



Scaling to 1,000,000 concurrent users on the JVM

JavaOne 2015 - CON7220

Jo Voordeckers

Sr. Software Engineer - Livefyre platform

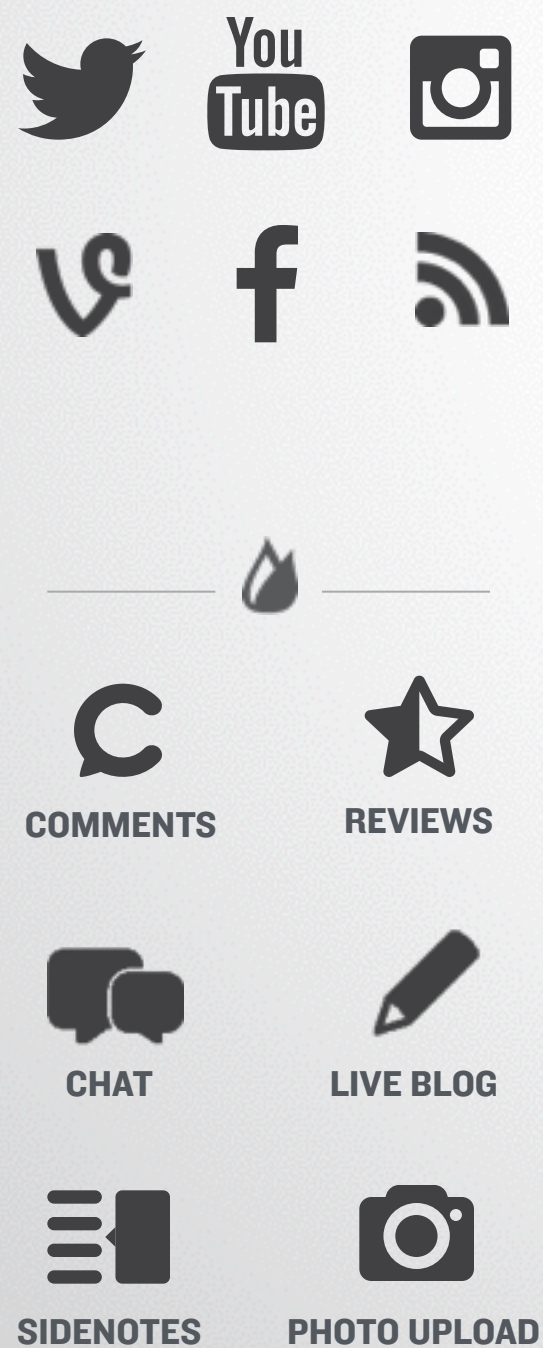
 @jovoordeckers

jvoordeckers@livefyre.com

Livefyre helps over 1,500 of the most influential
brands & media companies build an engaged audience

Collect

real-time streams of UGC
to scale content creation



Organize

to quickly find and organize
the best social content



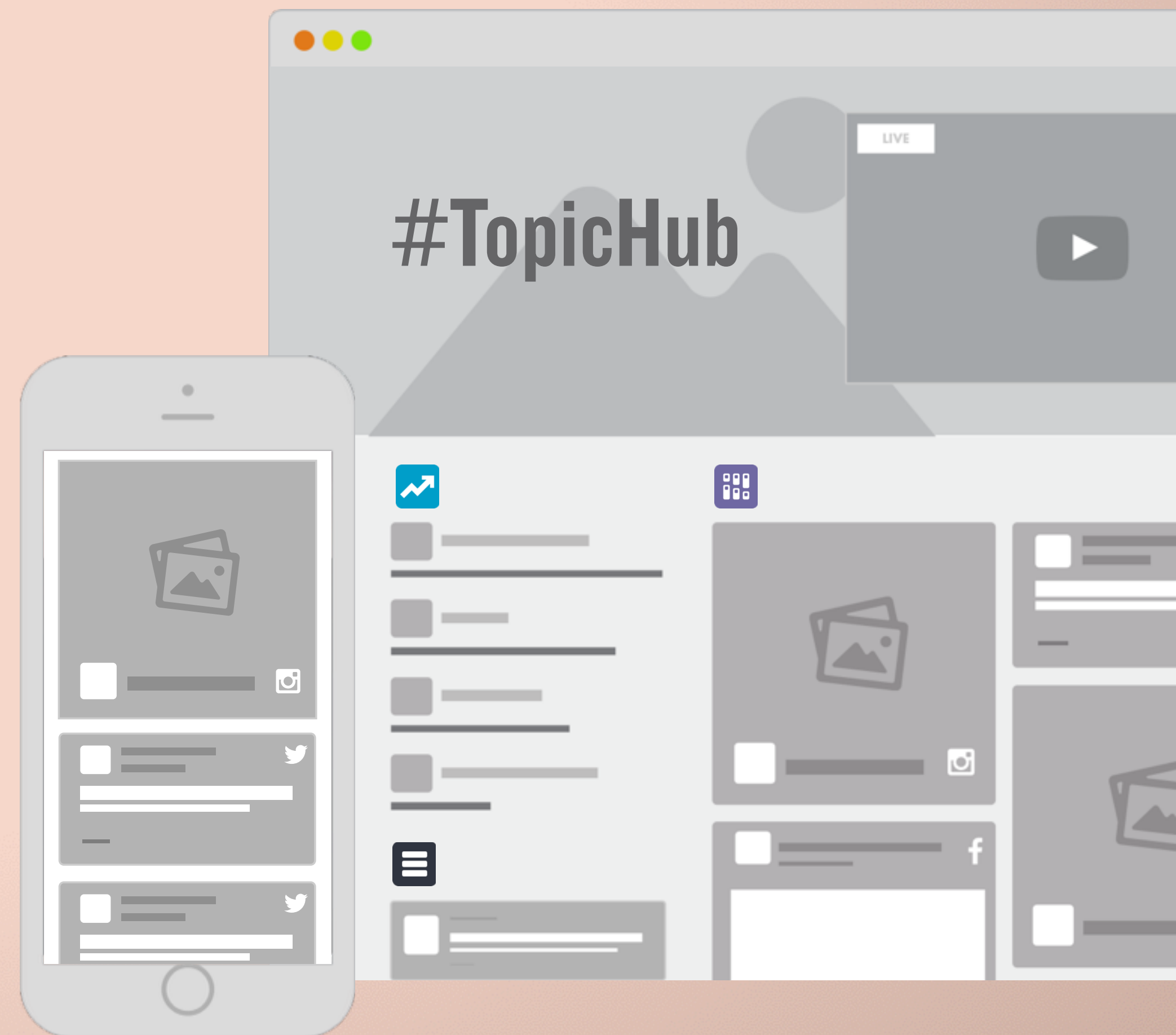
Publish

to your website with
no coding required



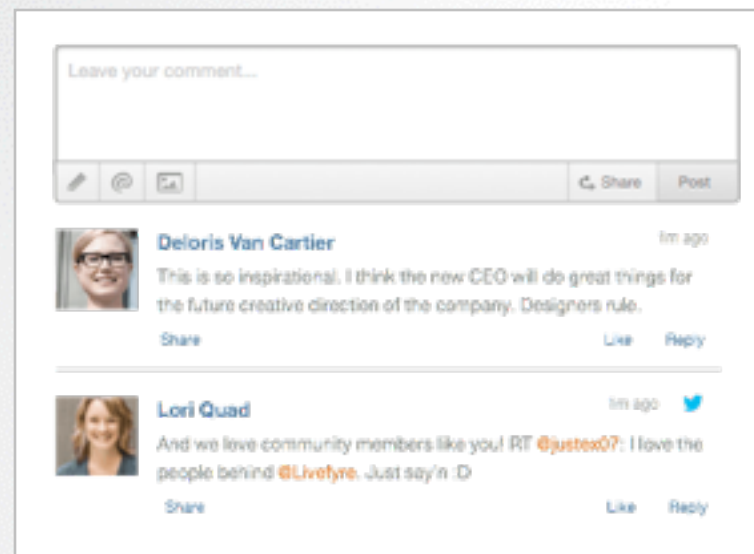
ENGAGE

audiences with best in class engagement tools
to increase time on site and build community



Real-Time Social Applications

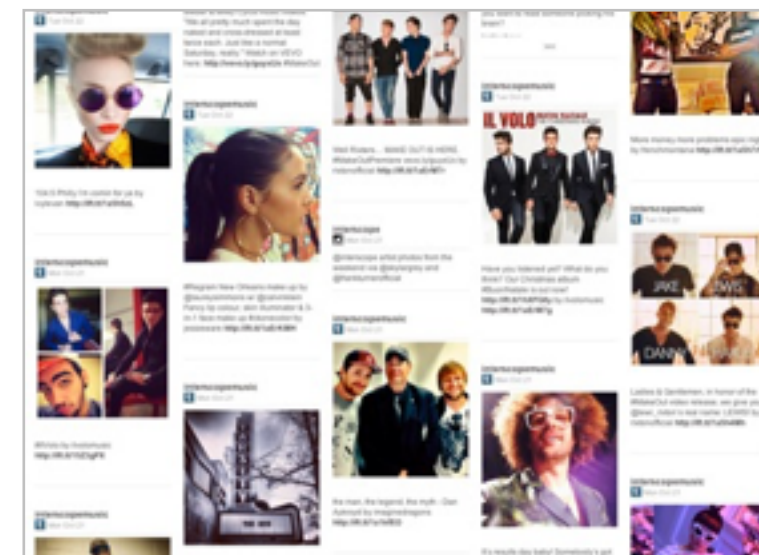
Comments



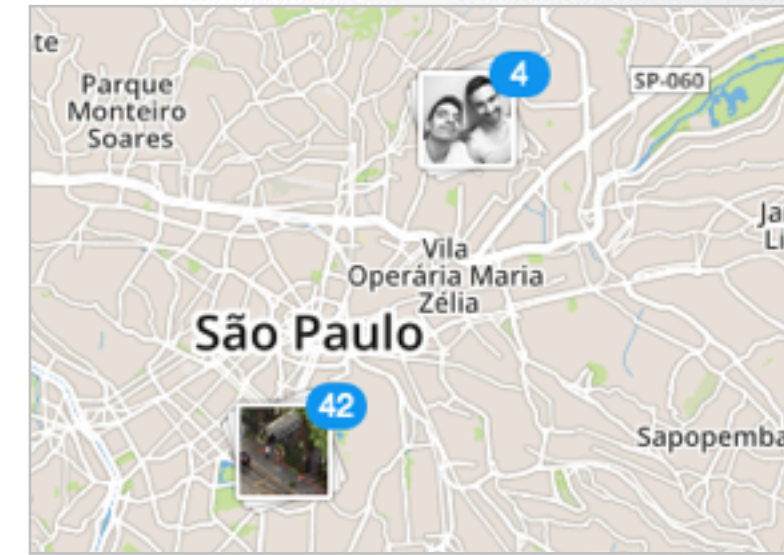
Sidenotes



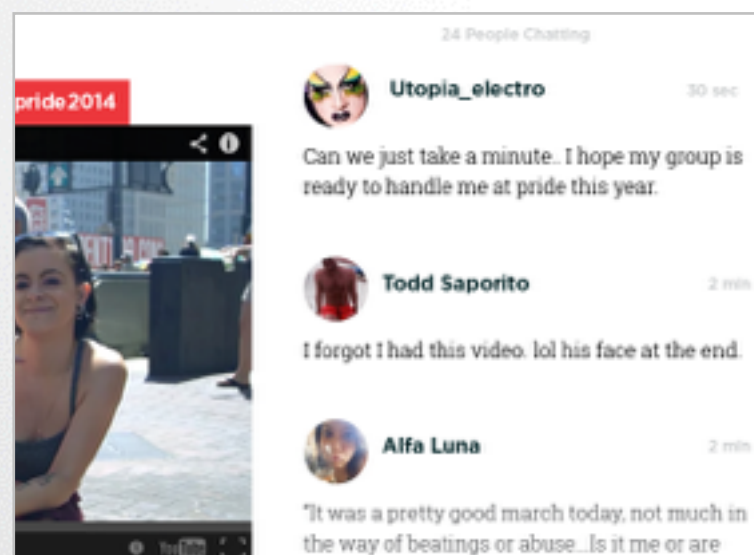
Media Wall



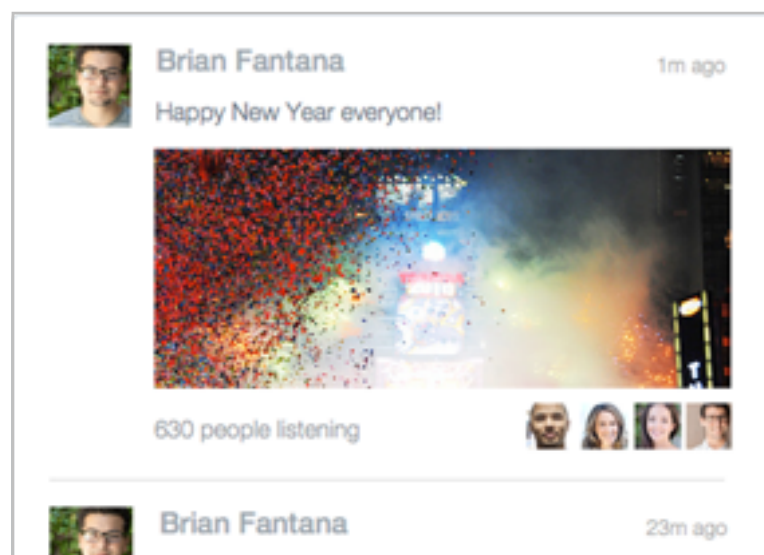
Social Maps



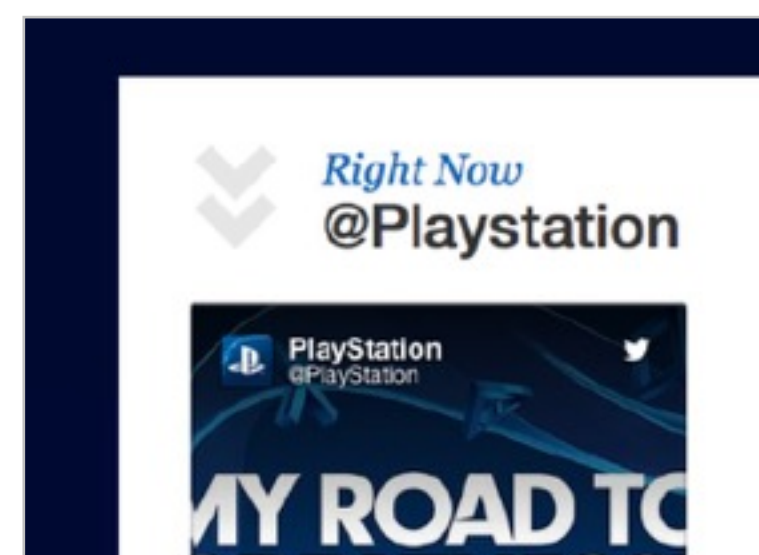
Chat



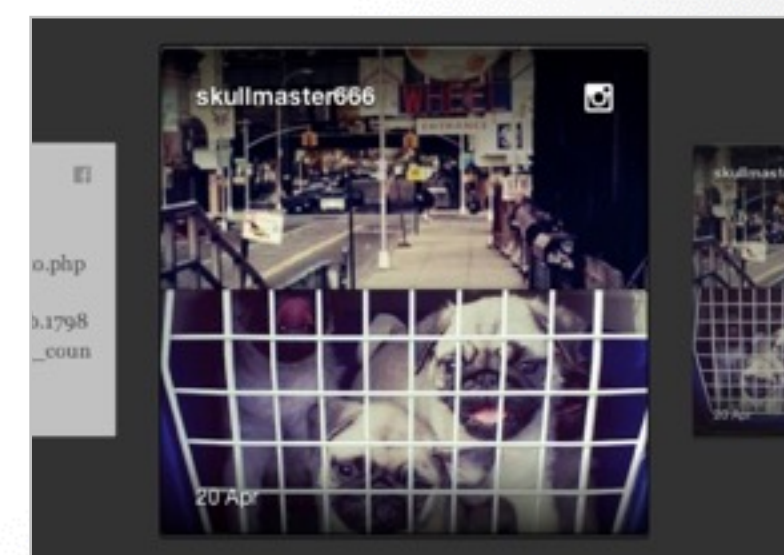
Live Blog



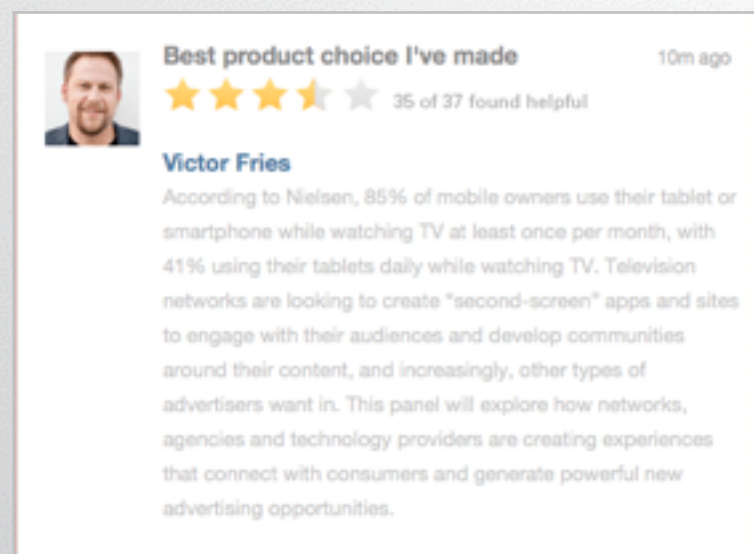
Feed



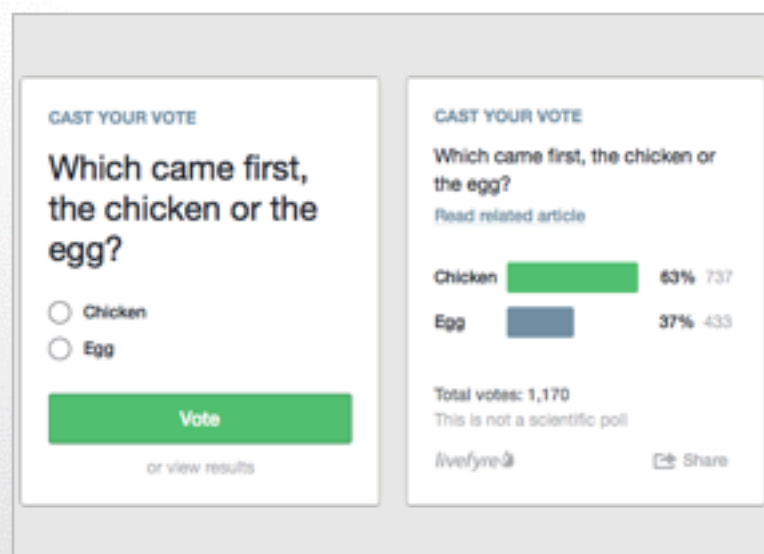
Gallery



Reviews



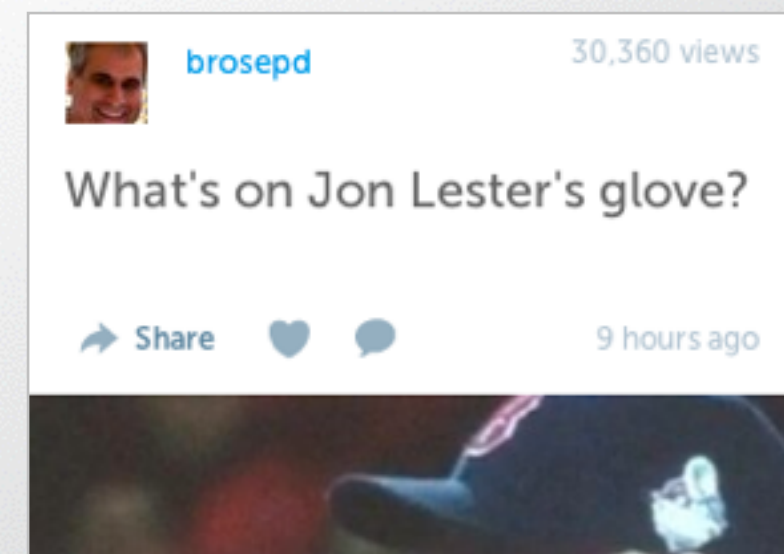
Polls



Trending



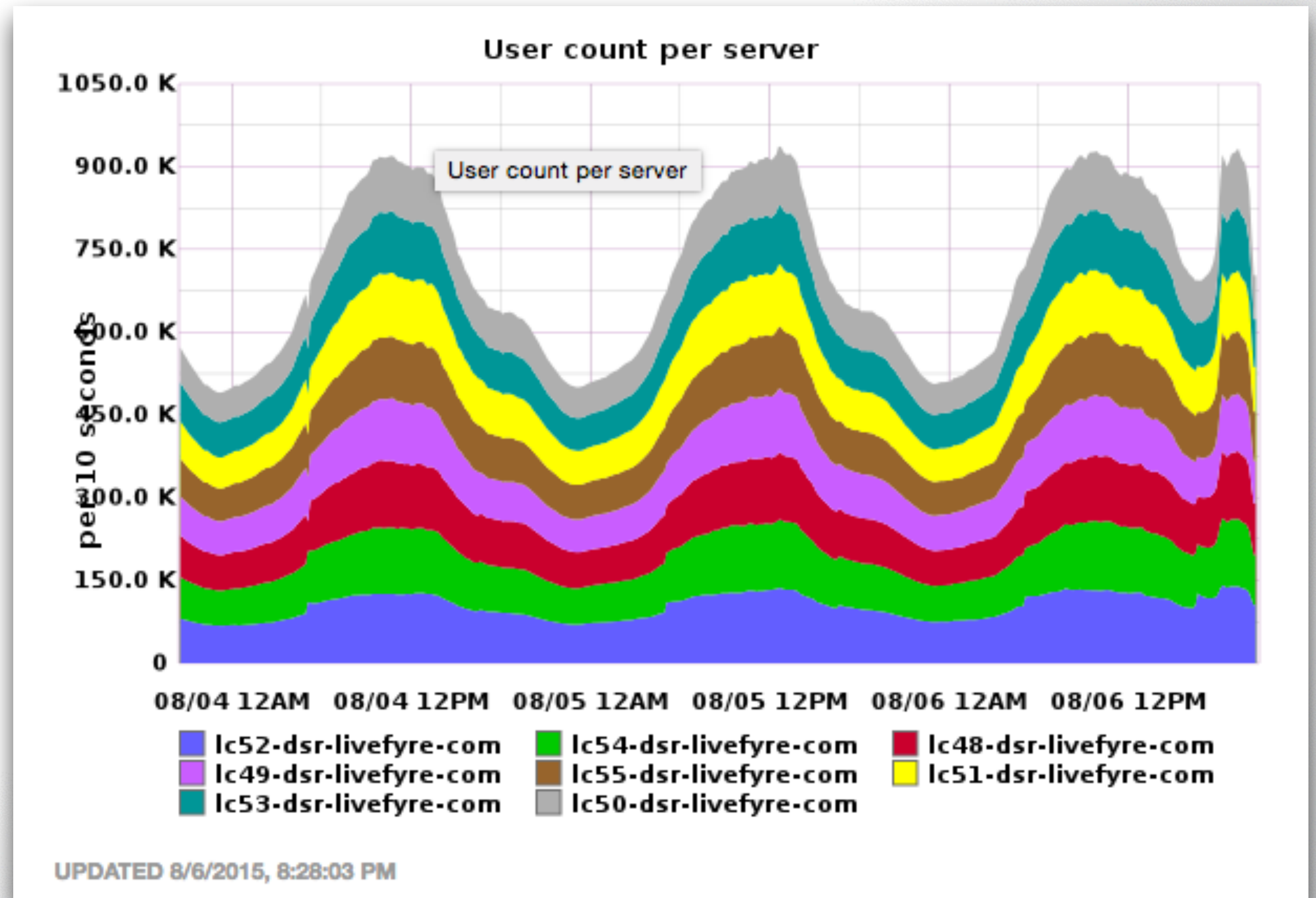
Storify



1 / CHALLENGE

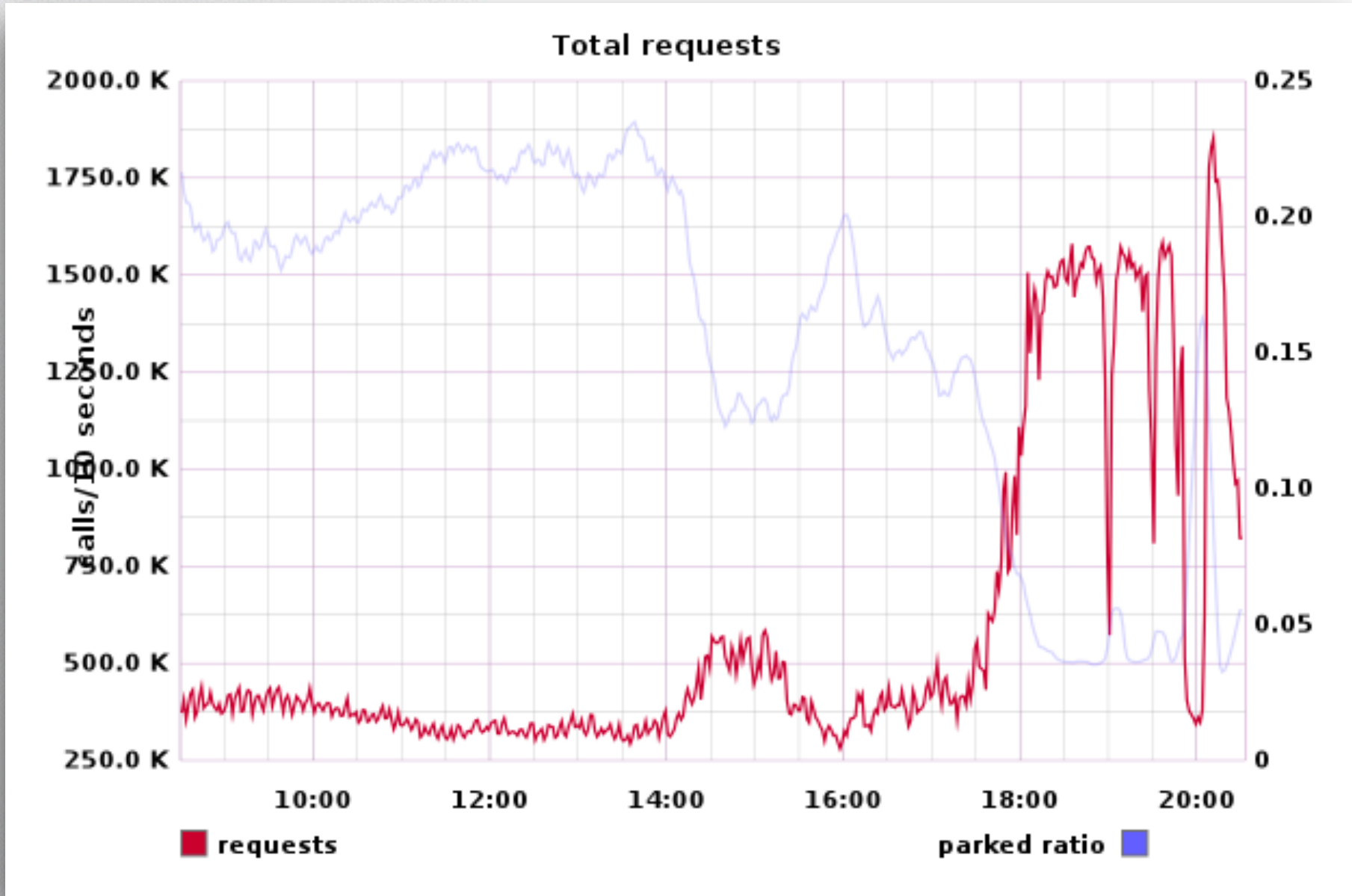
Real-time challenge

- 1,000,000 concurrent users
 - 150,000 per JVM
 - 100,000 req/s
 - 6-8x c3.2xlarge
 - long-poll + ws
- 100s - 1,000s of listeners per stream
 - up to 250,000 listeners
 - read-heavy
 - updates < 2s



Real-time challenge

- Presidential Debate on Fox News
 - from 50,000 req/s
 - to 200,000 req/s
 - 150,000+ listeners to the stream



Full Coverage

AUGUST 6, 2015 AT 5:00 PM & 9:00 PM ET, FROM CLEVELAND, OHIO

Fox News Republican Presidential Primary Debates

The first Republican presidential primary debates, hosted by Fox News and Facebook in conjunction with the Ohio Republican Party, will be held at the Quicken Loans Arena in Cleveland, Ohio, starting at 5:00 PM ET. This is the same venue as the 2016 Republican National Convention scheduled for next summer. See all the latest debate and election coverage on FoxNews.com.

August 6th: Watch Online and On Air

5pm ET
First Debate

6pm ET
Online Pre-Show

9pm ET
Primetime Debate

11pm ET
Online Post-Debate Show

Download The App

FOX NEWS

Latest News and Features

GOP underdogs go after Clinton, Obama; downplay long odds at first debate

Seven Republican primary underdogs downplayed their long political odds Thursday in Cleveland...

Christie and Huckabee debate entitlement reform

Republican presidential candidates clash in Cleveland

Is ISIS an ideological or military problem?

Sen. Ted Cruz responds #GOPDebate

Chris Christie, Rand Paul spar over NSA

New Jersey governor on protecting the homeland #GOPDebate

Choose your provider

COX DIRECTV optimum. DISH NETWORK

xfinity Charter AT&T U-verse Verizon FIOS


Other 3 Rivers Commu Login

Live Chat - What would you like us to ask the candidates?

Sign in

150380 people listening

johnncampbell just now
After this showing I'm now committing to Trump. Allowing the establishment machine to run this show was over the top and Hume and Kelley doing it was far to obvious.

 © LIVEFYRE 2015

2/**BESTTER,ST**} PRACTICES

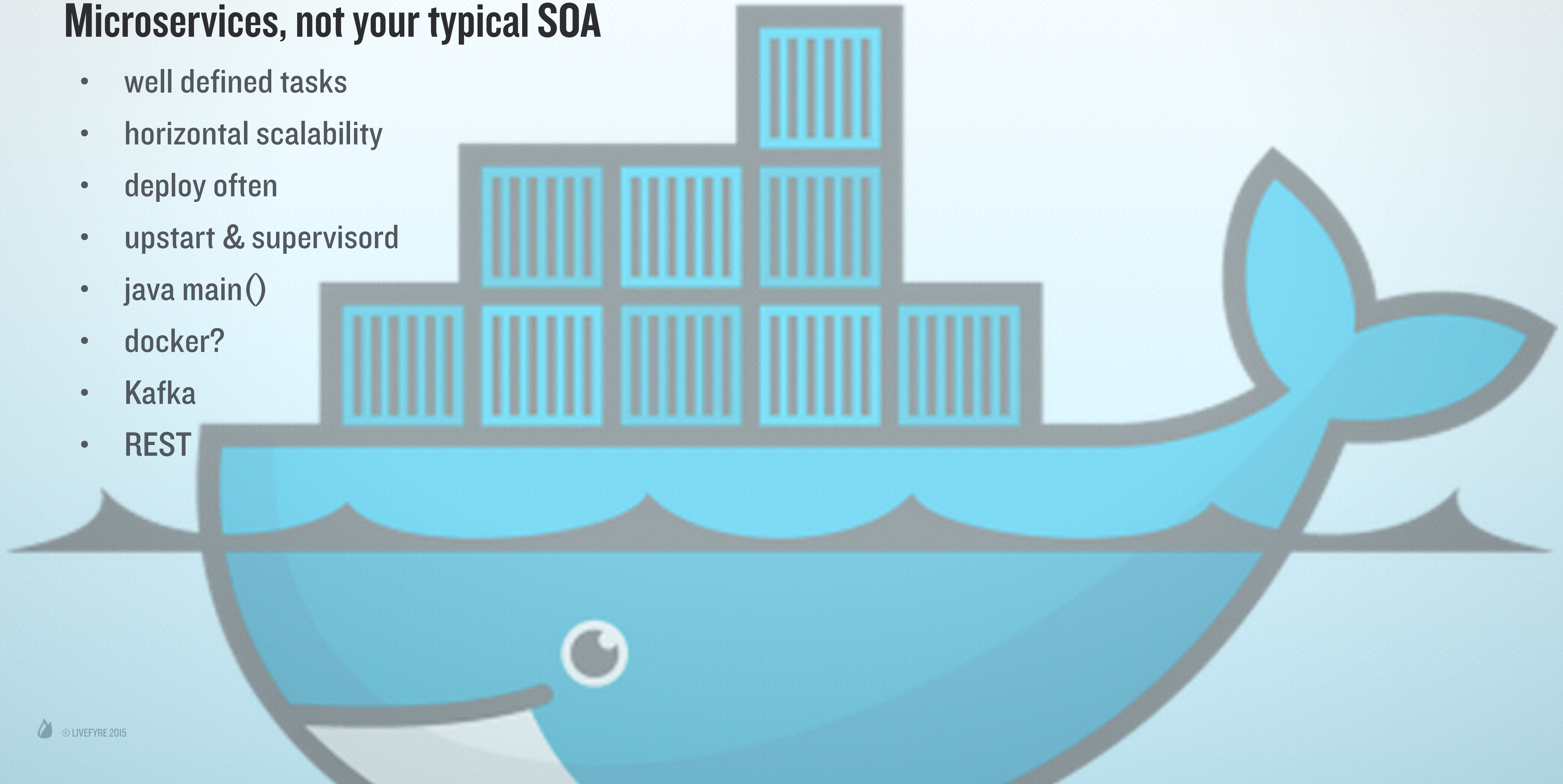
Don't use the “tech stack du jour”

- use the **right** tools for **your** problem
- embrace polyglot
 - Java, Scala, Jython
 - Python
 - NodeJS
- **KISS + YAGNI**

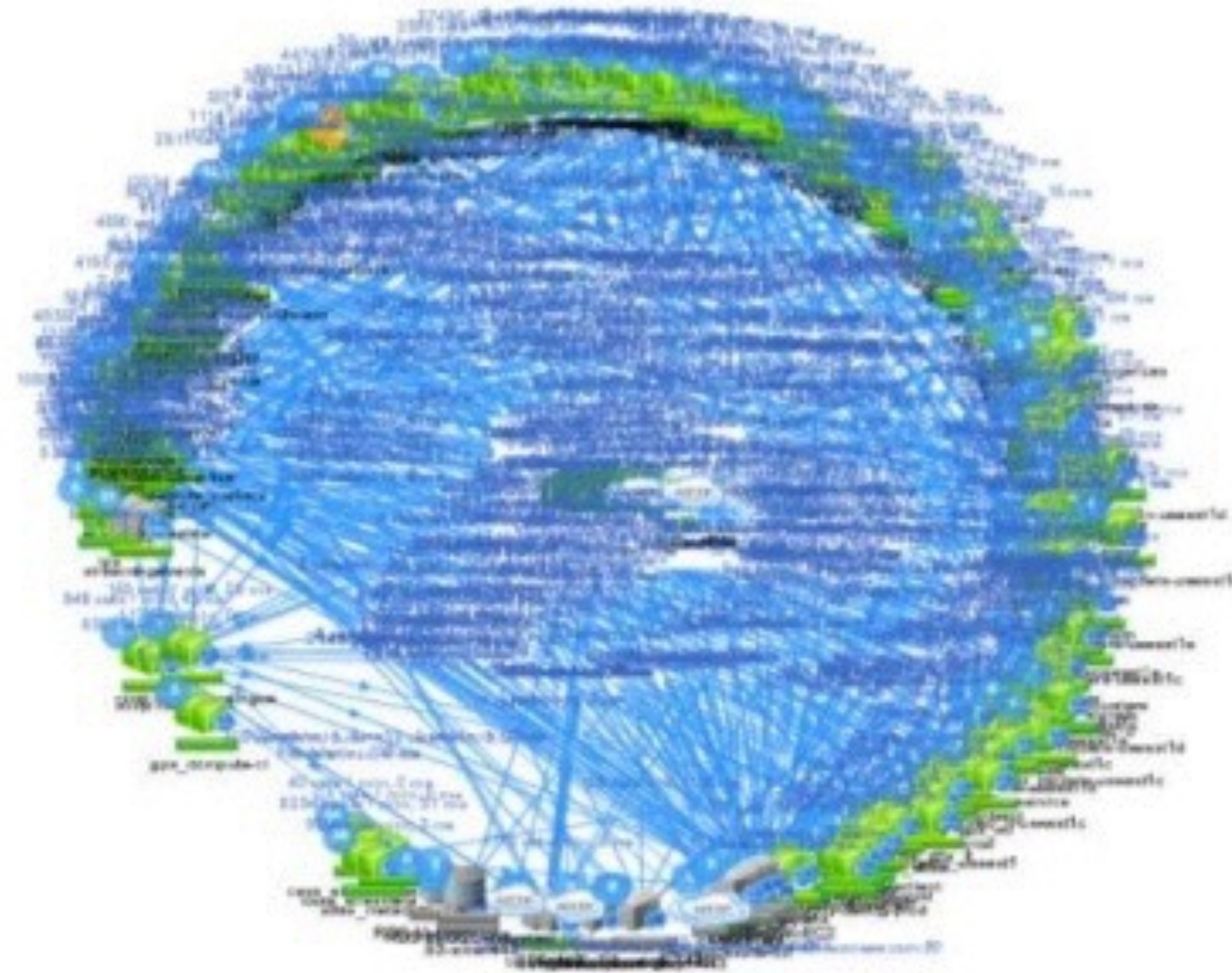


Microservices, not your typical SOA

- well defined tasks
- horizontal scalability
- deploy often
- upstart & supervisord
- java main()
- docker?
- Kafka
- REST



Netflix & Micro-Services



@bruce_m_wong

12

Monitor all the things!

are we sad

- error vs success rates and timing
- queue depth or lag
- system resources
- sample high velocity
- **/ping** and **/deep-ping**

access patterns

- optimize scaling strategy
- anticipate events



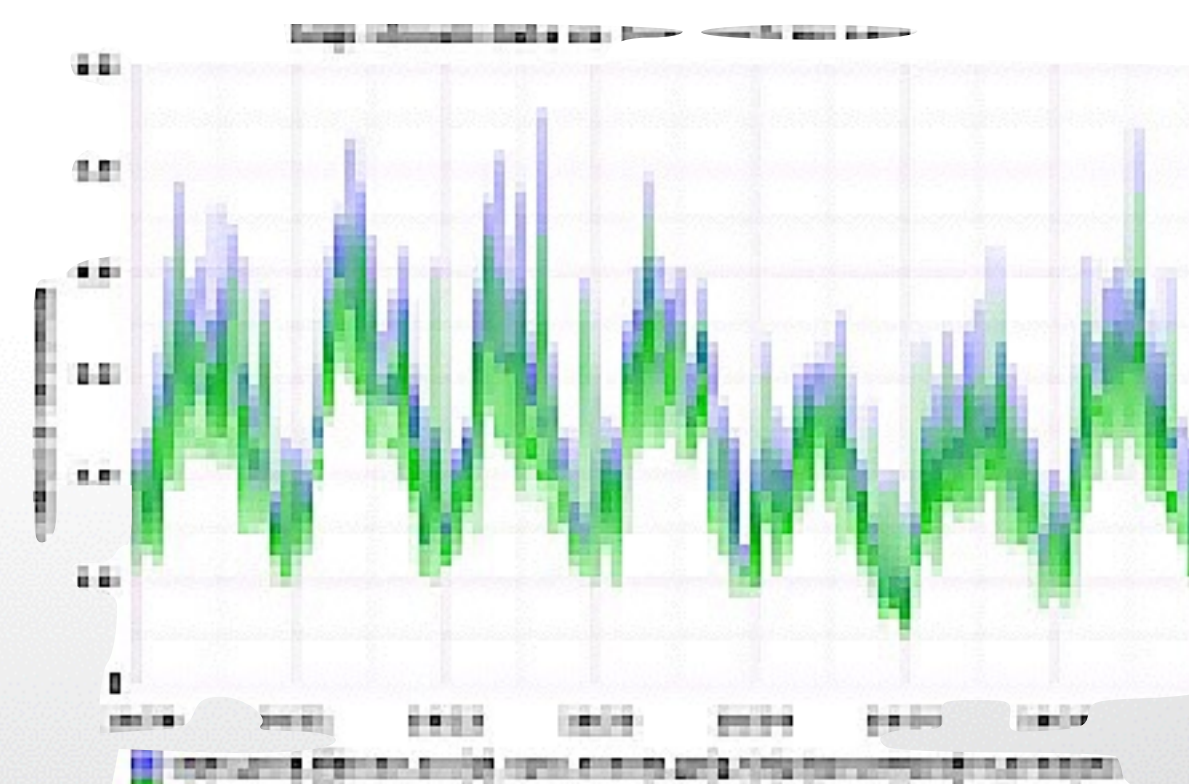
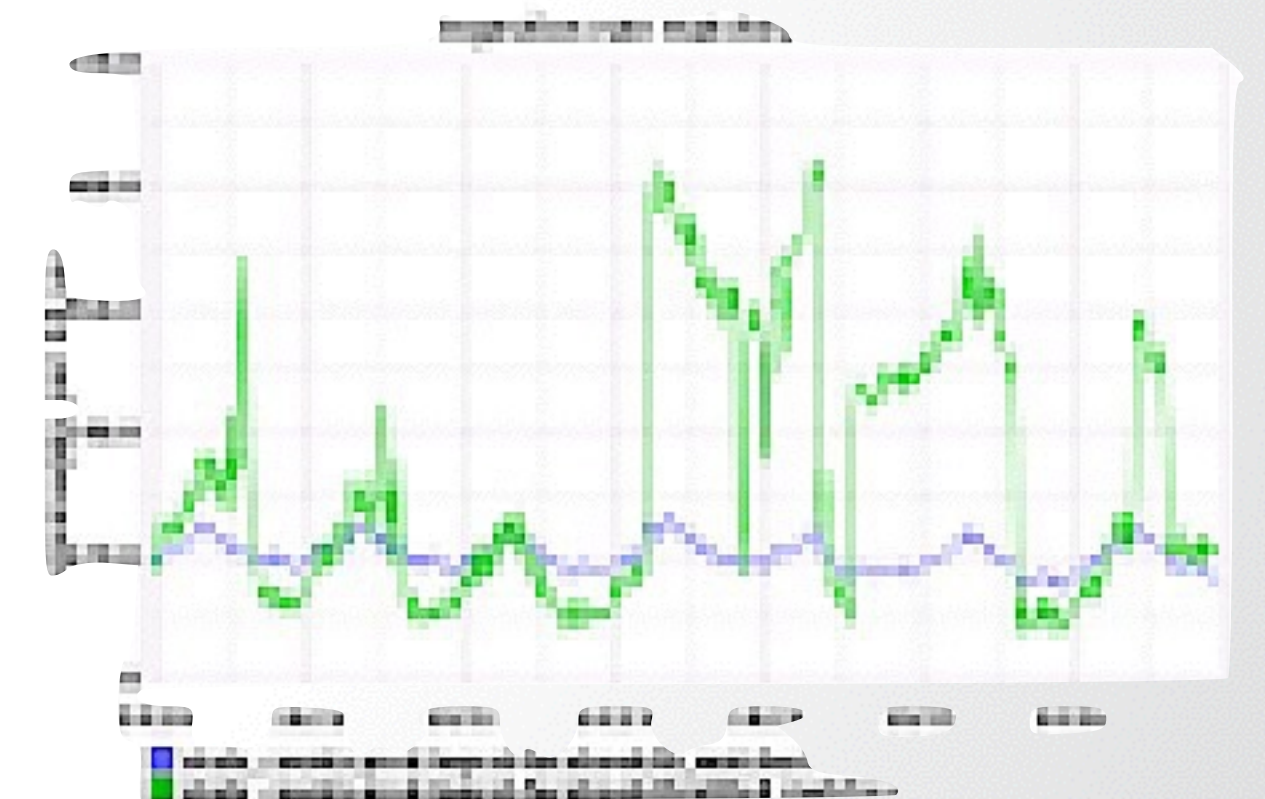
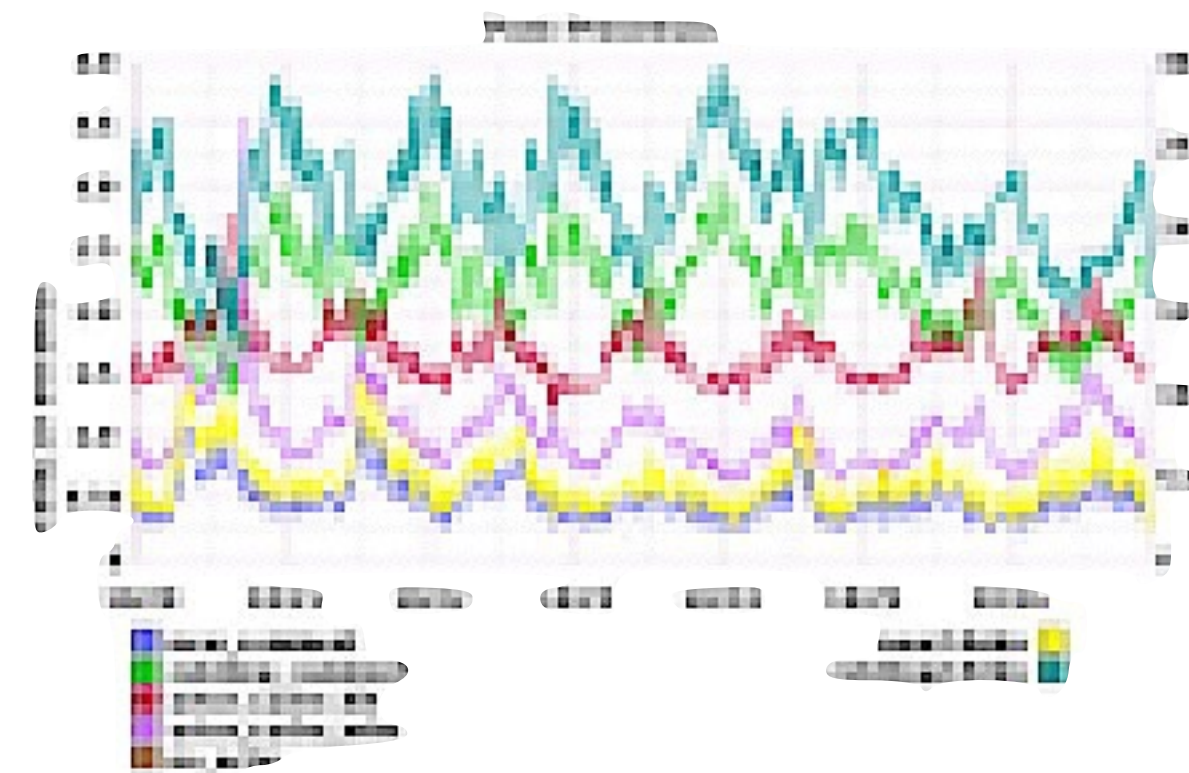
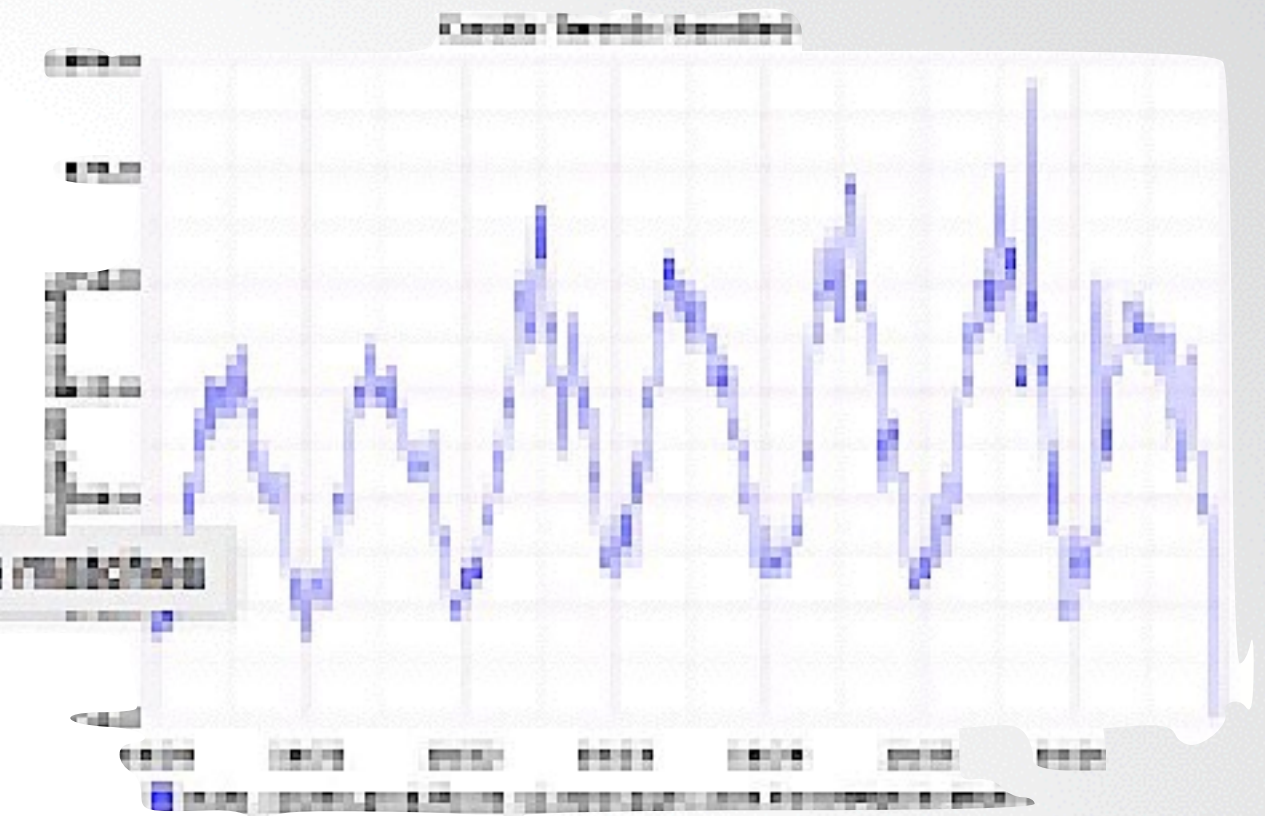
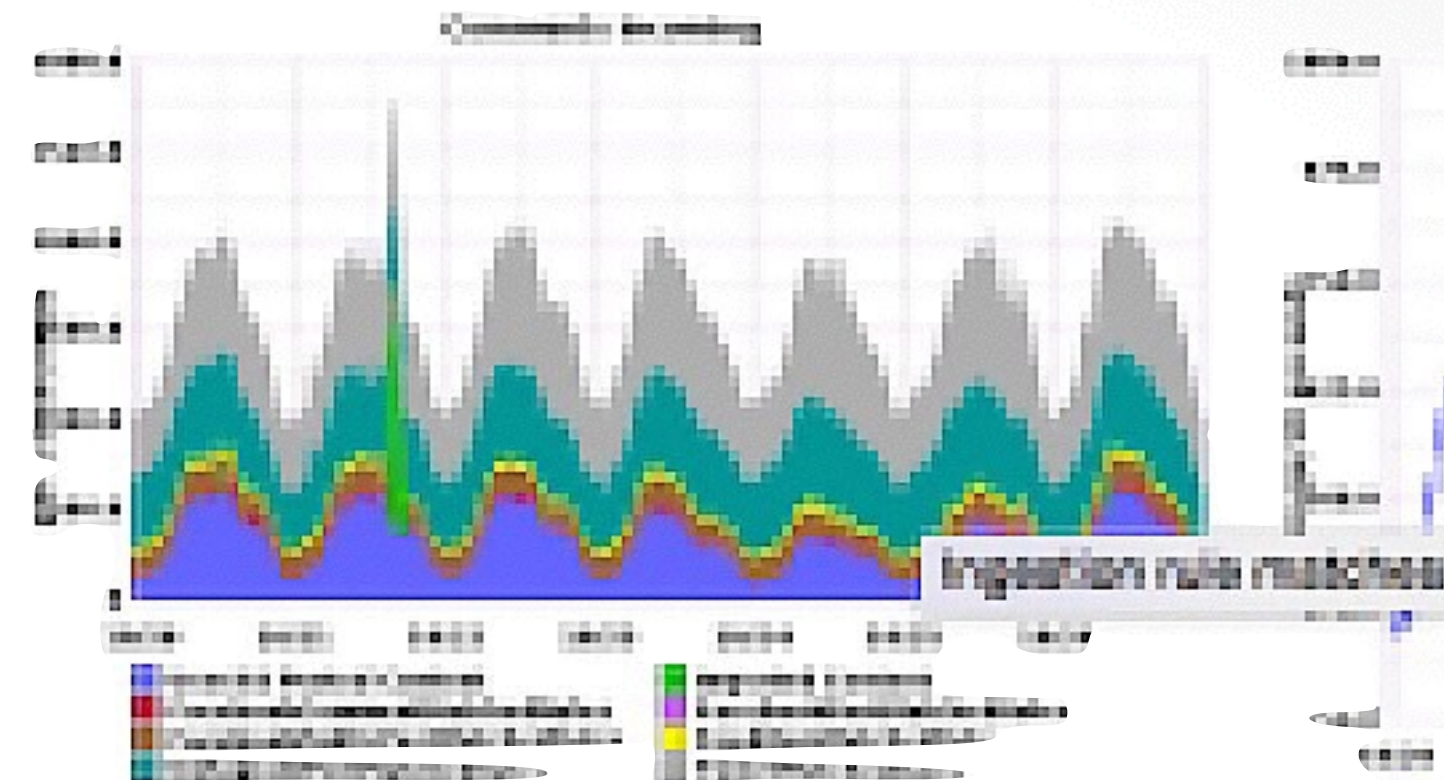
Mo services mo problems

Dashboards

- service vs system health
- correlate “strange events”
- capacity planning
- app specific

Tools

- statsd + graphite + grafana / gdash
- sentry log4j appender
- nagios + pagerduty



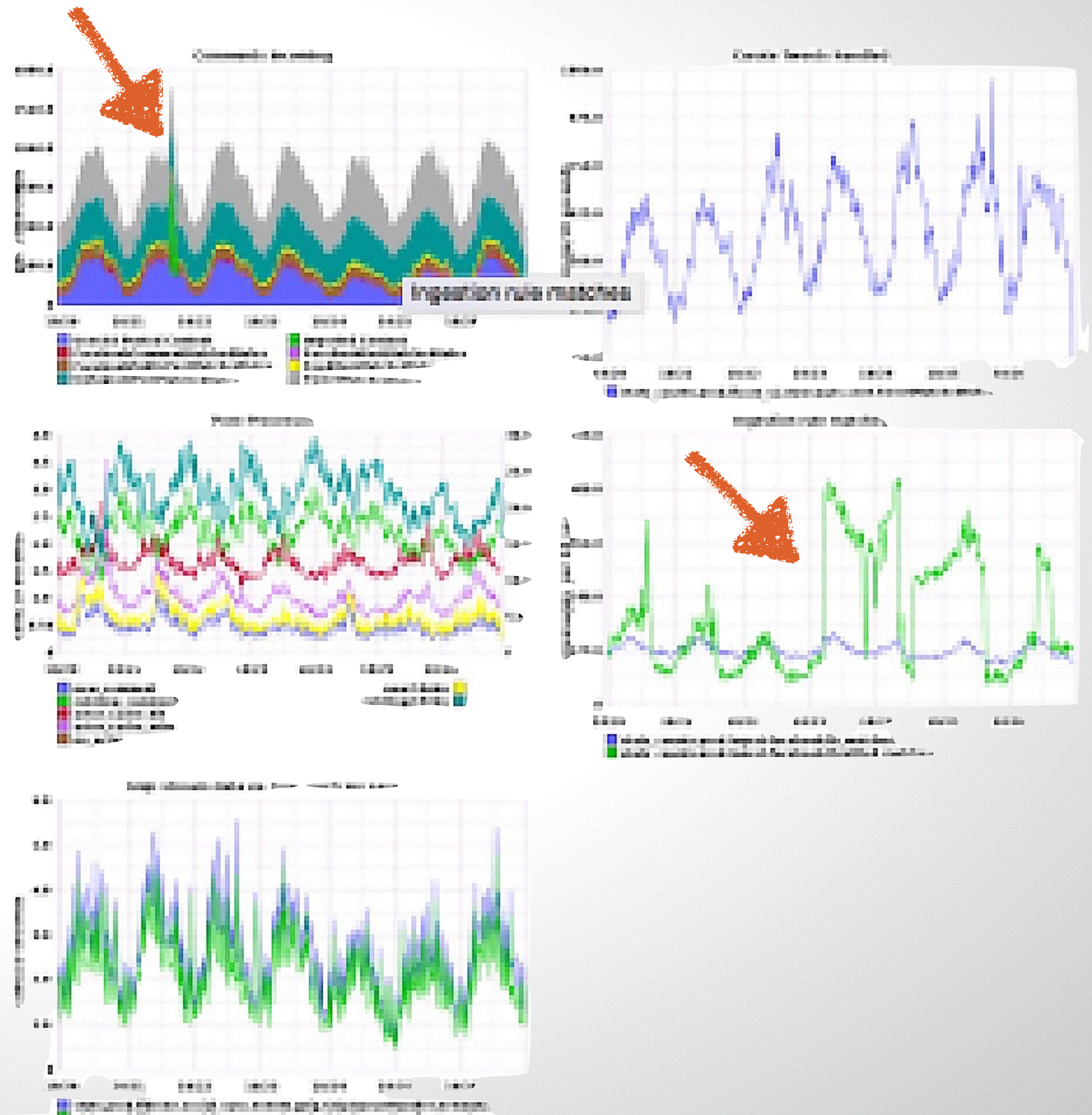
Mo services mo problems

Dashboards

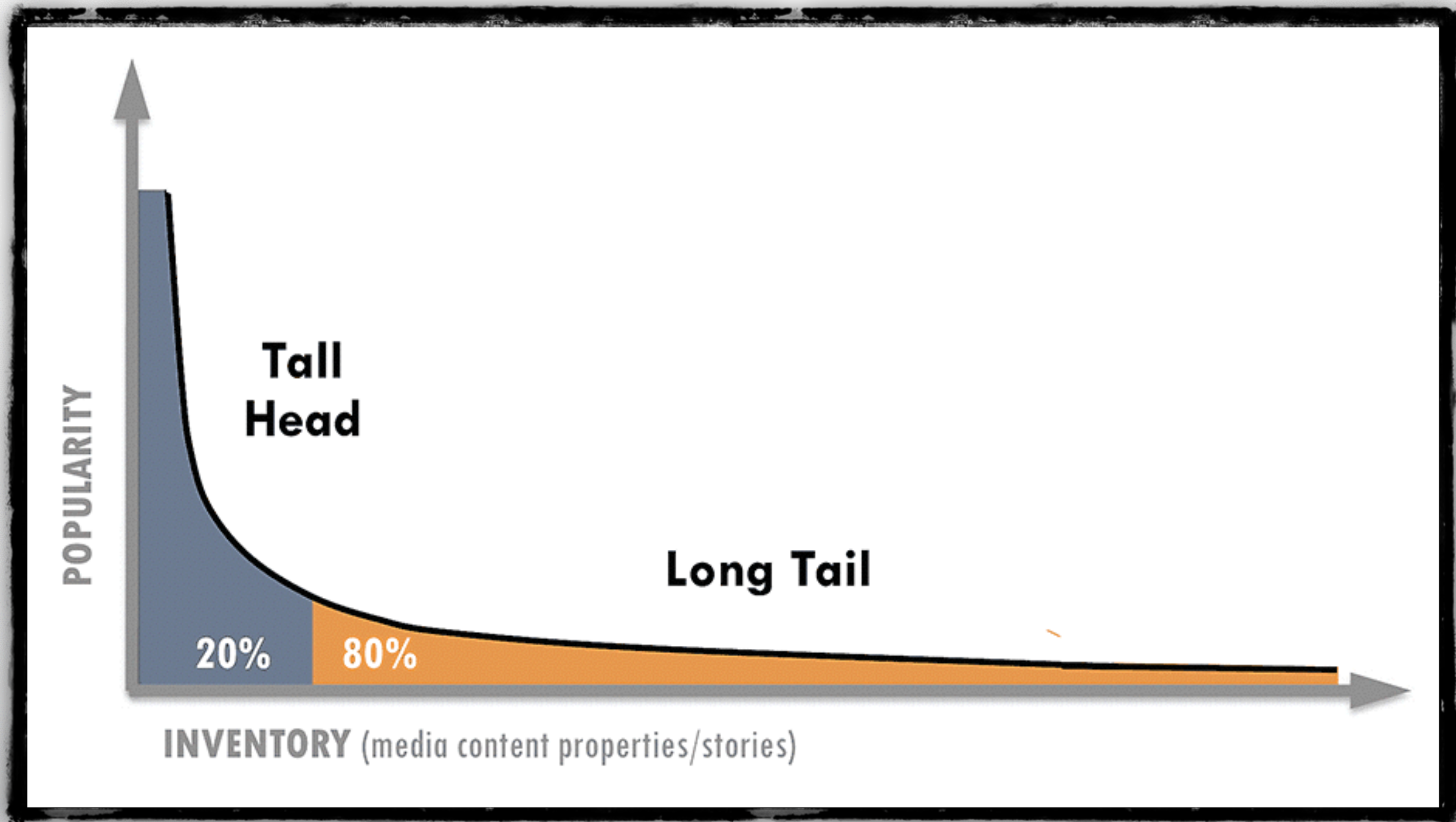
- service vs system health
- correlate “strange events”
- capacity planning
- app specific

Tools

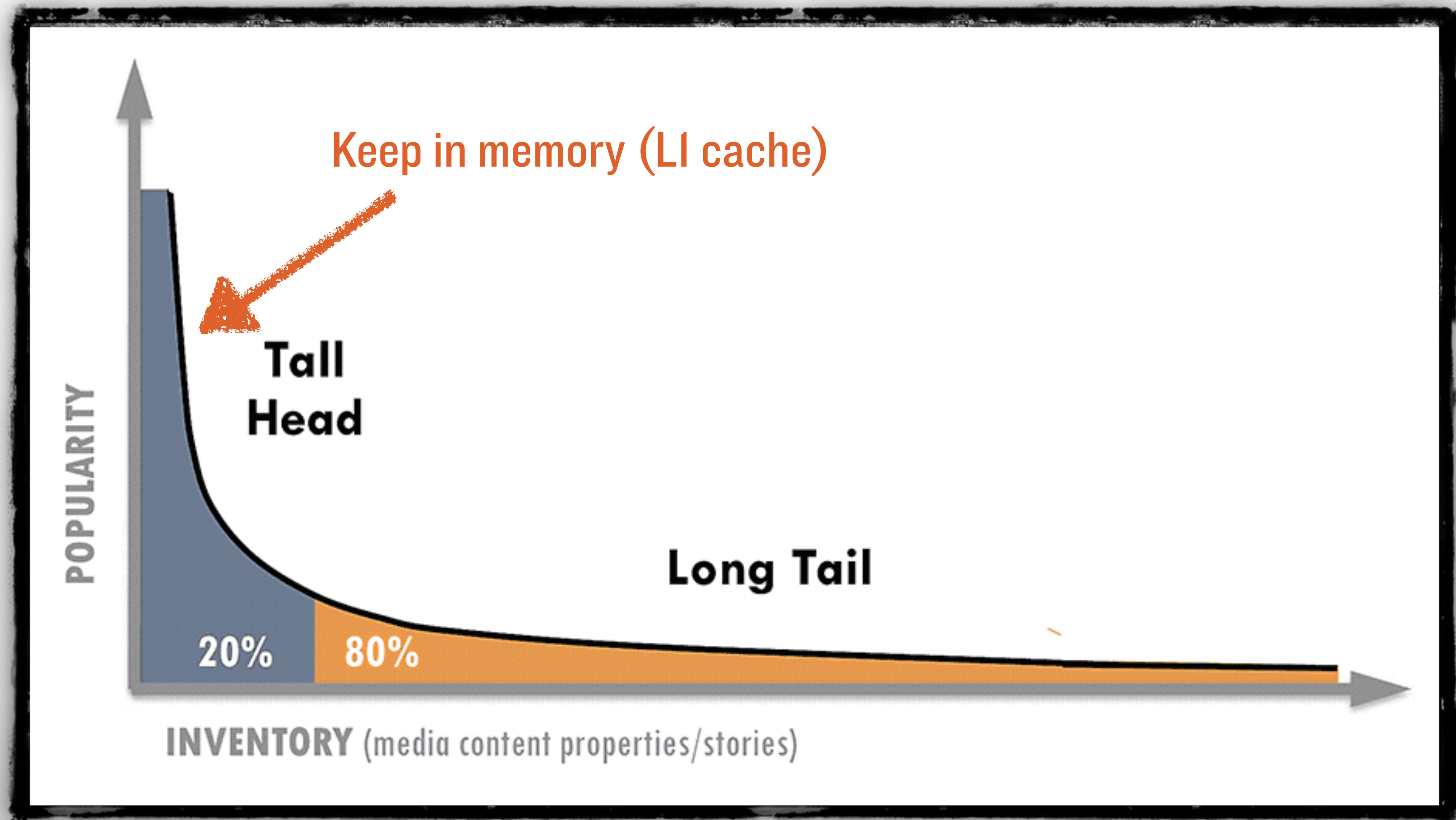
- statsd + graphite + grafana / gdash
- sentry log4j appender
- nagios + pagerduty



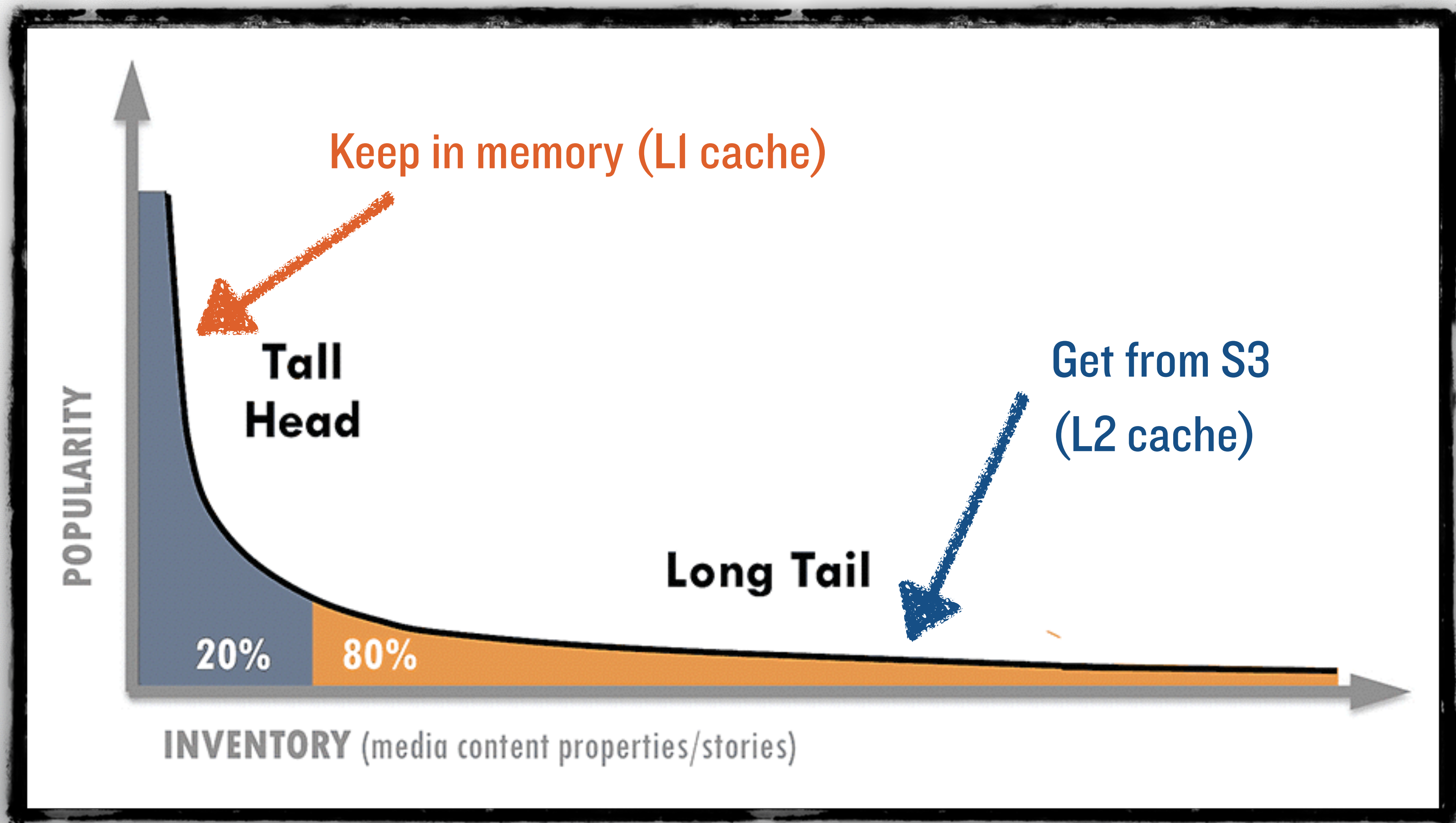
Request distribution or “data access pattern”



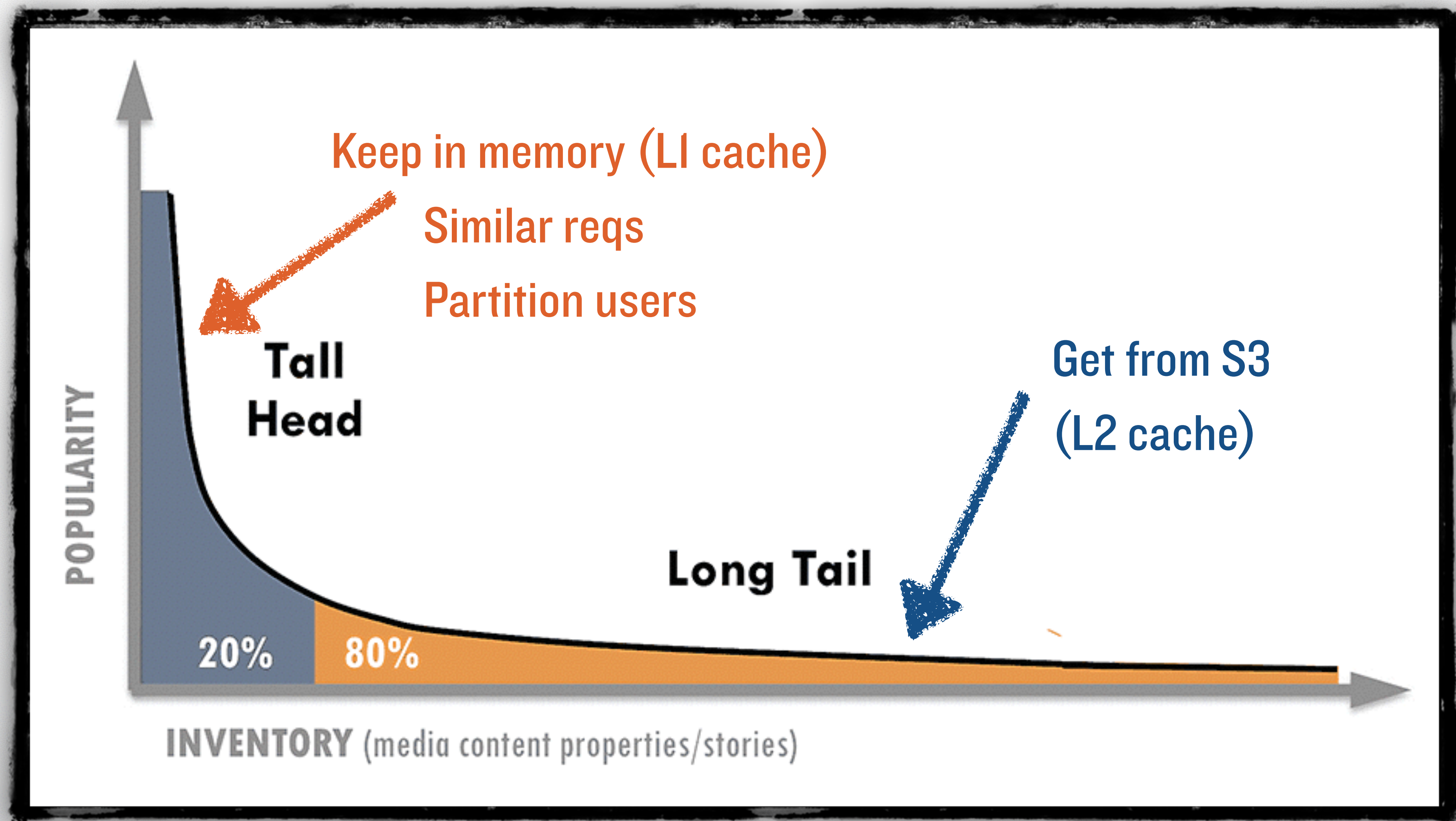
Request distribution or “data access pattern”



Request distribution or “data access pattern”



Request distribution or “data access pattern”



Forcing square pegs in a round hole

- choose the right data stores
 - Database
 - Queue
- sweet spot
 - type of data
 - type of queries
 - some optimized for write
 - some optimized for indexing
- trade off of speed and consistency



Call me maybe - a story of unreliable communication

CARLY RAE JEPSEN

Aphyr

BlogPhotographyCodeAbout

Call me Maybe: Percona XtraDB Cluster

Percona's CTO Vadim Tkachenko wrote a [response](#) to my [Galera](#)

Call me maybe: Aerospike

Previously, on [Jepsen](#), we reviewed [Elasticsearch's](#)

Call me maybe: Elasticsearch

This post  covers [Elasticsearch 1.1.0](#). In

Computational techniques in Knossos

Earlier versions of [Jepsen](#) found glaring inconsistencies, but

Call me Maybe: MariaDB Galera Cluster

Previously, on [Jepsen](#), we saw [Chronos fail to run jobs after a network](#)

Call me maybe: Elasticsearch 1.5.0

Previously, on [Jepsen](#), we demonstrated [stale and dirty reads in](#)

Call me maybe: etcd and Consul

In the previous post, we discovered the potential for [data loss in](#)

Strong consistency models

[Network partitions are going to happen](#). Switches, NICs, host

Call me Maybe: Chronos

[Chronos](#) is a distributed task scheduler (cf. cron) for the [Mesos](#) cluster management

Call me maybe: MongoDB stale reads

In [May of 2013](#), we showed that MongoDB 2.4.3 would lose

Call me maybe: RabbitMQ

RabbitMQ is a  distributed

Call me maybe: Redis redux

In a [recent blog post](#), antirez detailed a new operation in Redis:

Call me maybe: Strangeloop Hangout

Since the Strangeloop talks won't be available for a few months, I

Call me maybe: RICON East talk

 Call Me Maybe: Carly Rae Jepsen

Call me maybe: Cassandra

Previously on [Jepsen](#), we learned about [Kafka's proposed](#)

Call me maybe: Kafka

In the last [Jepsen](#) post, we learned about [NuoDB](#). Now it's time to switch gears and

Call me maybe: NuoDB

Previously on [Jepsen](#), we explored [Zookeeper](#). Next up: [Kafka](#).

Call me maybe: Zookeeper

In this [Jepsen](#) post, we'll explore [Zookeeper](#). Up next:

Automating Jepsen

If you, as a database vendor, implement a few features in your API, I can probably

The network is reliable

I've been discussing [Jepsen](#) and partition tolerance with [Peter Bailis](#) over the past few

Asynchronous replication with failover

In response to my [earlier post](#) on Redis inconsistency, Antirez

Call me maybe: final thoughts

Previously in [Jepsen](#), we discussed [Riak](#). Now we'll review and

Call me maybe: Riak

Previously in [Jepsen](#), we discussed [MongoDB](#). Today, we'll see how last-write-wins

Call me maybe: MongoDB

Previously in [Jepsen](#), we discussed [Redis](#). In this post, we'll see [MongoDB drop a](#)

Call me maybe: Redis

Previously on [Jepsen](#), we explored two-phase

Call me maybe: Postgres

Previously on [Jepsen](#), we introduced the

Call me maybe: Carly Rae Jepsen and the perils of network partitions

<https://aphyr.com/tags/Jepsen>

3/**BUILDING BLOCKS**



Throttling - Leaky bucket algorithm

- capped output flow
regardless of input flow
- accrue output allowance over time
- drop requests if insufficient allowance
- cost function

```
# 1 item per interval
allowance = rate = 1
# 10 sec interval
throttle_interval = 10
# 1req/10sec = 0.1 qps
qps = rate / throttle_interval
last_check = time()

def throttle(item):
    current = time() # or item.created_at
    size = cost(item) # [0..1]
    time_passed = current - last_check
    last_check = current
    allowance += time_passed * qps
    # Cap to rate
    allowance = min(rate, allowance)

    if allowance < size:
        return True
    allowance -= size
    return False
```


Counting 'Heavy Hitters' - Space Saving Algorithm

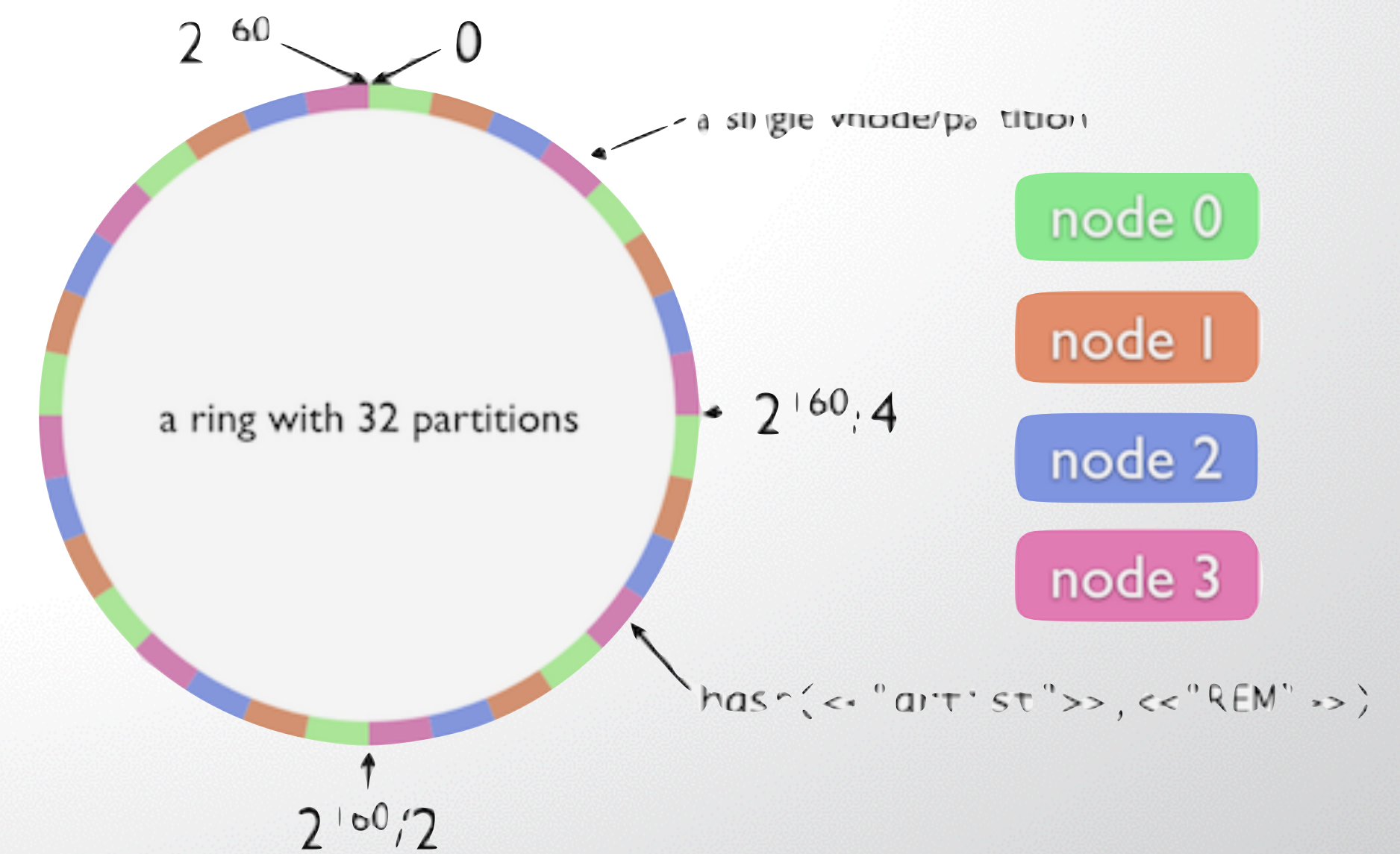
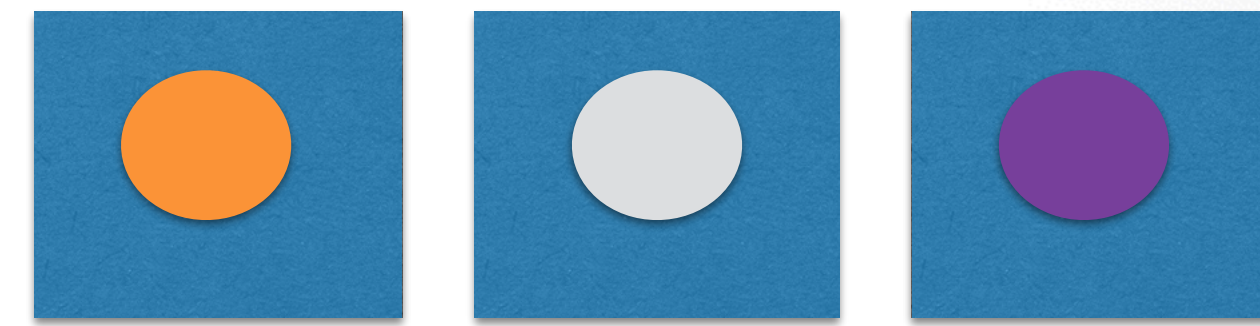
```
counts = { } # map of item to count
errors = { } # map of item to error count

for item in stream:
    if len(counts) < k:
        counts[item] += weight
    else:
        if item in counts:
            counts[item] += 1
        else:
            prev_min = item_with_min_count(counts)
            counts[item] = counts[prev_min] + 1
            errors[item] = counts[prev_min]
            counts.remove_key(prev_min)
```

- unbounded stream
- TOP-K in constant space
 - $k * (\text{item}, \text{count}, \text{error})$
- overestimates on replace
 - $\min(\text{count})$
- MIN Heap + HashMap

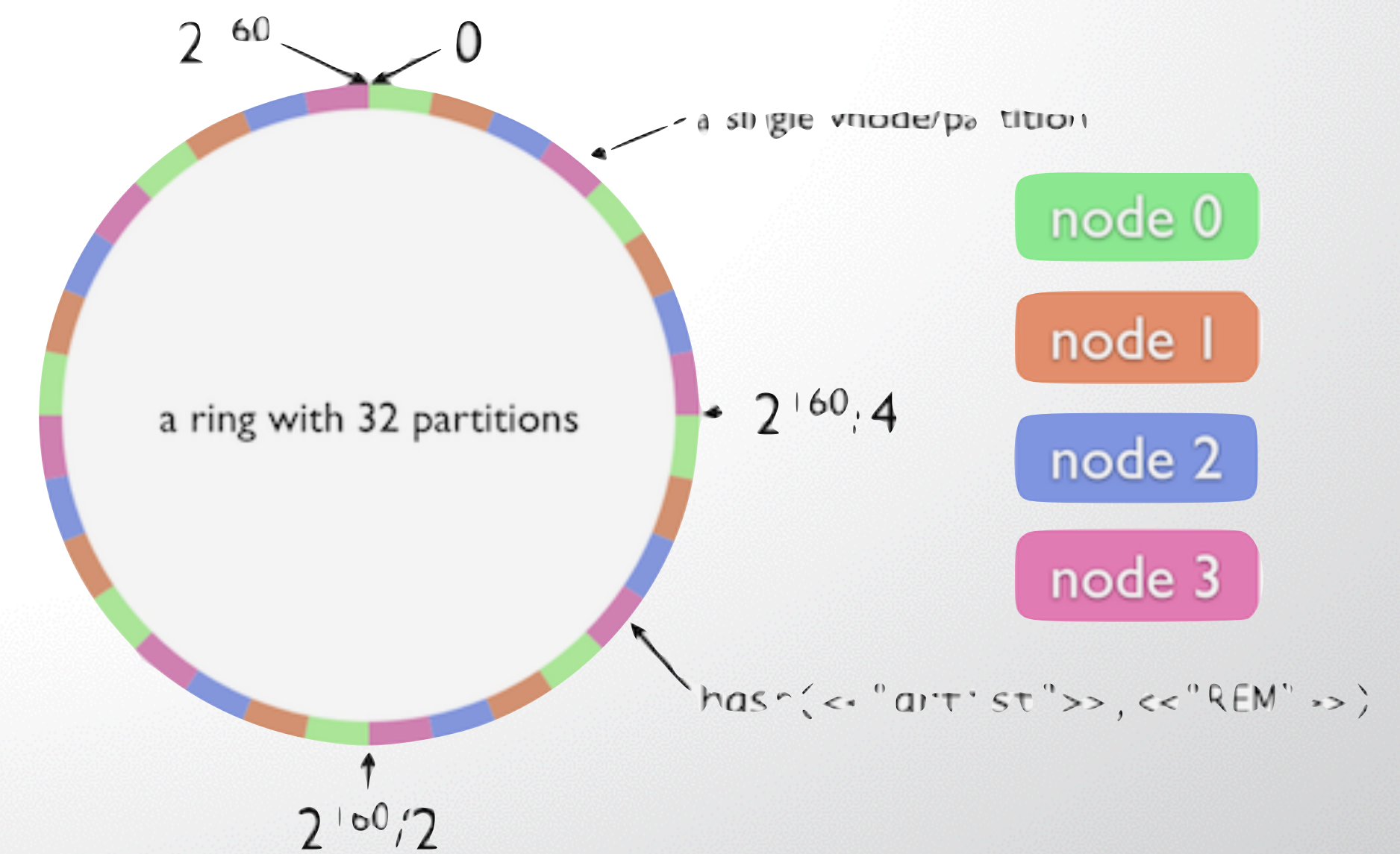
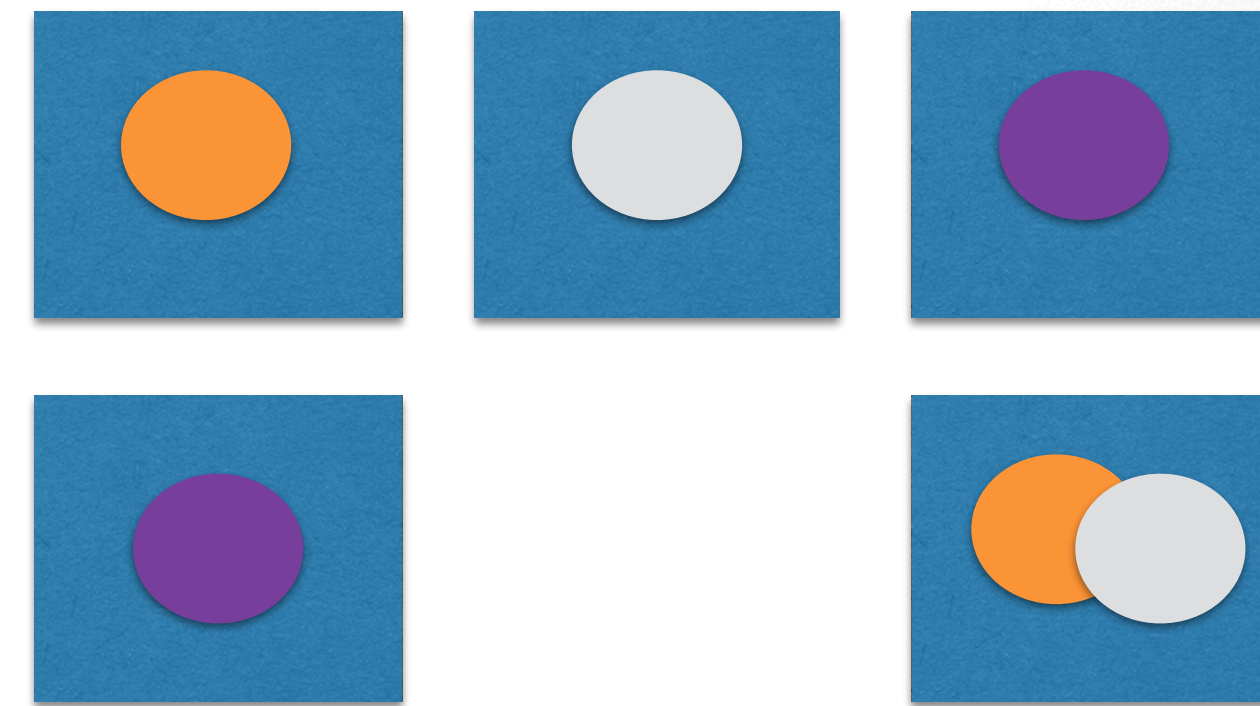
Partitioning - Consistent Hashing

- **`article_id % server_count`**
 - what if hosts added/removed ?
 - thundering herd!
- **`Hashing.consistentHash(item, server_count)`**
 - minimizes shuffling
- **ConsistentHashRing with virtual nodes**
 - **TreeSet with 100 replicas per node**
 - `hash("node1:1") .. hash("node1:100")`
 - `hash("node2:1") .. ("node2:100") ,...`
 - **`SortedMap.get(hash(item))` or**
 - **`SortedMap.tailMap(hash(item)).firstKey()`**



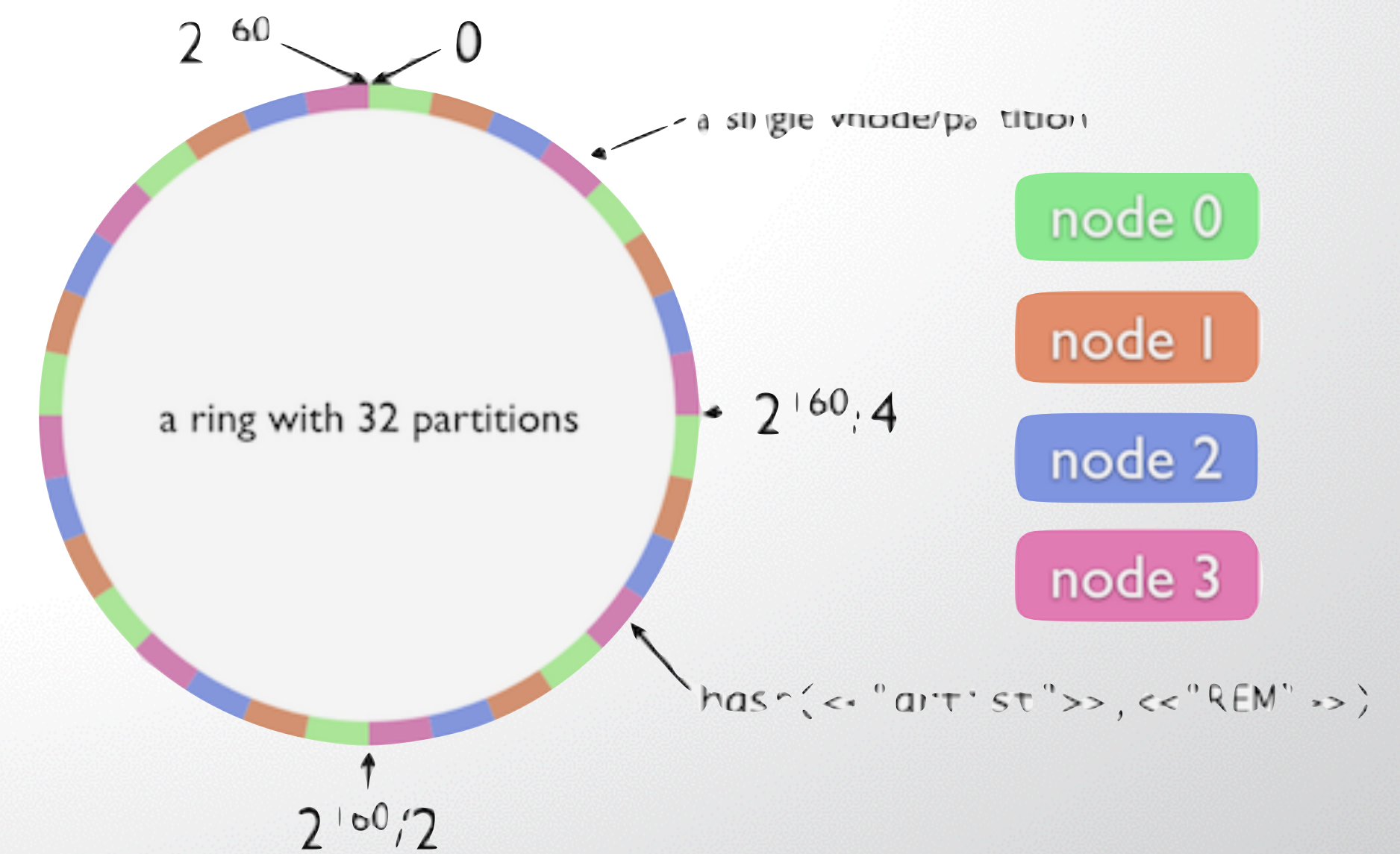
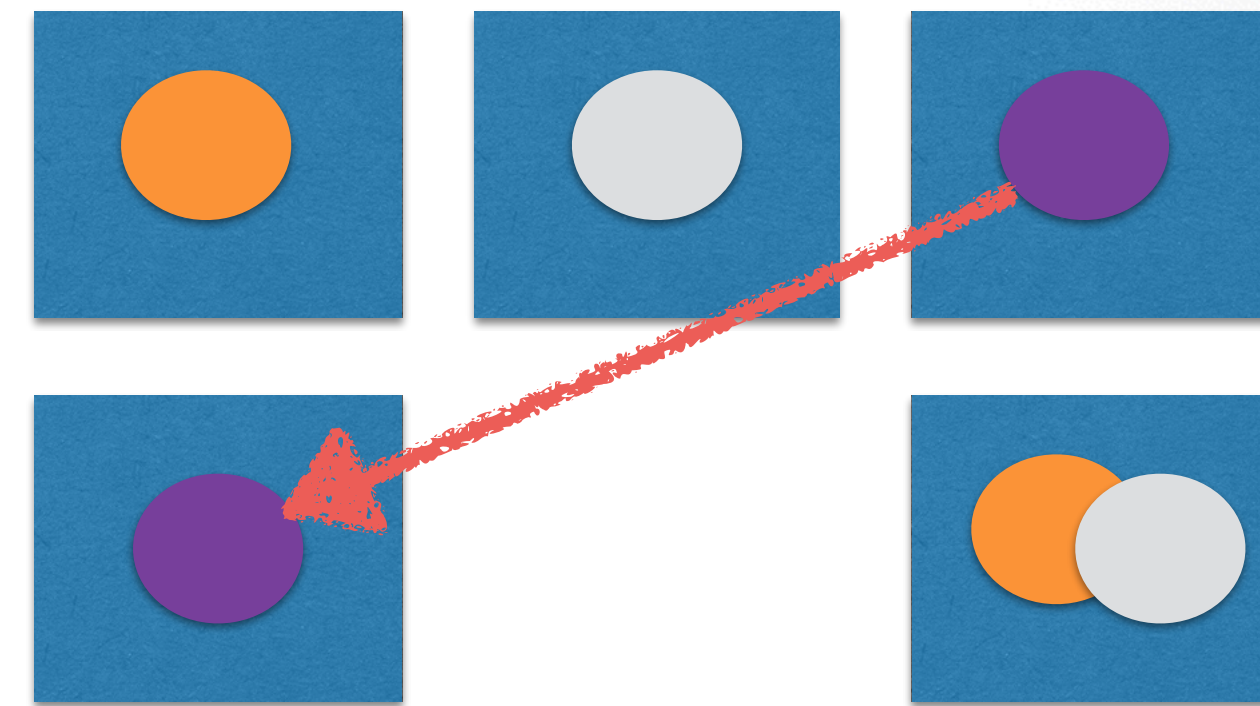
Partitioning - Consistent Hashing

- **`article_id % server_count`**
 - what if hosts added/removed?
 - thundering herd!
- **`Hashing.consistentHash(item, server_count)`**
 - minimizes shuffling
- **ConsistentHashRing with virtual nodes**
 - **TreeSet** with 100 replicas per node
 - `hash("node1:1") .. hash("node1:100")`
 - `hash("node2:1") .. ("node2:100")`, ...
 - **`SortedMap.get(hash(item))`** or
 - **`SortedMap.tailMap(hash(item)).firstKey()`**



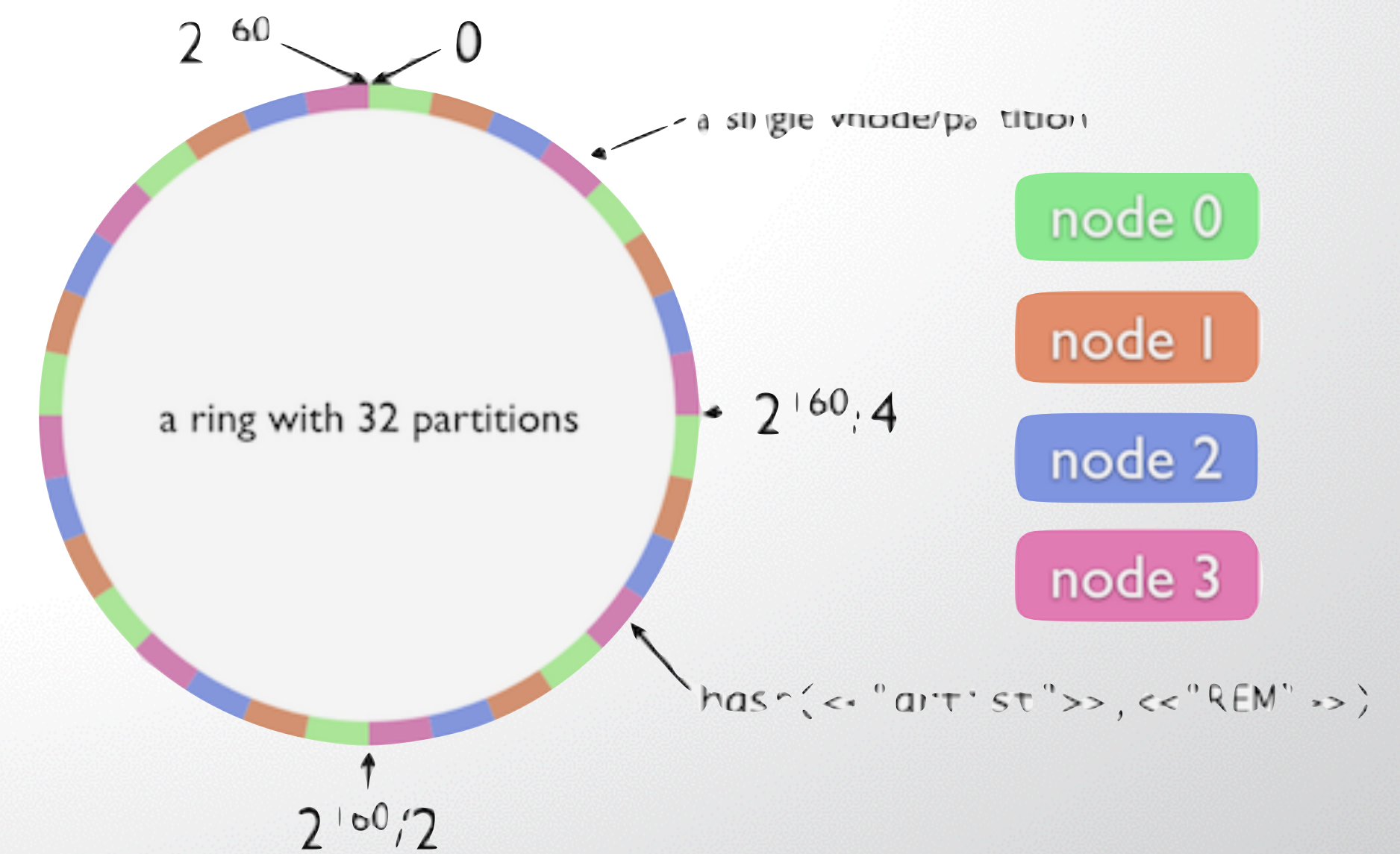
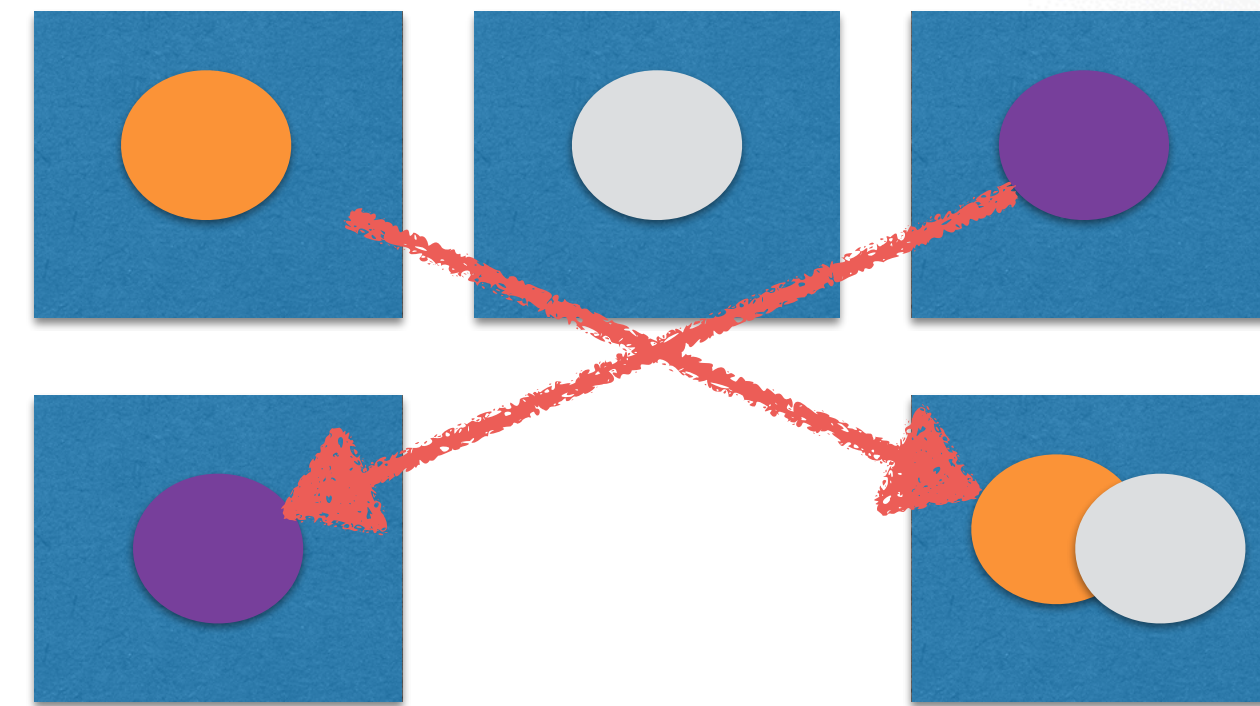
Partitioning - Consistent Hashing

- **`article_id % server_count`**
 - what if hosts added/removed?
 - thundering herd!
- **`Hashing.consistentHash(item, server_count)`**
 - minimizes shuffling
- **ConsistentHashRing with virtual nodes**
 - **TreeSet** with 100 replicas per node
 - `hash("node1:1") .. hash("node1:100")`
 - `hash("node2:1") .. ("node2:100")`, ...
 - **`SortedMap.get(hash(item))`** or
 - **`SortedMap.tailMap(hash(item)).firstKey()`**



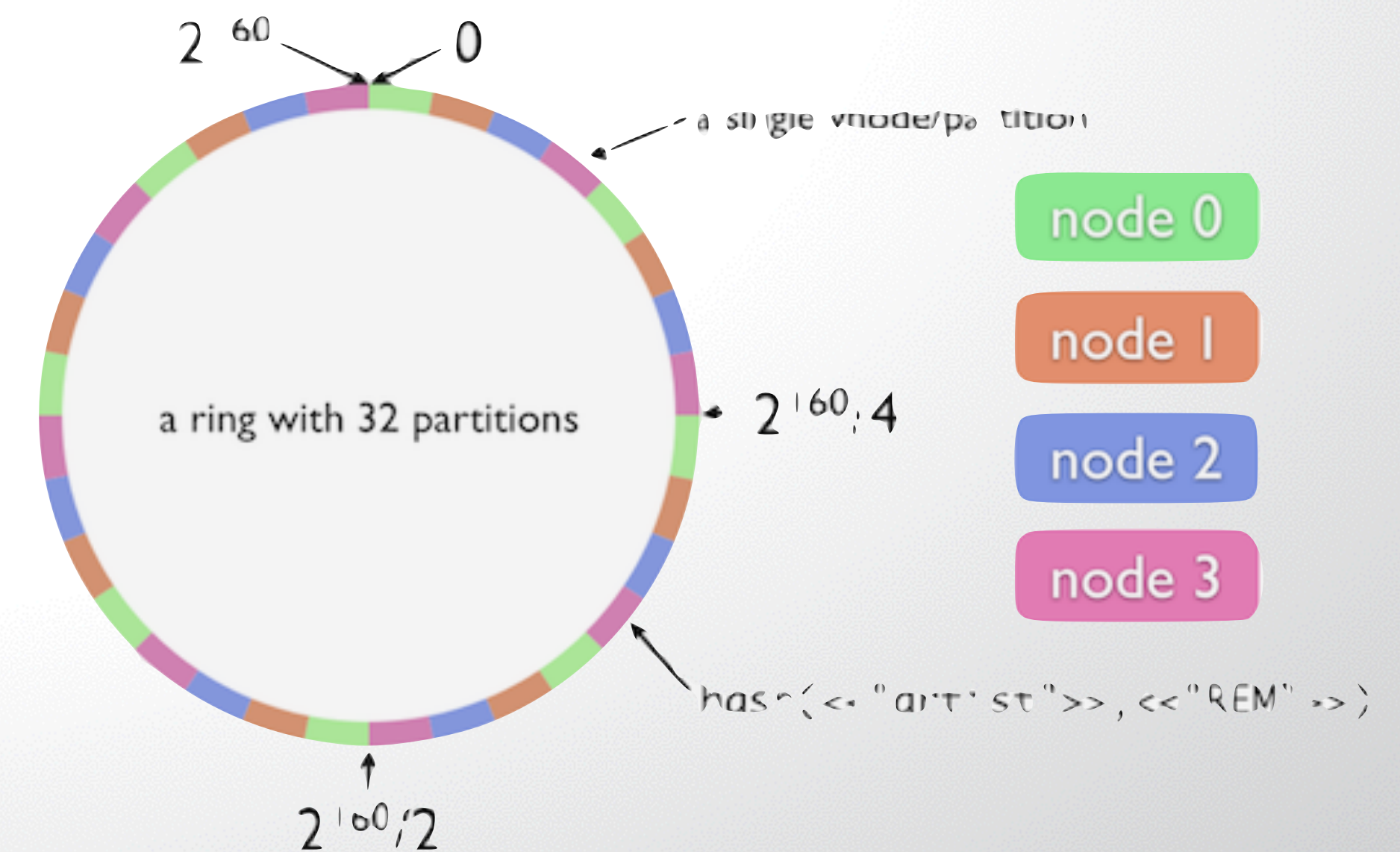
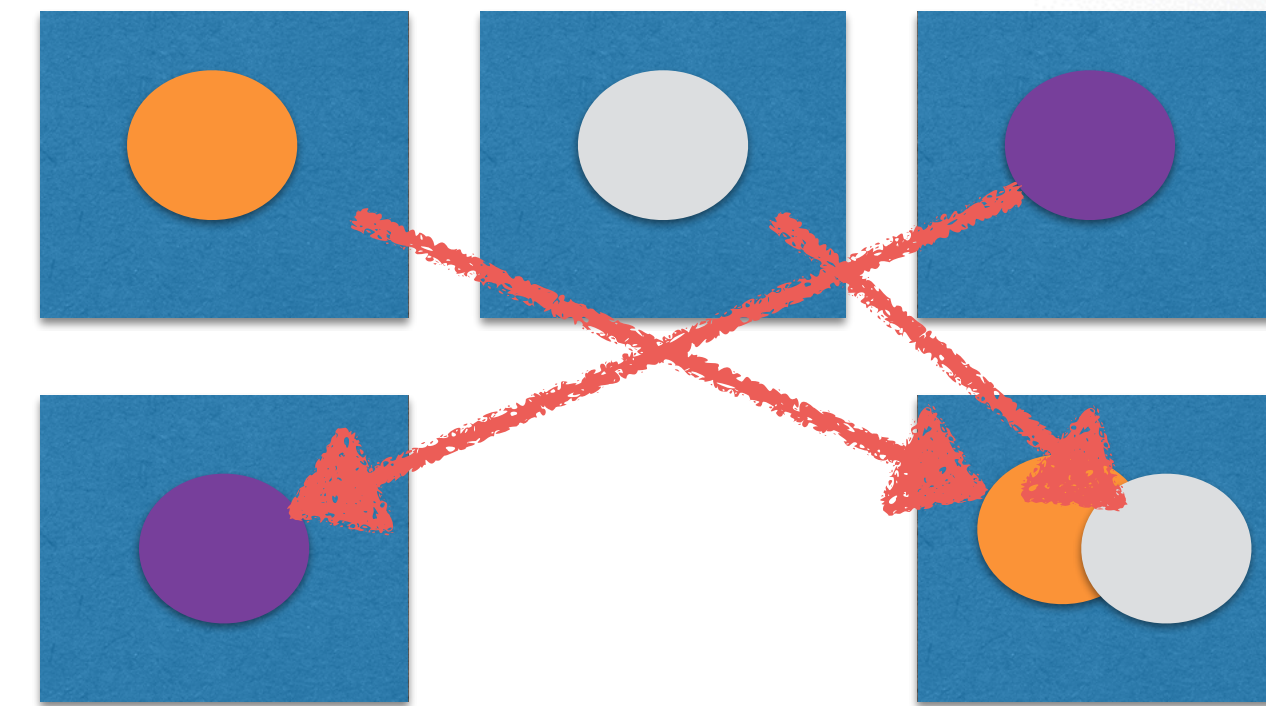
Partitioning - Consistent Hashing

- **`article_id % server_count`**
 - what if hosts added/removed ?
 - thundering herd!
- **`Hashing.consistentHash(item, server_count)`**
 - minimizes shuffling
- **ConsistentHashRing with virtual nodes**
 - **TreeSet** with 100 replicas per node
 - `hash("node1:1") .. hash("node1:100")`
 - `hash("node2:1") .. ("node2:100") ,...`
 - **`SortedMap.get(hash(item))`** or
 - **`SortedMap.tailMap(hash(item)).firstKey()`**



Partitioning - Consistent Hashing

- **`article_id % server_count`**
 - what if hosts added/removed?
 - thundering herd!
- **`Hashing.consistentHash(item, server_count)`**
 - minimizes shuffling
- **ConsistentHashRing with virtual nodes**
 - **TreeSet** with 100 replicas per node
 - `hash("node1:1") .. hash("node1:100")`
 - `hash("node2:1") .. ("node2:100")`, ...
 - **`SortedMap.get(hash(item))`** or
 - **`SortedMap.tailMap(hash(item)).firstKey()`**



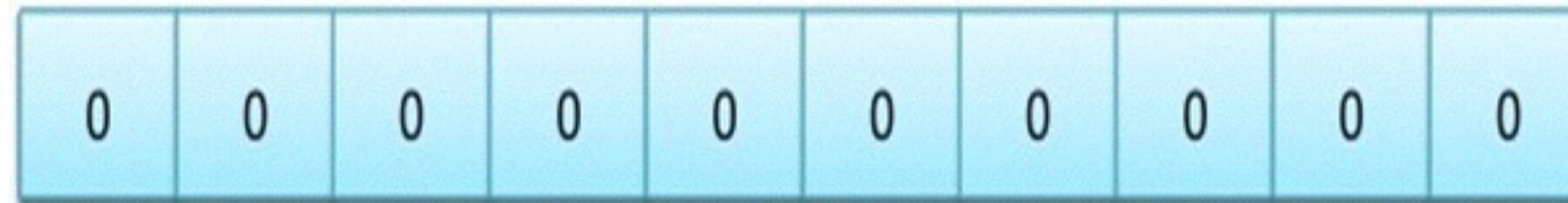
Membership test - Bloom Filters

- very memory efficient
- almost as fast as CHM
- small % false pos
- ZERO false neg
- append only
 - see Cuckoo Filter
- `BloomFilter.create()`



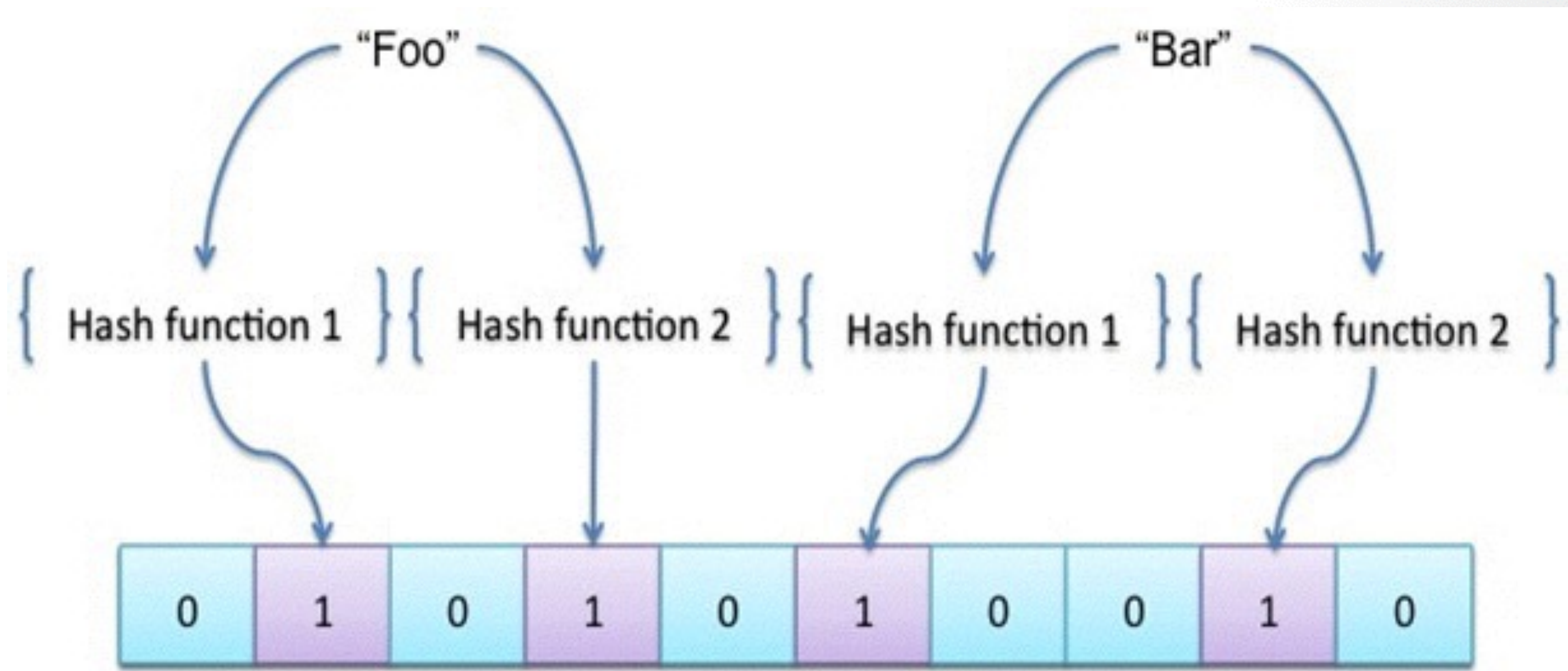
Membership test - Bloom Filters

- very memory efficient
- almost as fast as CHM
- small % false pos
- ZERO false neg
- append only
 - see Cuckoo Filter
- `BloomFilter.create()`



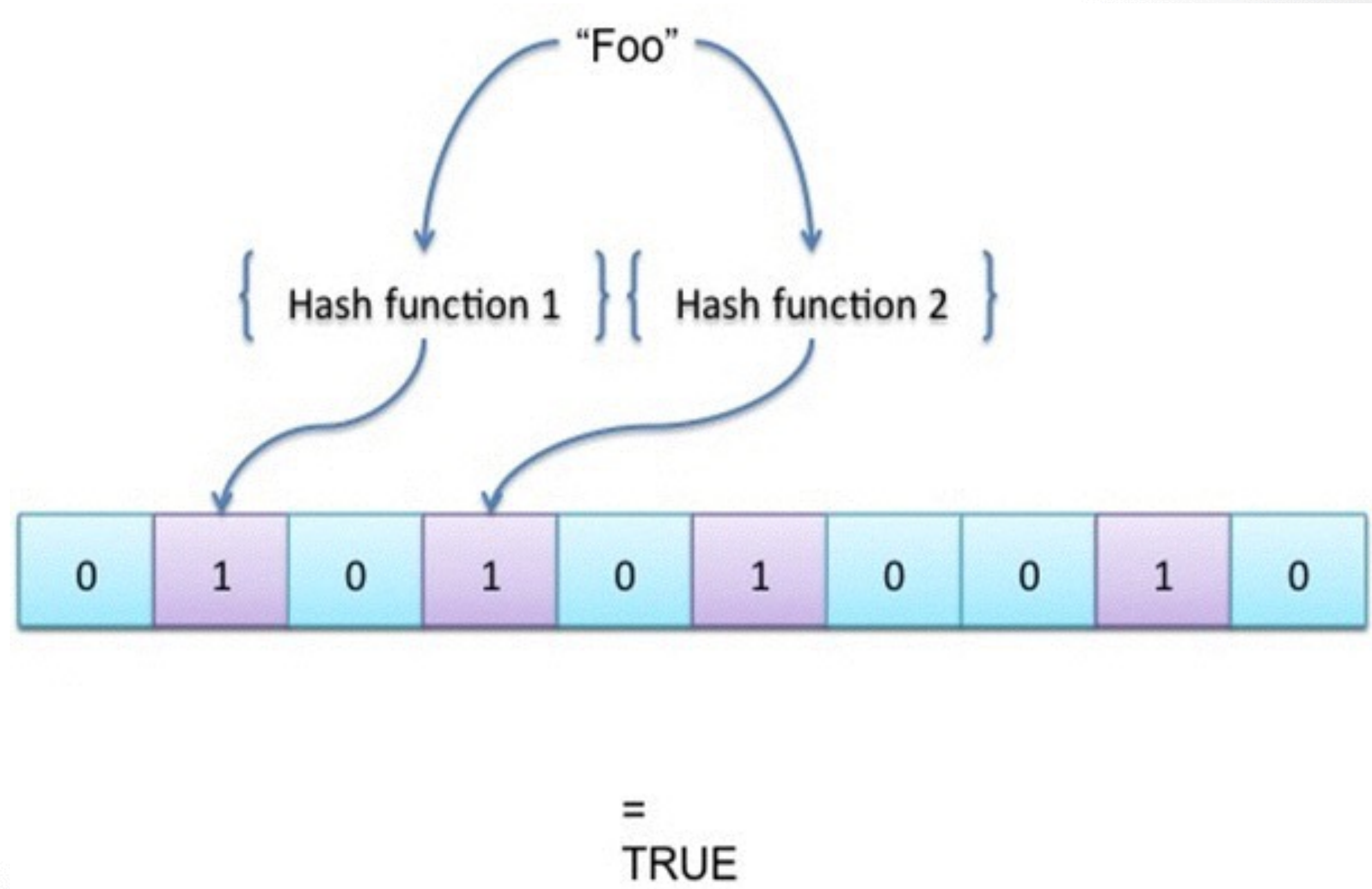
Membership test - Bloom Filters

- very memory efficient
- almost as fast as CHM
- small % false pos
- ZERO false neg
- append only
 - see Cuckoo Filter
- `BloomFilter.create()`



Membership test - Bloom Filters

- very memory efficient
- almost as fast as CHM
- small % false pos
- ZERO false neg
- append only
 - see Cuckoo Filter
- `BloomFilter.create()`



Concurrency for shared resources - Striped Lock

- **ConcurrentHashMap**'s secret
- eg: **ConcurrentBloomFilter**
 - up to n threads non-blocking
 - n shards with a **ReadWriteLock** and **BloomFilter**
 - ConsistentHash index into shards
- **Striped** in Guava

Random Sampling

```
float sampleRate = 0.10f; // 10%  
if (ThreadLocalRandom.current().nextFloat() < sampleRate) {  
    statsd.increment("high.velocity.request.success");  
}
```

- for high velocity events
- NEVER for sparse events



Distributed Consensus - Zookeeper

- metadata store
- set membership
- distributed lock
- leader election
- Netflix Curator
- **DON'T TRY THIS AT HOME!**

Async IO

- Get up to 1M connections, capped by bandwidth
- Netty
 - EPOLL on Linux
 - (Composite)ByteBuf
 - ChannelGroup
 - HashedWheelTimer
 - **READ THE SOURCE!**
- Others work as well:
 - Vert.x, NodeJS, Python Gevent



Data processing pipelines

- Kafka Queues with many partitions
- Auto-scale group of workers
- commit batches of work to ZK (restart, lag)
- Emit stats (success, error, timing)
- Custom dashboard
 - sampled data from the stream
 - inject data in the stream (debug)
- Future:
 - Spark Streaming
 - Mesos + Marathon + Chronos



Mechanical Sympathy

- Disruptor, lock-free Queue
- BlockingQueue - backpressure!
- JCTools - Multi Prod Single Cons Queue
- CAS - Atomic* & Unsafe
- OpenHFT
 - off-heap storage
 - cpu affinity for JVM threads
 - zero allocation hashing
- mechanical-sympathy.blogspot.com



THANK YOU



Jo Voordeckers

SR. SOFTWARE ENGINEER - LF PLATFORM

Email: jvoordeckers@livefyre.com

 [@jvoordeckers](https://twitter.com/jvoordeckers)

livefyre 

San Francisco, CA

New York, NY

London, UK

 [@livefyre](https://twitter.com/livefyre)

press.livefyre.com

blog.livefyre.com