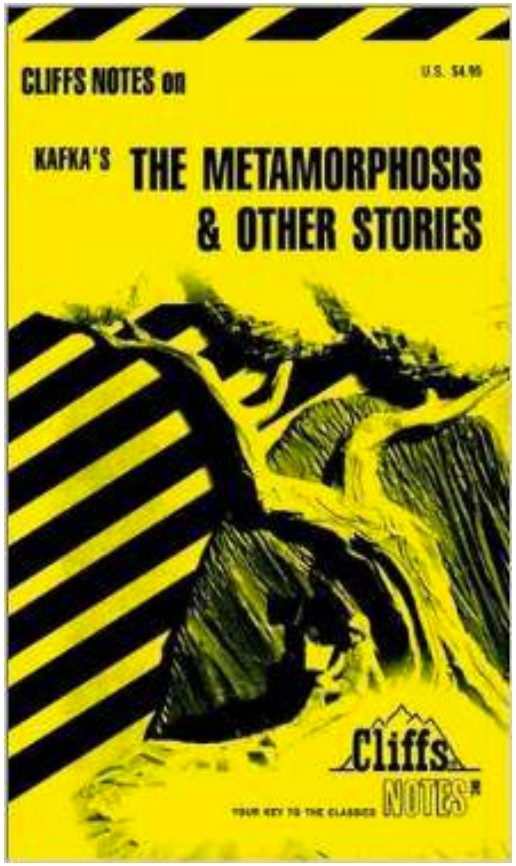
Putting Apache Kafka to Use

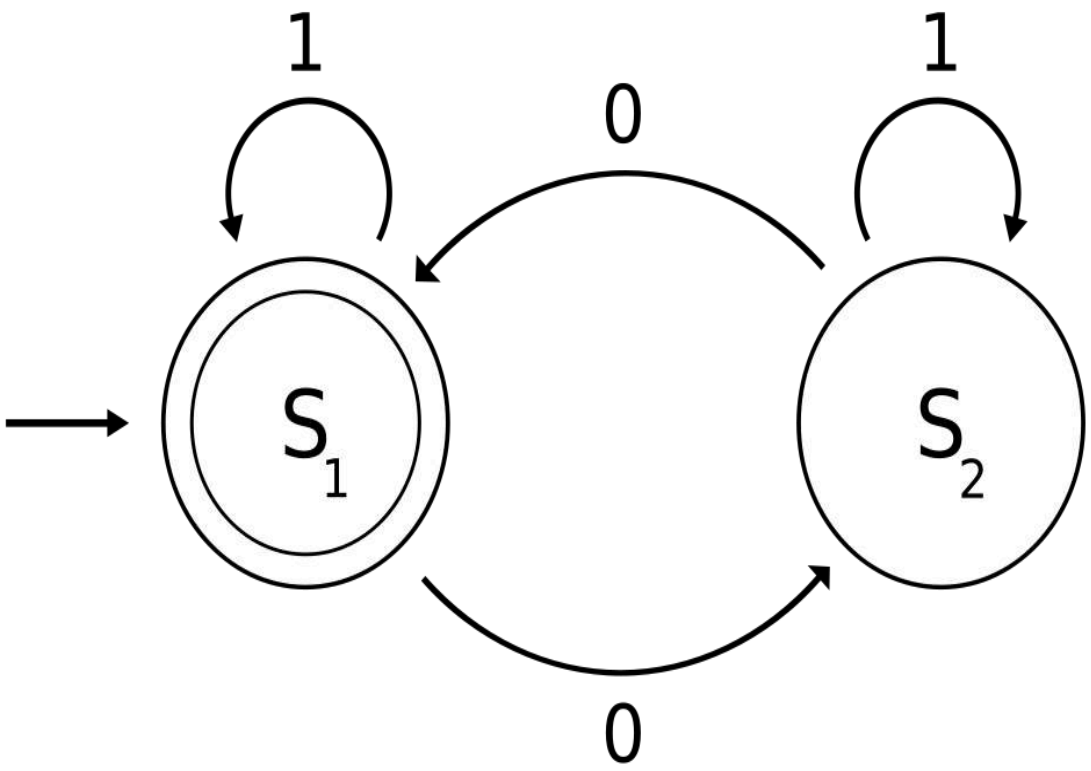
Building a Real-time Data Platform for Event Streams

JAY KREPS, CONFLUENT

A Couple of Themes



Theme 1: Rise of Events



Theme 2: Immutability Everywhere

Level	Example	Immutable Alternative
Mutable local state	Counter in a for loop	Functional Programming
Mutable process-wide state	ConcurrentHashMap	Functional Programming
Mutable on disk structures	B-Tree	LSM
Distributed systems	Dynamo-like key-value store	State machine replication
Mutability in databases	RDBMS	Event Sourcing
Company-wide data flow	Double write	Kafka

Theme 3: Datacenter-Level Thinking

The Datacenter Needs an Operating System

Matei Zaharia, Benjamin Hindman, Andy Konwinski, Ali Ghodsi,
Anthony D. Joseph, Randy Katz, Scott Shenker, Ion Stoica
University of California, Berkeley

1 Introduction

Clusters of commodity servers have become a major computing platform, powering not only some of today's most popular consumer applications—Internet services such as search and social networks—but also a growing number of scientific and enterprise workloads [2]. This rise in cluster computing has even led some to declare that "the datacenter is the new computer" [16, 24]. However, the tools for managing and programming this new computer are still immature. This paper argues that, due to the growing diversity of cluster applications and users, the datacenter increasingly needs an operating system.[1]

We take a broad view of an operating system as both a software layer that manages and abstracts hardware and a package of tools, such as programming languages and debuggers, that facilitate the use of a computer. Tradi-

and Pregel steps). However, this is currently difficult because applications are written independently, with no common interfaces for accessing resources and data.

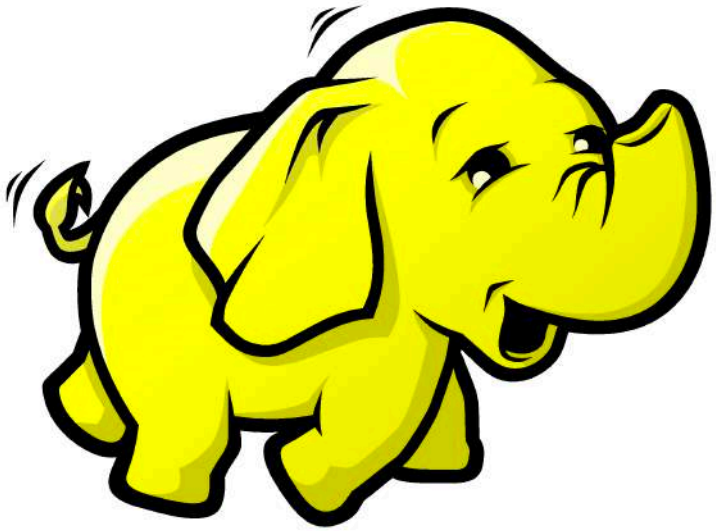
In addition, clusters are serving increasing numbers of concurrent users, which require responsive time-sharing. For example, while MapReduce was initially used for a small set of batch jobs, organizations like Facebook are now using it to build data warehouses where hundreds of users run near-interactive ad-hoc queries [29].

Finally, programming and debugging cluster applications remains difficult even for experts, and is even more challenging for the growing number of non-expert users (*e.g.,* scientists) starting to leverage cloud computing.

While cluster users are well-aware of these problems, current solutions are often ad-hoc. For example, in the Hadoop stack [3], MapReduce acts as a common exe-

Experience at LinkedIn

2009: We want all our data in Hadoop!



What is all our data?

Initial approach: “gut it out”

Problems

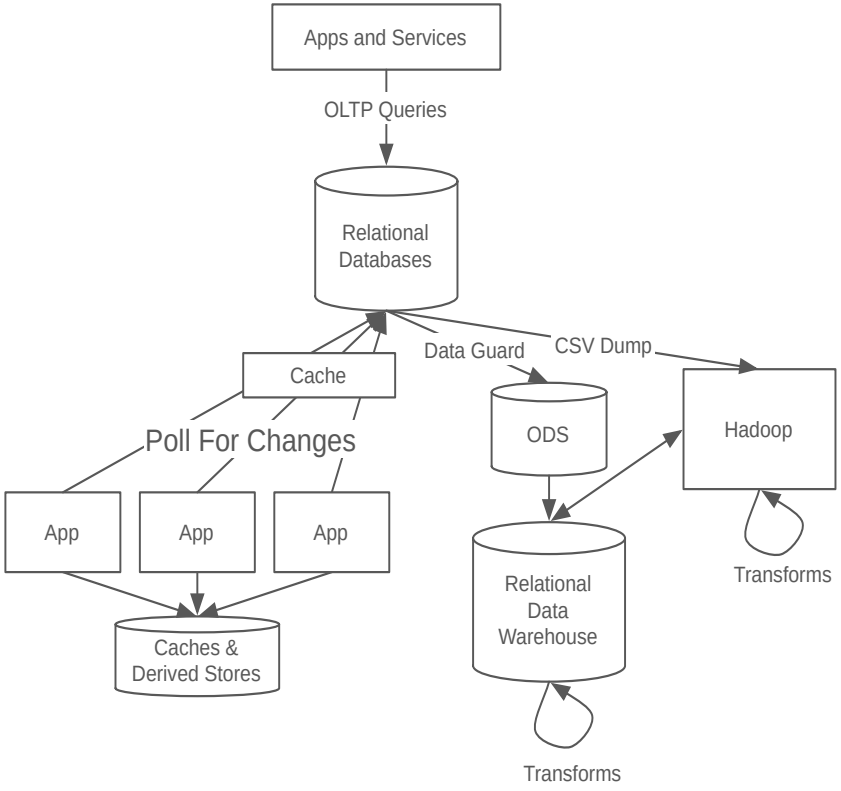
- Data coverage
- Many source systems
  - Relational DBs
  - Log files
  - Metrics
  - Messaging systems
- Many data formats
- Constant change
  - New schemas
  - New data sources

Needed: organizational scalability

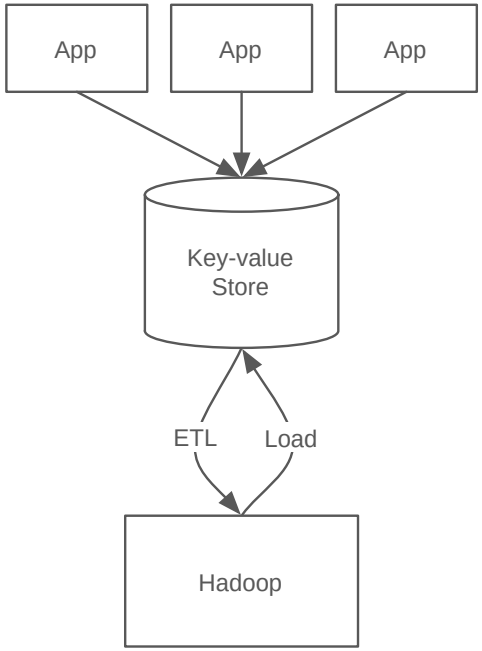
$$\Theta(N) => \Theta(1)$$

How does everything else work?

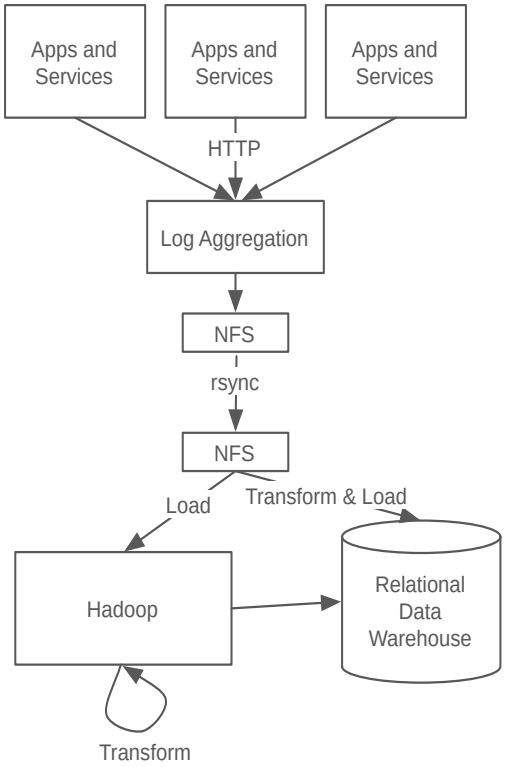
Relational database changes



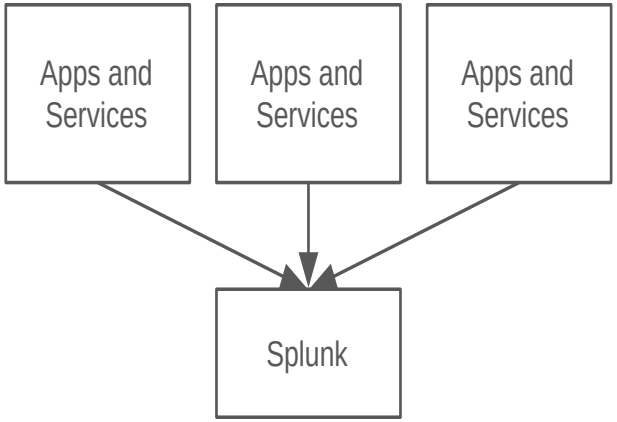
NoSQL



User events

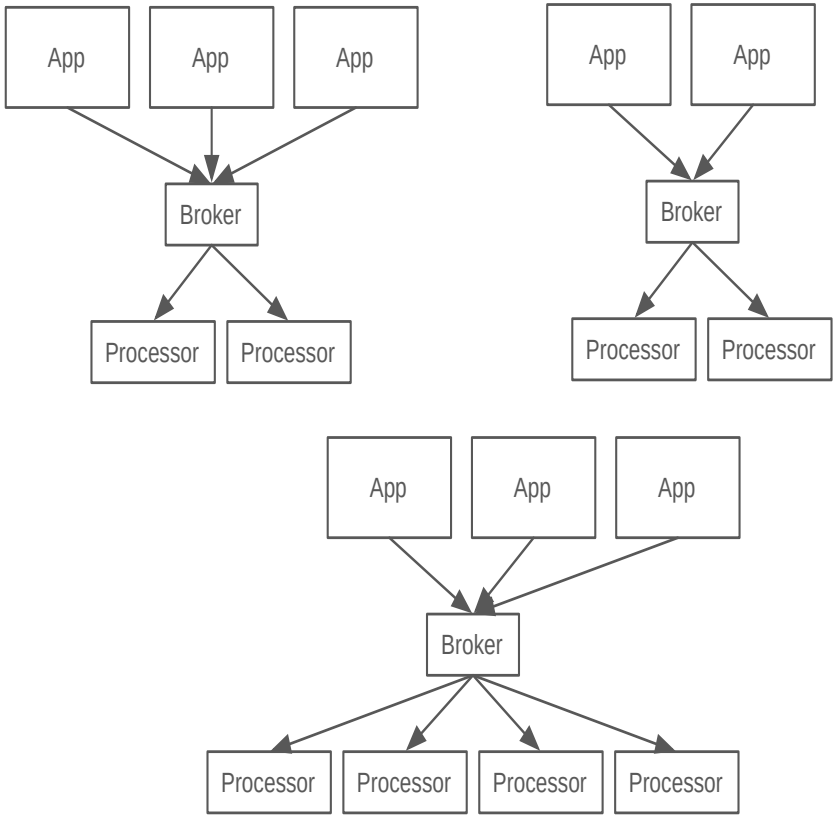

Apps and Services
Apps and Services
Apps and Services
HTTP
Log Aggregation
NFS
rsync
NFS
Load
Transform & Load
Hadoop
Relational Data Warehouse
Transform

Application Logs

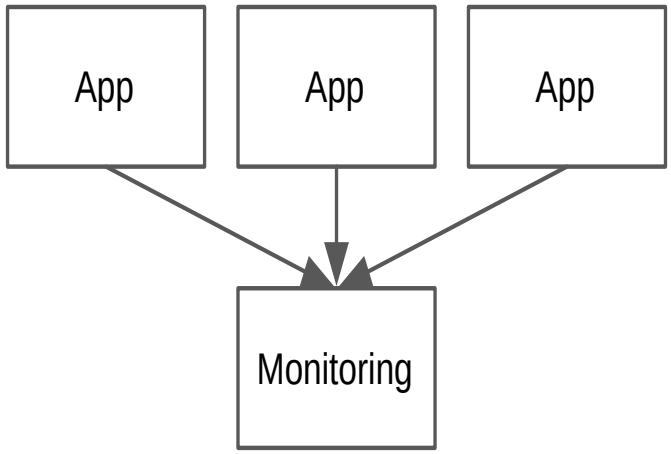


Messaging

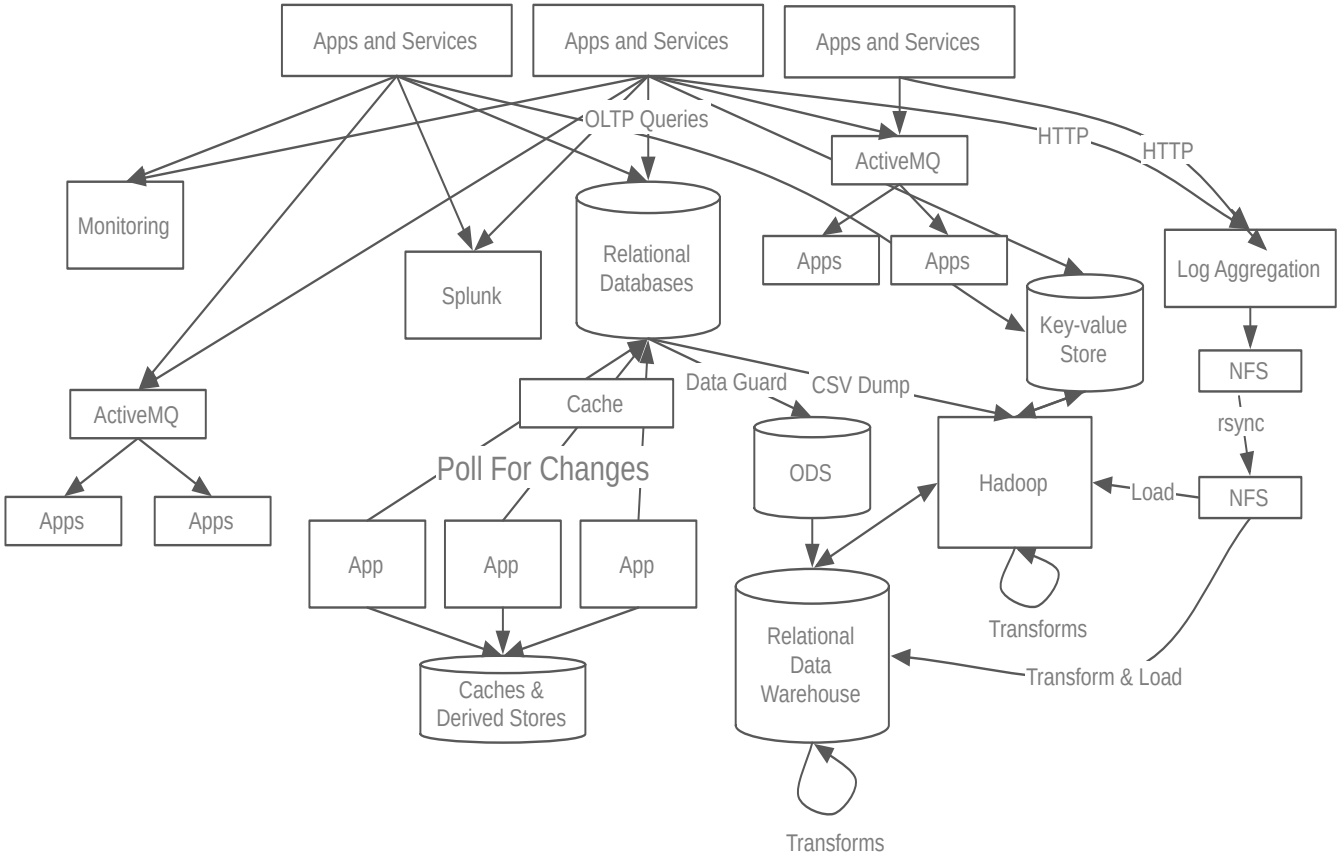

ActiveMQ


App
App
App
Broker
Processor
Processor
App
App
Broker
Processor
Processor
App
App
App
Broker
Processor
Processor
Processor
Processor

Metrics and operational data



This is a giant mess

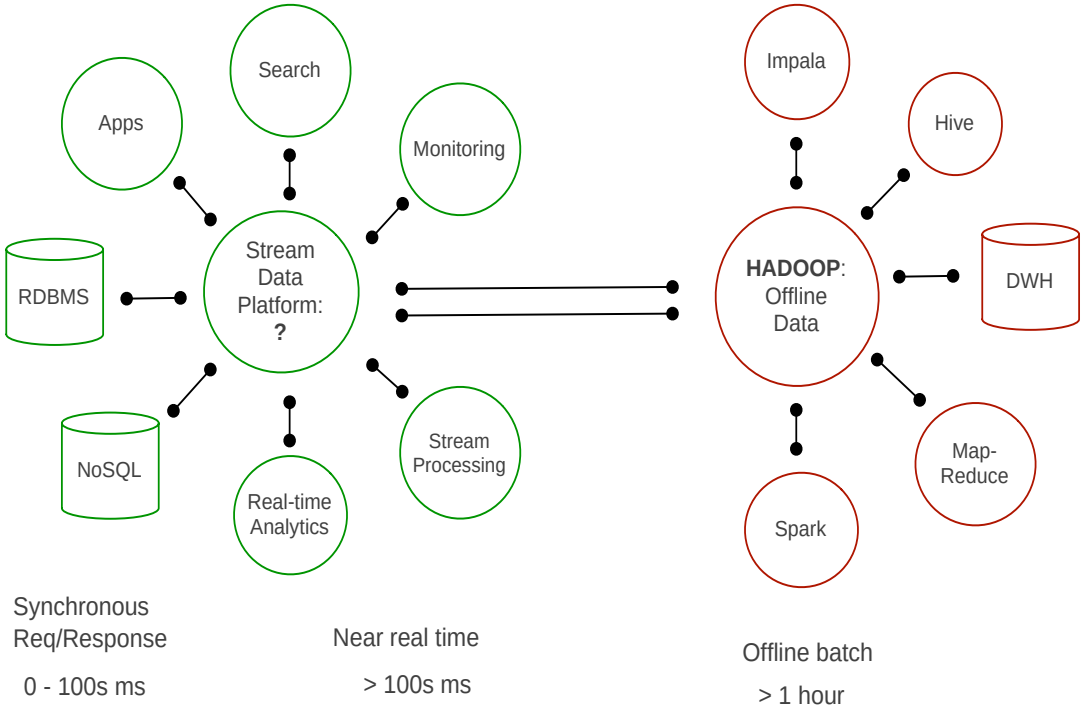


Impossible ideas

- Publish data from Hadoop to a search index
- Run a SQL query to find the biggest latency bottleneck
- Run a SQL query to find common error patterns
- Low latency monitoring of database changes or user activity
- Incorporate popularity in real-time display and relevance algorithms
- Products that incorporate user activity

An infrastructure solution?

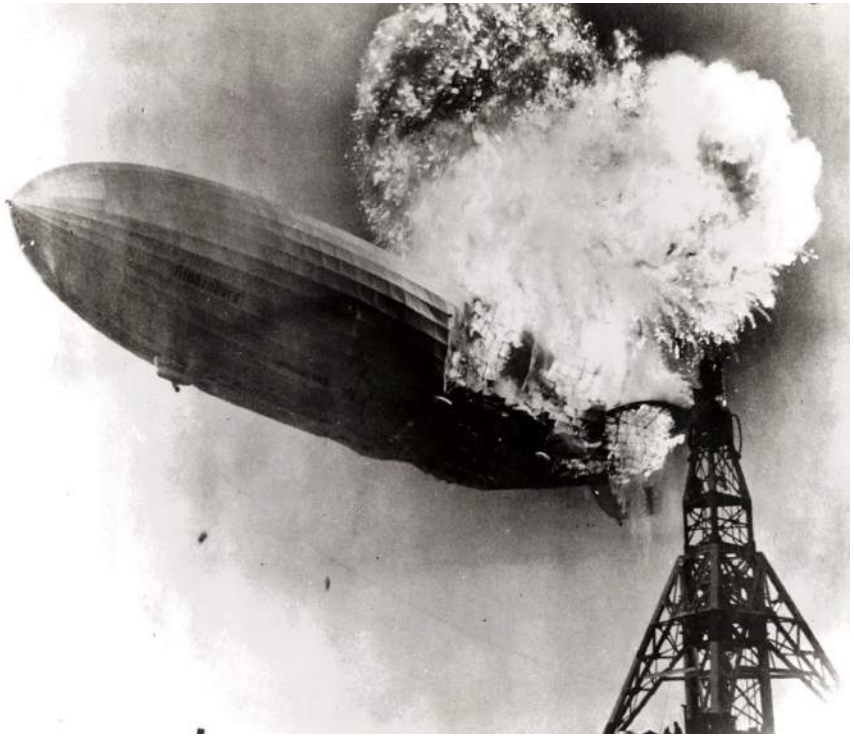
Idea: Stream Data Platform



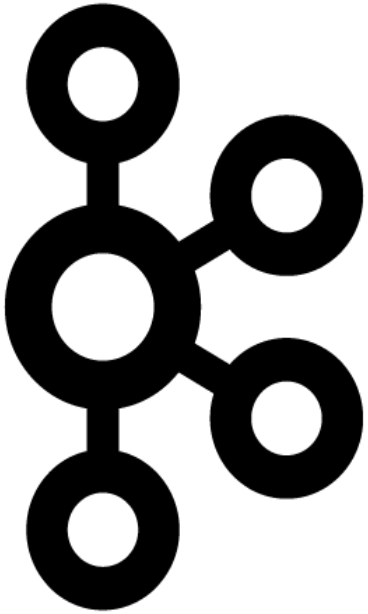
First Attempt: Messaging systems!

Problems

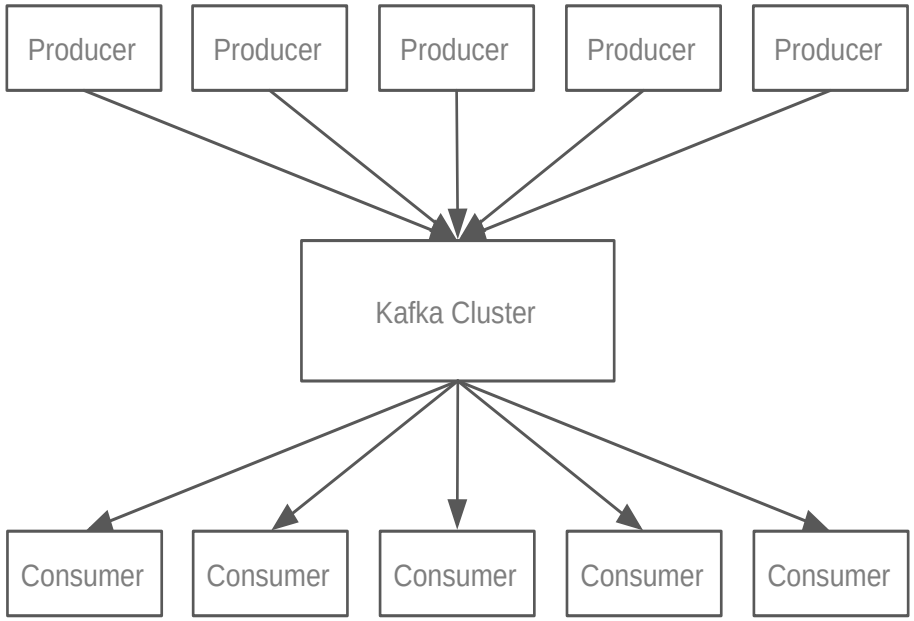
- Throughput
- Batch systems
- Persistence
- Stream Processing
- Ordering guarantees
- Partitioning



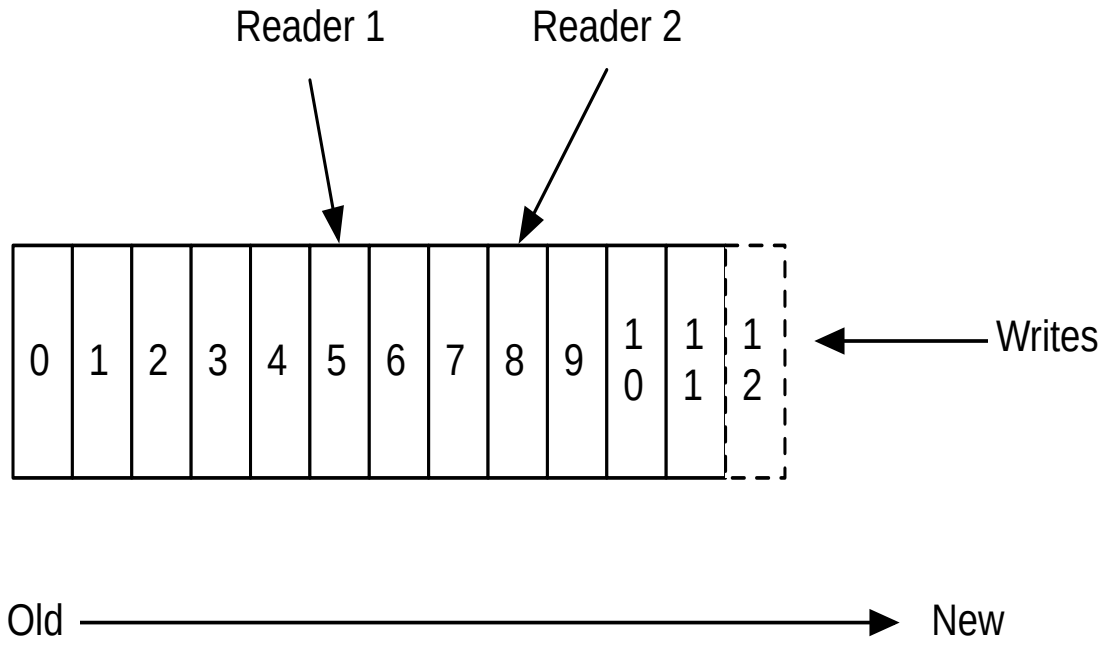
Second Attempt: Build Kafka!



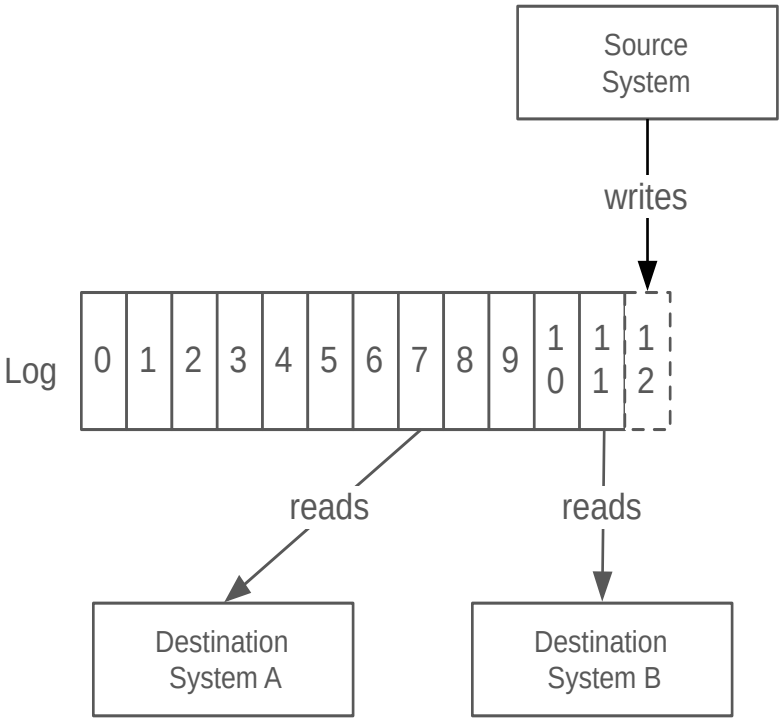
What does it do?



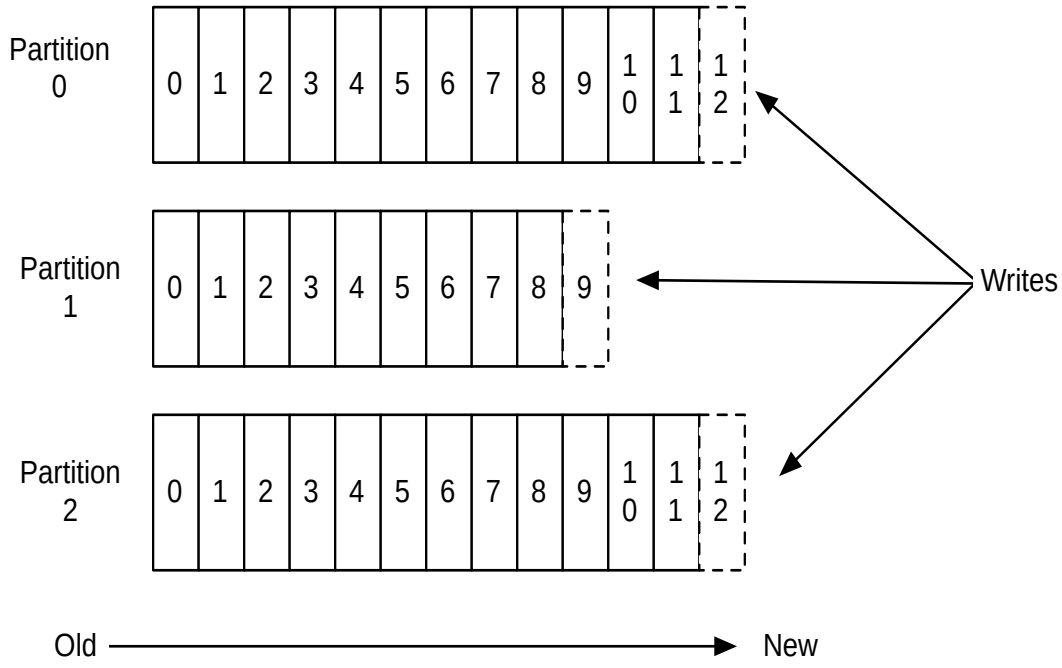
Commit Log Abstraction



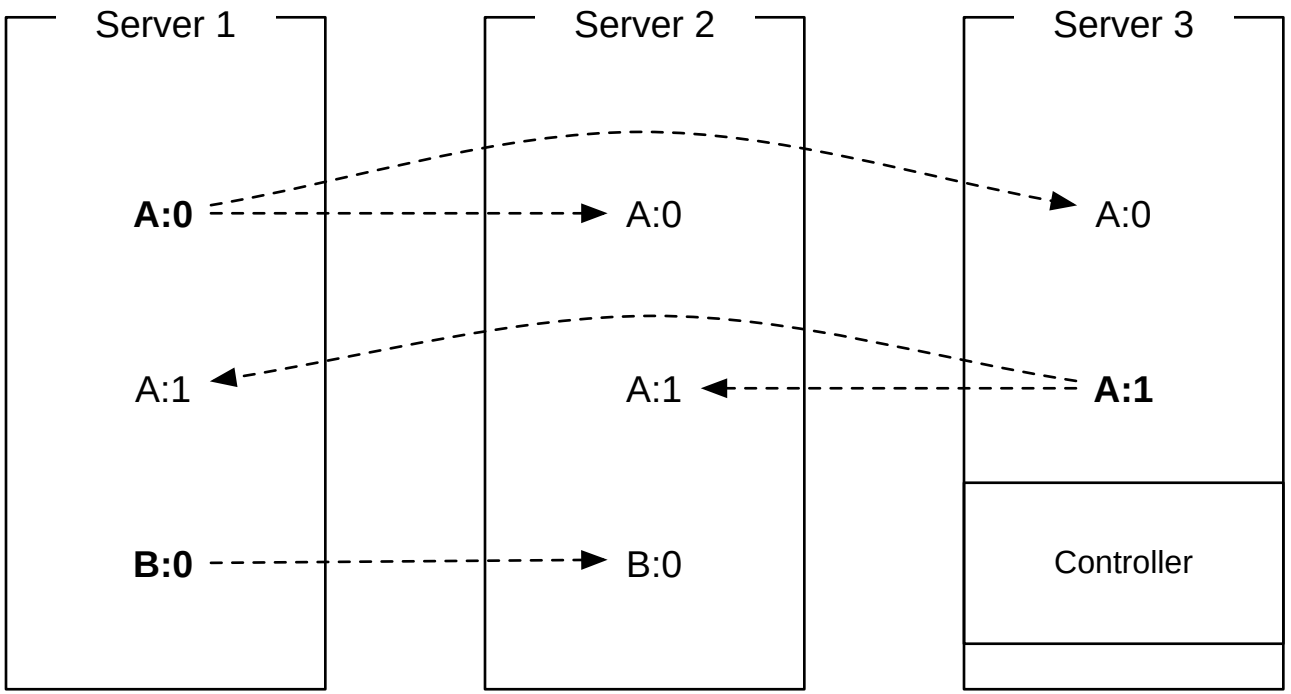
Logs & Publish-Subscribe Messaging



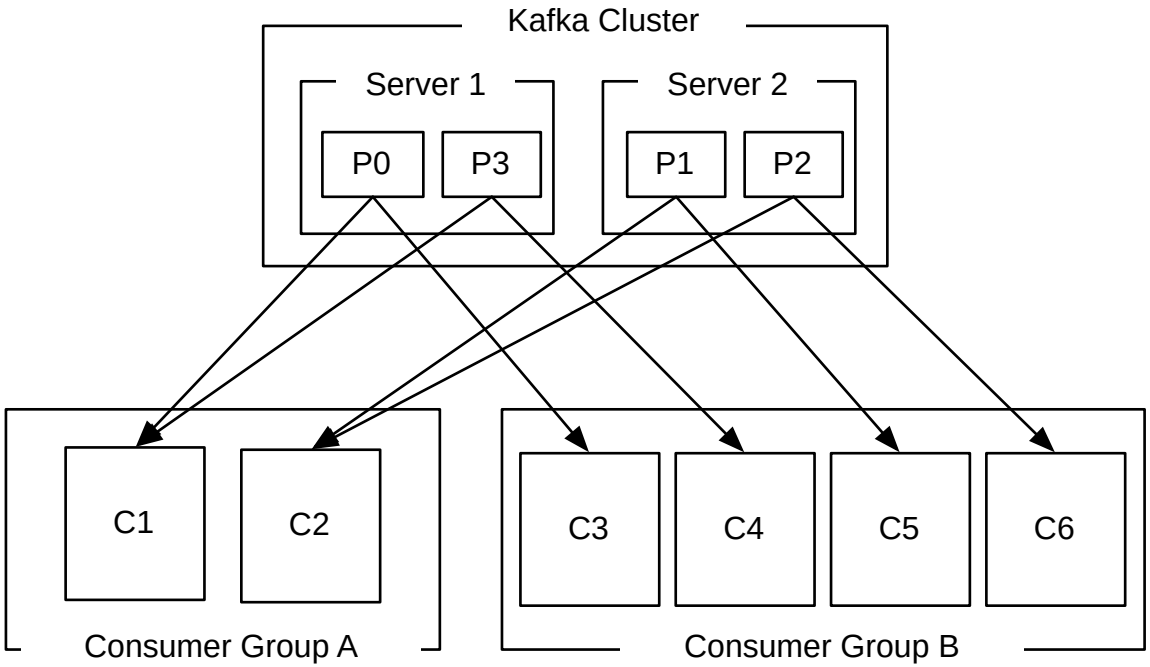
A Kafka Topic



Replication



Scaling Consumers



Kafka: A Modern Distributed System for Streams

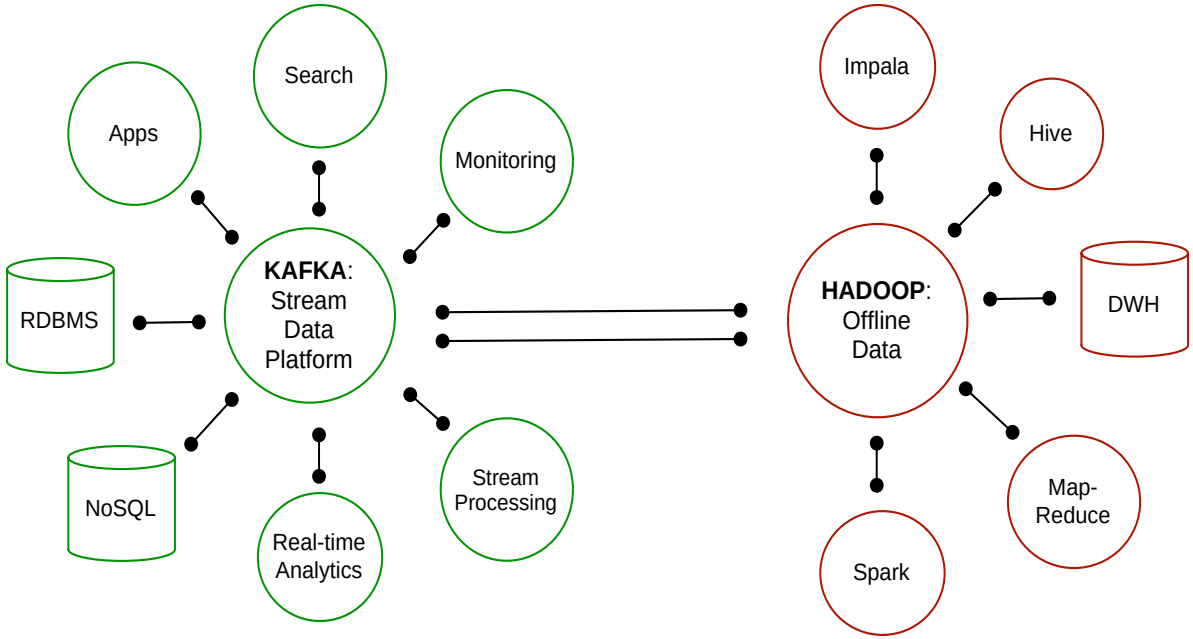
Scalability of a filesystem
- Hundreds of MB/sec/server throughput
- Many TB per server

Guarantees of a database
- Messages strictly ordered
- All data persistent

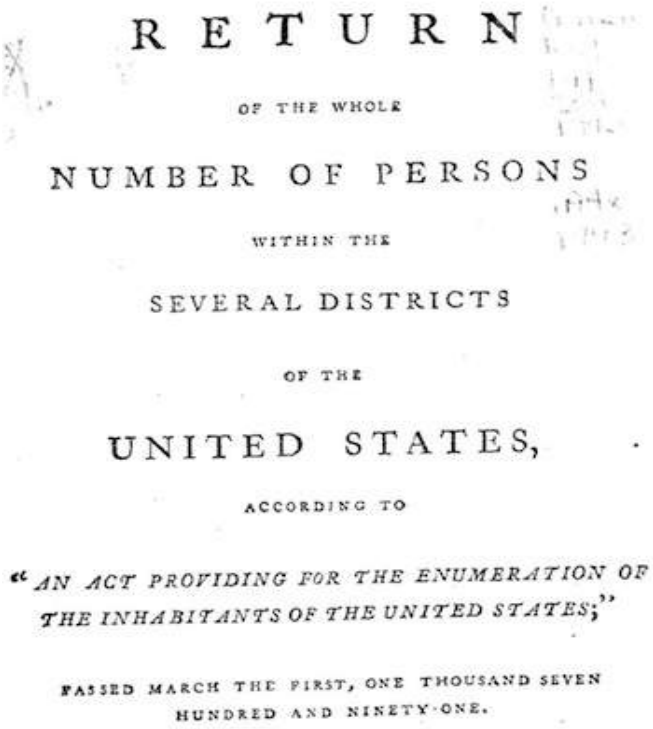
Distributed by default
- Replication
- Partitioning model

Producers, Consumers, and Brokers all fault tolerant and horizontally scalable

Stream Data Platform



Batch Data => Batch Processing



RETURN

OF THE WHOLE

NUMBER OF PERSONS

WITHIN THE

SEVERAL DISTRICTS

OF THE

UNITED STATES,

ACCORDING TO

"*AN ACT PROVIDING FOR THE ENUMERATION OF THE INHABITANTS OF THE UNITED STATES;*"

PASSED MARCH THE FIRST, ONE THOUSAND SEVEN HUNDRED AND NINETY-ONE.

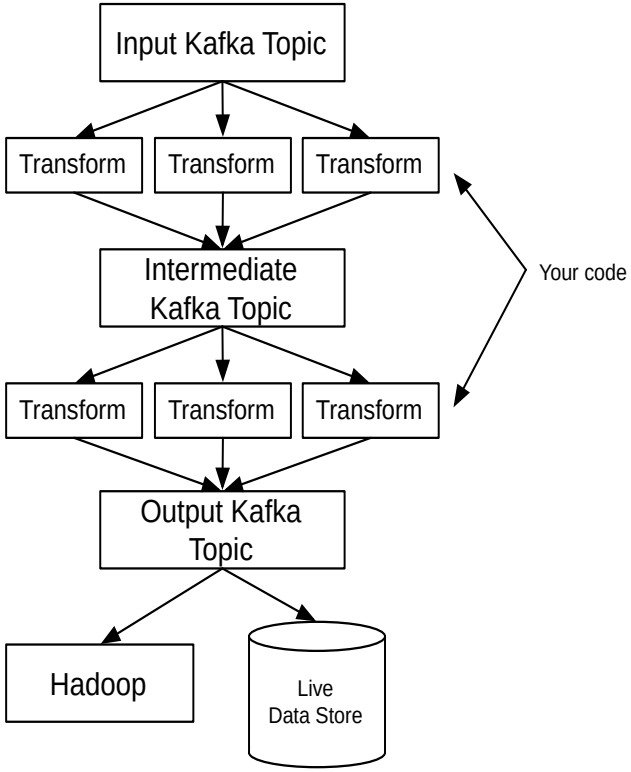
Stream processing is a
generalization
of batch processing
and request/response processing

Request/Response processing: One input => One output

Batch processing:
All inputs => All outputs

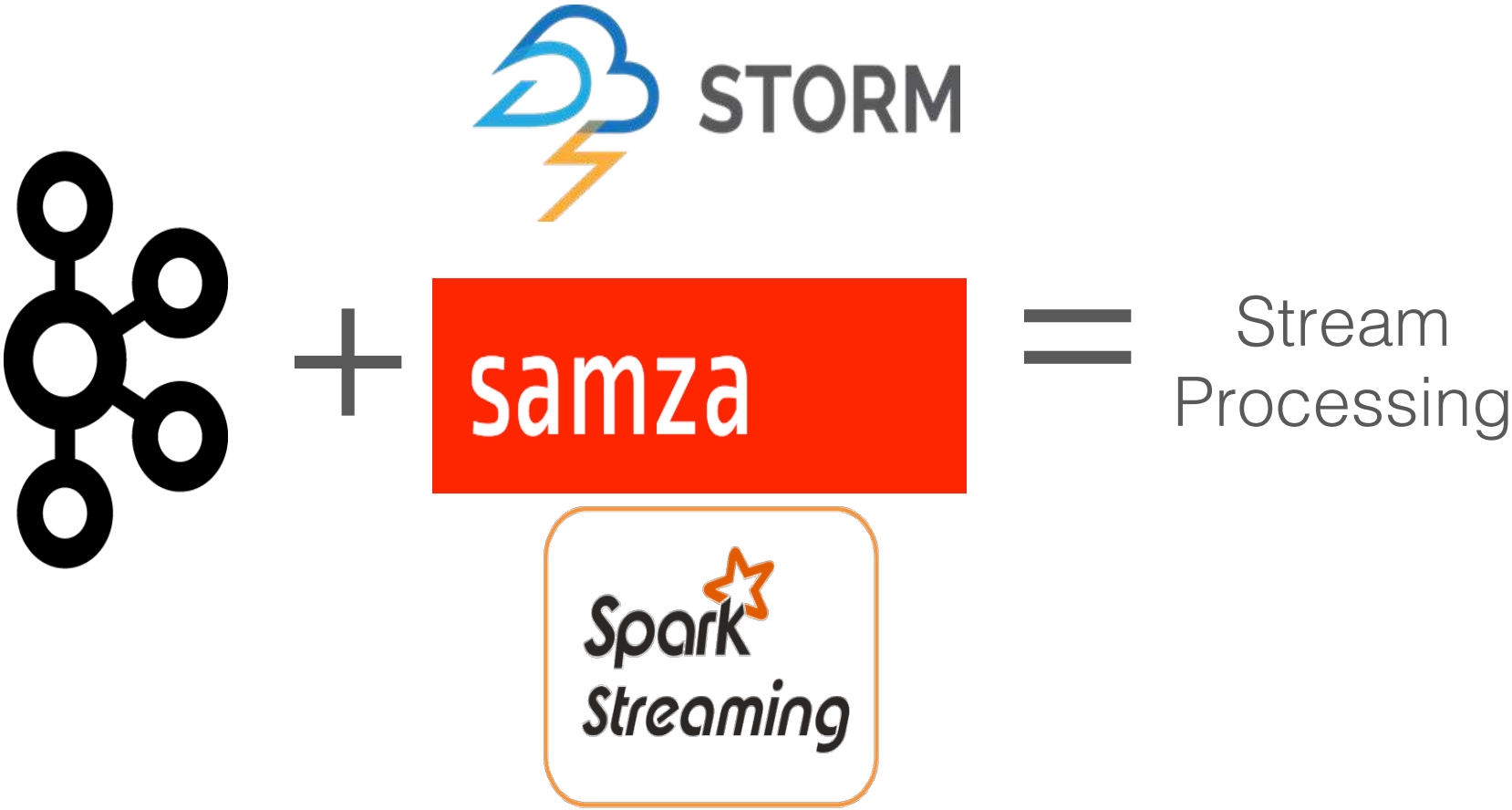
Stream Processing:
Some inputs => some outputs
(you choose how much “some” is)

Stream Processing a la carte




```
cat input | grep “foo” | wc -l
```

Stream Processing with Frameworks



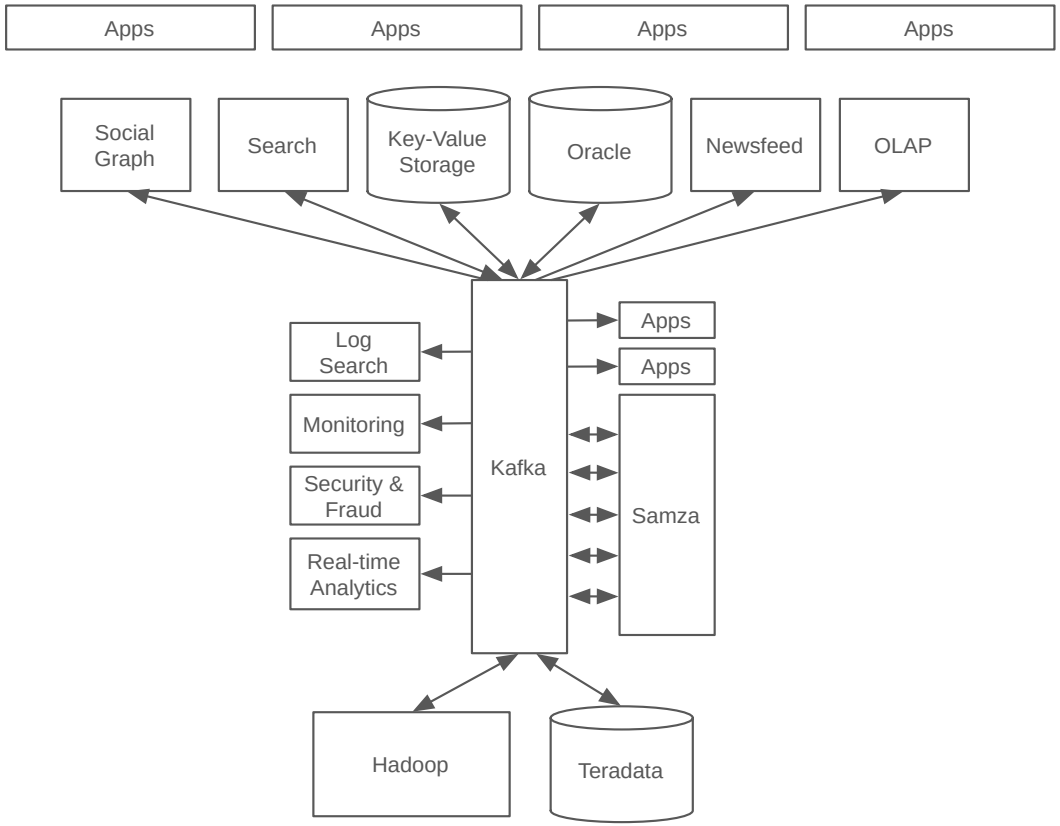
Unix Pipes, Modernized

```
cat /usr/share/dict/words | wc -l
```

On Schemas

Bad Schemas < No Schemas < Good Schemas

Put it all together



At LinkedIn

- Everything in the company is a real-time stream
- > 800 billion messages written per day
- > 2.9 trillion messages read per day
- ~ 1 PB of stream data
- Tens of thousands of producer processes
- Backbone for data stores
  - Search
  - Social Graph
  - Newsfeed
  - Primary storage (in progress)
- Basis for stream processing

Elsewhere


intuit
cisco
Goldman Sachs
yelp
Pinterest
airbnb
dish NETWORK
Adobe
NETFLIX
ebay
UBER
PayPal
box
Square
WIKIPEDIA
The Free Encyclopedia
verizon
salesforce
Cerner

Why this is the future

1. System diversity is increasing
2. Data diversity and volume is increasing
3. The world is getting faster
4. The technology exists

- Mission: Make this a practical reality everywhere
- Product
  - Apache Kafka
  - Schemas and metadata management
  - Connectors for common systems
  - Monitor data flow end-to-end
  - Stream processing integration

Questions?

- Confluent
  - @confluentinc
  - http://confluent.io
  - http://blog.confluent.io/2015/02/25/stream-data-platform-1
- Apache Kafka
  - @apachekafka
  - http://kafka.apache.org
  - http://linkd.in/199iMwY
- Me
  - @jaykreps