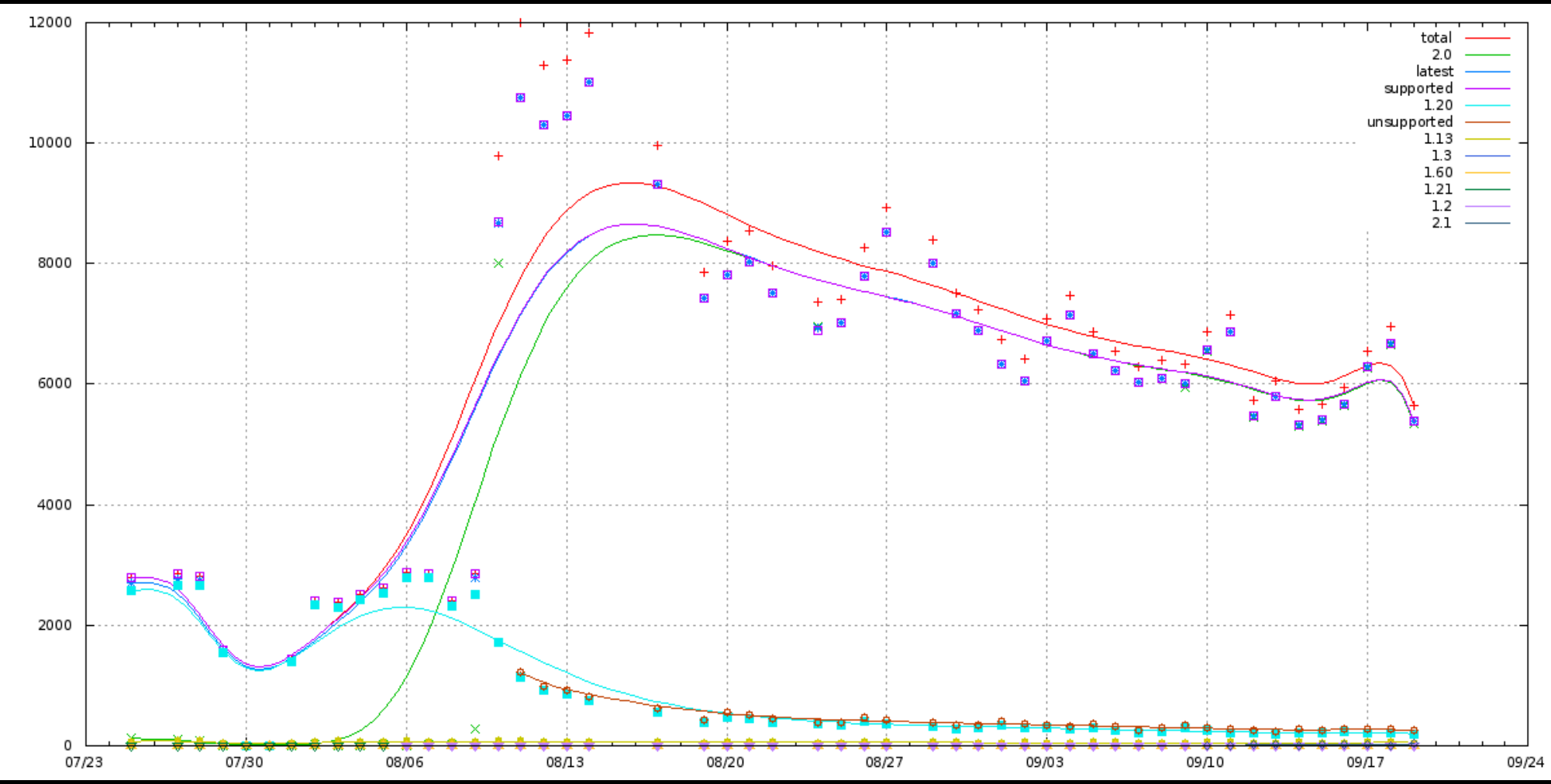
	disk	get 50	get 95	get 99	keys	node_gets	node_puts	put 50	put 95	put 99	read_repairs	ring	vnode_gets	vnode_puts
		0.98	2.89	4.99		122.32	24.22	1.53	5.03	6.49				
api1.tx	46.29	0.98	4.46	7.55	16852104.00	20.98	6.35	1.52	7.00	9.05	0.10	?	123.60	11.55
api2.tx	53.30	0.94	1.29	1.64	28095752.00	54.48	6.27	1.76	2.22	2.48	0.02	?	133.27	11.00
api3.tx	52.36	1.02	3.55	6.45	31521044.00	29.33	7.27	1.49	6.18	9.41	0.05	?	128.02	11.05
api4.tx	52.49	0.90	1.42	1.62	29831640.00	1.67	0.25	1.35	2.56	2.71	0.05	?	128.60	11.95
api5.tx	51.19	0.93	3.89	7.76	19558314.00	4.95	1.38	1.48	7.21	8.15	0.03	?	134.57	11.48
api7.tx	56.82	1.10	2.71	4.93	27040880.00	10.90	2.70	1.58	4.99	7.17	0.03	?	130.07	11.68


cpu
disk /
disk /boot
disk /disk1
disk /disk2
disk /ssd1
disk /ssd2
disk /ssd3
load
memory
api1.tx 0.05 0.13 0.67 0.01 0.58 0.06 0.34
api2.tx 0.07 0.14 0.67 0.01 0.62 0.09 0.46
api3.tx 0.08 0.14 0.67 0.01 0.61 0.11 0.48
api4.tx 0.05 0.14 0.67 0.01 0.62 0.08 0.48
api5.tx 0.06 0.14 0.67 0.01 0.63 0.09 0.38
api7.tx 0.06 0.14 0.67 0.01 0.64 0.08 0.44
be1.tx 0.13 0.03 0.67 0.02 0.67
be2.tx
hub3.tx 0.02
importer.tx 0.07
solr2.tx 0.24
13.29% user+nice+sytem
115 13112 ruby reaper/reaper.rb
24.2 9590 /usr/sbin/named -u bind
21.0 21088 /usr/bin/redis-server /etc/redis/redis.conf
19.5 11175 ruby tablet_feed/merge.rb
5.0 30724 java -Djava.library.path=./native -cp bin:.:lib/* -Xmx1G fabric.Fabric config/config.be1.tx.json be1.tx
3.6 31459 ruby tablet_suggested_users/updater.rb
2.6 27971 ruby tablet_featured/updater.rb
1.4 11187 ruby tablet_thumbnailer/cluster.rb
1.4 11184 ruby tablet_thumbnailer/cluster.rb
1.4 11180 ruby tablet_thumbnailer/cluster.rb
connected_cl
127.00
1.00

Understand application behavior


Facebook
Twitter
Vodpod
600
500
400
300
200
100
0
06/18
07/02
07/16
07/30
08/13
08/27
09/10
09/24

When can we. . . ?


12000
10000
8000
6000
4000
2000
0
07/23
07/30
08/06
08/13
08/20
08/27
09/03
09/10
09/17
09/24
total
2.0
latest
supported
1.20
unsupported
1.13
1.3
1.60
1.21
1.2
2.1

It’s 23:15 PST.

It’s 23:15 PST.

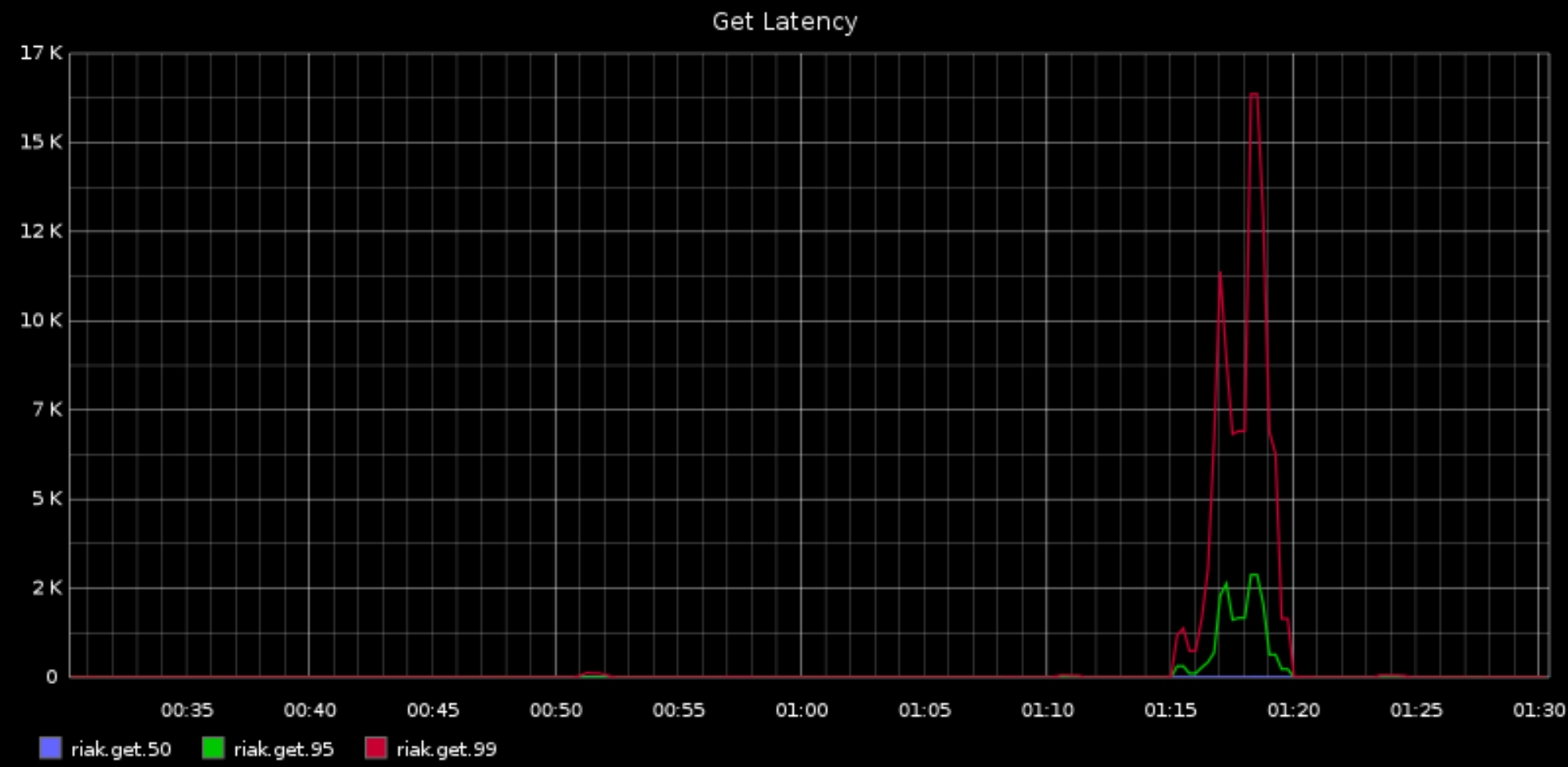
Do you know where YOUR database is?

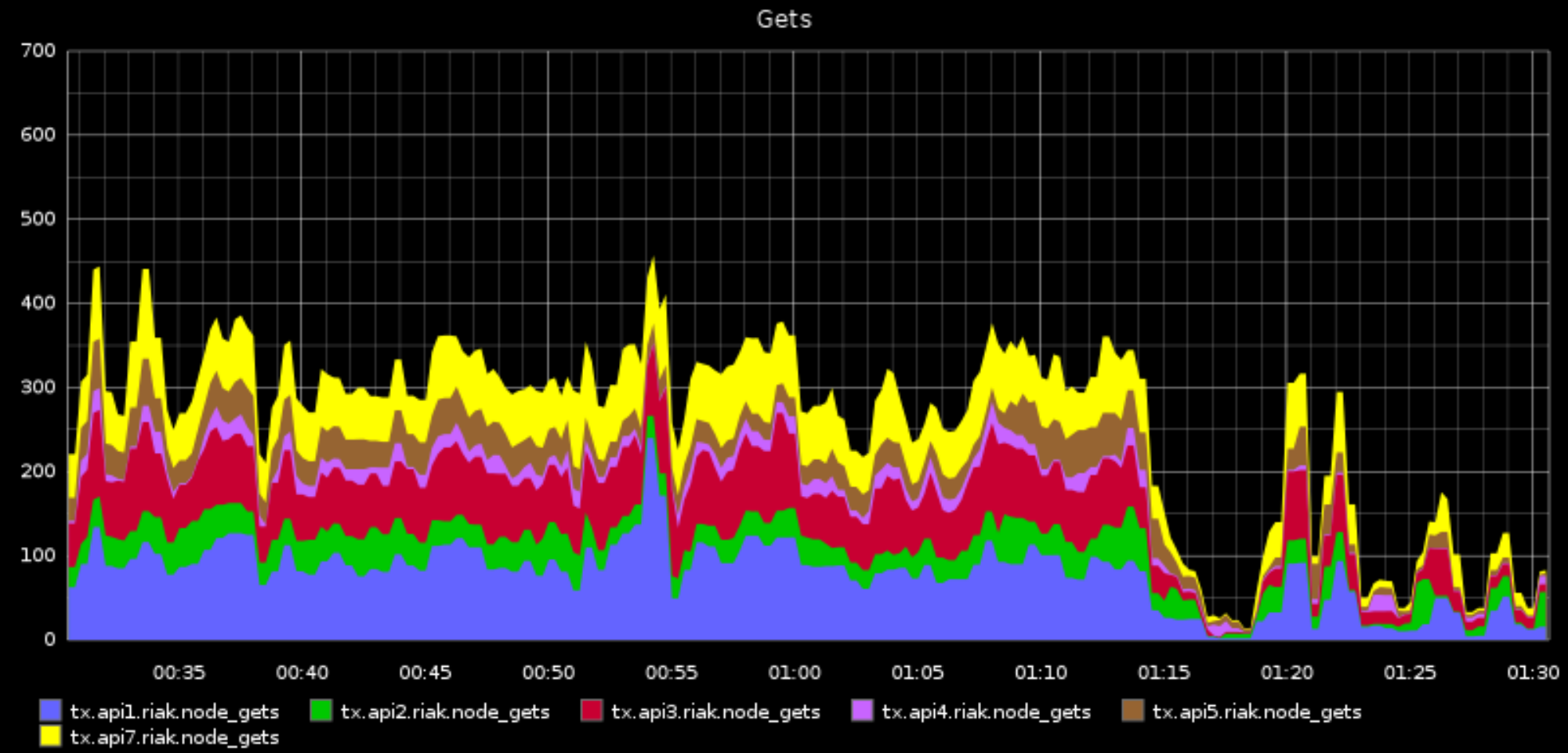
from ustate <ustate@showyou.com>

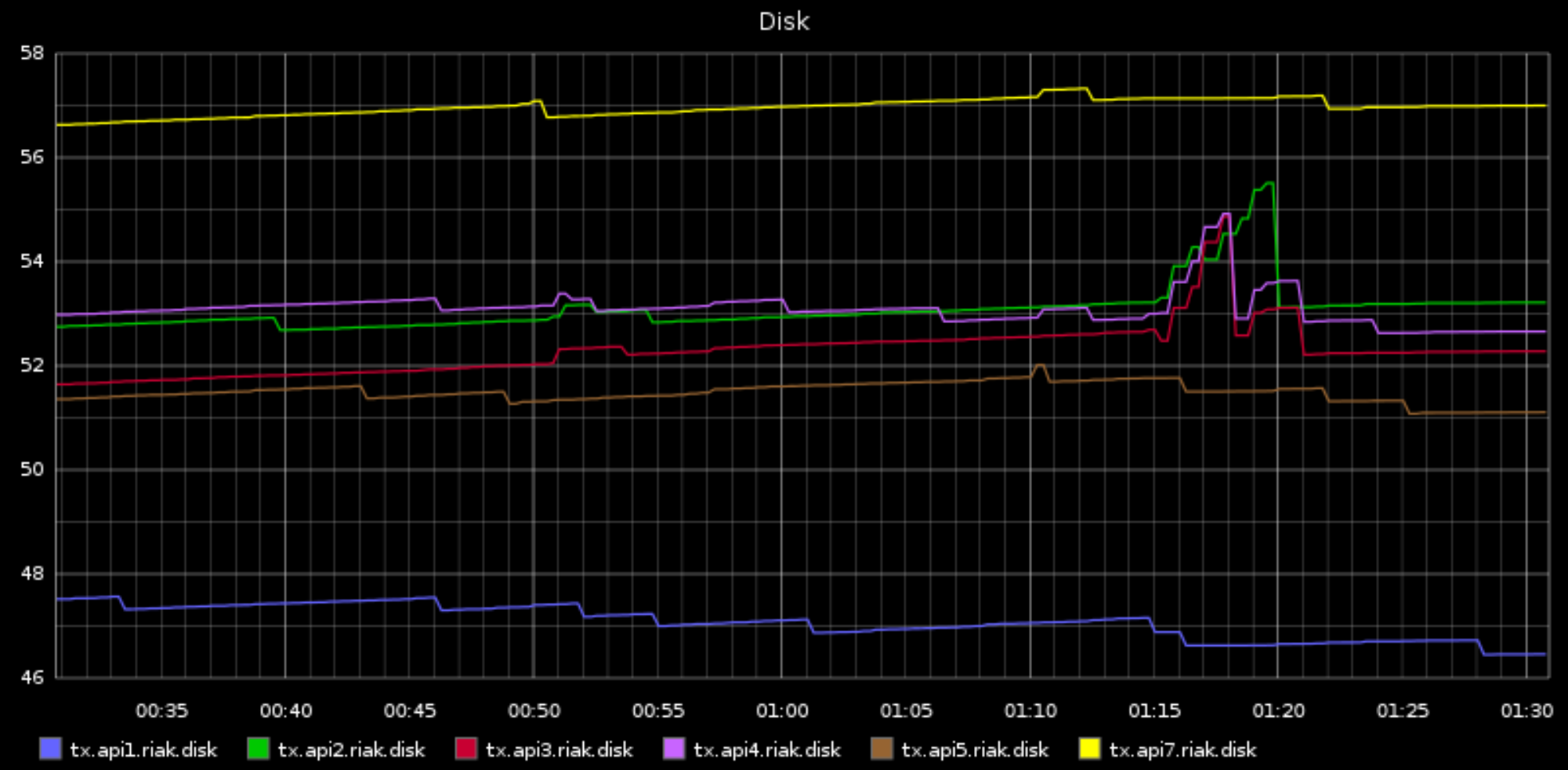
subject **api5.tx riak get 99 is warn**

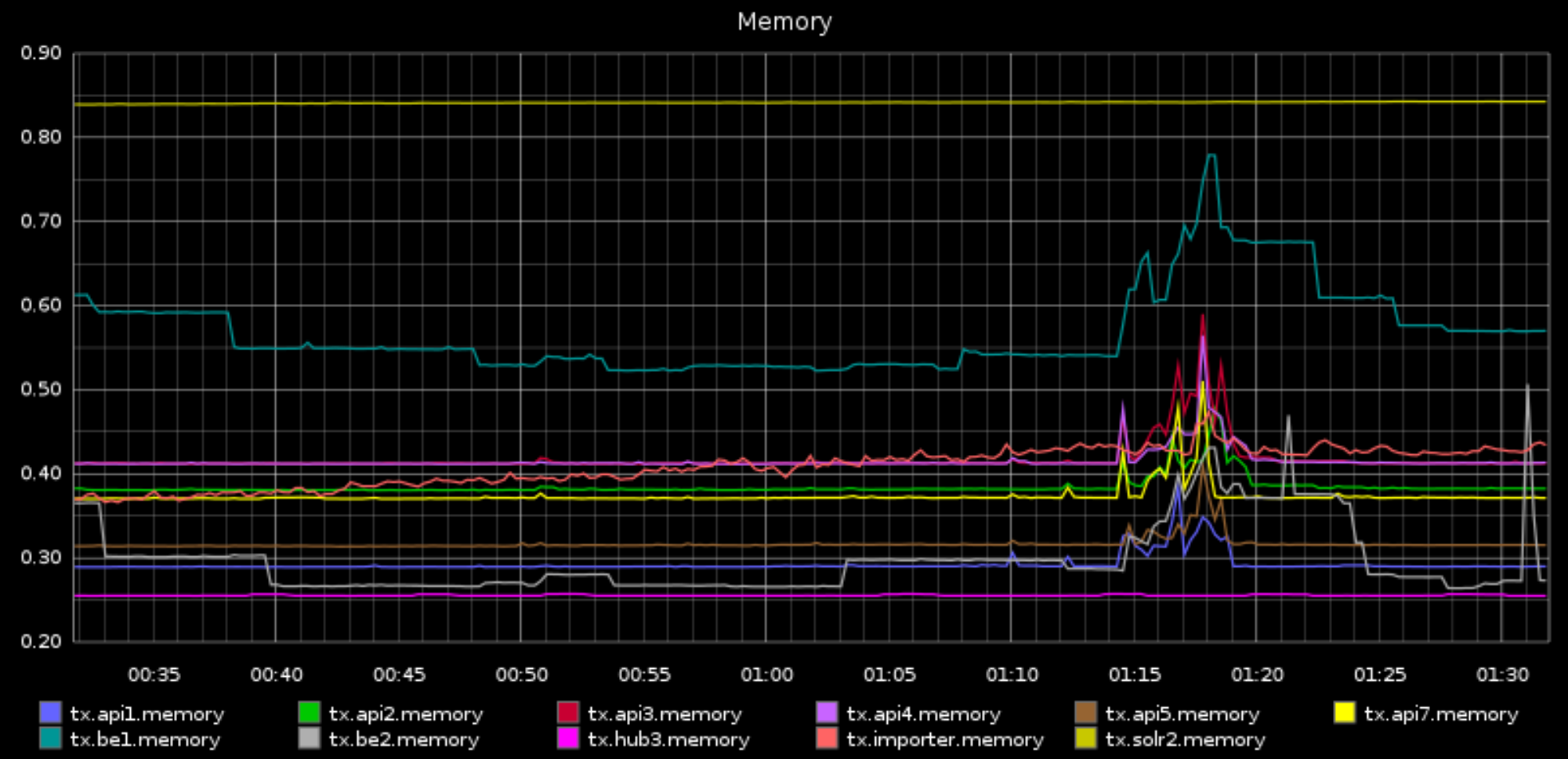
to kyle@remixation.com

```
api5.tx riak get 99 is warn: 16977.38 ms
```


Get Latency
17 K
15 K
12 K
10 K
7 K
5 K
2 K
0
00:35
00:40
00:45
00:50
00:55
01:00
01:05
01:10
01:15
01:20
01:25
01:30
riak.get.50
riak.get.95
riak.get.99


Gets
700
600
500
400
300
200
100
0
00:35
00:40
00:45
00:50
00:55
01:00
01:05
01:10
01:15
01:20
01:25
01:30
tx.api1.riak.node_gets
tx.api2.riak.node_gets
tx.api3.riak.node_gets
tx.api4.riak.node_gets
tx.api5.riak.node_gets
tx.api7.riak.node_gets


Disk
58
56
54
52
50
48
46
00:35
00:40
00:45
00:50
00:55
01:00
01:05
01:10
01:15
01:20
01:25
01:30
tx.api1.riak.disk
tx.api2.riak.disk
tx.api3.riak.disk
tx.api4.riak.disk
tx.api5.riak.disk
tx.api7.riak.disk


Memory
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
00:35
00:40
00:45
00:50
00:55
01:00
01:05
01:10
01:15
01:20
01:25
01:30
tx.api1.memory
tx.api2.memory
tx.api3.memory
tx.api4.memory
tx.api5.memory
tx.api7.memory
tx.be1.memory
tx.be2.memory
tx.hub3.memory
tx.importer.memory
tx.solr2.memory

```ruby
#!/usr/bin/ruby
require 'bitcask'
require 'pp'
require 'bert'

hints = Dir.glob('/var/lib/riak/bitcask/*/*.hint')
big = {}

hints.each do |f|
  h = Bitcask::HintFile.new f
  puts "Checking #{h}..."
  h.each do |e|
    if e.value_sz > 10 * 1024 * 1024
      big[e.key] = e.value_sz
    end
  end
end

big.sort do |a,b|
  b[1] <=> a[1]
end.map do |key, size|
  puts "#{size}\t#{BERT.decode(key).inspect}"
end
```

1,1 All

```
Checking #<Bitcask::HintFile:0x00000001259990>...
Checking #<Bitcask::HintFile:0x0000000186a898>...
Checking #<Bitcask::HintFile:0x00000012534a0>...
Checking #<Bitcask::HintFile:0x000000015c1880>...
Checking #<Bitcask::HintFile:0x0000000142b750>...
Checking #<Bitcask::HintFile:0x0000000179de88>...
Checking #<Bitcask::HintFile:0x000000017ad1d0>...
Checking #<Bitcask::HintFile:0x000000017d20e8>...
Checking #<Bitcask::HintFile:0x00000001808490>...
Checking #<Bitcask::HintFile:0x00000001480340>...
Checking #<Bitcask::HintFile:0x00000001946e38>...
Checking #<Bitcask::HintFile:0x000000018d0b98>...
Checking #<Bitcask::HintFile:0x000000015dd350>...
Checking #<Bitcask::HintFile:0x0000000197fe18>...
Checking #<Bitcask::HintFile:0x000000019lef00>...
Checking #<Bitcask::HintFile:0x00000001950cf8>...
Checking #<Bitcask::HintFile:0x00000001947108>...
Checking #<Bitcask::HintFile:0x0000000190d4a8>...
Checking #<Bitcask::HintFile:0x000000015dd8c8>...
70194714        t["tablet_feed_items", "nf_kwyclef77_vy%3AwkKiKwBCocw"]
54633928        t["tablet_feed_items", "nf_krohankennedy_vv%3A18346645"]
21737208        t["tablet_feed_items", "nf_kaabannink_v14221484"]
11567775        t["tablet_feed_items", "nf_kstarwalkersng_vy%3AwkKiKwBCocw"]
deploy@api3:~$
```

http://github.com/aphyr/ustate

Recap

- Robust, discrete components
- Highly distributed
- Message passing
- Eventual consistency
- Comprehensive monitoring

Thanks!

- Basho (esp. Pharkmillups!)
- Formspring
- Bump