Kafka

high-throughput, persistent,
multi-reader streams

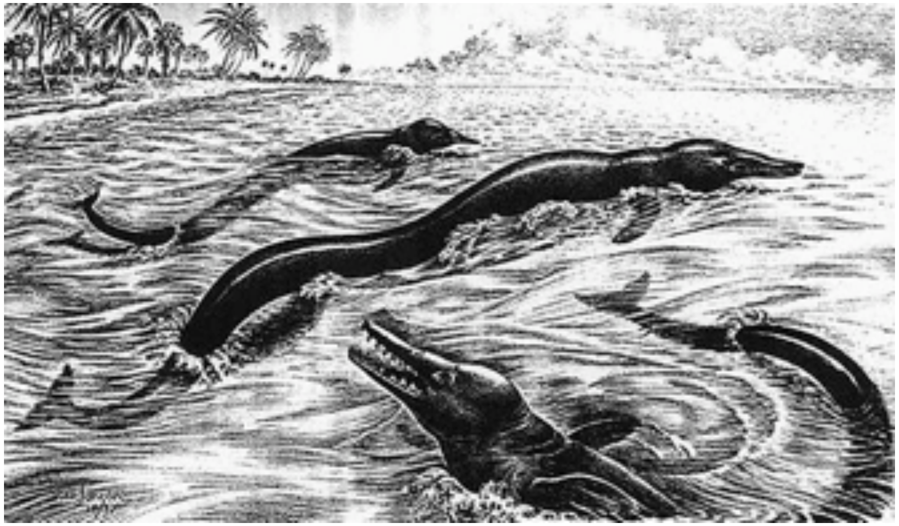
http://sna-projects.com/kafka

Linked in

- LinkedIn SNA (Search, Network, Analytics)
- Worked on a number of open source projects at LinkedIn (Voldemort, Azkaban, …)
- Hadoop, data products

Problem

How do you model and process stream data for a large website?



Examples

Tracking and Logging – Who/what/when/where

Metrics – State of the servers

Queuing – Buffer between online and “nearline” processing

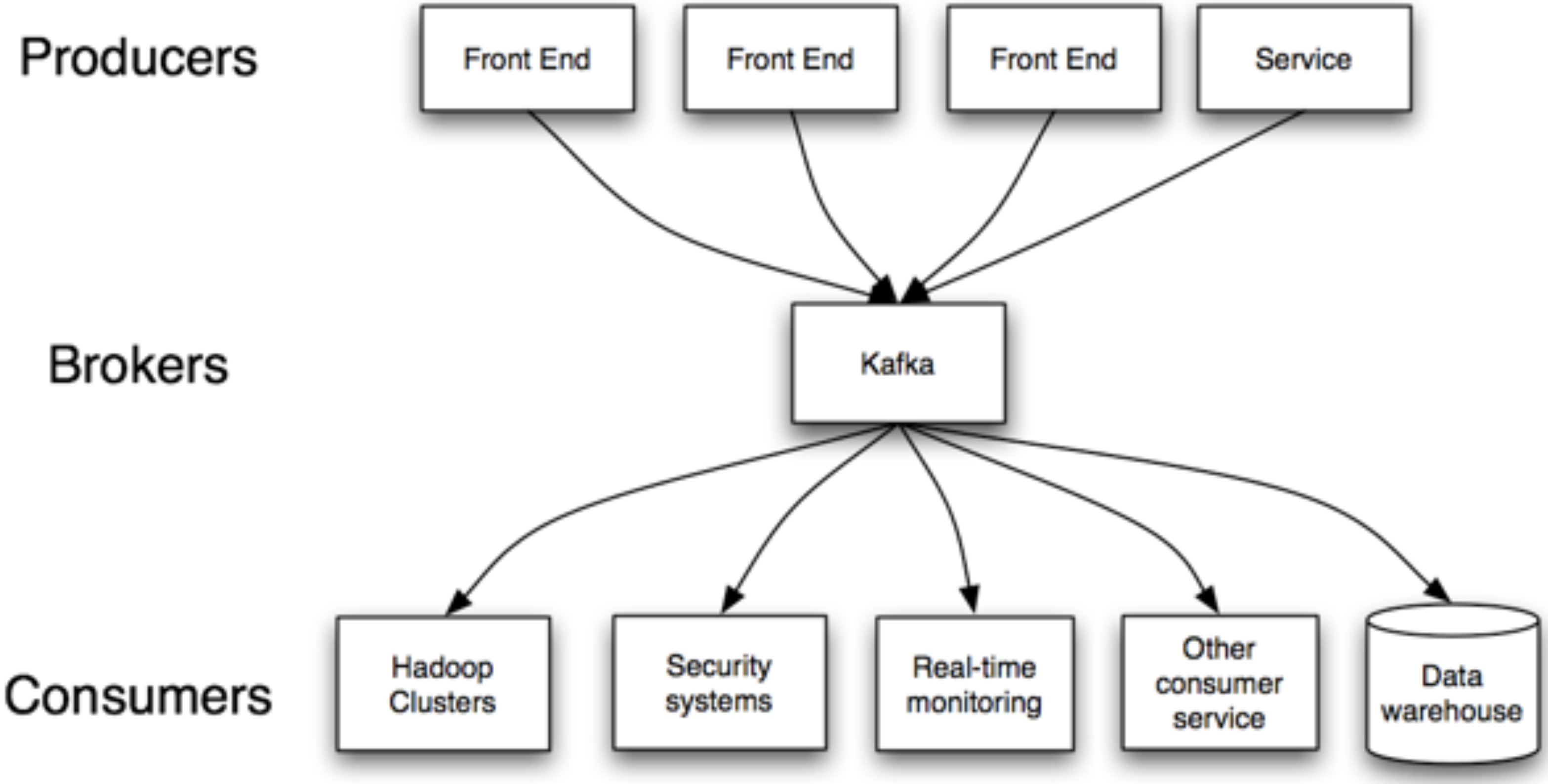
Change capture – Database updates

Messaging Examples: numerous JMS brokers, RabbitMQ, ZeroMQ

The Hard Parts

persistence, scale,
throughput, replication,
semantics, simplicity

Tracking



Producers

Front End

Front End

Front End

Service

Brokers

Kafka

Consumers

Hadoop Clusters

Security systems

Real-time monitoring

Other consumer service

Data warehouse

Tracking Basics

- Example “events” (around 60 types)
  - Search
  - Page view
  - Invitation
  - Impressions
- Avro serialization
- Billions of events
- Hundreds of GBs/day
- Most events have multiple consumers
  - Security services
  - Data warehouse
  - Hadoop
  - News feed
  - Ad hoc

Existing messaging systems

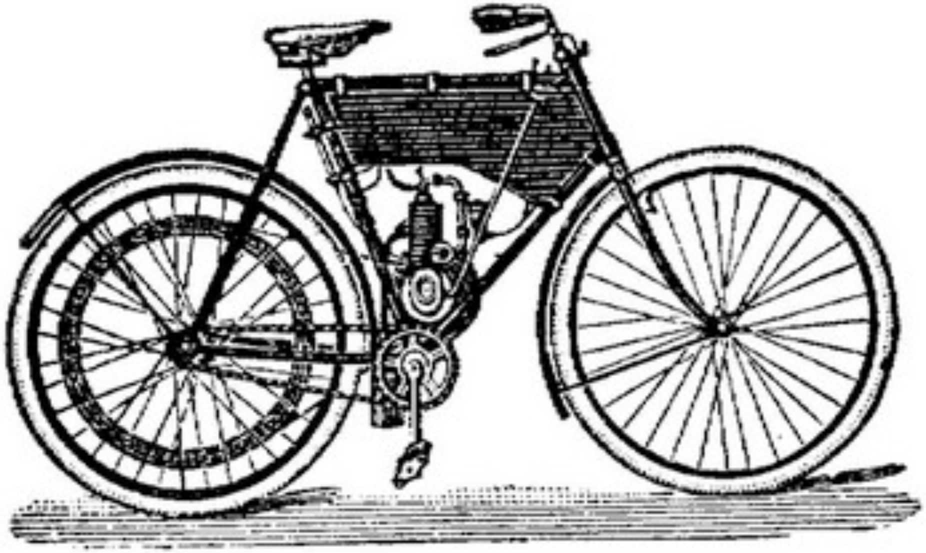
- JMS
  - An API not an implementation
  - Not a very good API
    - Weak or no distribution model
    - High complexity
    - Painful to use
  - Not cross language
- Existing systems seem to perform poorly with large datasets

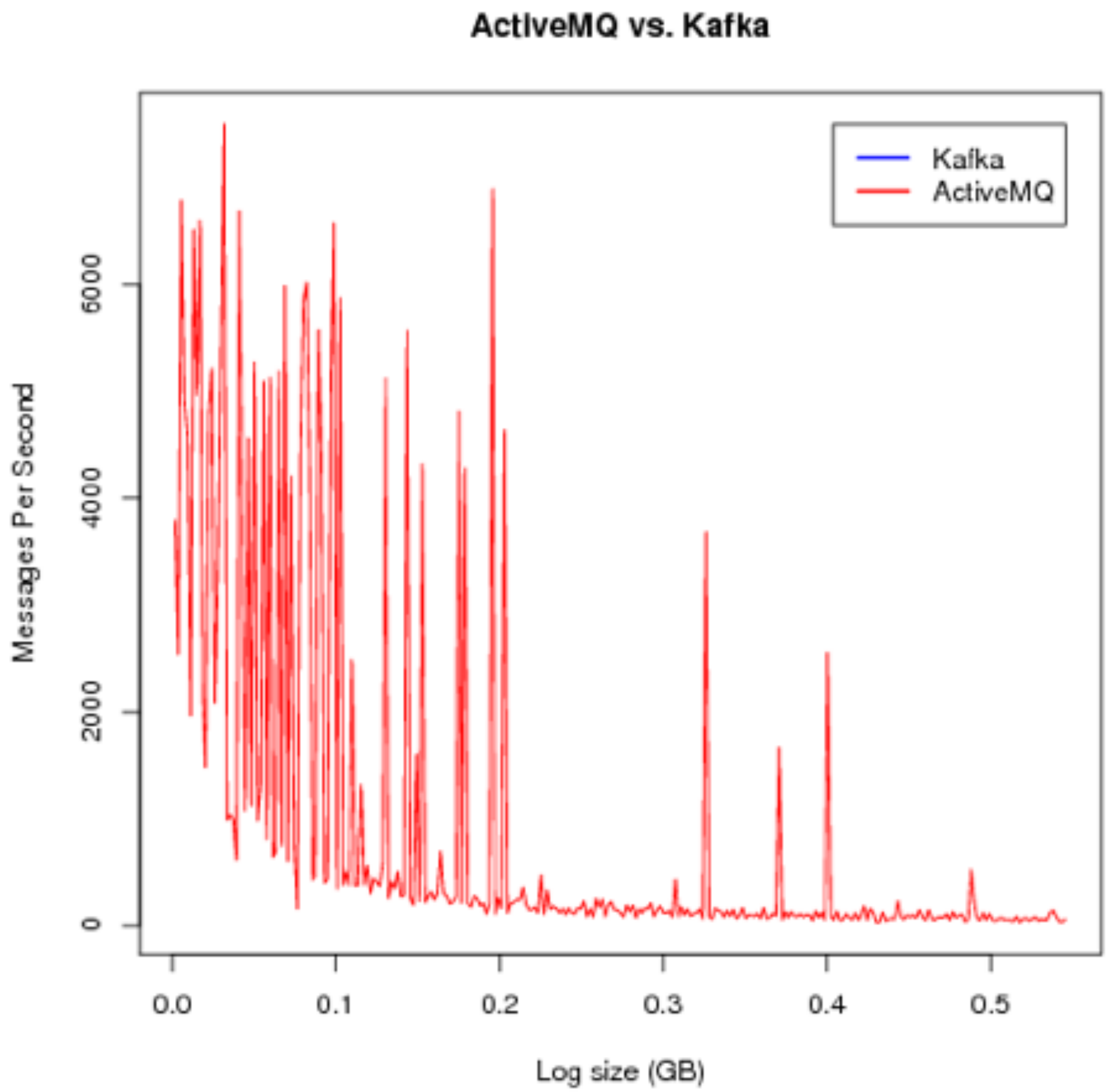
Ideas

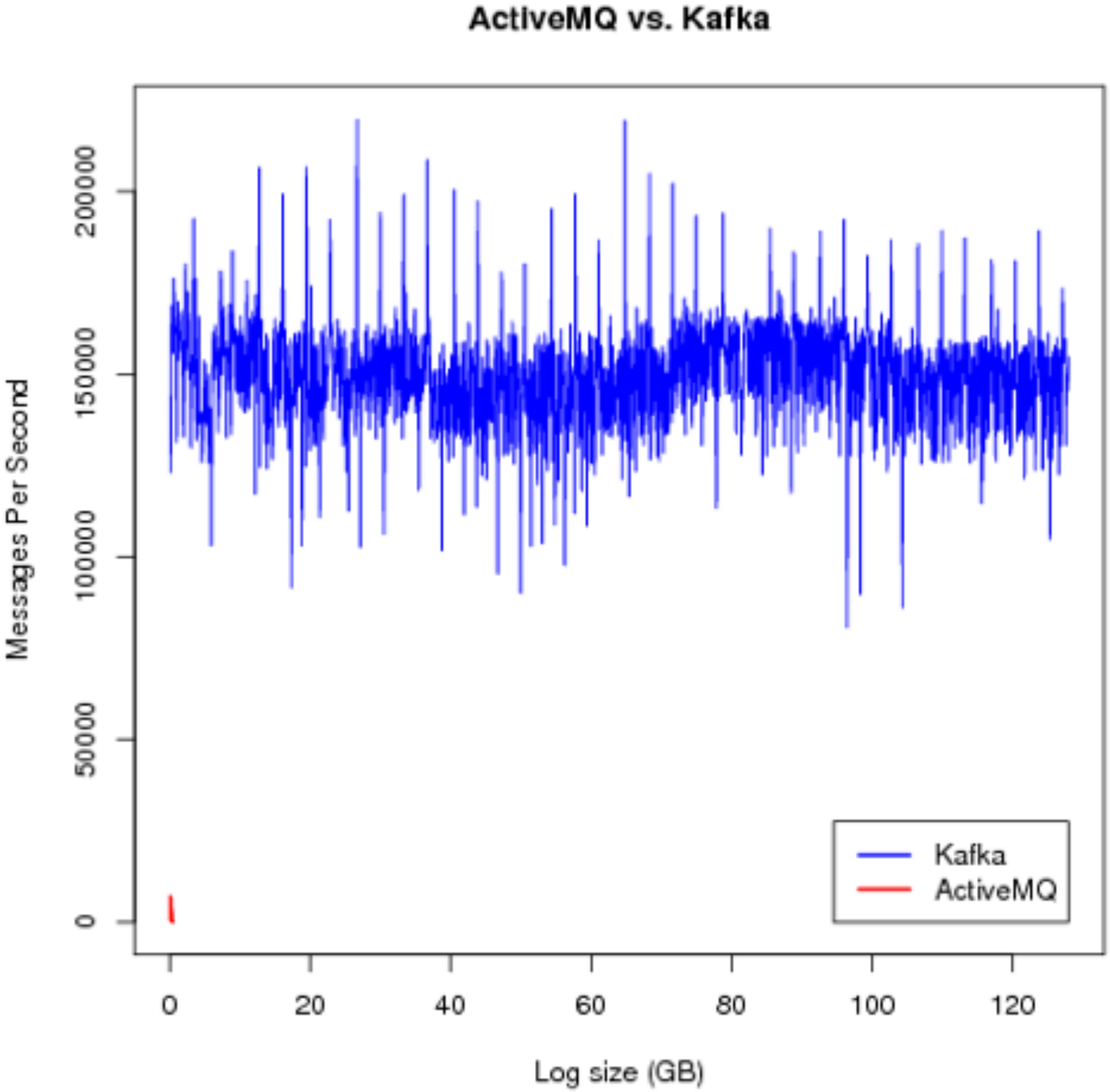
1. Eschew random-access persistent data structures
2. Allow very parallel consumption (e.g. Hadoop)
3. Explicitly distributed
4. Push/Pull not Push/Push

Performance Test

- Two Amazon EC2 large instances
  - Dual core AMD 2.0 GHz
  - 1 7200 rpm SATA drive
  - 8GB memory
- 200 byte messages
- 8 Producer threads
- 1 Consumer thread
- Kafka
  - Flush 10k messages
  - Batch size = 50
- ActiveMQ
  - syncOnWrite = false
  - fileCursor

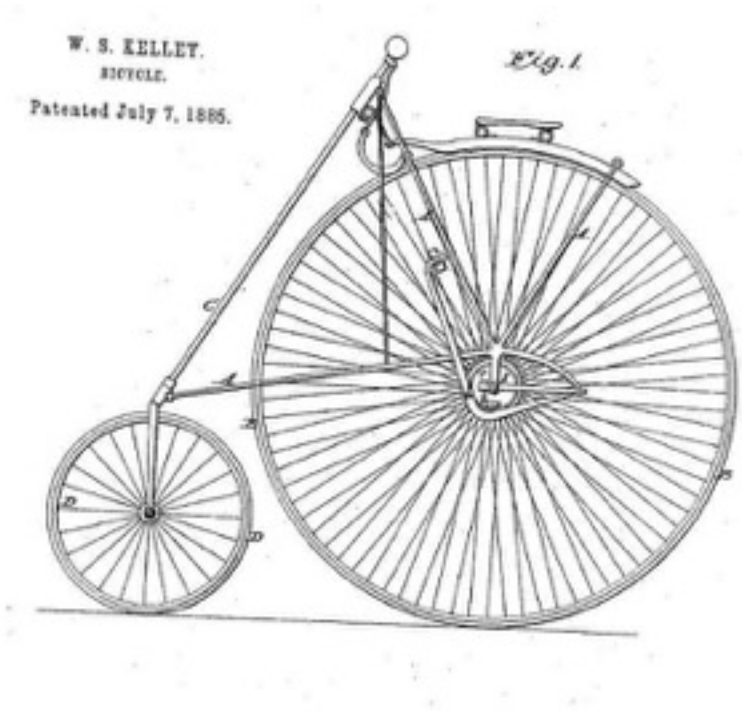



ActiveMQ vs. Kafka
Kafka
ActiveMQ
Messages Per Second
0
2000
4000
6000
0.0
0.1
0.2
0.3
0.4
0.5
Log size (GB)


ActiveMQ vs. Kafka
Messages Per Second
0
50000
100000
150000
200000
Log size (GB)
0
20
40
60
80
100
120
Kafka
ActiveMQ

Performance Summary

- Producer
  - 111,729.6 messages/sec
  - 22.3 MB per sec
- Consumer
  - 193,681.7 messages/sec
  - 38.6 MB per sec
- On on our hardware
  - 50MB/sec produced
  - 90MB/sec consumed



How can we get high performance with persistence?

Some tricks

- Disks are fast when used sequentially
  - Single thread linear read/write speed: > 300MB/sec
  - Reads are faster still, when cached
  - Appends are effectively O(1)
  - Reads from known offset are effectively O(1)
- End-to-end message batching
- Zero-copy network implementation (sendfile)
- Zero copy message processing APIs

Implementation

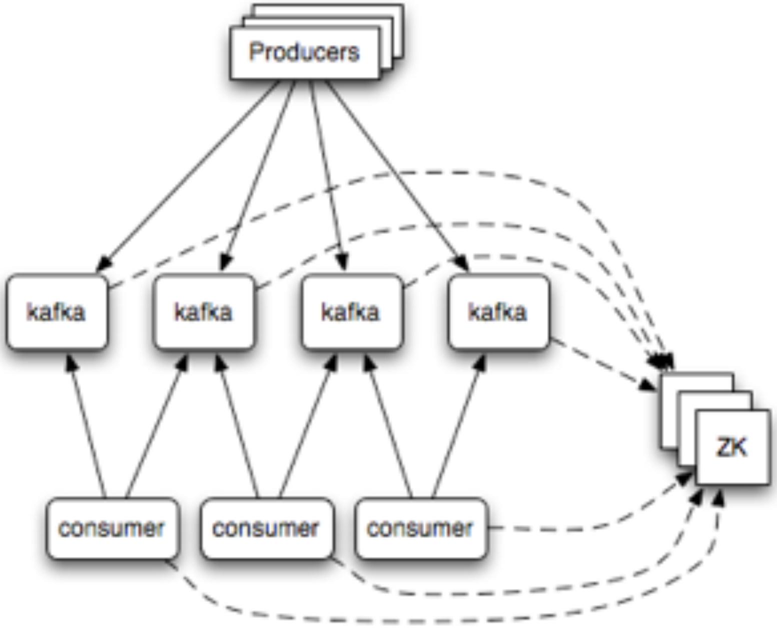
- ~5k lines of Scala
- Standalone jar
- NIO socket server
- Zookeeper handles distribution, client state
- Simple protocol
  - Python, Ruby, and PHP clients contributed

Königl. Preuss. Jagdstaffel Boelcke *

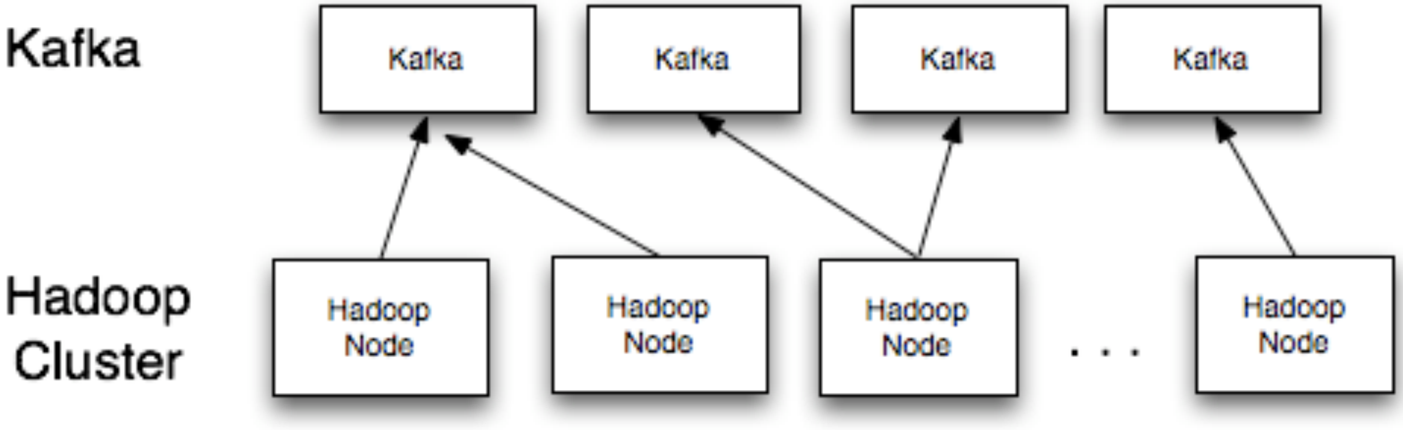
Brief-
stempel

Distribution

- Producer randomly load balanced
- Consumer balances M brokers, N consumers



Hadoop InputFormat



Consumer State

- Data is retained for N days
- Client can calculate next valid offset from any fetch response
- All server APIs are stateless
- Client can reload if necessary

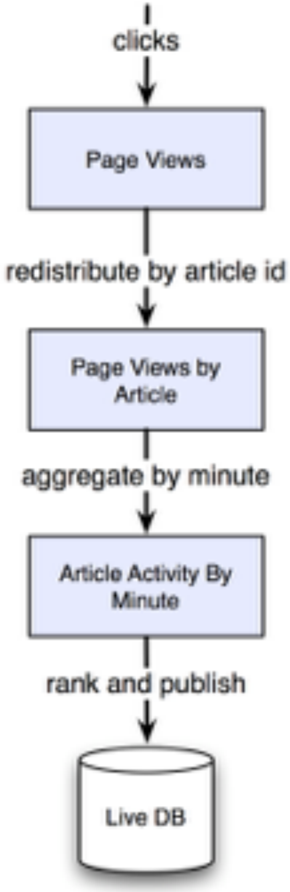
APIs

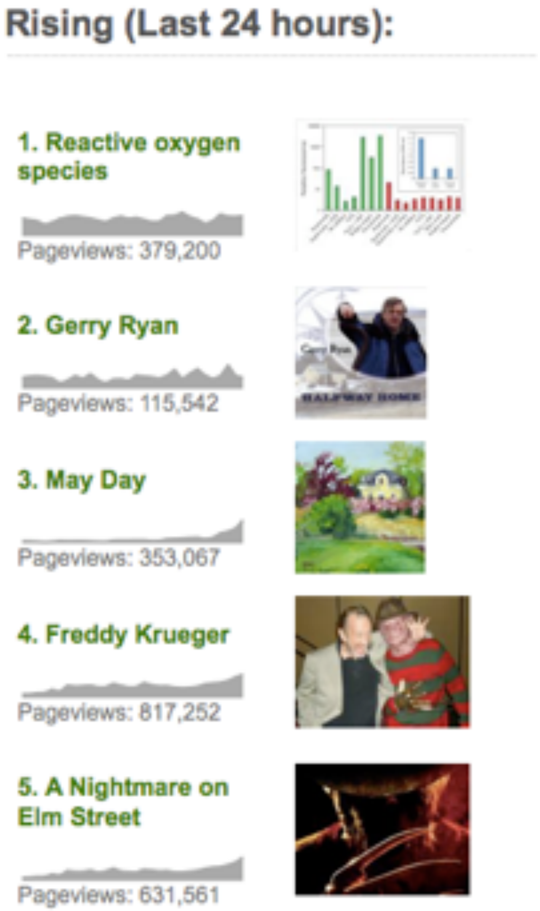
```
// Sending messages
client.send(“topic”, messages)


// Receiveing messages
Iterable stream =
  client.createMessageStreams(…).get(“topic”).get(0)
for(message: stream) {
  // process messages
}
```

Stream processing (0.06 release)

Data published to persistent topics, and redistributed by primary key between stages





Rising (Last 24 hours):

1. Reactive oxygen species

Pageviews: 379,200

2. Gerry Ryan

Pageviews: 115,542

3. May Day

Pageviews: 353,067

4. Freddy Krueger

Pageviews: 817,252

5. A Nightmare on Elm Street

Pageviews: 631,561

Also coming soon

- End-to-end block compression
- Contributed php, ruby, python clients
- Hadoop InputFormat, OutputFormat
- Replication

The End

http://sna-projects.com/kafka

https://github.com/kafka-dev/kafka

kafka-dev@google-groups.com

jay.kreps@gmail.com