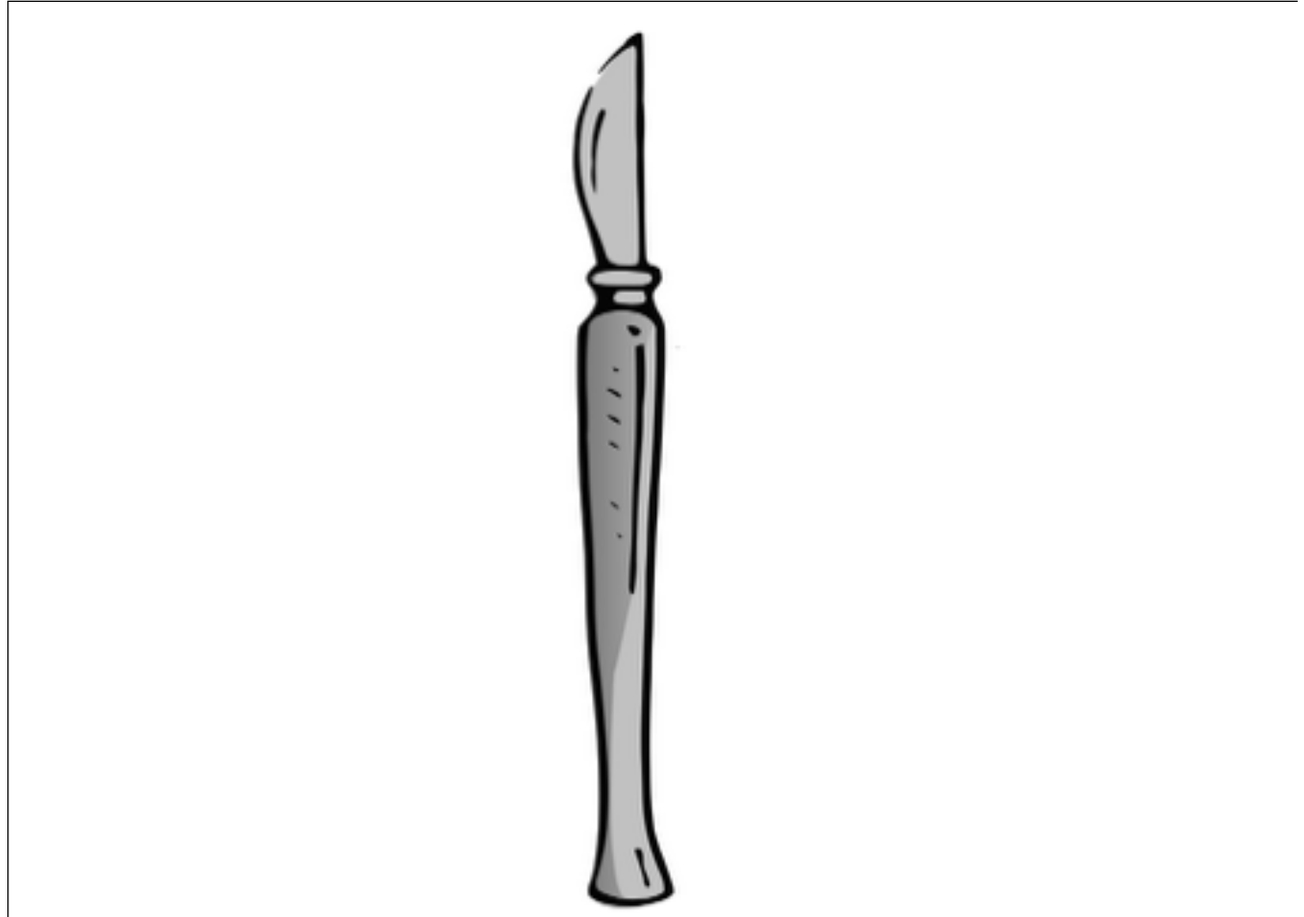
PLATFORM ENGINEERING

Lessons Learned in Scaling Twitter

@bmdhacks
qcon london 2014



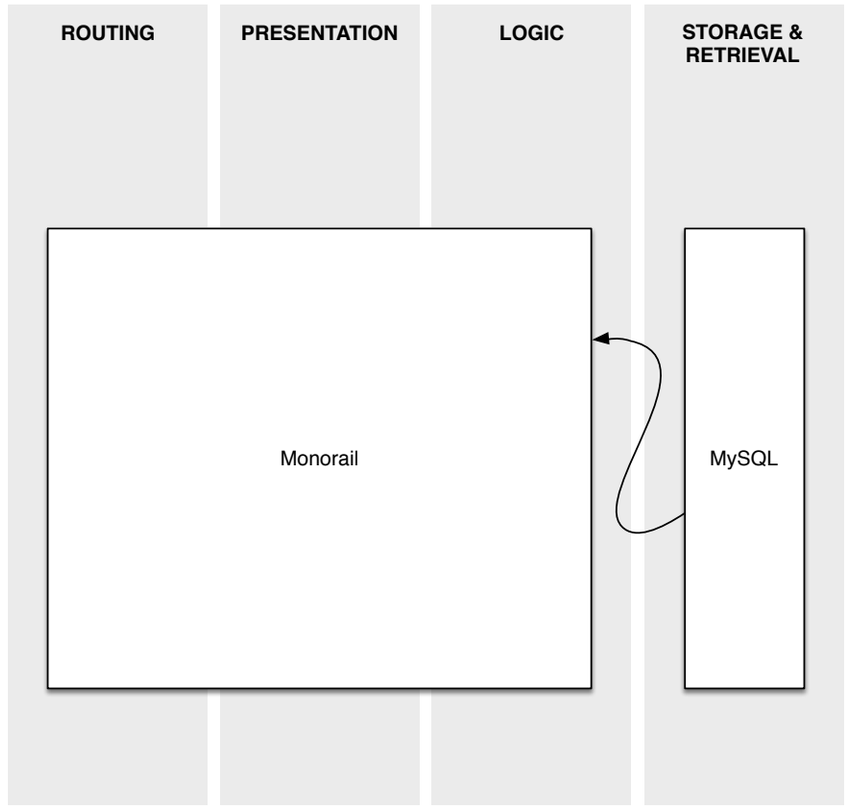
Scalability, actually all Software Engineering can be described in how you divide a problem into components. It is the scalpel you use to divide large problems into tractable ones. The abstractions are the boundaries of these cuts, and a well-divided problem is both scalable and manageable.

Lessons Learned

- Incrementally Implement SOA
- Separate semantics from execution
- Use statistics to monitor behavior

Service Oriented Architecture

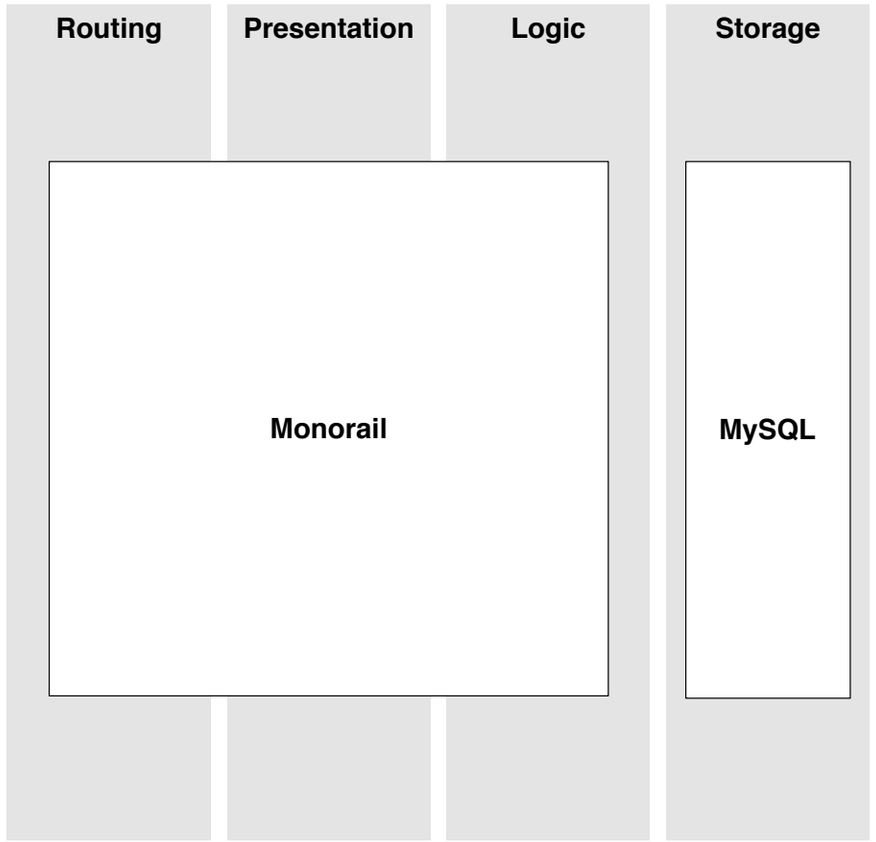
@bmdhacks
qcon london 2014

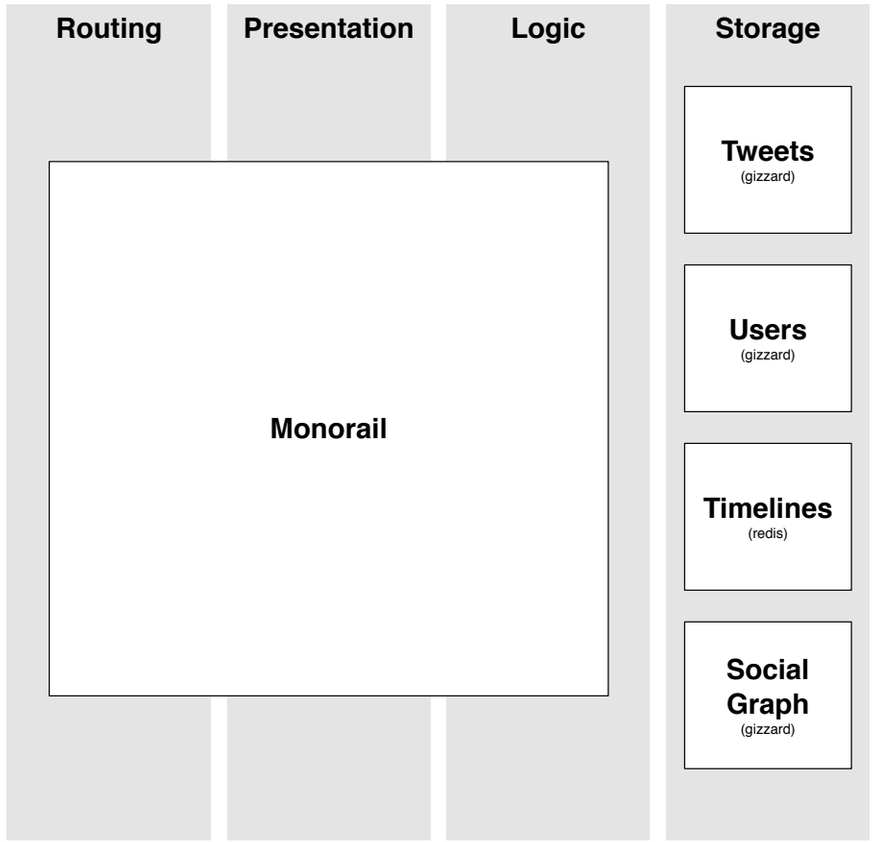


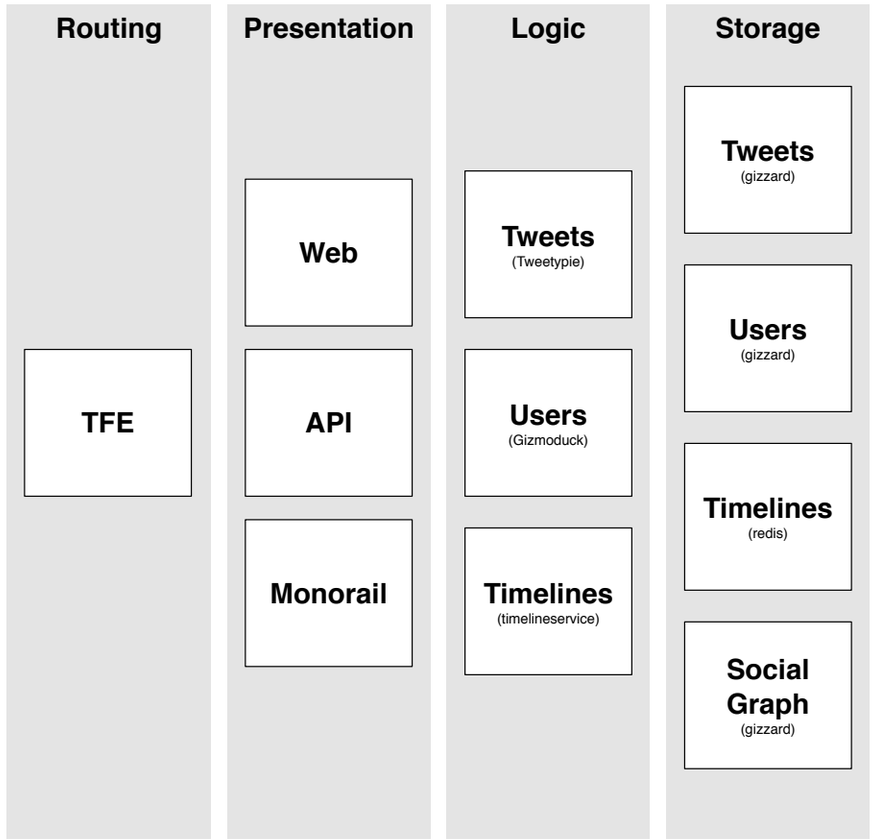
The monorail was our term for a single Rails application running on ruby 1.8.3

challenges

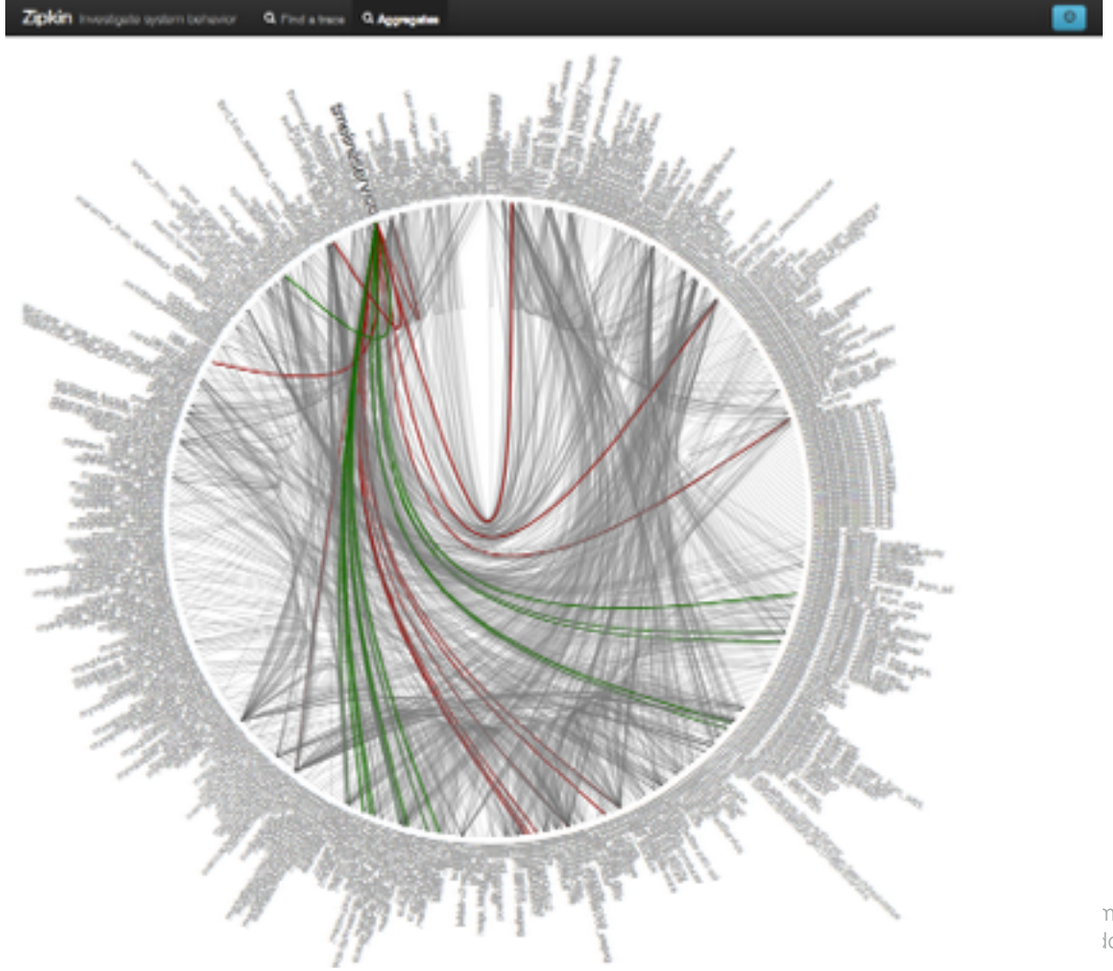
- difficult to scale storage model
- any change deploys to all servers
- poor concurrency, runtime performance
- leaky abstractions / tight-coupling


Routing
Presentation
Logic
Storage
Monorail
MySQL


Routing
Presentation
Logic
Storage
Monorail
Tweets
(gizzard)
Users
(gizzard)
Timelines
(redis)
Social Graph
(gizzard)



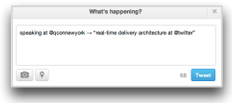
The final result of all services at twitter

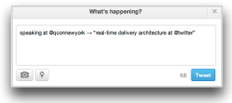


ndhacks
lon 2014

Actually this is a map of all services at twitter

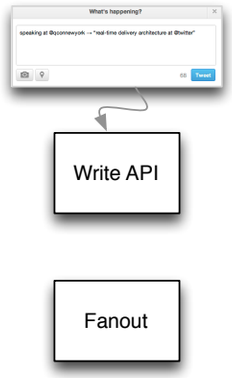
Sending a Tweet Today

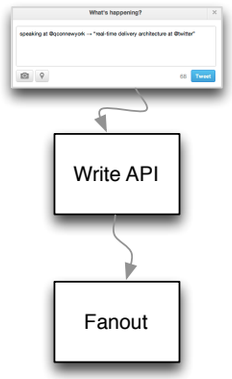

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
68
Tweet

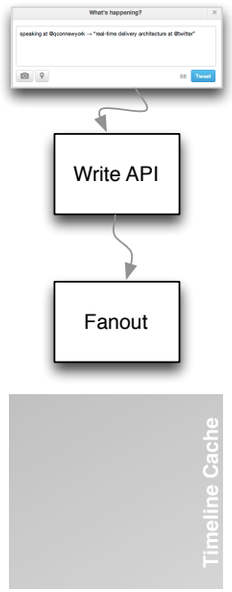

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
68
Tweet

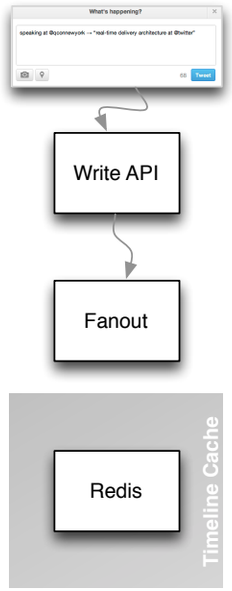
Write API

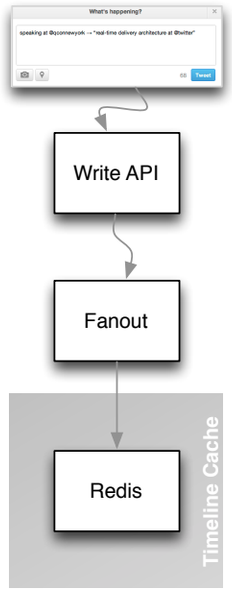

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
68
Tweet
Write API

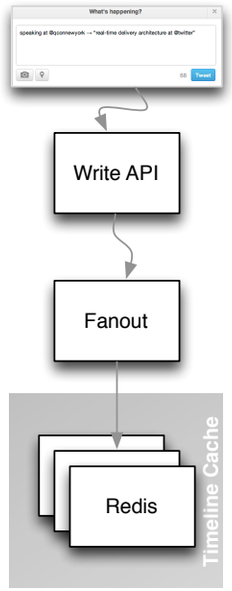

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
68
Tweet
Write API
Fanout

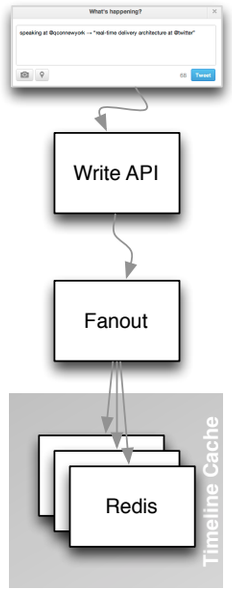

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Fanout

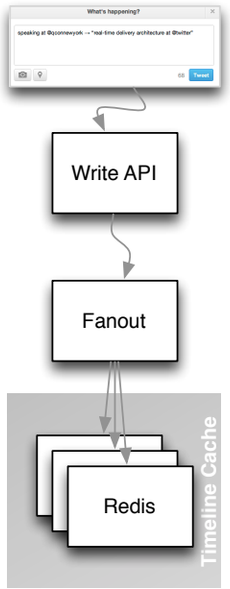

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Fanout
Timeline Cache

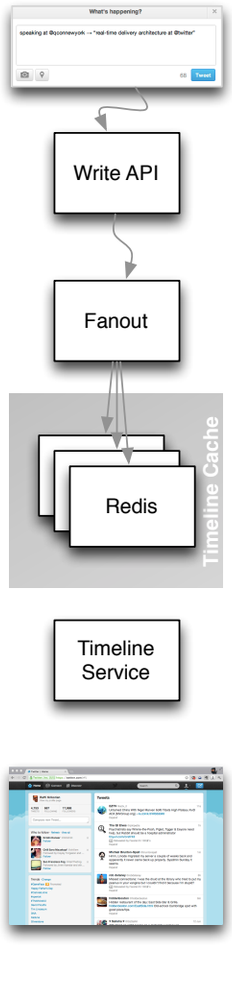

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Fanout
Redis
Timeline Cache

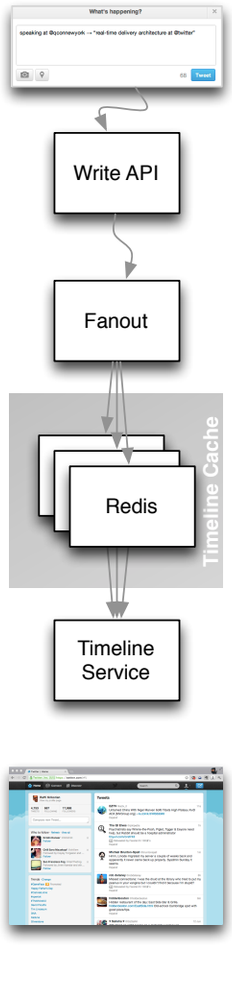

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Fanout
Redis
Timeline Cache

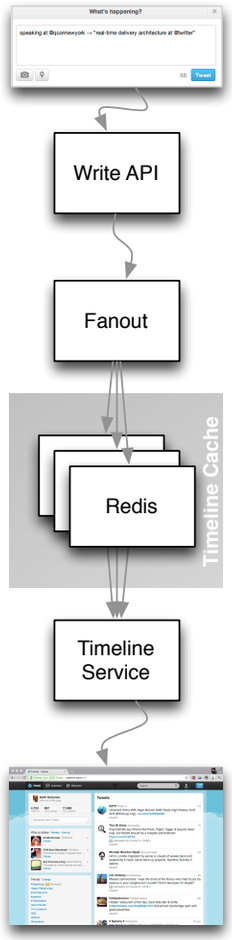

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Fanout
Redis
Timeline Cache

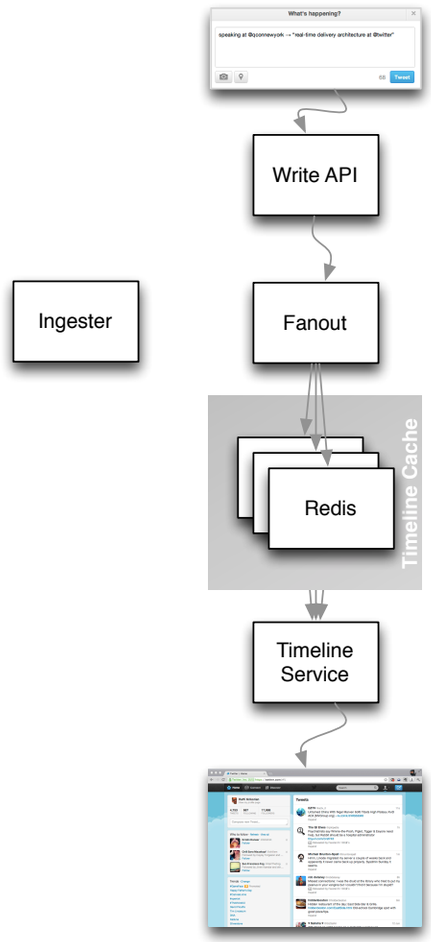

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Fanout
Redis
Timeline Cache

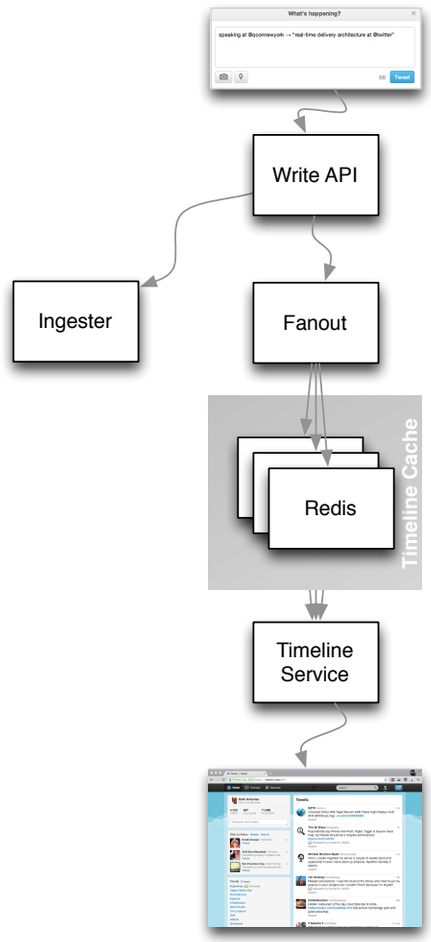

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Fanout
Redis
Timeline Cache

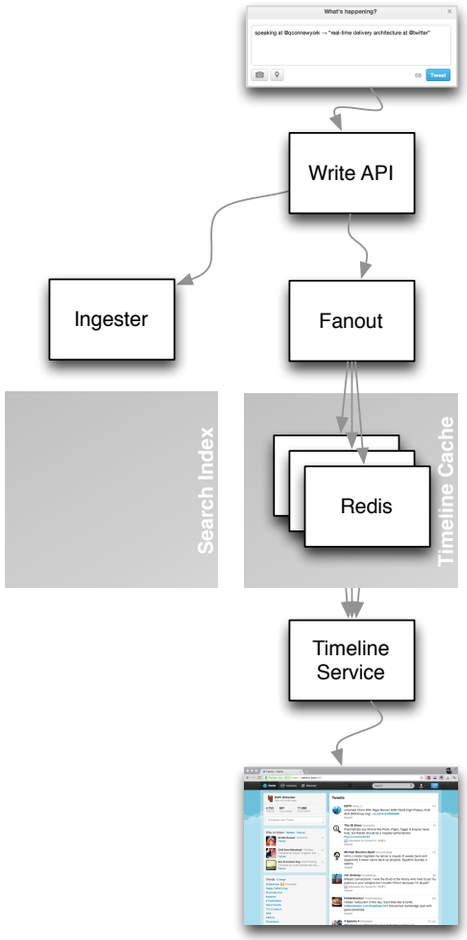

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Fanout
Redis
Timeline Cache
Timeline Service

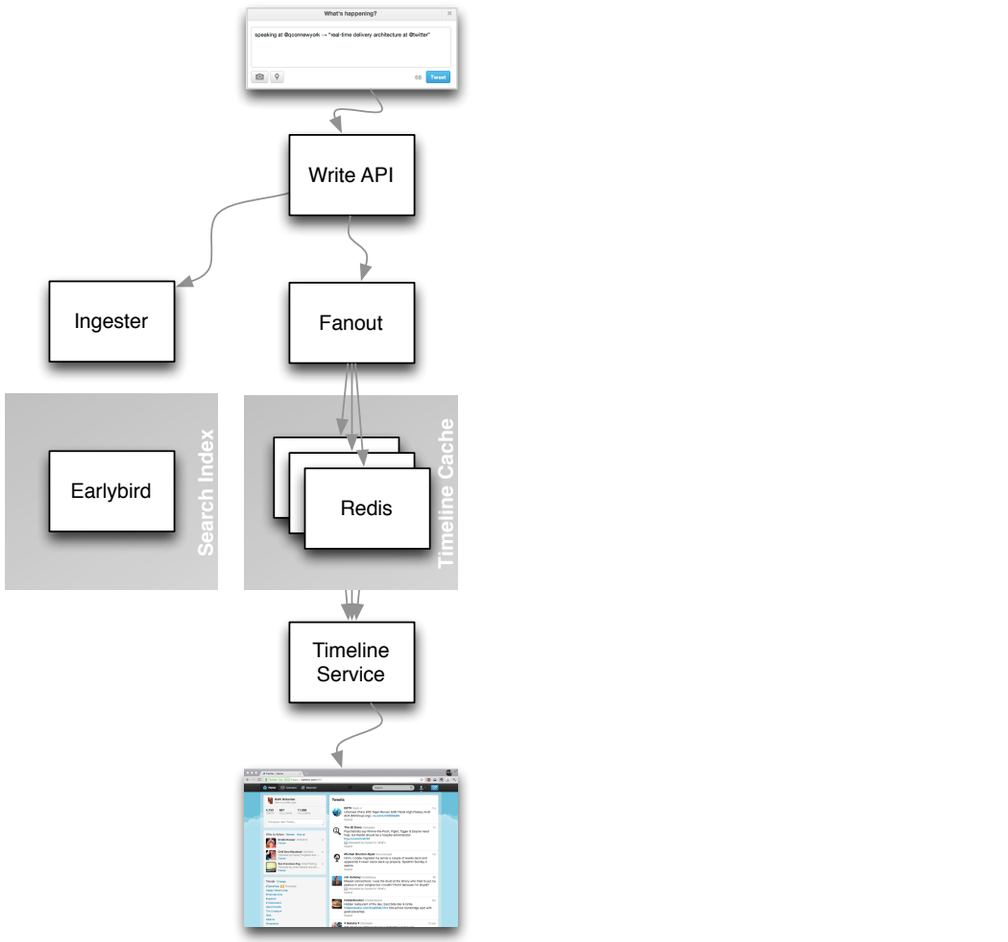

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Fanout
Redis
Timeline Cache
Timeline Service

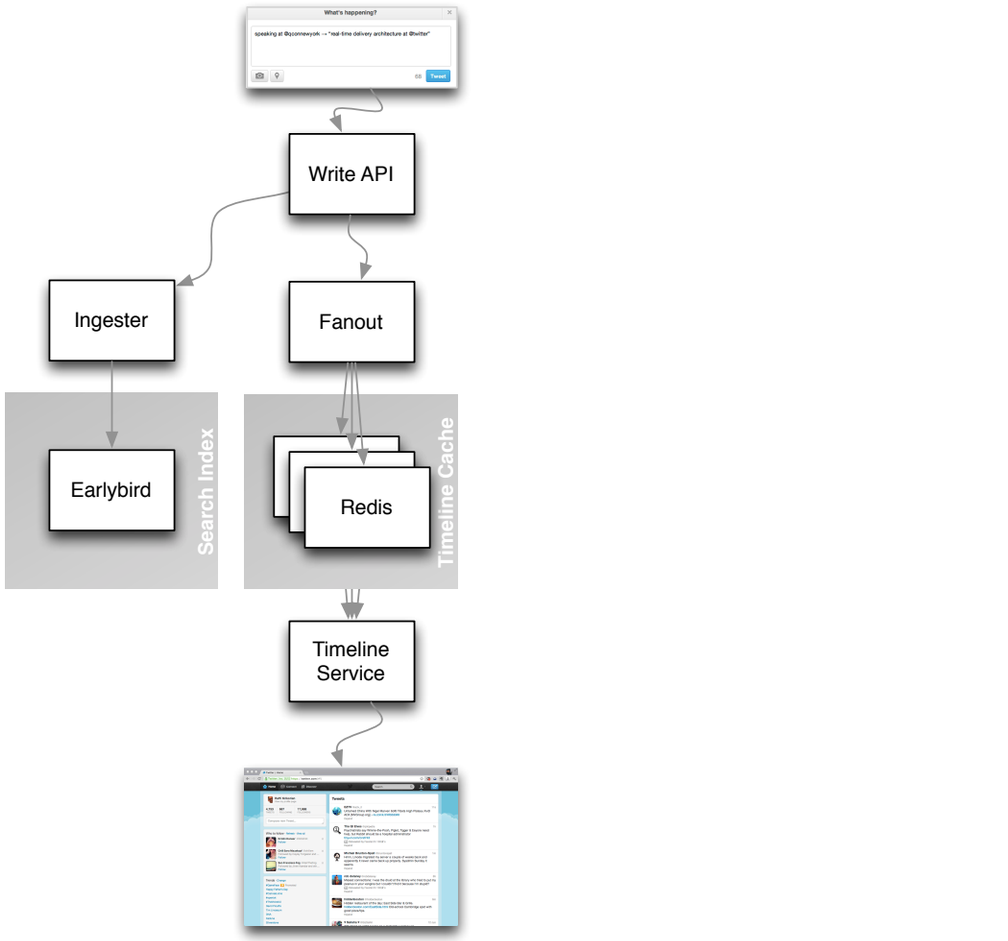

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Fanout
Redis
Timeline Cache
Timeline Service

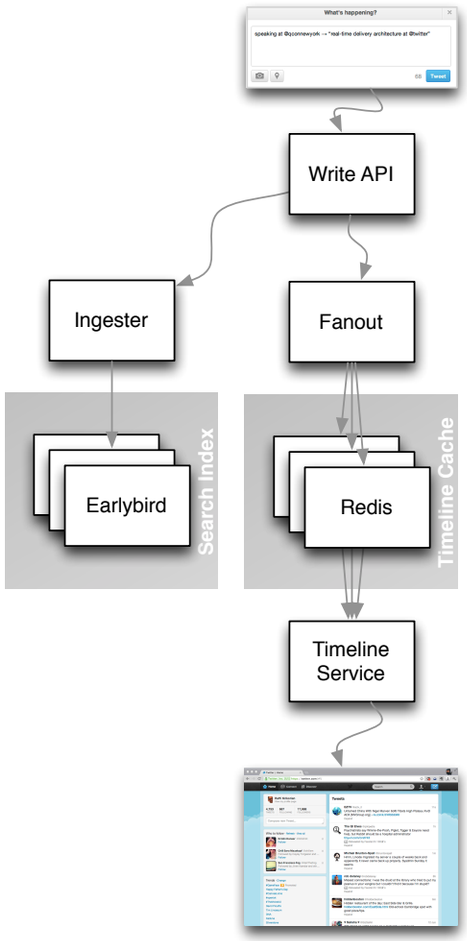

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Ingester
Fanout
Redis
Timeline Cache
Timeline Service

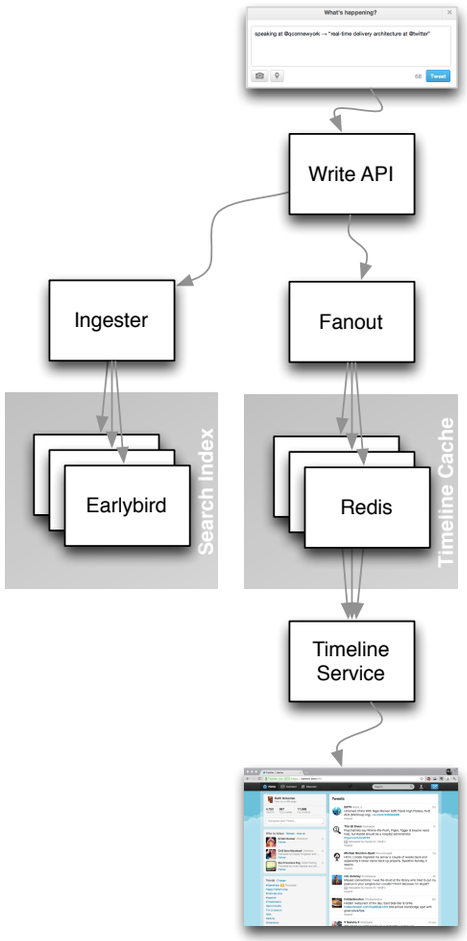

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Ingester
Fanout
Redis
Timeline Cache
Timeline Service

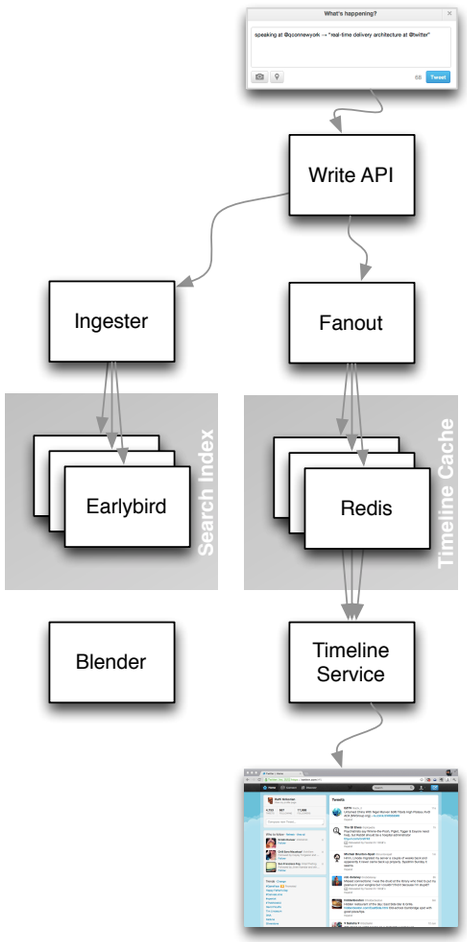

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Ingester
Fanout
Search Index
Redis
Timeline Cache
Timeline Service

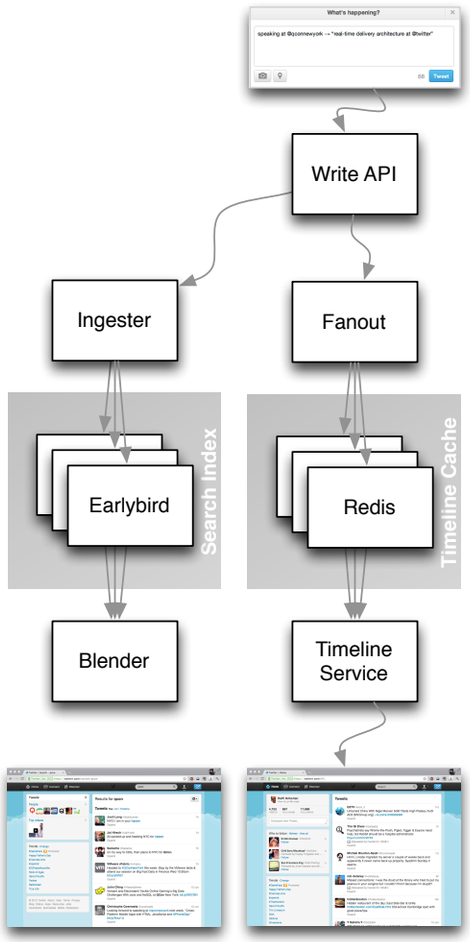

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Ingester
Fanout
Earlybird
Search Index
Redis
Timeline Cache
Timeline Service

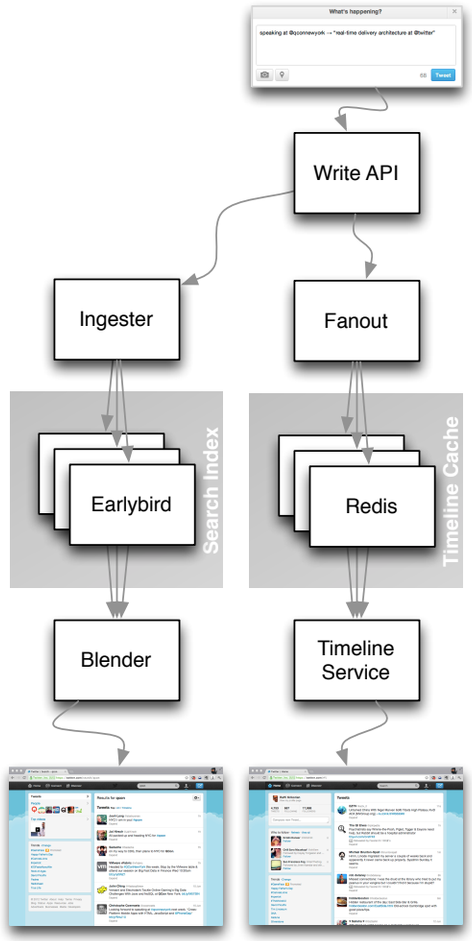

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Ingester
Fanout
Earlybird
Search Index
Redis
Timeline Cache
Timeline Service

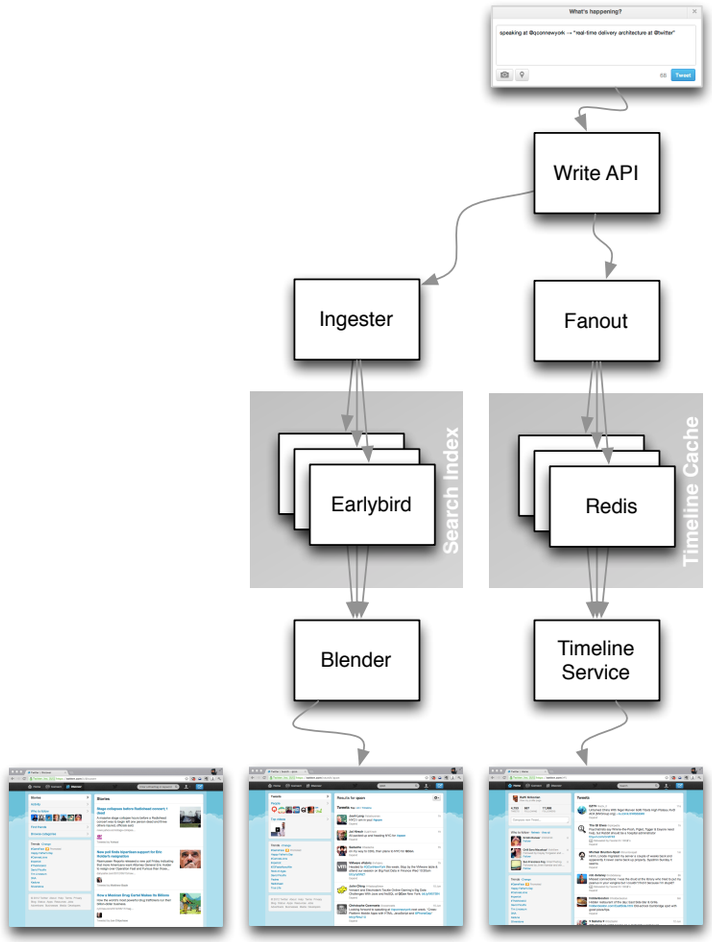

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Ingester
Fanout
Earlybird
Search Index
Redis
Timeline Cache
Timeline Service

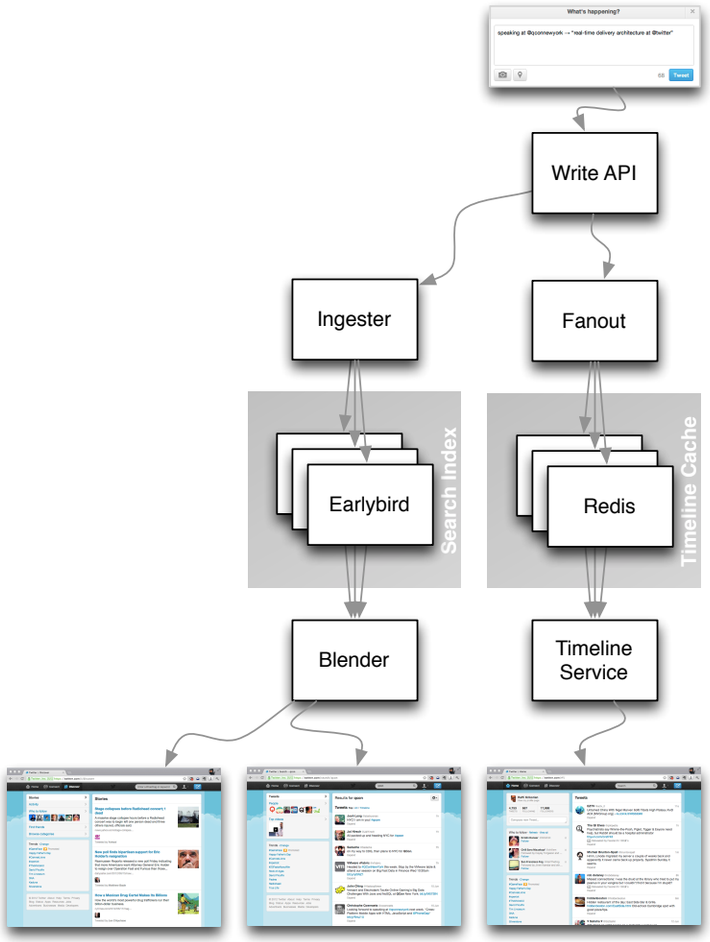

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Ingester
Fanout
Earlybird
Search Index
Redis
Timeline Cache
Timeline Service

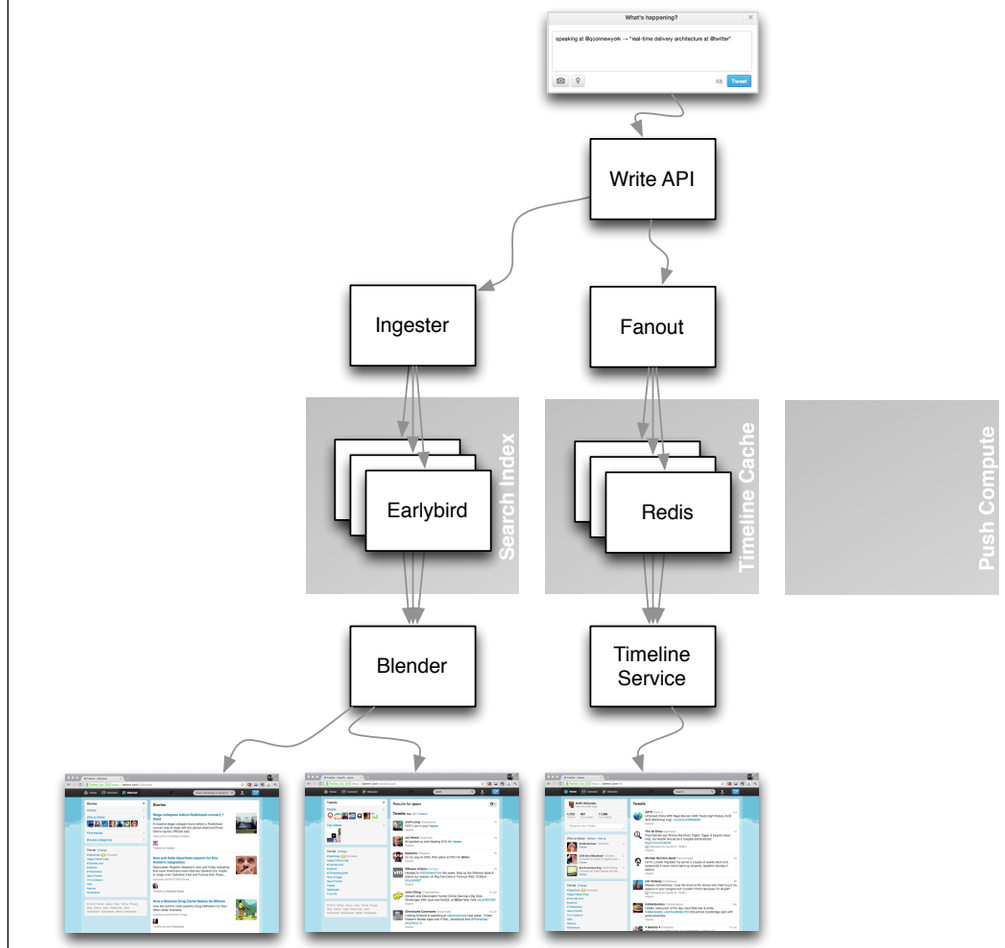

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Ingester
Fanout
Earlybird
Search Index
Redis
Timeline Cache
Blender
Timeline Service

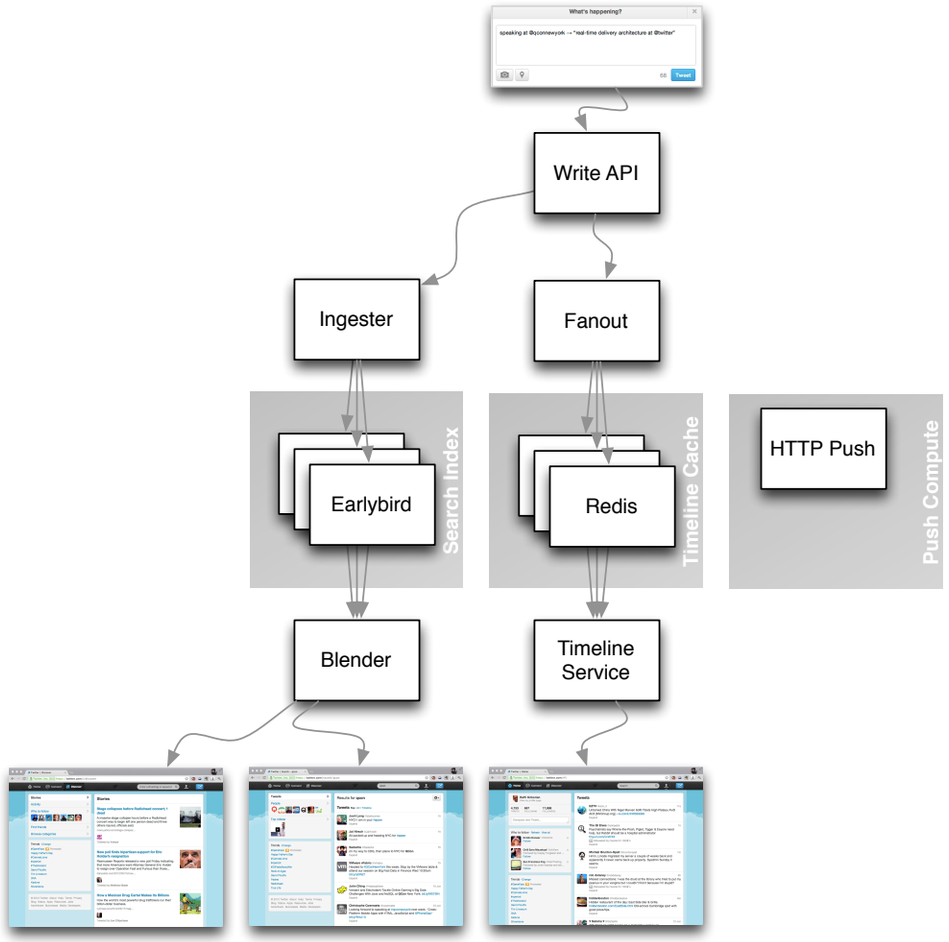

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Ingester
Fanout
Search Index
Earlybird
Timeline Cache
Redis
Blender
Timeline Service

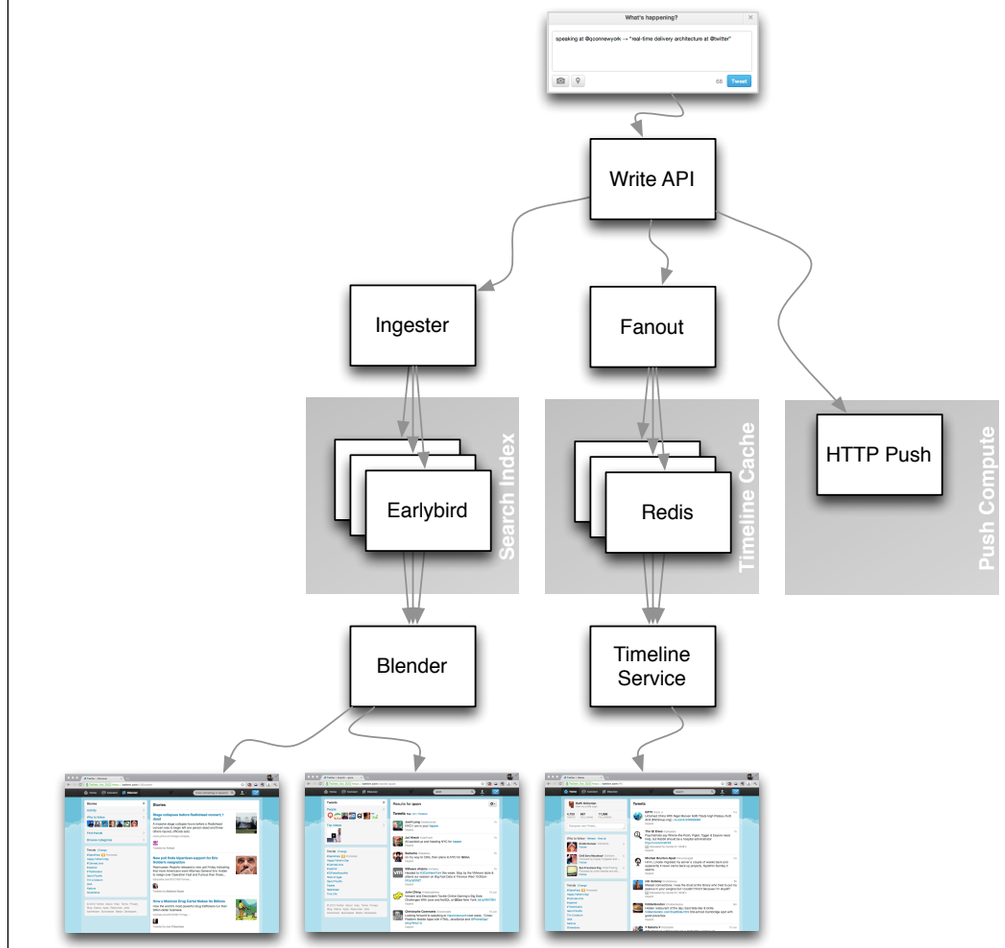

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Ingester
Fanout
Earlybird
Search Index
Redis
Timeline Cache
Blender
Timeline Service

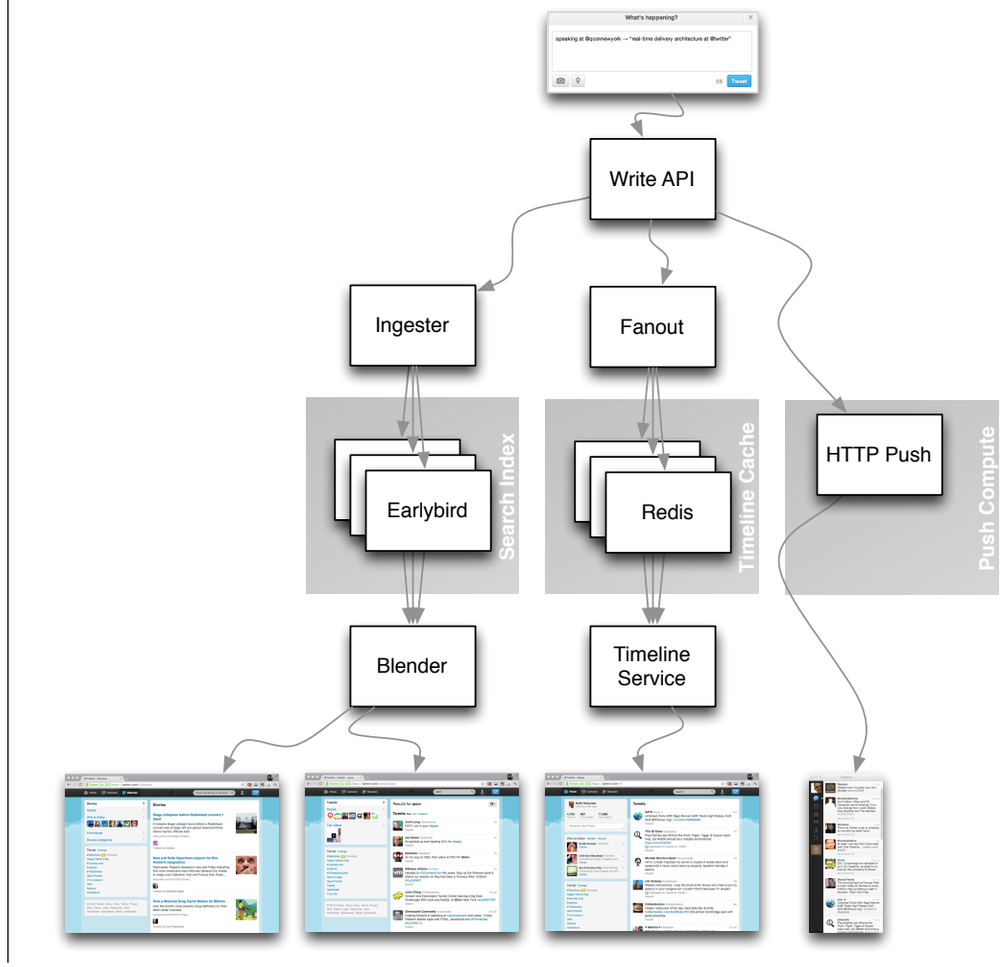

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Ingester
Fanout
Earlybird
Search Index
Redis
Timeline Cache
Blender
Timeline Service

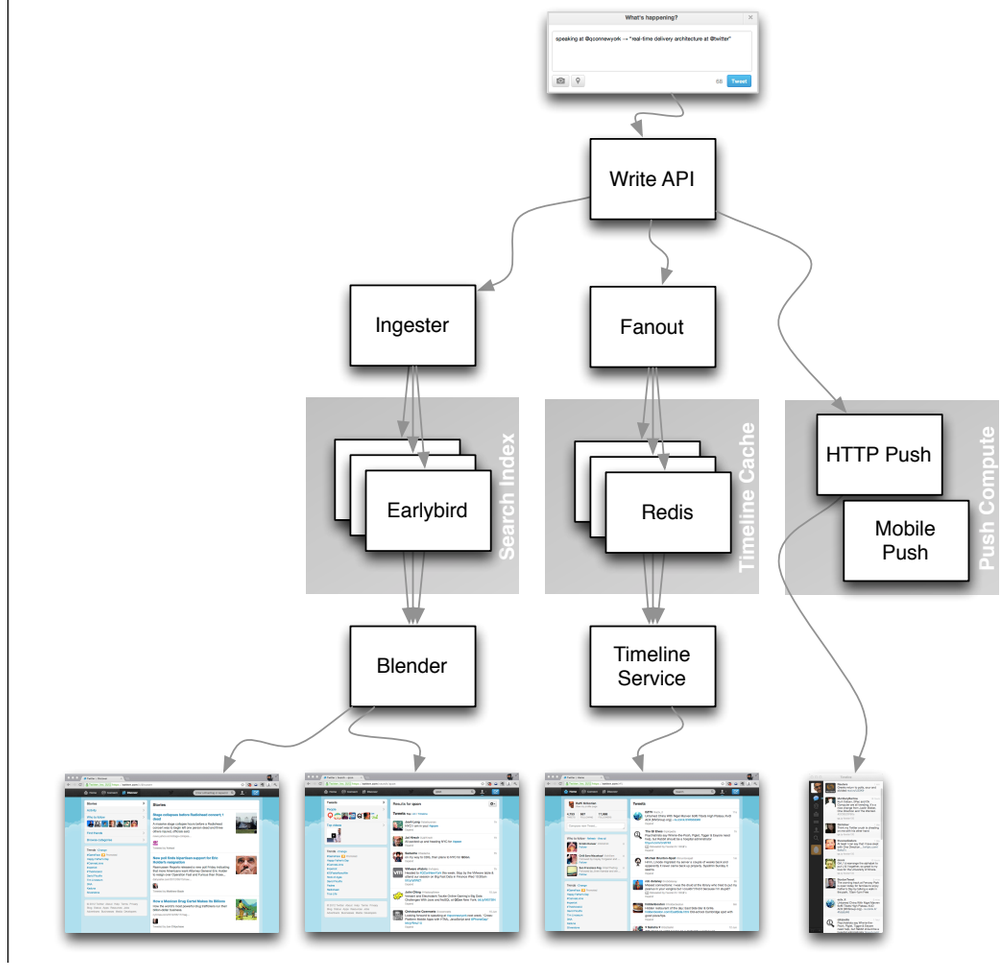

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Ingester
Fanout
Earlybird
Search Index
Redis
Timeline Cache
Blender
Timeline Service

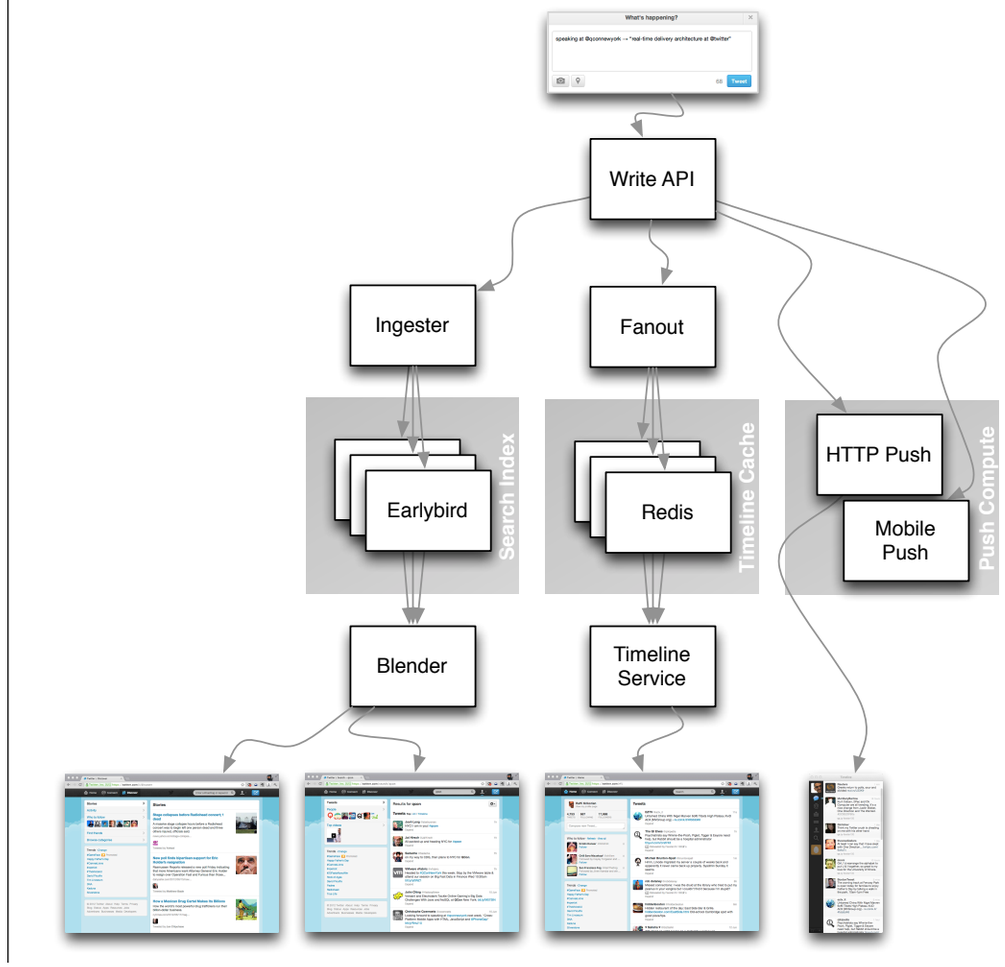

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Ingester
Fanout
Earlybird
Search Index
Redis
Timeline Cache
Push Compute
Blender
Timeline Service

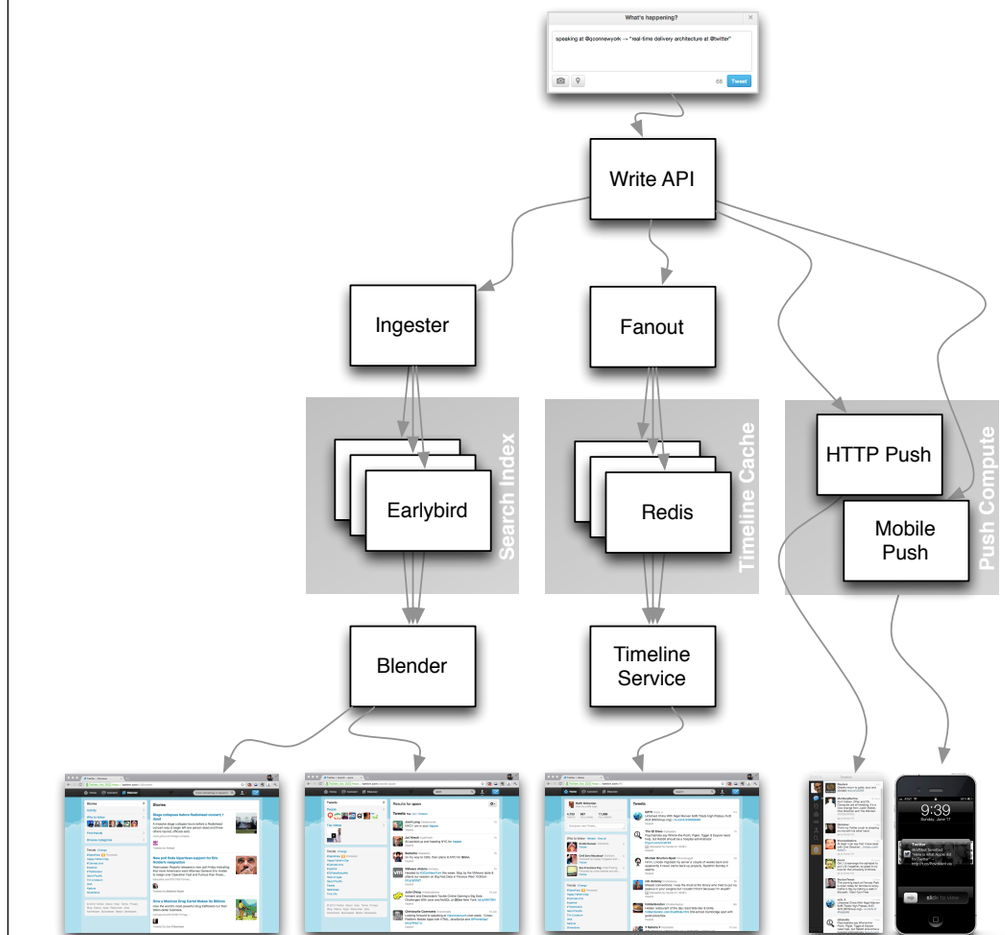

What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Ingester
Fanout
Earlybird
Search Index
Redis
Timeline Cache
HTTP Push
Push Compute
Blender
Timeline Service


What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Ingester
Fanout
HTTP Push
Earlybird
Search Index
Redis
Timeline Cache
Push Compute
Blender
Timeline Service


What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Ingester
Fanout
Earlybird
Search Index
Redis
Timeline Cache
HTTP Push
Push Compute
Blender
Timeline Service


What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
66
Tweet
Write API
Ingester
Fanout
Search Index
Earlybird
Timeline Cache
Redis
HTTP Push
Mobile Push
Push Compute
Blender
Timeline Service


What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Ingester
Fanout
Search Index
Earlybird
Timeline Cache
Redis
HTTP Push
Mobile Push
Push Compute
Blender
Timeline Service


What's happening?
speaking at @qconnewyork → "real-time delivery architecture at @twitter"
Tweet
Write API
Ingester
Fanout
Search Index
Earlybird
Timeline Cache
Redis
HTTP Push
Mobile Push
Push Compute
Blender
Timeline Service
9:39

Timeline Query Statistics

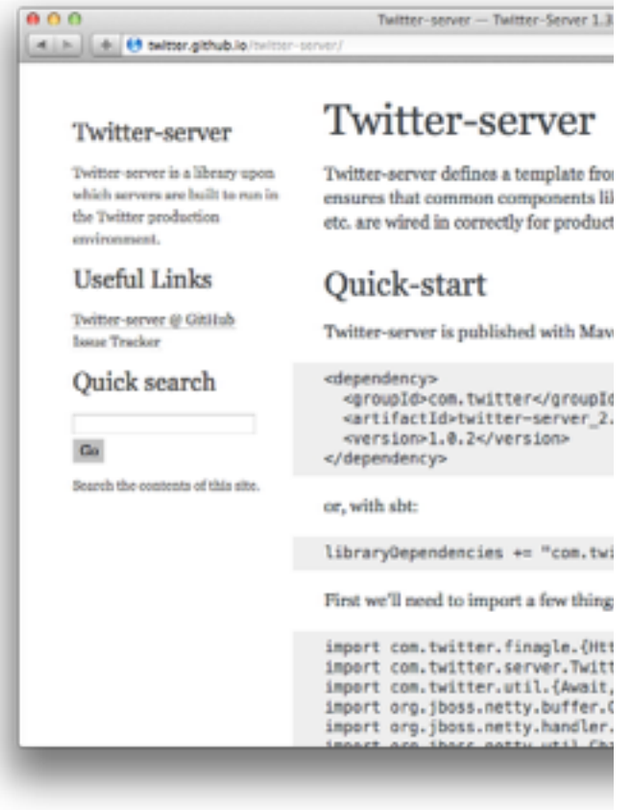
- >240m active users
- 300k qps poll-based timelines
  @1ms p50 / 4ms p99

How to build a Service

@bmdhacks
qcon london 2014

Twitter-server

- configuration
- administration
- logging
- lifecycle
- metrics



Finagle is an extensible RPC system for the JVM, used to construct high-concurrency servers

finagle

- service discovery
- load balancing
- retrying
- thread/connection pooling
- stats collection
- distributed tracing

Finagle is an extensible RPC system for the JVM, used to construct high-concurrency servers

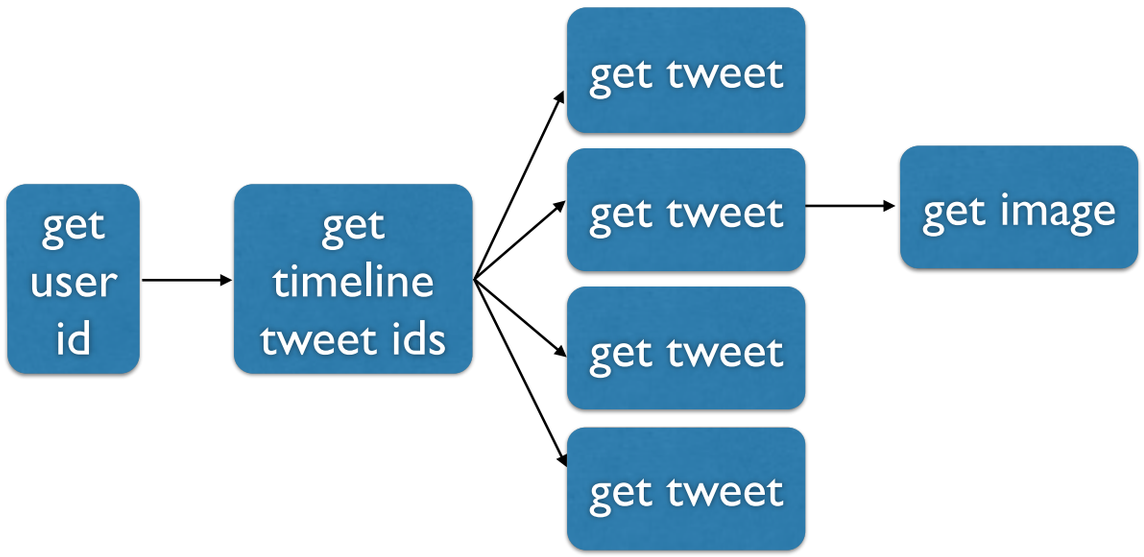
Your Service as a Function

```
trait Service[Req,Rep] extends (Req => Future[Rep])
```

Future[T]

- Composable
- Concurrent
- Easy to reason about

Composable


get user id
get timeline tweet ids
get tweet
get tweet
get tweet
get tweet
get image

Composable

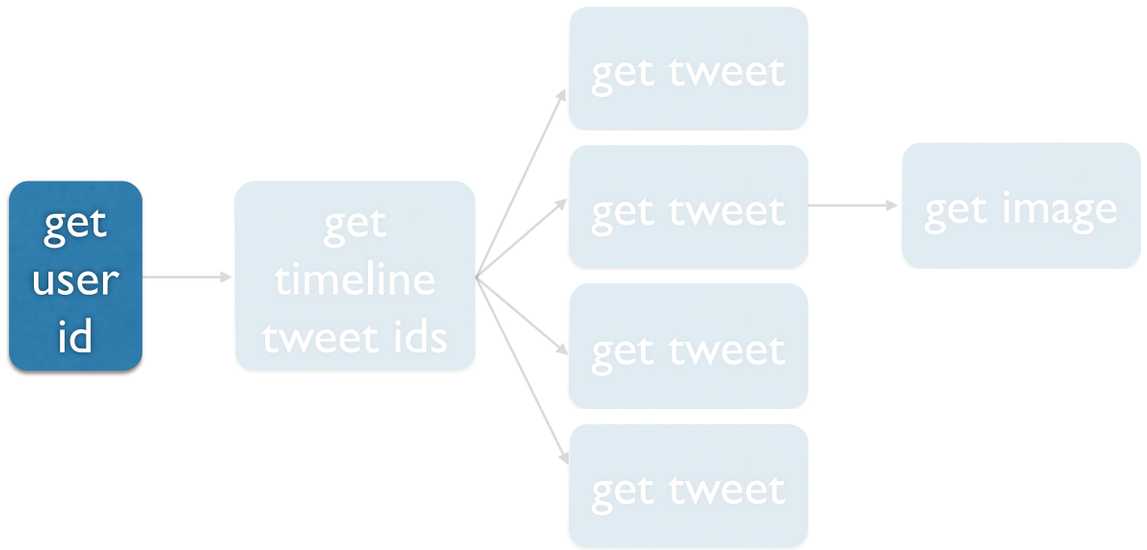
```
val user_id = Future(23)

val ftl = user_id.flatMap { id => getTimeline(id) }

val tweets: Future[Seq[Tweet]] =
 ftl.flatMap { tl =>
    val future_tweets = tl.tweet_ids.map { id =>
      val tweet = getTweet(id)
      tweet.map { t =>
        if (t.hasImage) getImage(t) else t
      }
    }
    Future.collect(future_tweets)
  }
```

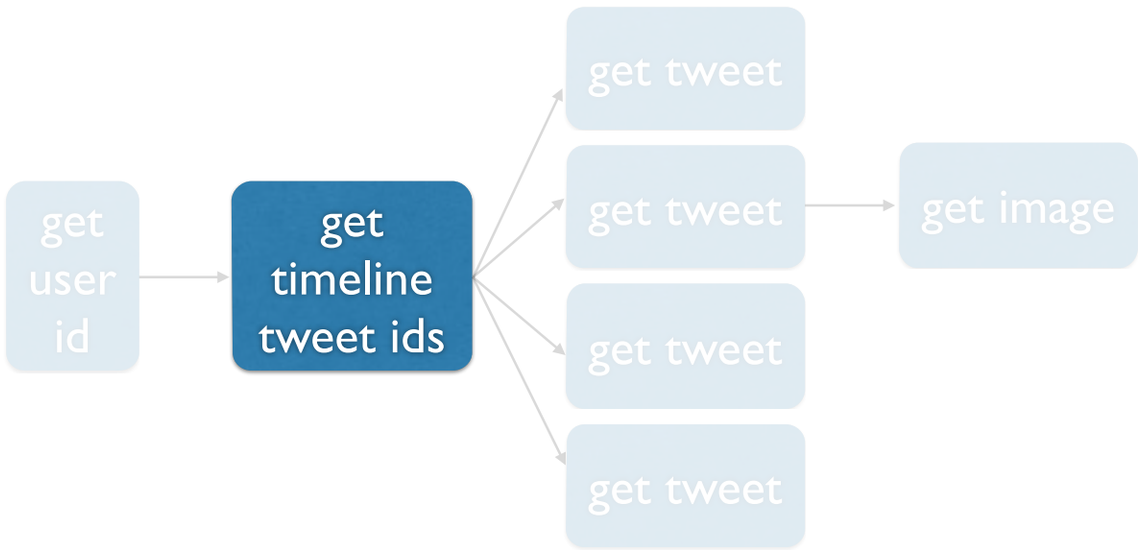
Composable

```
val user_id = Future(23)
```



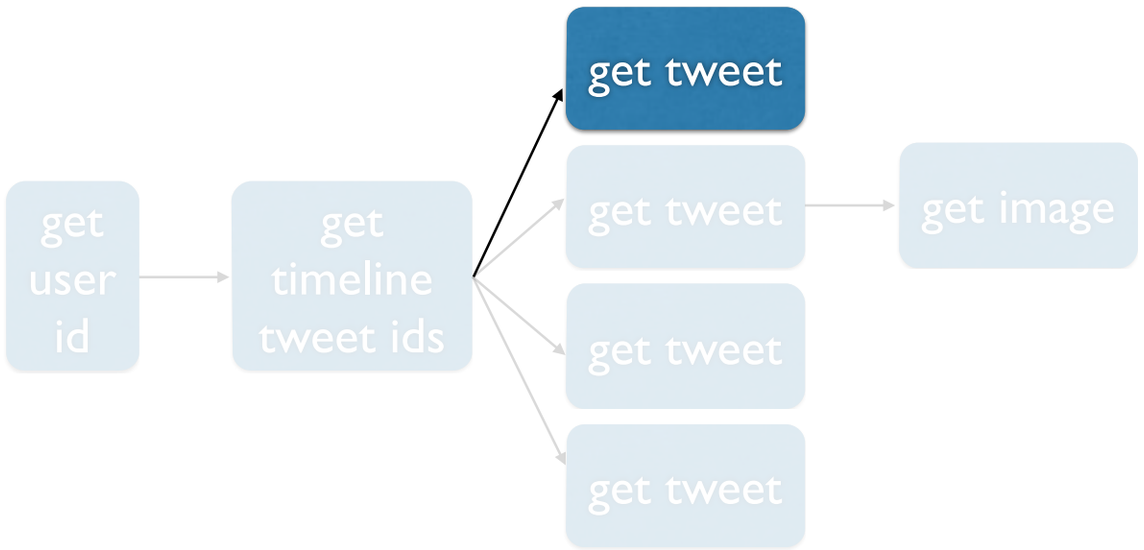
Composable

```
val ftl = user_id.flatMap { id => getTimeline(id) }
```



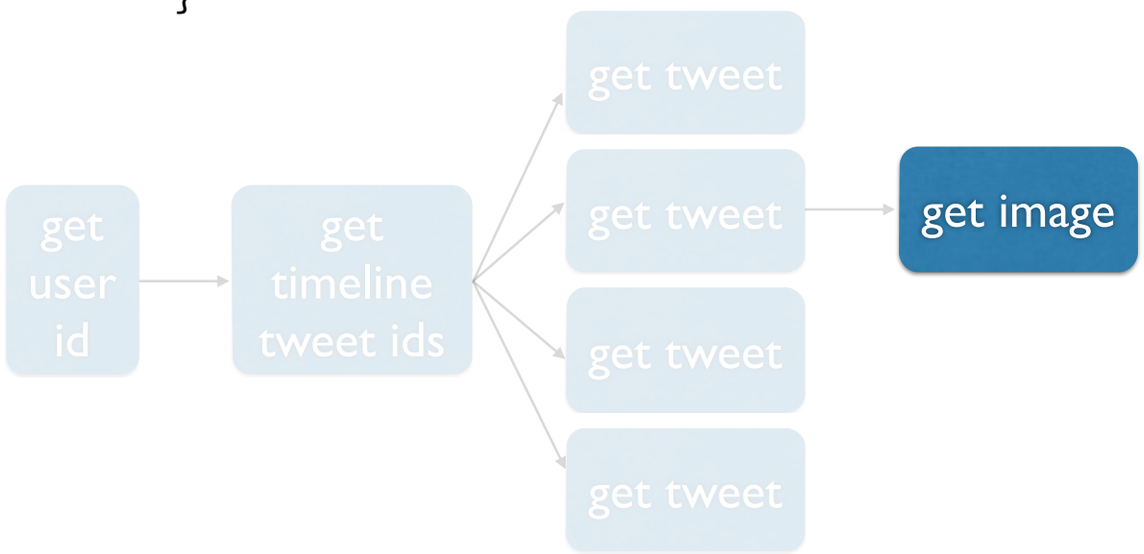
Composable

```
val future_tweets = tl.tweet_ids.map { id =>
  val tweet = getTweet(id)
```



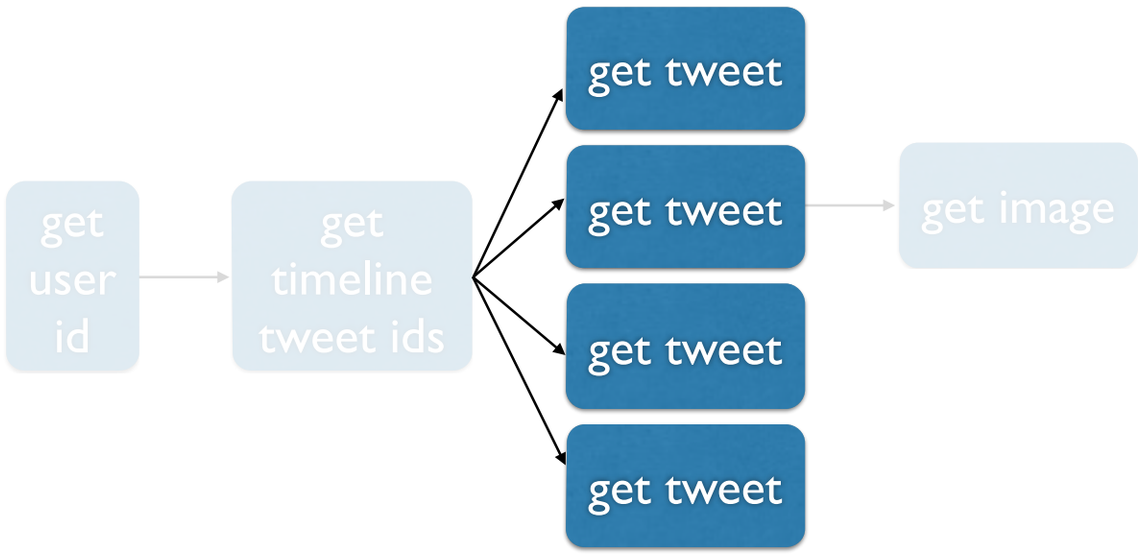
Composable

```
tweet.map { t =>
  if (t.hasImage) getImage(t) else t
}
```



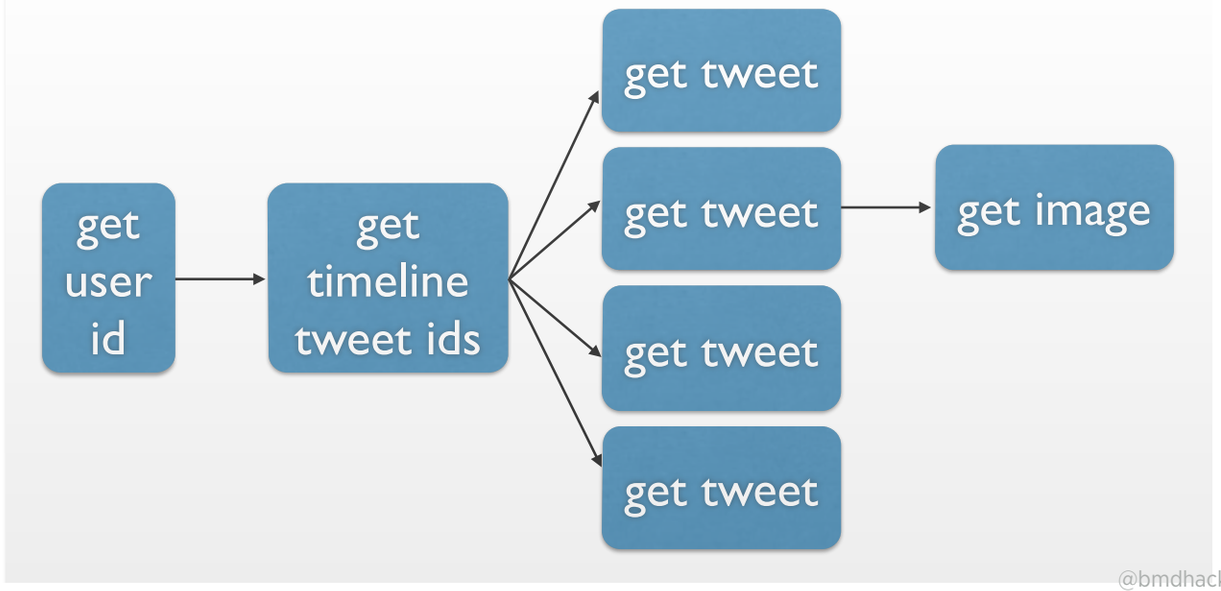
Composable

```
Future.collect(future_tweets)
```



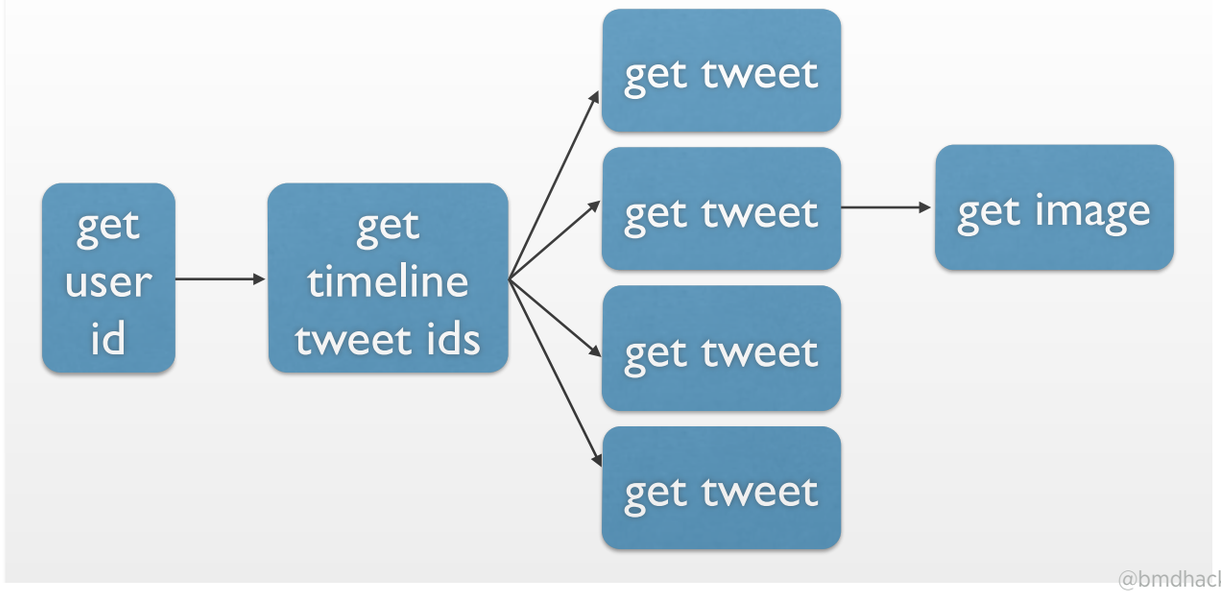
Composable

```
val timeline: Future[Timeline]
```

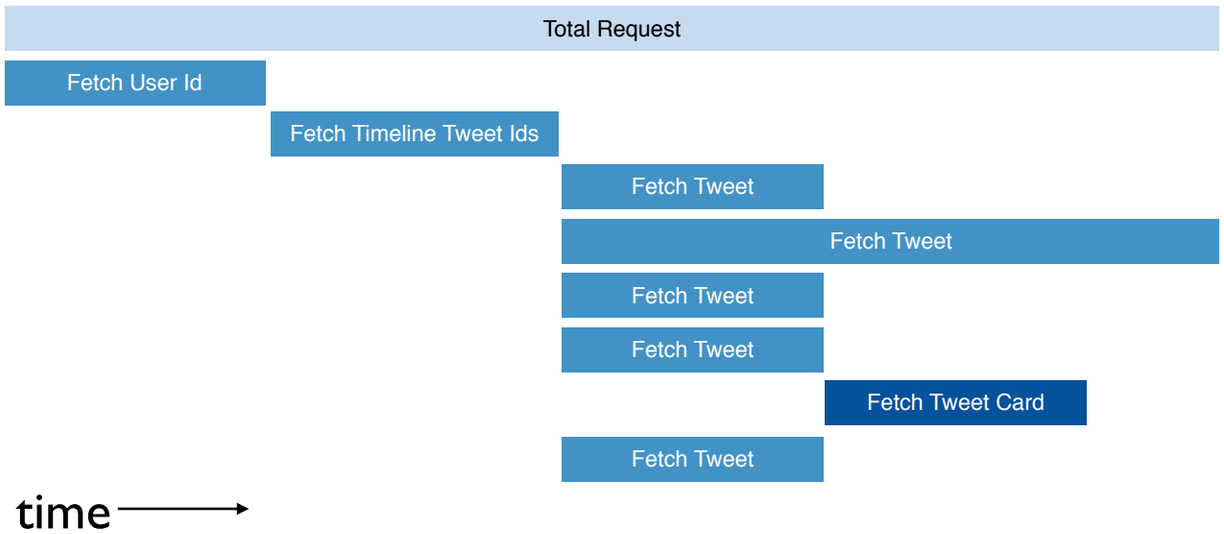


Concurrent

```
val timeline: Future[Timeline]
```



Concurrent


Total Request
Fetch User Id
Fetch Timeline Tweet Ids
Fetch Tweet
Fetch Tweet
Fetch Tweet
Fetch Tweet
Fetch Tweet Card
Fetch Tweet
time

What we want to do

How the threads execute it

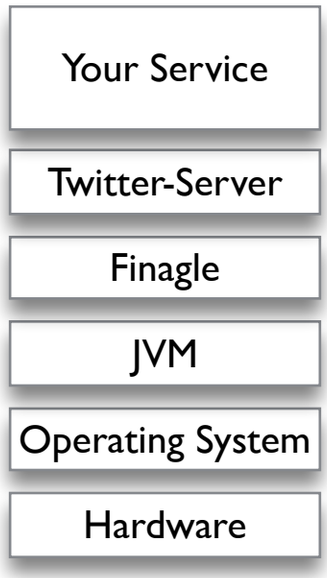
Separate semantics from execution

Your Service as a Function

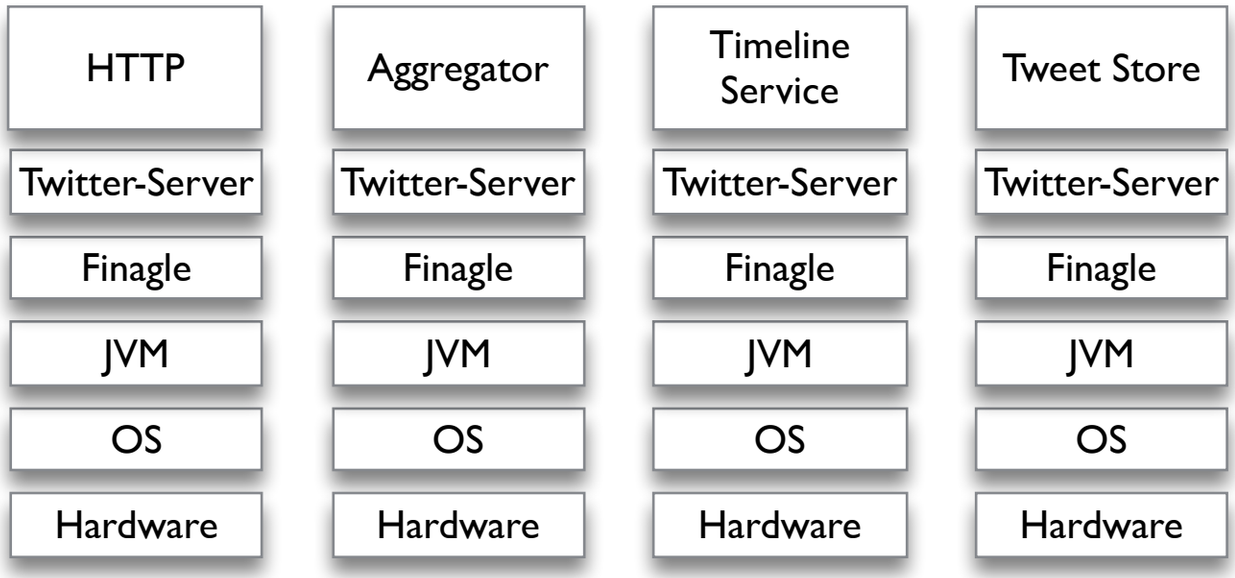
```
trait Service[Req,Rep] extends (Req => Future[Rep])
```

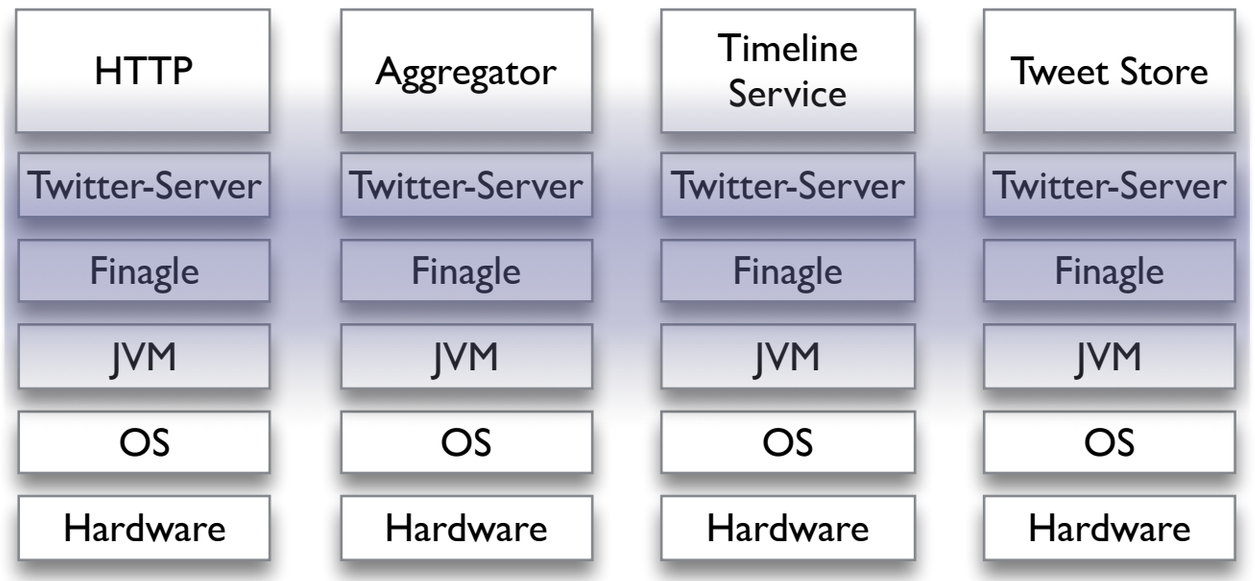
(Also your Client)

```
trait Service[Req,Rep] extends (Req => Future[Rep])
```

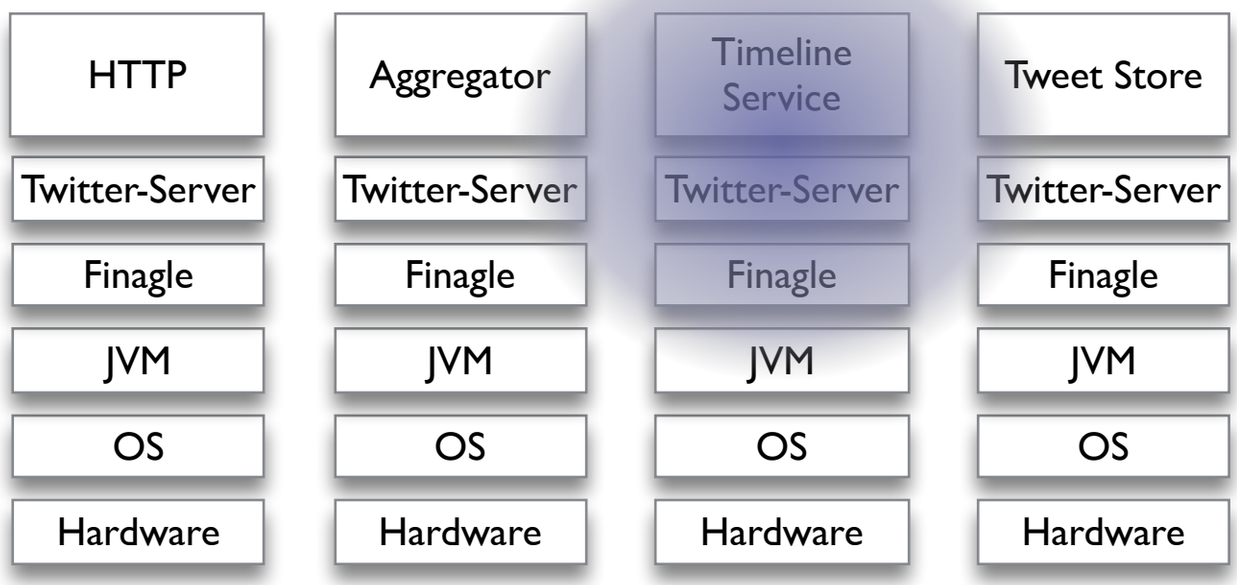

Your Service
Twitter-Server
Finagle
JVM
Operating System
Hardware

Monorail


HTTP
Aggregator
Timeline Service
Tweet Store
Twitter-Server
Twitter-Server
Twitter-Server
Twitter-Server
Finagle
Finagle
Finagle
Finagle
JVM
JVM
JVM
JVM
OS
OS
OS
OS
Hardware
Hardware
Hardware
Hardware



Expertise can be applied in both directions of our stack. My team, CSL, works on Twitter-Server and Finagle. These are used by all services, however we don’t have a ton of insight into what those services do.



Other engineers focus on specific services, and extend their knowledge down the stack for their specific use case.

@bmdhacks
qcon london 2014

Use Statistics to Monitor Behavior

@bmdhacks
qcon london 2014

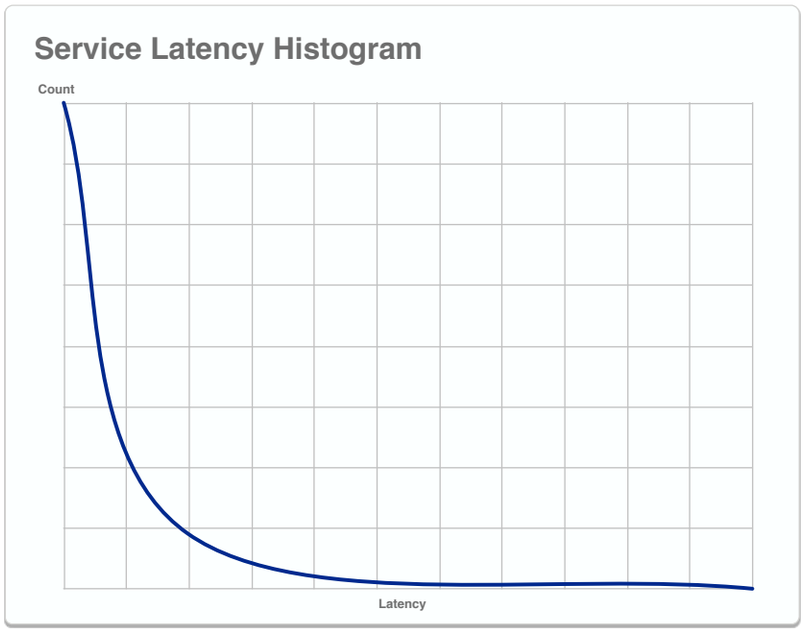
Why Stats Matter

- Increased traffic velocity makes individual events less important
- More vertical components requires more measurement
- Horizontal SOA needs more measurement

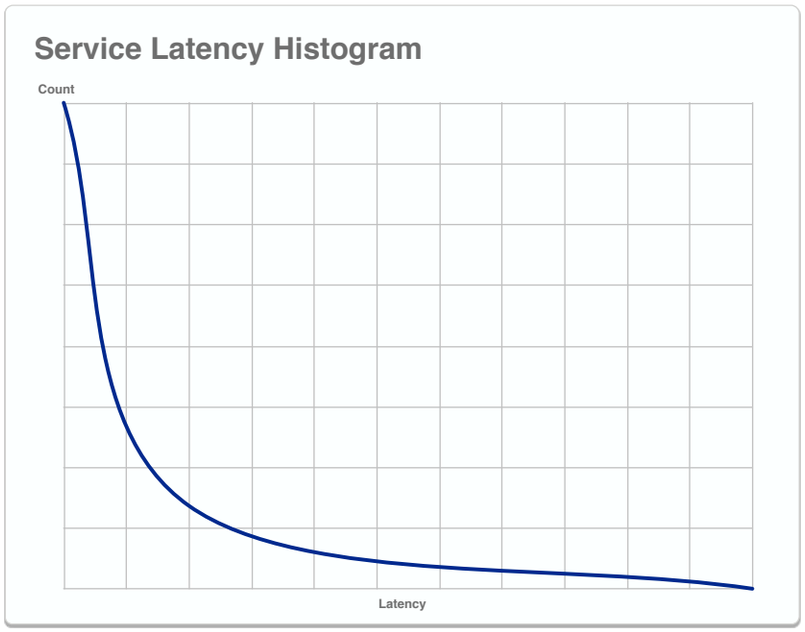
Failures are OK

- 300,000rps
- 99.99% success rate
- 30 failures per second
- Failure Rate is more informative than individual failures

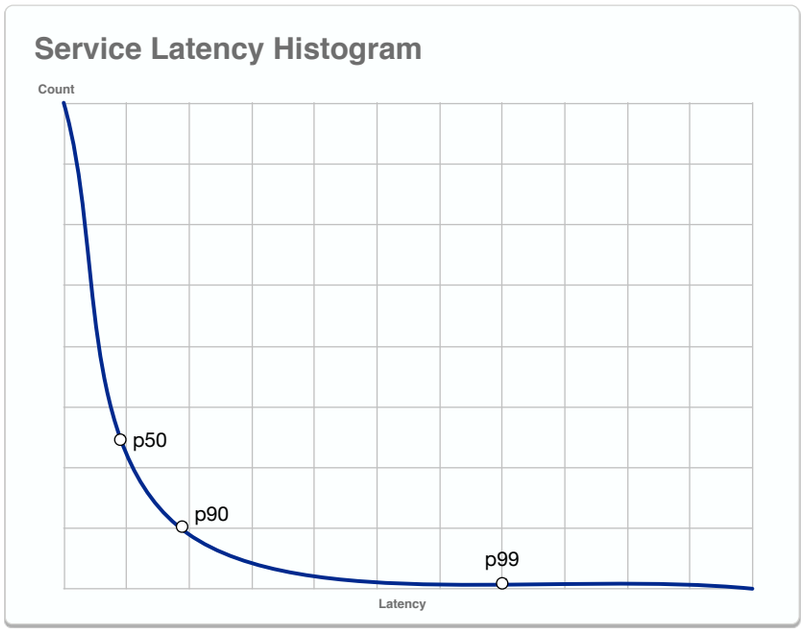
Why Stats Matter



Why Stats Matter



Why Stats Matter



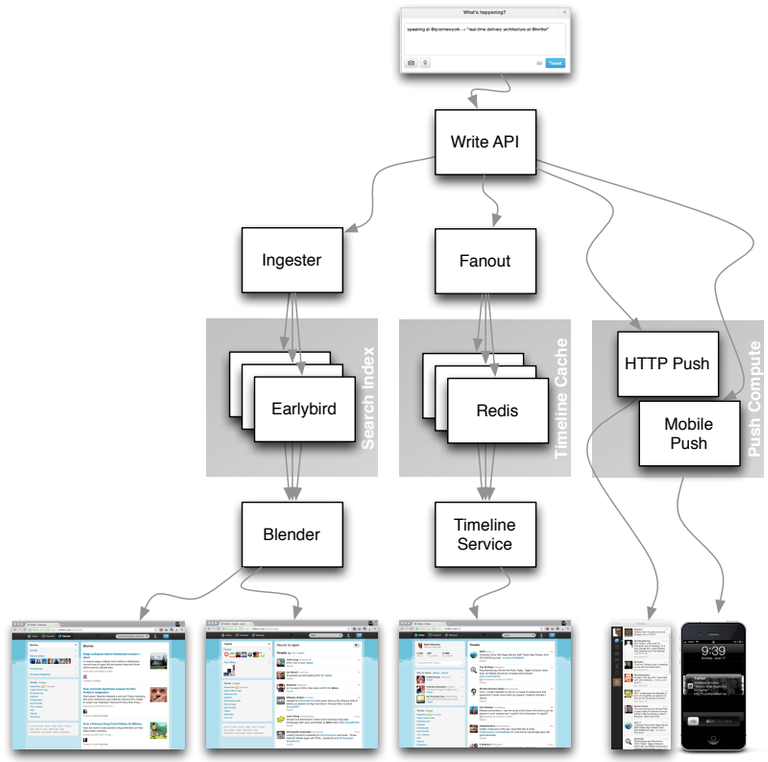
Why Stats Matter

	Before	After
Median	10ms	11ms
p90	50ms	55ms
p99	100ms	400ms
p999	500ms	500ms

Measure the Long Tail

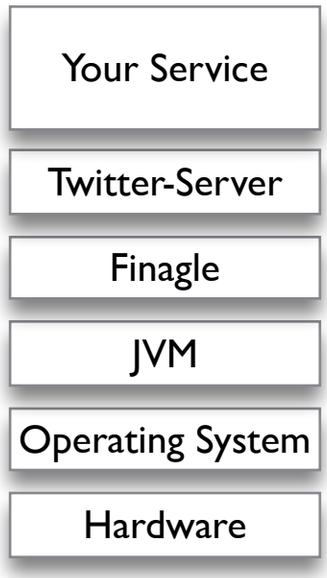
- Tail effects are cumulative
- This is why request-level concurrency is important

Tail effects are cumulative



This is also why concurrency is important.

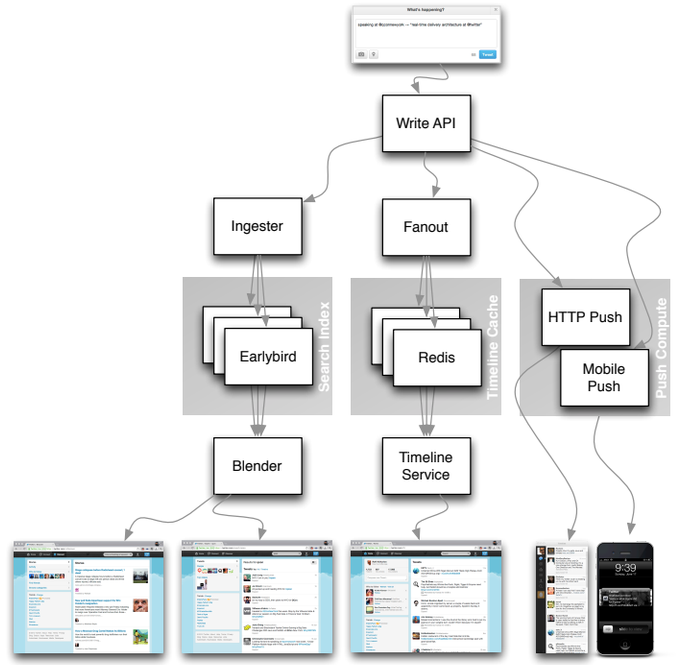
Measuring the Vertical Stack



Viz: Measuring the Vertical Stack

Viz: Measuring the Vertical Stack

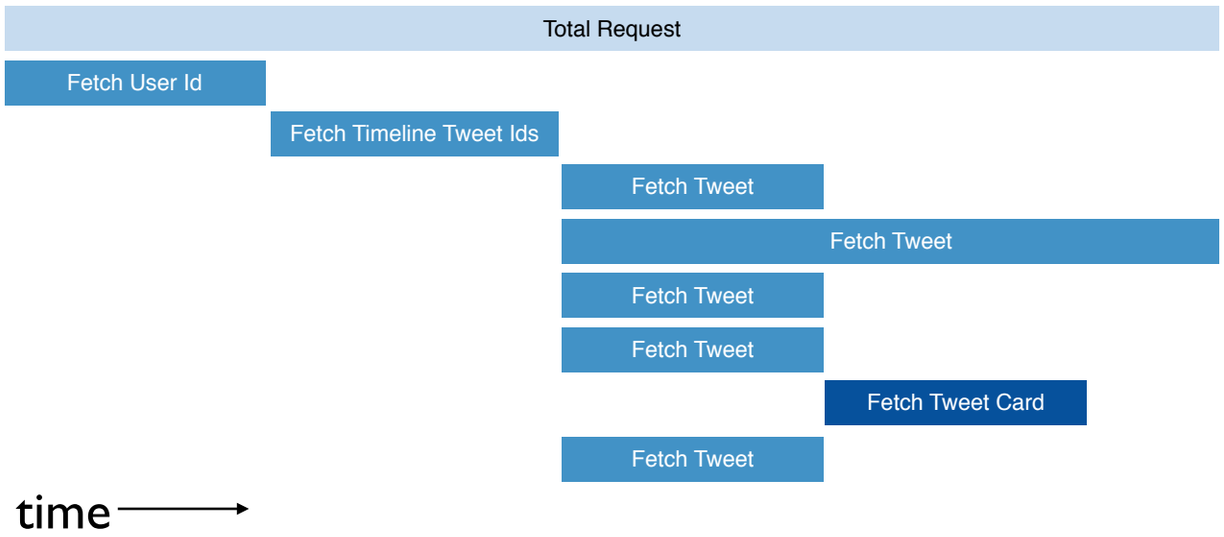
Measuring the Horizontal SOA



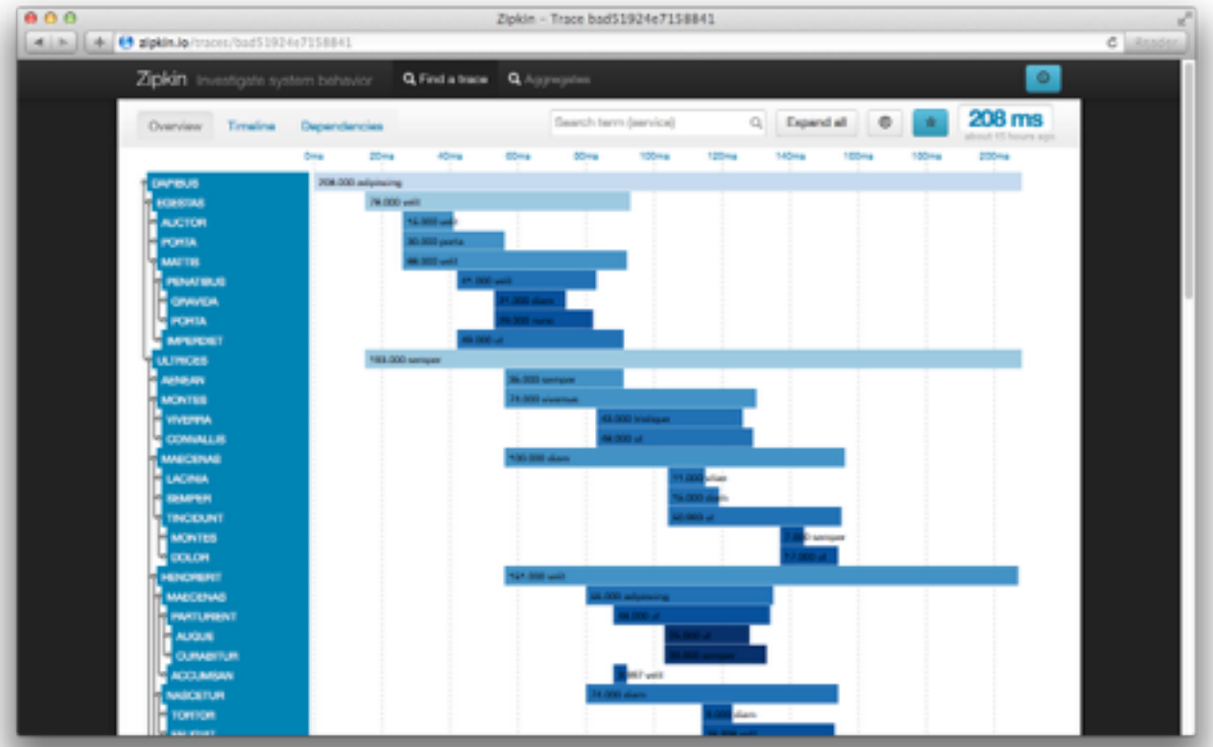
Zipkin

- It’s awesome

Remember This?

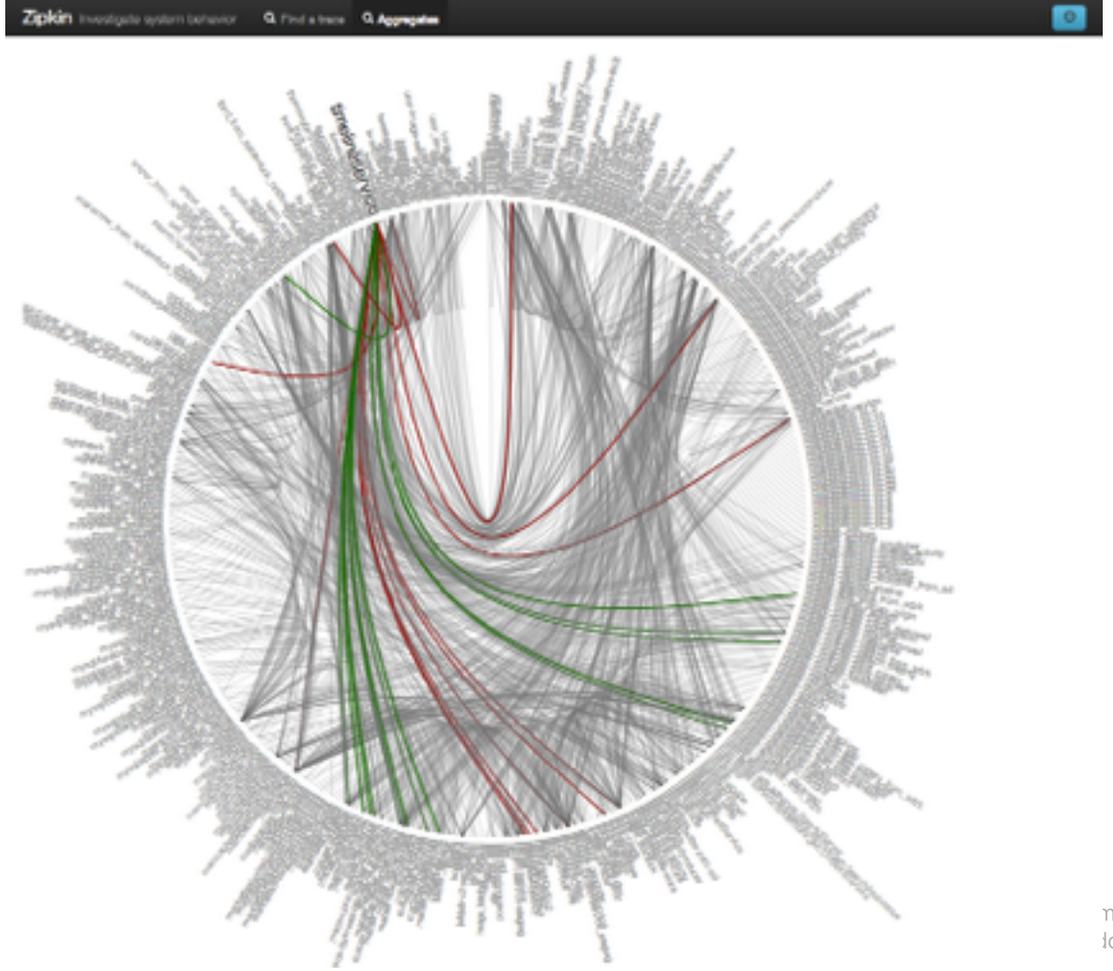


Zipkin

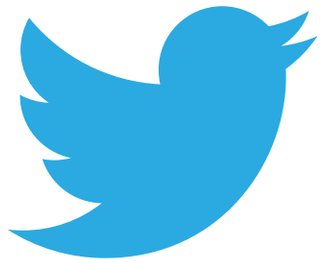

Zipkin - Trace bad51924e7158841
zipkin.io/traces/bad51924e7158841
Zipkin
Investigate system behavior
Find a trace
Aggregates
Overview
Timeline
Dependencies
Search term (service)
Expand all
208 ms

Zipkin

- Distributed Tracing System
- Open Source
- Modeled after Google's 'Dapper' paper



ndhacks
lon 2014



thanks!

References

- Timelines at Scale
  http://www.infoq.com/presentations/Twitter-Timeline-Scalability

- Finagle
  http://twitter.github.io/finagle/

- Twitter-Server
  http://twitter.github.io/twitter-server/

- Zipkin
  http://twitter.github.io/zipkin/

- Twitter's Observability Stack
  https://blog.twitter.com/2013/observability-at-twitter

- Your Server as a Function
  http://monkey.org/~marius/funsrv.pdf

QCon

Please evaluate
my talk via the
mobile app!