Big Data Engineer Bootcamp

Part 1

Agenda

- More on Big Data
- Introduction to Kafka
- Introduction to Zookeeper
- Introduction to Cassandra
- Q&A

Agenda

- **More on Big Data**
- Introduction to Kafka
- Introduction to Zookeeper
- Introduction to Cassandra
- Q&A

More on Big Data

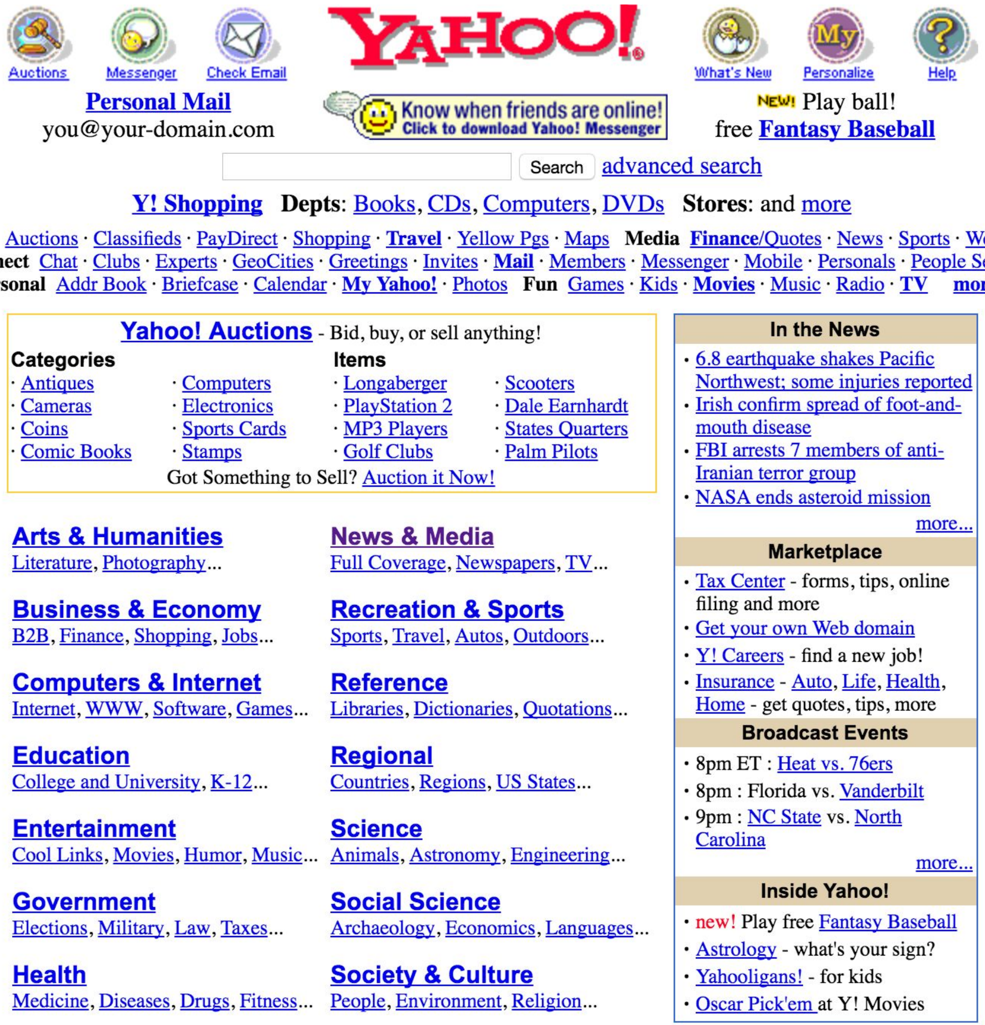
- **History of Big Data**
- Frameworks and Tools

History of Big Data

- 1991 - internet was born (yay)
- 1996 - digital storage is so cheap, cheaper than paper
- 1998 - Google was founded (yay)
- 1999 - the word Big Data first used in a paper
- 2001 - idea of 3V of data described by Doug Laney

Web 1.0

- Front End
  - HTML
  - CSS
  - JavaScript
- Dynamic Content
  - PHP
  - JSP
  - ASP.NET
  - Ruby
  - Perl
- Database
  - RDBMS
- Site Owner Generates Content



Personal Mail
you@your-domain.com

Know when friends are online!
Click to download Yahoo! Messenger

NEW! Play ball!
free **Fantasy Baseball**

advanced search

Y! Shopping **Depts**: Books, CDs, Computers, DVDs **Stores**: and more

Shop Auctions · Classifieds · PayDirect · Shopping · **Travel** · Yellow Pgs · Maps **Media** **Finance/Quotes** · News · Sports · Weather
Connect Chat · Clubs · Experts · GeoCities · Greetings · Invites · **Mail** · Members · Messenger · Mobile · Personals · People Search
Personal Addr Book · Briefcase · Calendar · **My Yahoo!** · Photos **Fun** Games · Kids · **Movies** · Music · Radio · **TV** **more...**

Yahoo! Auctions - Bid, buy, or sell anything!

Categories		Items	
· Antiques	· Computers	· Longaberger	· Scooters
· Cameras	· Electronics	· PlayStation 2	· Dale Earnhardt
· Coins	· Sports Cards	· MP3 Players	· States Quarters
· Comic Books	· Stamps	· Golf Clubs	· Palm Pilots

Got Something to Sell? Auction it Now!

Arts & Humanities
Literature, Photography...

News & Media
Full Coverage, Newspapers, TV...

Business & Economy
B2B, Finance, Shopping, Jobs...

Recreation & Sports
Sports, Travel, Autos, Outdoors...

Computers & Internet
Internet, WWW, Software, Games...

Reference
Libraries, Dictionaries, Quotations...

Education
College and University, K-12...

Regional
Countries, Regions, US States...

Entertainment
Cool Links, Movies, Humor, Music...

Science
Animals, Astronomy, Engineering...

Government
Elections, Military, Law, Taxes...

Social Science
Archaeology, Economics, Languages...

Health
Medicine, Diseases, Drugs, Fitness...

Society & Culture
People, Environment, Religion...

In the News

- 6.8 earthquake shakes Pacific Northwest; some injuries reported
- Irish confirm spread of foot-and-mouth disease
- FBI arrests 7 members of anti-Iranian terror group
- NASA ends asteroid mission

more...

Marketplace

- Tax Center - forms, tips, online filing and more
- Get your own Web domain
- Y! Careers - find a new job!
- Insurance - Auto, Life, Health, Home - get quotes, tips, more

Broadcast Events

- 8pm ET : Heat vs. 76ers
- 8pm : Florida vs. Vanderbilt
- 9pm : NC State vs. North Carolina

more...

Inside Yahoo!

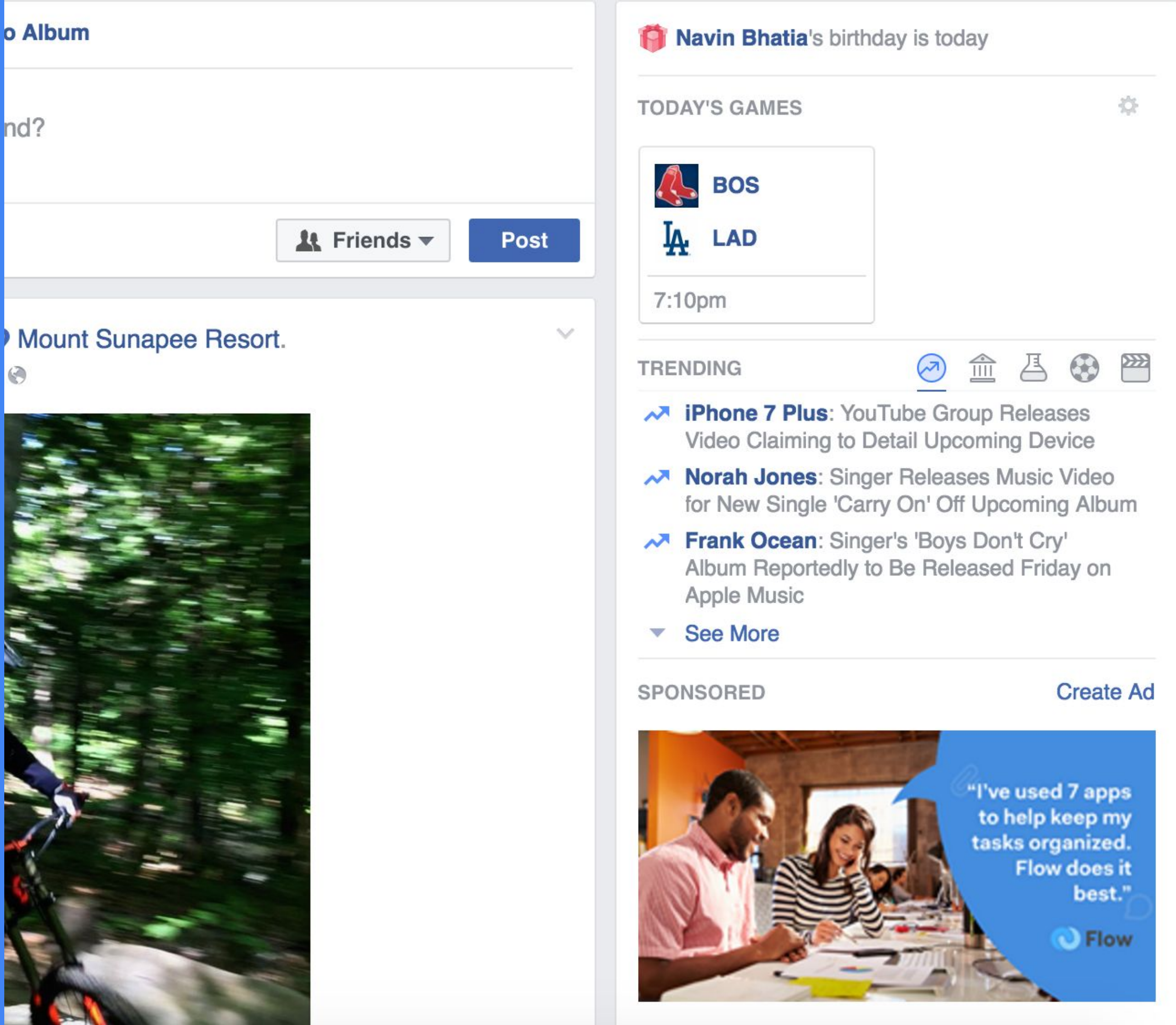
- new! Play free Fantasy Baseball
- Astrology - what's your sign?
- Yahooligans! - for kids
- Oscar Pick'em at Y! Movies

History of Big Data

- 2004 - Facebook was founded (yay)
- 2005 - Web 2.0
- 2007 - iPhone and AWS was released (yay)
- 2008 - 14.7 EB of data being generated
- 2014 - Mobile system surpass desktops

Web 2.0

- Front End
  - AngularJS, etc
- Dynamic Content
  - PHP
  - JSP
  - ASP.NET
  - Ruby
  - JavaScript
- Data Processing
  - Hadoop, etc
- Database
  - NoSQL
- Users Generate Content



Mount Sunapee Resort.

Navin Bhatia's birthday is today

iPhone 7 Plus: YouTube Group Releases Video Claiming to Detail Upcoming Device

Norah Jones: Singer Releases Music Video for New Single 'Carry On' Off Upcoming Album

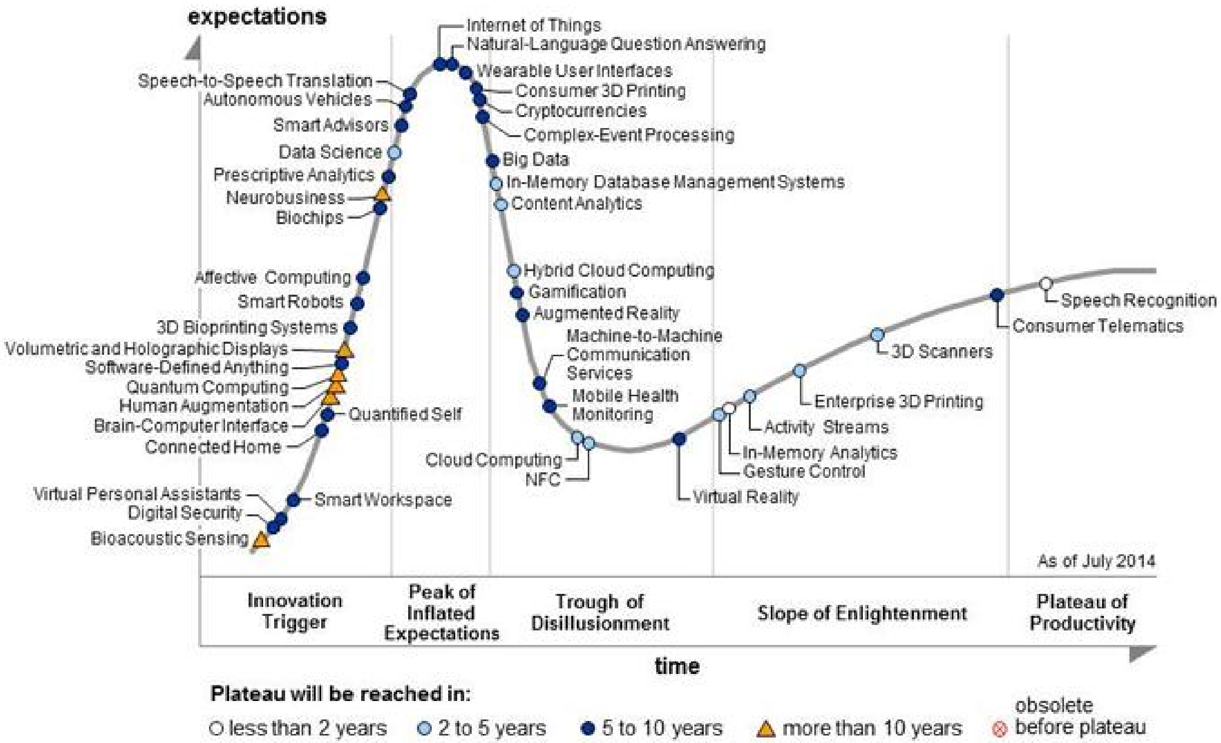
Frank Ocean: Singer's 'Boys Don't Cry' Album Reportedly to Be Released Friday on Apple Music

See More
SPONSORED
Create Ad
“I've used 7 apps to help keep my tasks organized. Flow does it best.”
Flow

Gartner
Cycle of Hype
2014

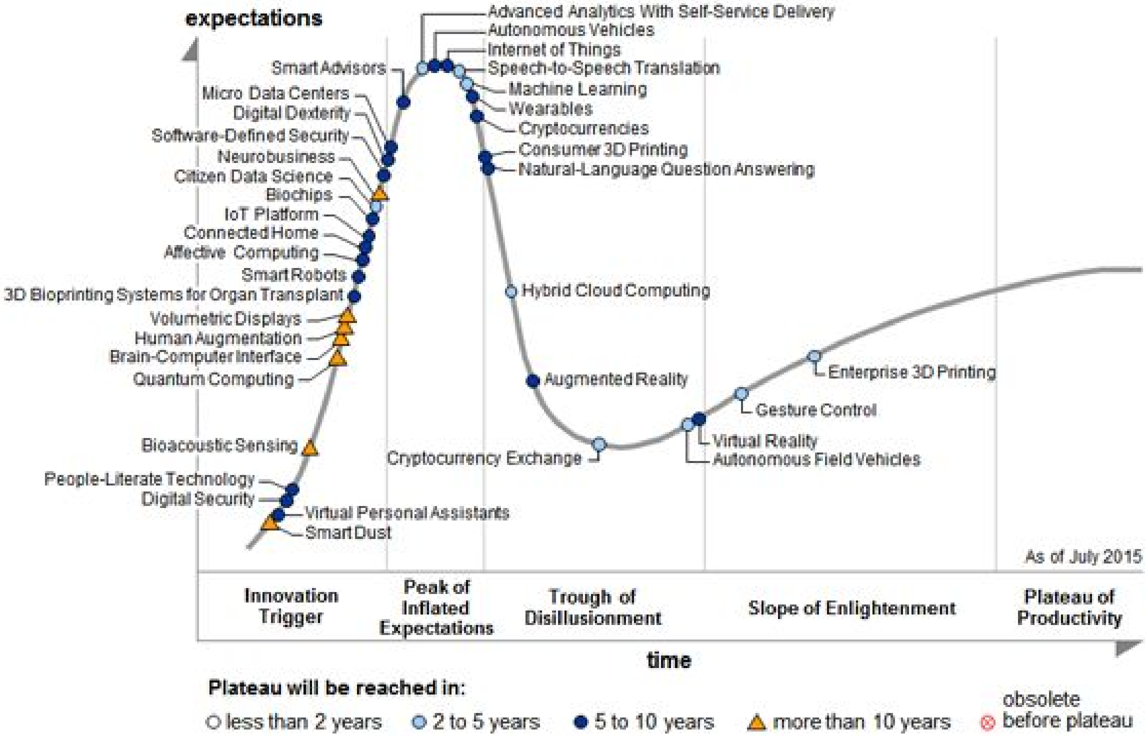
- Every year Gartner publish a list buzzwords and their trends
- Divided into 5 categories
- Divided into 5 time ranges

- Big Data is doing OK



Gartner
Cycle of Hype
2015

- Big Data is gone!?



History of Big Data

- 2014 - end of a buzzword
- 2015 - the wide deployment stage

More on Big Data

- History of Big Data
- **Frameworks and Tools**

Explosion of Frameworks and Tools

Document Management Systems
dotclear
WordPress
GetSimple
XOOPS
OpenCms
OpenKM
MODX
TWiki
typo
eXo
Alfresco
Portal
moodle
bricolage
PHP-Fusion
LogicalDOC
Orchard
umbraco
PHP-Nuke
exponent
Plone
Drupal
Data Mining
orange
mlpy
fastlab
KNIME
Neuroph
R
pentaho
SHARK
The Shogun Machine Learning Toolbox
mahout
HIVE
Column Oriented Data store
Apache HBase
accumulo
HYPERTABLE INC
MonetDB
Document Oriented Database
mongoDB
Couchbase
CouchDB
OrientDB
ArangoDB
Cluster Compute
hadoop
Spark
GridGain
Sector/Sphere
Triple Store
Apache Jena
mulgara
Apache Marmotta
SPARKLE DB
Orchestration & Coordination
Activiti
Apache AURORA
APACHE ODE
talend
OOZIE
Graph & XML Database
Neo4j
BASE X
sedna
existdb
Distributed File System
GLUSTER COMMUNITY
lustre
MooseFS
Stream Processing
Storm
GIGASPACES
elastics
Solandra
Sphi
Dist
hazel
DHT
cassandra
MySQL

Frameworks and Tools

- Google leads the way
  - Google File System
  - MapReduce
  - Bigtable
  - Chubby
  - Pregel
  - Dremel
  - Tenzing
  - Spanner
  - F1
  - Borg
  - Omega

Frameworks and Tools

- Community follows
  - Apache Hadoop HDFS
  - Apache Hadoop MapReduce
  - Apache HBase
  - Apache Zookeeper
  - Apache Pig
  - Apache Hive
  - Apache Drill
  - Apache Impala
  - Apache Giraph
  - Apache Mesos
  - Apache Spark

Key Projects

- Apache Zookeeper
- Apache Hadoop
- Apache Kafka
- Apache Mesos

Apache Zookeeper

- Make building distributed system easier
- Enabled a wide range of open-source projects
- Ground of truth for distributed system

Apache Hadoop

- Make distributed computation easier
- Expose simple API
    - Map
    - Reduce

Apache Kafka

- Make data transportation easier
- Make SOA model more reliable
- De facto data transportation framework

Apache Mesos

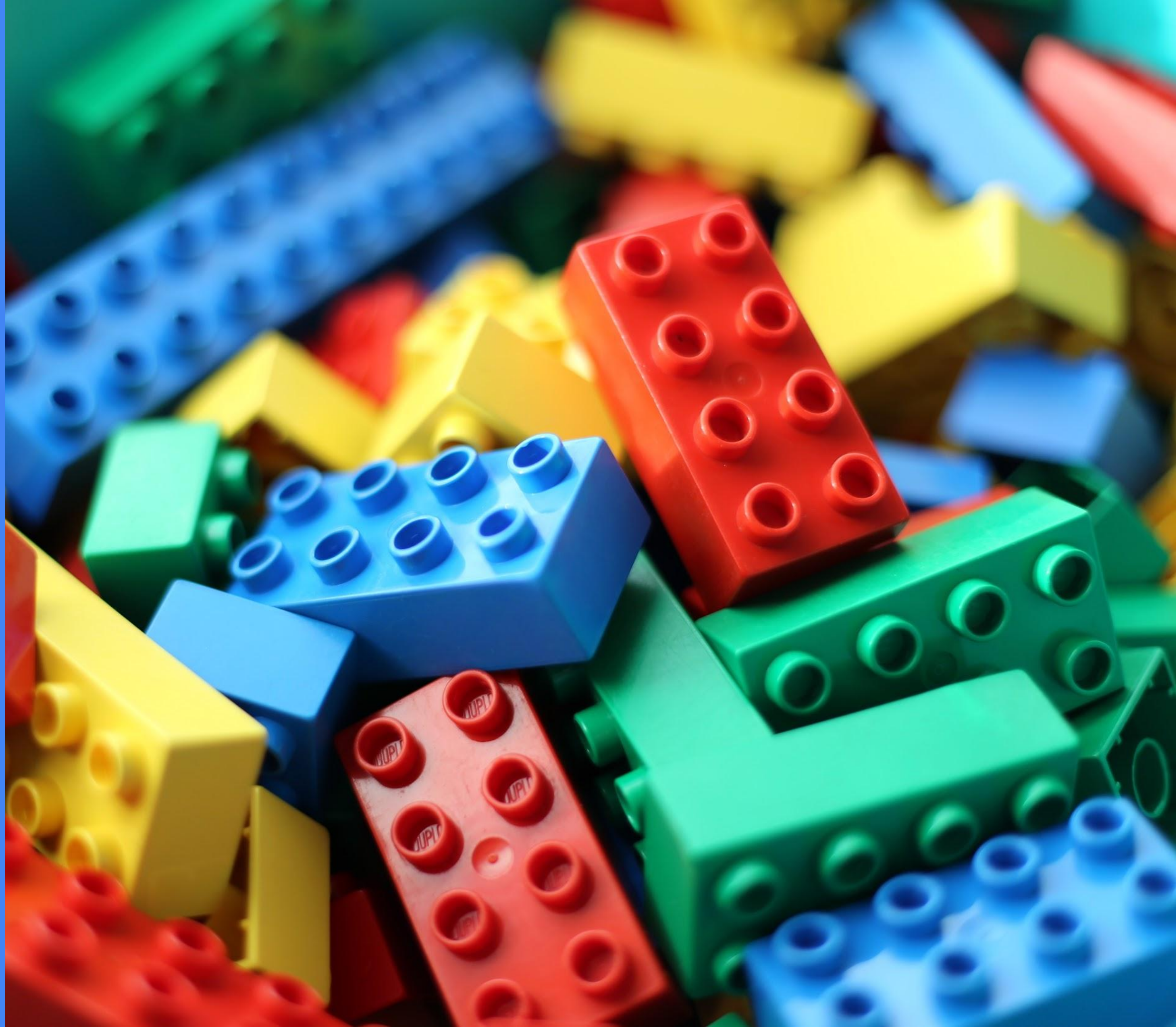
- Make distributed task scheduling easier
- Dramatically increase resource utilization

Frameworks and Tools

- Computation
    - Hadoop, Spark, Samza, Flink, Hive, Pig, Drill, etc
- Transportation
    - Kafka, Flume, Sqoop, Scribe, RabbitMQ, ZeroMQ, IronMQ, etc
- Storage
    - HBase, Cassandra, CouchDB, MongoDB, etc
- Coordination
    - Zookeeper, Consul, Etcd, Eureka, etc
- Scheduling
    - Mesos, Yarn, Oozie, etc

Explosion of Frameworks and Tools

- Don’t Panic
- Building Big Data Platform become LEGO building



Introduction to Zookeeper

Because distributed system is a ZOO

Agenda

- **Use Cases**
- What is Zookeeper
- Architecture
- Zookeeper Usage

An Example Computing Problem

Given a truck of fruits, calculate the quantity for each kind of fruit

Standalone Program

- Runs on a single machine
- Predictable
- Isolated
- Performance is limited by the machine

Image credit: Docker Inc

Distributed System/Program

- Runs on multiple machines
- Distribute the workload
- Hard to coordinate

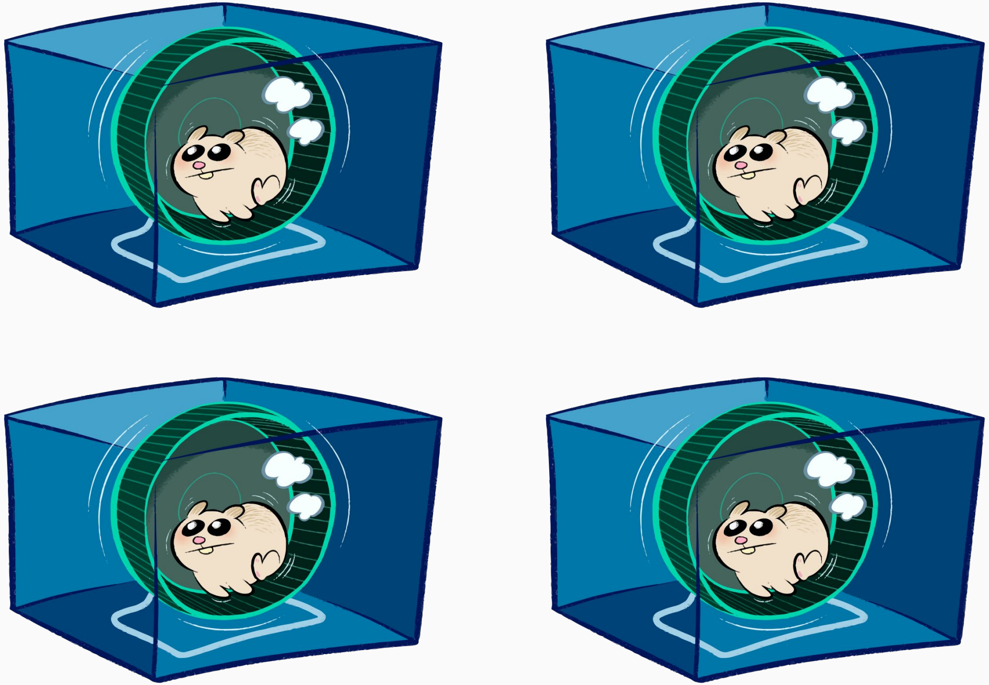


Image credit: Docker Inc

Master Slave Model

- Master knows a list of tasks
- Master in charge of distributing tasks
- Slaves in charge of executing tasks
- Slaves need to report status back to Master

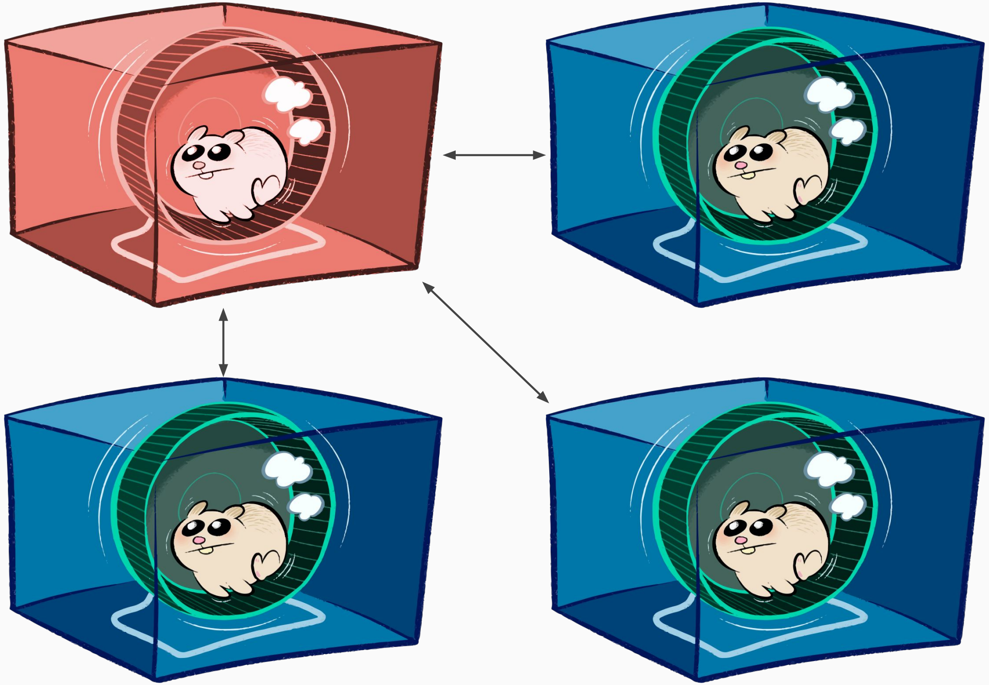


Image credit: Docker Inc

Master Slave Model Common Issues

- How to come up with a Master?
- What happens when master crash?
- What happens when worker crash?
- What if master and worker cannot communicate?

Master Slave Model Common Tasks

- Master Election
  - The process of deciding who is the master
- Crash Detection
  - The master need to detect when workers crash
- Group Membership
  - The master must learn who is available for tasks
- Metadata Management
  - All the nodes must be able to reliably store/retrieve status

Agenda

- Use Cases
- **What is Zookeeper**
- Architecture
- Zookeeper Usage

What is Zookeeper

- An open source distributed system that provides
  - Strong consistency, ordering, and durability guarantees
  - The ability to implement typical synchronization primitives
  - A simpler way of dealing with concurrency

- Inspired by Google Chubby
- Developed in Yahoo using Java

Agenda

- Use Cases
- What is Zookeeper
- **Architecture**
- Zookeeper Usage

Architecture

- **Coordinate Strategy**
- Concepts
- Internal

Ways to Coordinate

- Message passing
  - Processes exchange messages directly through a network
- Shared storage
  - Read or write to shared storage

Coordinating with Messages

- Communicate to each other by passing messages

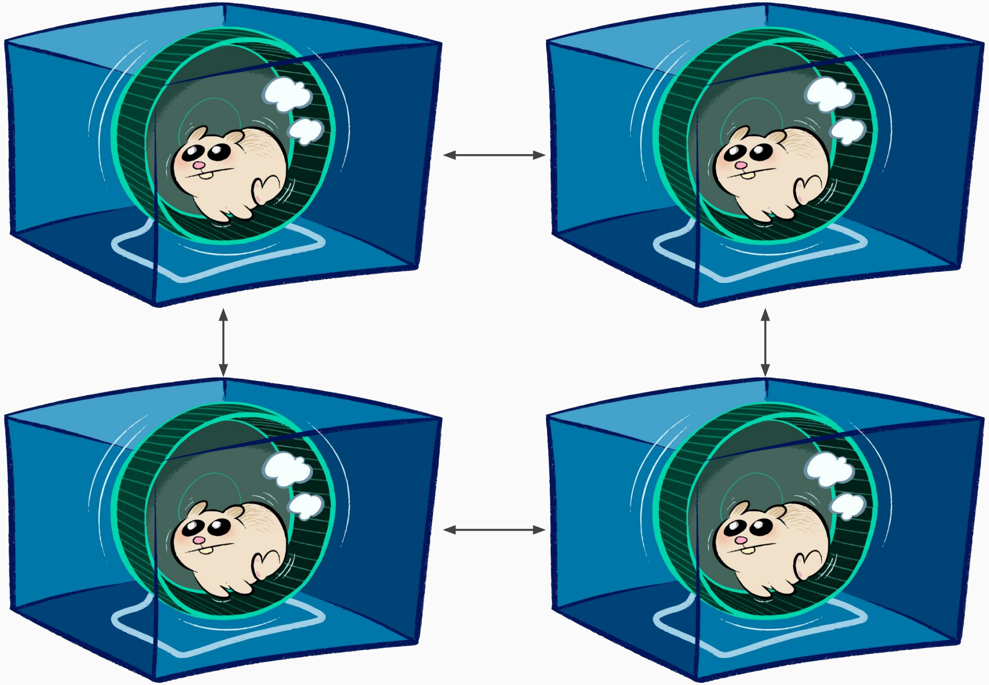


Image credit: Docker Inc

Coordinating with Message Passing

- Network Delays
  - Messages might get delayed arbitrarily
- Speed of Processing
  - Some machines might process things faster
- Clock Difference
  - The time on the machines are different

- Cannot tell the difference between real crash or network issue

A Split-brain Example

- In a system, we have one master, one backup master, and many workers
- Part of the workers are having trouble to reach master
- Connect with backup master instead

Coordinate with Shared Storage

- More straightforward
- Still rely on network to transfer data

- Zookeeper use this model to implement coordination

Architecture

- Coordinate Strategy
- **Concepts**
- Internal

Data Tree

- Organized in hierarchical structure
- Similar to file system

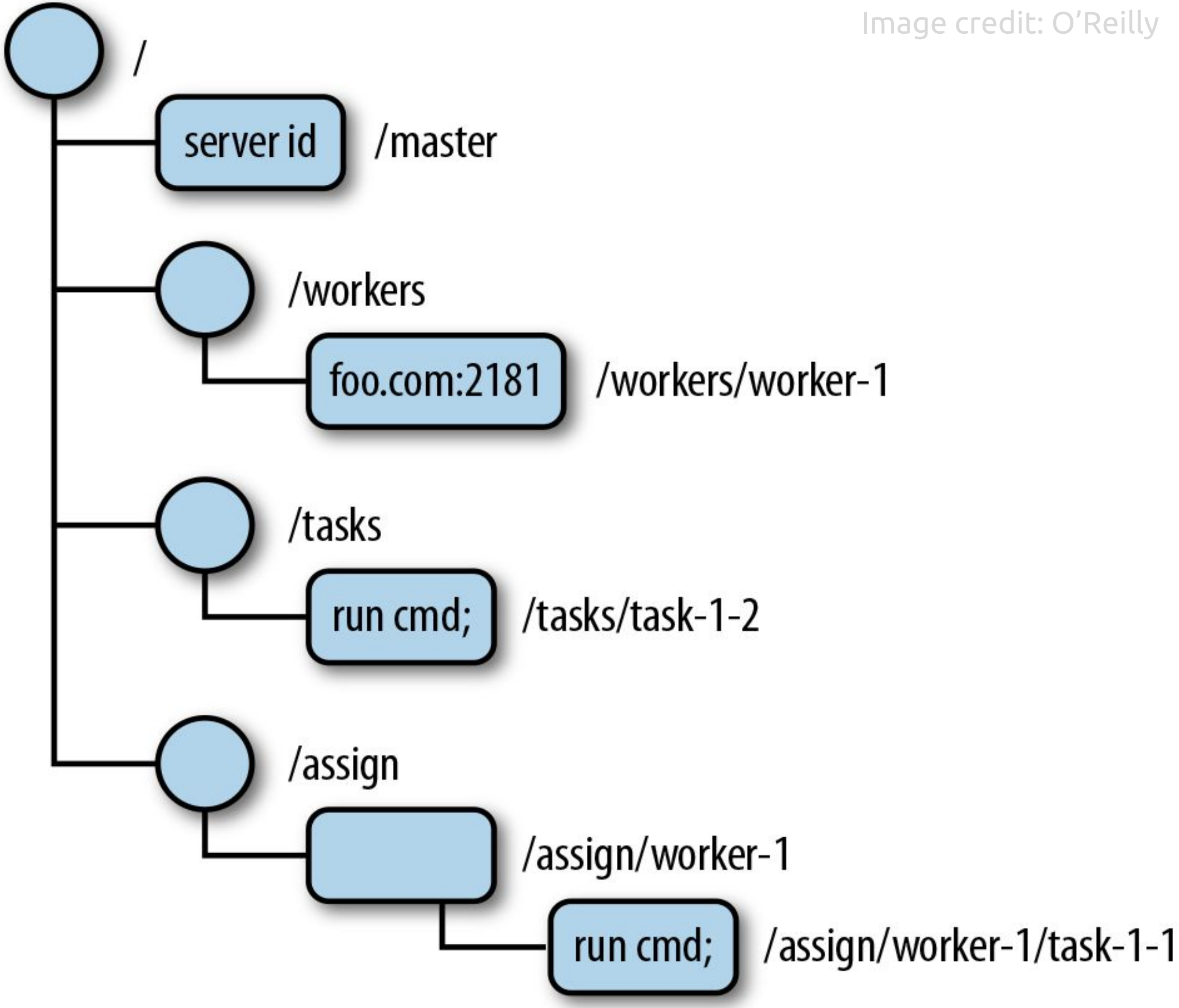


Image credit: O'Reilly

API

API Name	Usage
`create /path data`	Create a znode /path with data
`delete /path`	Delete znode /path
`exists /path`	Check if /path exists
`setData /path data`	Set znode /path to data
`getData /path`	Get the data in /path znode
`getChildren /path`	Return a list of children under /path

znode

- Basic unit for Data Tree
- Persistent znode
  - Data is persisted unless **`delete`** is called
- Ephemeral znode
  - Node is deleted if the client created it loses connection to zookeeper
- Sequential znode
  - Zookeeper will assign sequence number and append it to the path

Watcher

- Help client to know changes to znodes
- Avoid race conditions in polling
- Notifications are one-time operation

Watcher

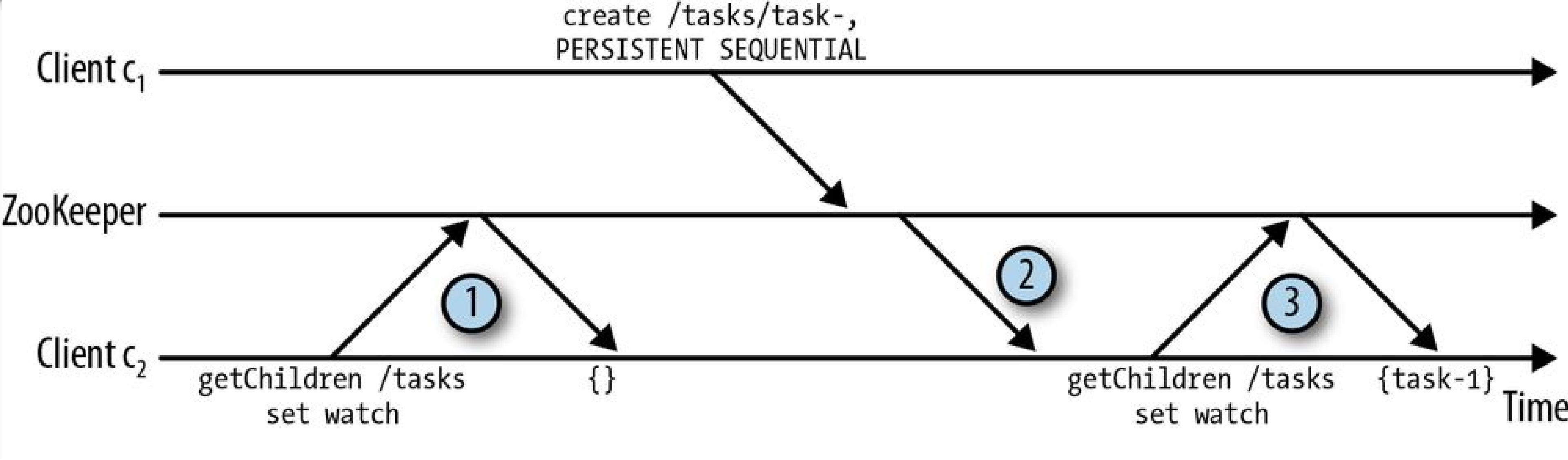


Image credit: O'Reilly

Architecture

- Coordinate Strategy
- Concepts
- **Internal**

Client - Server Interaction

- Connection
  - TCP connection
  - Client only connects to server with newer/equal state
  - Configurable session timeout
- Read
  - Read can be read from any of the servers
- Write
  - Write requests will be forwarded to the leader of zookeeper cluster

zxid

- A 64-bit integer id
- Generated when there is a new update of znode

Leader Election within Zookeeper

- Zookeeper cluster AKA Ensemble
- Server has following mode
  - Leader
  - Follower
  - Observer
- Server has following state
  - LOOKING
  - LEADING
  - FOLLOWING

Leader Election within Zookeeper

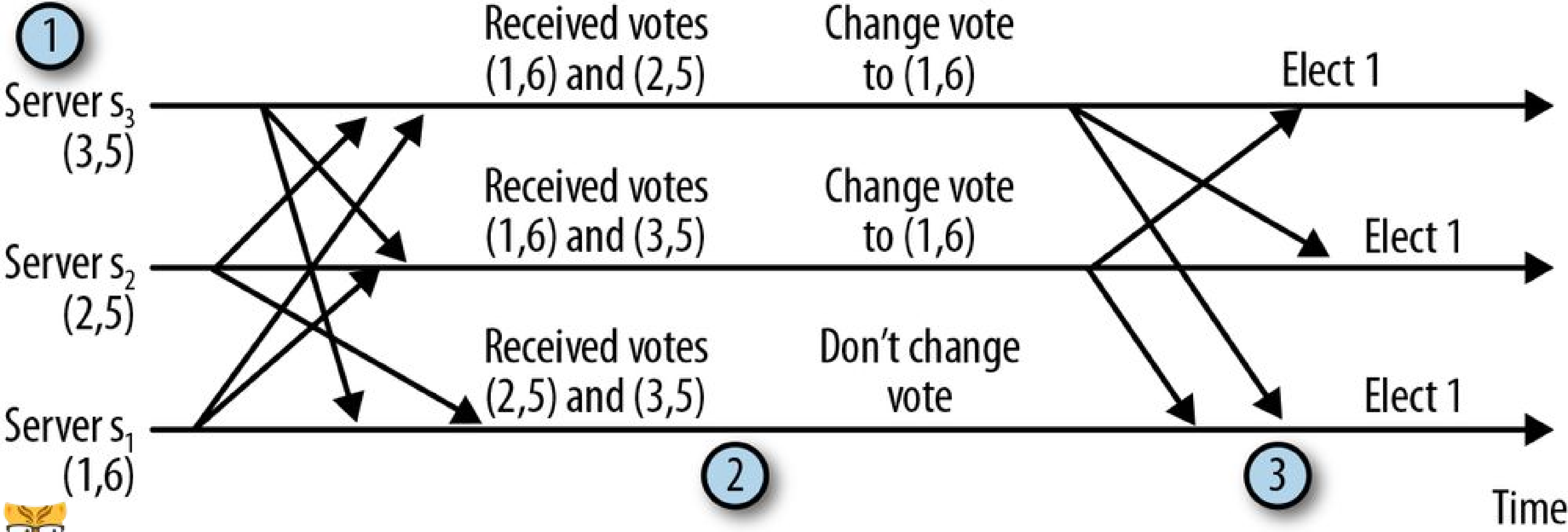


Image credit: O'Reilly

State Replication within Zookeeper

- Follow a consensus protocol called Zab
- Leader
  - Receives Write requests
  - Convert requests into transaction
  - Leader send PROPOSAL message to all follower
  - Once receiving acknowledges from a quorum, leader sends COMMIT message

State Replication within Zookeeper

- Follower
  - Received PROPOSAL message from leader
  - Check if the leader is the correct leader
  - Check if the transaction is in correct order
  - Send acknowledgement back to leader
  - Received COMMIT message from leader
  - Apply change to the Data Tree

Agenda

- Use Cases
- What is Zookeeper
- Architecture
- **Zookeeper Usage**

Zookeeper Usage

- Master Election
- Crash Detection
- Group Membership
- Metadata Management

Master Election

- Have all the process go and create sequential ephemeral znode
- Znode with smallest sequence number is the leader
- Setup watcher for changes

Crash Detection

- Have slaves create ephemeral znodes
- Setup watcher on the znodes

Group Membership

- The members need to create ephemeral znodes under group node

Metadata Management

- Store metadata in persistent znode
- Use persistent znode as source of truth

Similar Systems

- Consul (https://www.consul.io/)
  - Use Raft for Consensus
- Etcd (https://coreos.com/etcd/)
  - Use Raft for Consensus
- Eureka (https://github.com/Netflix/eureka)

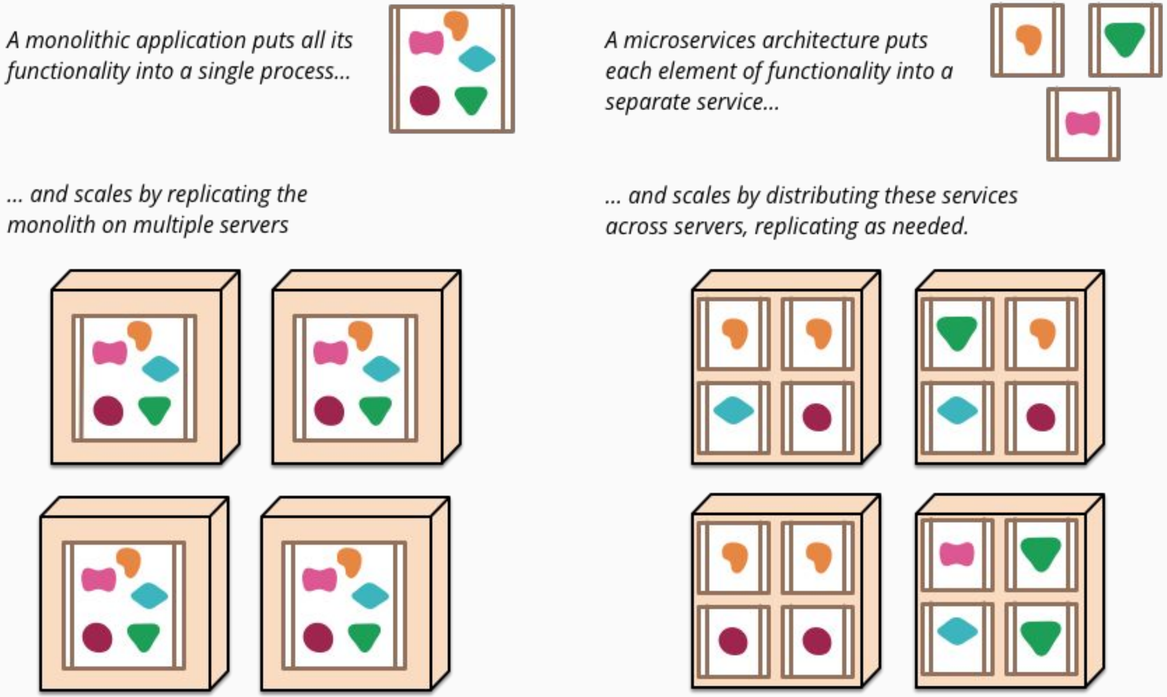
Introduction to Kafka

A High-throughput Distributed Messaging System

Agenda

- **Use Cases**
- What is Kafka
- Architecture
- Kafka Usage

Microservice Architecture



A monolithic application puts all its functionality into a single process...

... and scales by replicating the monolith on multiple servers

A microservices architecture puts each element of functionality into a separate service...

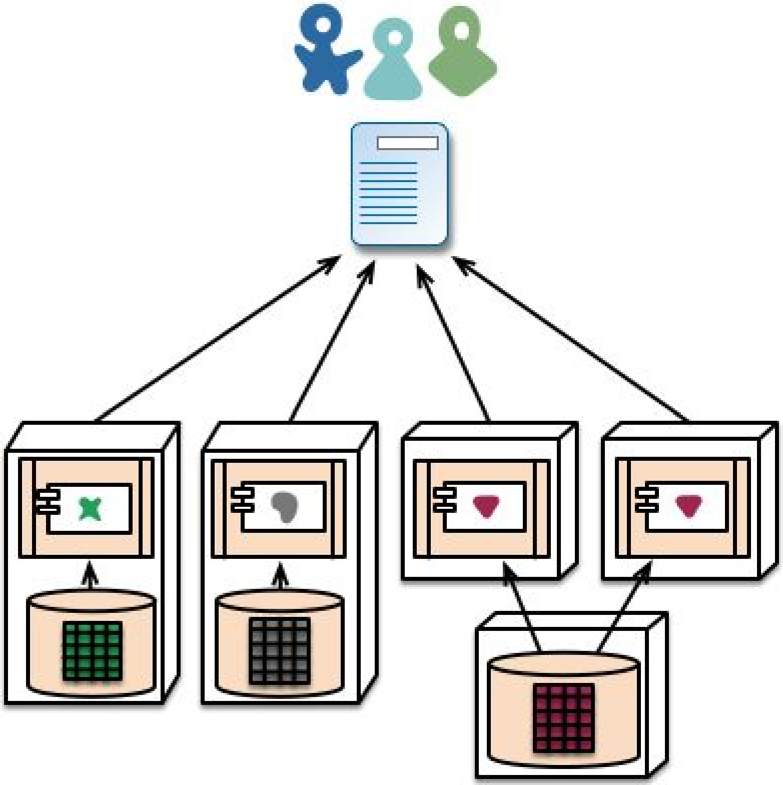
... and scales by distributing these services across servers, replicating as needed.

Image credit: Martin Fowler

Microservice Architecture

- Organize around business capabilities
- Decentralized data management
- Better deployment infrastructure

Image credit: Martin Fowler

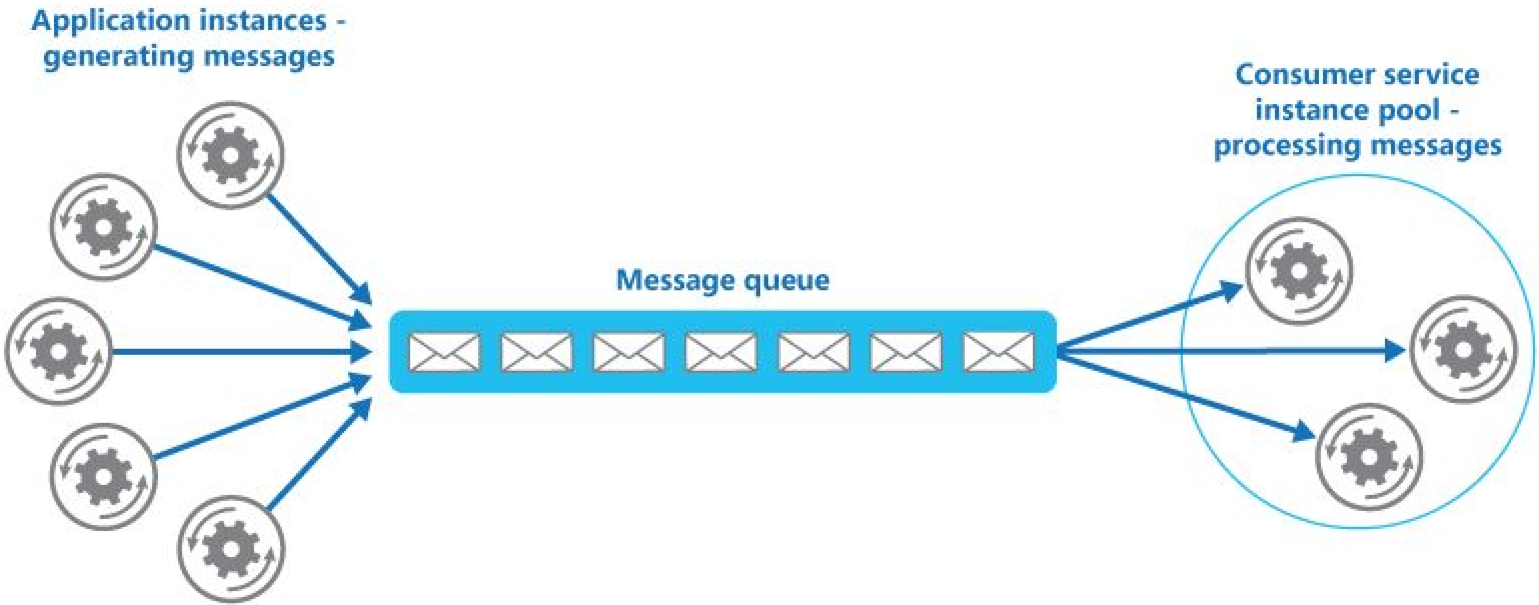


microservices - application databases

How Does Microservices Communicate

- Sync
  - RESTful API
  - RPC frameworks
- Async
  - Drop a message and back to work

Message Queue

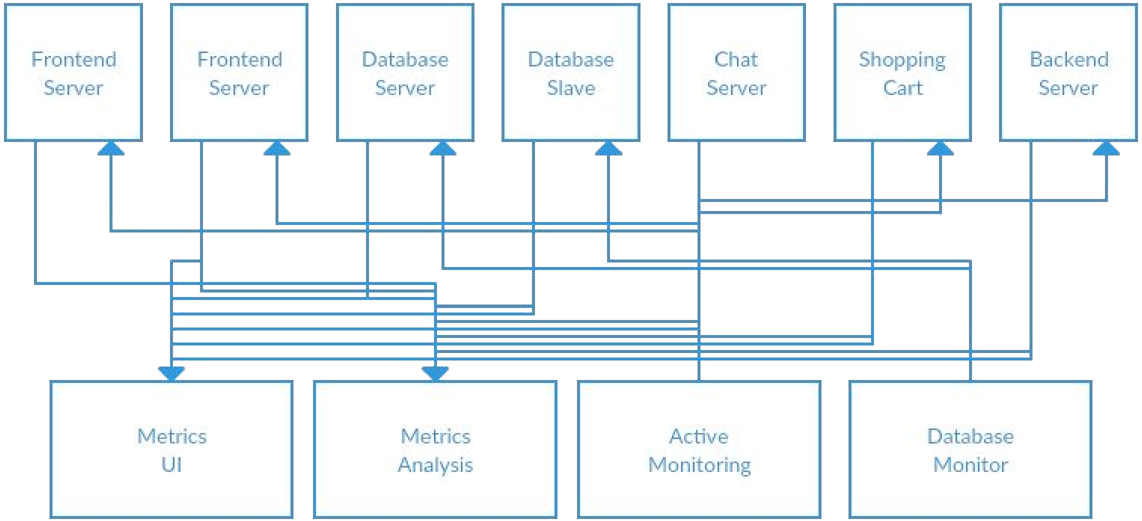

Application instances - generating messages
Message queue
Consumer service instance pool - processing messages

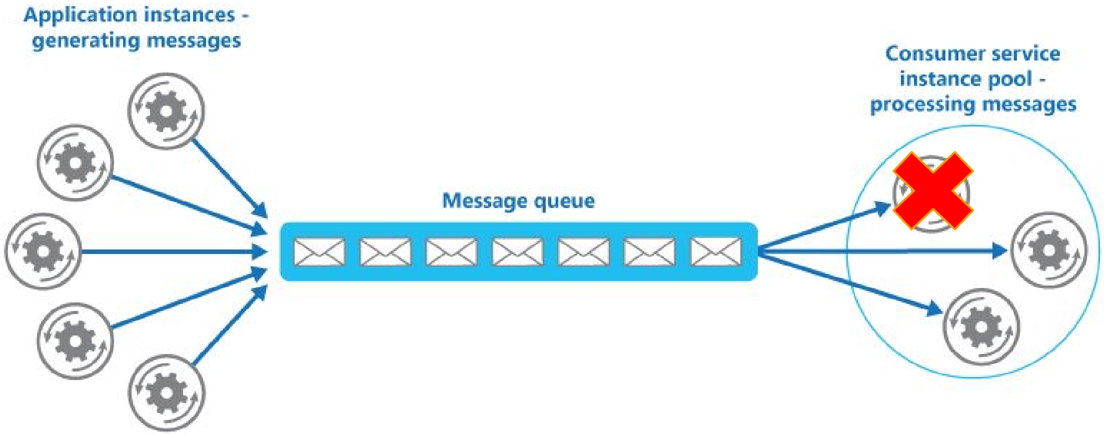
Message Queue Benefits

- Service Decoupling
- Increase Scalability
- Data Redundancy
- Deal with Peak Traffic
- Buffer Cushion for Failed Components

Message Queue Issues

- Spaghetti of queues
- Skip Messages
- Brokers need to record client positions




Application instances - generating messages
Message queue
Consumer service instance pool - processing messages

Agenda

- Use Cases
- **What is Kafka**
- Architecture
- Kafka Usage

What is Kafka

- An open source distributed messaging system
  - Fast - hundreds MB/s from thousands of clients
  - Scalable - easily scale up and down without downtime
  - Durable - Messages are persisted on disk to prevent data loss

- Developed in LinkedIn using Scala

Agenda

- Use Cases
- What is Kafka
- **Architecture**
- Kafka Usage

Architecture

- **Design Strategy**
- Concepts
- Internal

Pull vs Push

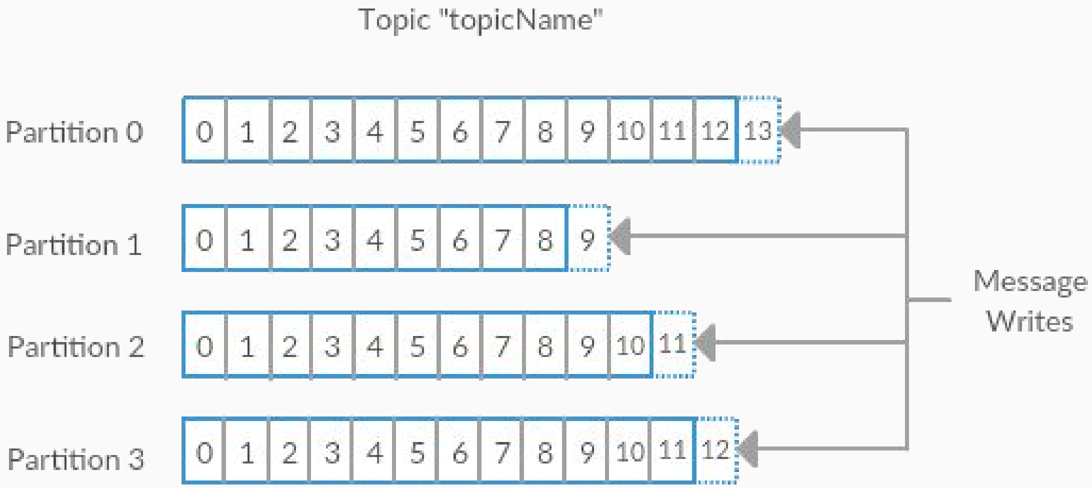
- Push model
  - High throughput
  - Complex server logic
- Pull model
  - Simple server logic
  - Reply feature

Architecture

- Design Strategy
- **Concepts**
- Internal

Topic and Partition

- Messages are organized logically into topics
- Physically divided into Partitions



Offset

- A incremental sequence number
- The position of a message in a partition

API

API Name	Usage
`publish topic data`	Publish data onto a topic
`consume topic offset`	Consume from a topic

Producer

- Producer in charge of sending messages to Kafka broker

Consumer

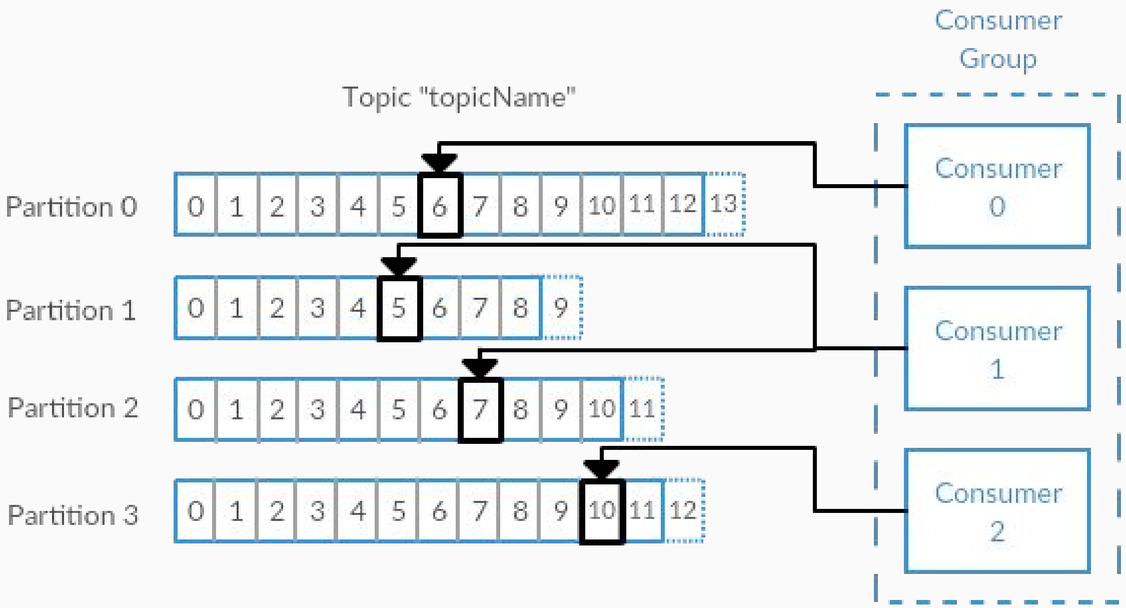
- Consumer pull messages from Kafka broker
- Can use offset to target a specific message on a partition
  - By default point to the latest offset
  - Can set the offset to an older value to read old data
- Consumer maintain message state
- No ACK

Consumer Group

- A group of consumers
- Mapped to partitions
    - Messages in certain partition can only be consumed by corresponding consumer

Consumer Group

- A group of consumers
- Mapped to partitions
  - Messages in certain partition can only be consumed by corresponding consumer



Architecture

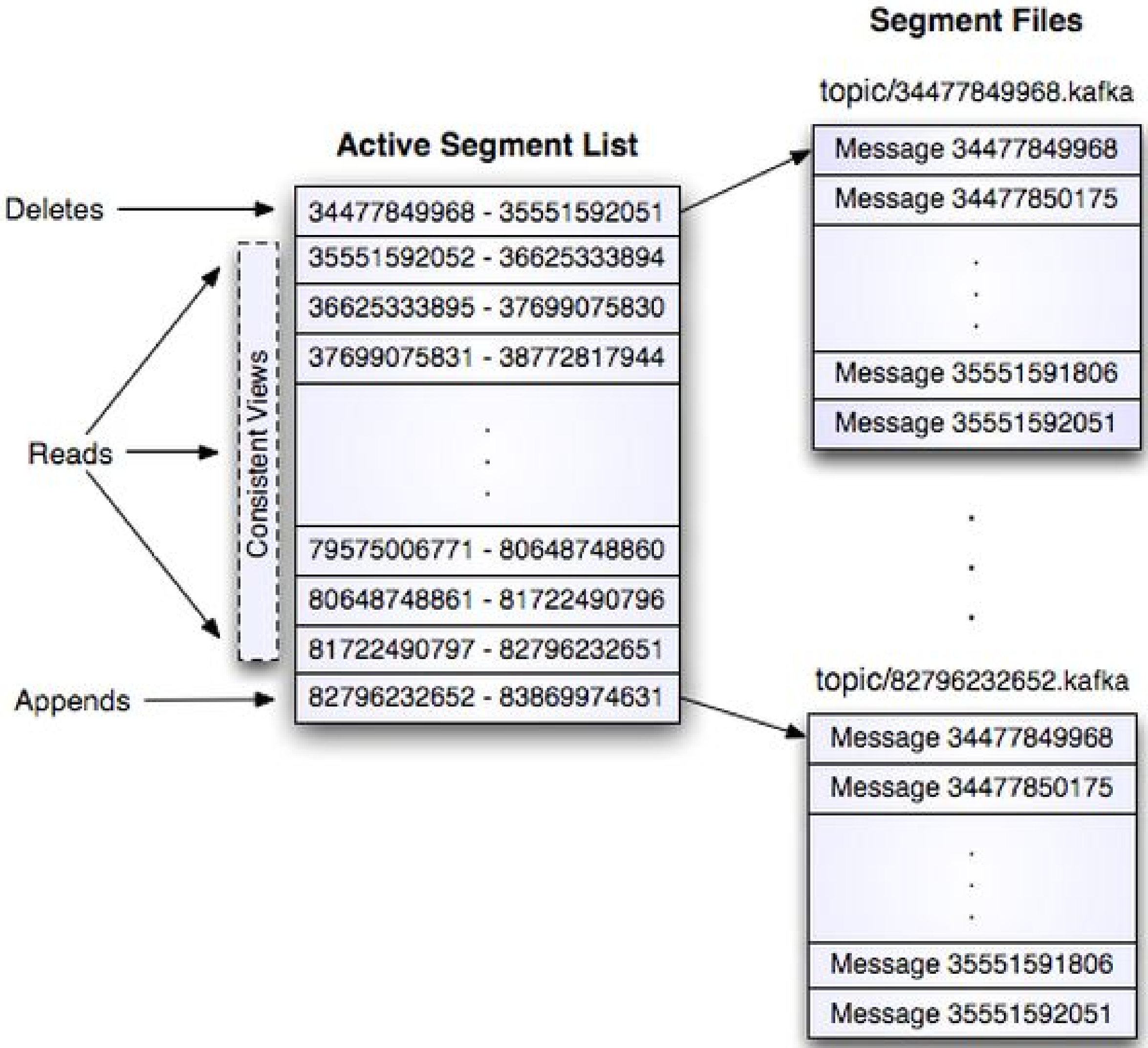
- Design Strategy
- Concepts
- **Internal**

Log File Format

- One partition is a physical folder
- One message:
  - offset
  - Message length
  - Magic value, 1 byte
  - CRC value, 4 bytes
  - Payload, N bytes

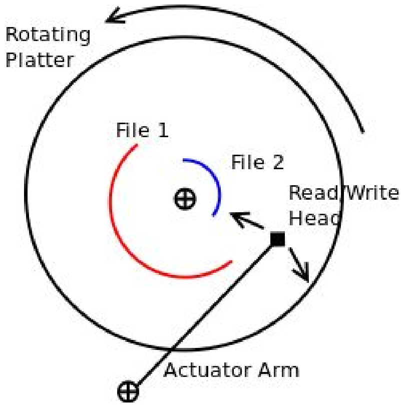
Log File Format

- Active Segment List maintains mapping to segment files
- Log entries in segment files



IO Optimization

- Append-only writing
- Reads do not block writes
- Super fast writing speed because one partition is one log file



IO Optimization - zerocopy

- OS reads data from disk into pagecache in kernel space
- Application reads data from kernel space into user space
- Application writes data back to kernel space into socket buffer
- OS copies data from socket buffer to NIC buffer

- zerocopy copies data into pagecache only once and reuse

Data Replication inside Kafka

- Producer write through Partition Leader
- Partition Leader write the message into local disk
- Partition Follower pull from Partition Leader
- Once Leader received ACK from all the

ISR (in-sync Replica)

- A nice balance between sync and async copy
- Configurable lag behind
  - If one replica is too slow, will be removed from ISR
- Follower can batch read from Leader

Agenda

- Use Cases
- What is Kafka
- Architecture
- **Kafka Usage**

Kafka Usage

- Log Aggregation
- Real-time Data Analysis

Log Aggregation

- Forward all the logs into specific Kafka topic
- Setup Consumer/Consumer Group on the topic and forward to indexing service
- Query on the data through Kibana, etc

Read-time Analysis

- Forward interested data into Kafka topic
- Integrate with Spark/Samza for steaming processing

Similar Systems

- RabbitMQ (https://www.rabbitmq.com/)
    - Erlang, based on AMQP
- ZeroMQ (http://zeromq.org/)
    - Middleware less
- Redis (http://redis.io/)
    - C/C++, can be used as lightweight queue
- ActiveMQ (http://activemq.apache.org/)
    - Java, based on AMQP
- SQS (https://aws.amazon.com/sqs/)
    - An AWS service, the visibility timeout feature is awesome

Introduction to Cassandra

Manage massive amounts of data, fast, without losing sleep

Agenda

- **Use Cases**
- What is Cassandra
- Architecture
- Cassandra Usage

An Example Storage Problem

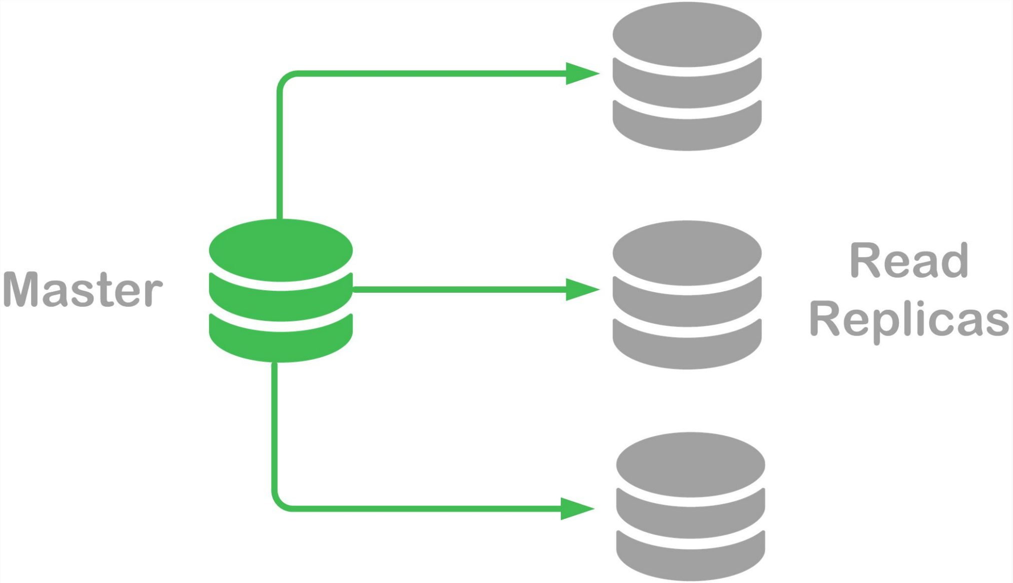
- We are creating a system for people to store:
  - First name
  - Last name
  - Phone number

- Initially we can just go with one simple table

An Example Storage Problem

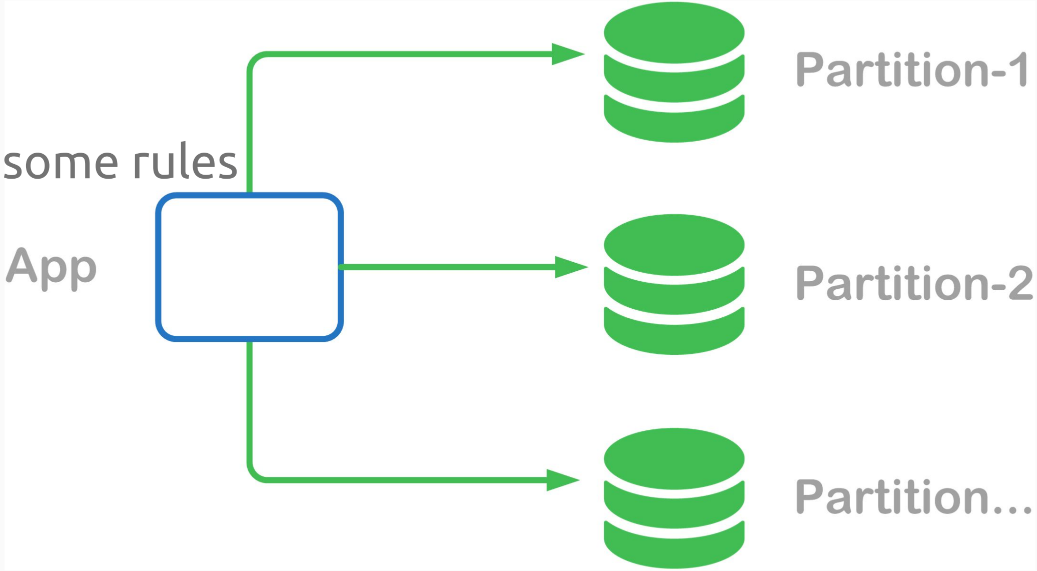
- Your application is getting popular, you noticed that people are getting slow read

- Read/Write separation



An Example Storage Problem

- Your app is so popular you are hitting the storage limit of database
- Shard/Partition data based on some rules



Agenda

- Use Cases
- **What is Cassandra**
- Architecture
- Cassandra Usage

What is Cassandra

- An open source distributed storage system that provides
  - High availability
  - No single point of failure

- Inspired by Amazon DynamoDB
- Developed in Facebook using Java

Agenda

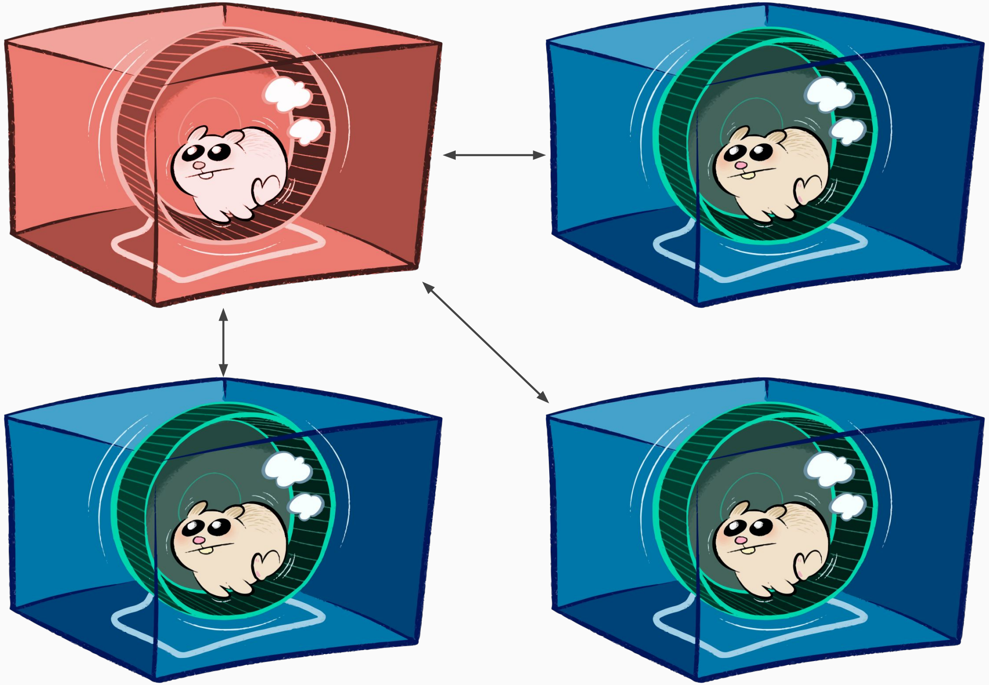
- Use Cases
- What is Cassandra
- **Architecture**
- Cassandra Usage

Architecture

- **Design Strategy**
- Concepts
- Internal

Revisit Master Slave Model

- Single point of failure
- Even backup master might fail
- Capacity depends on master



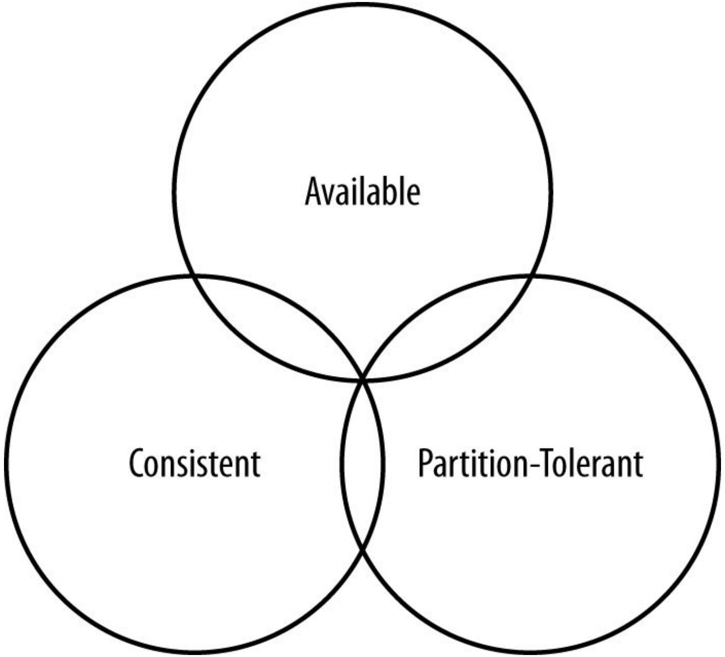
Gossip Protocol

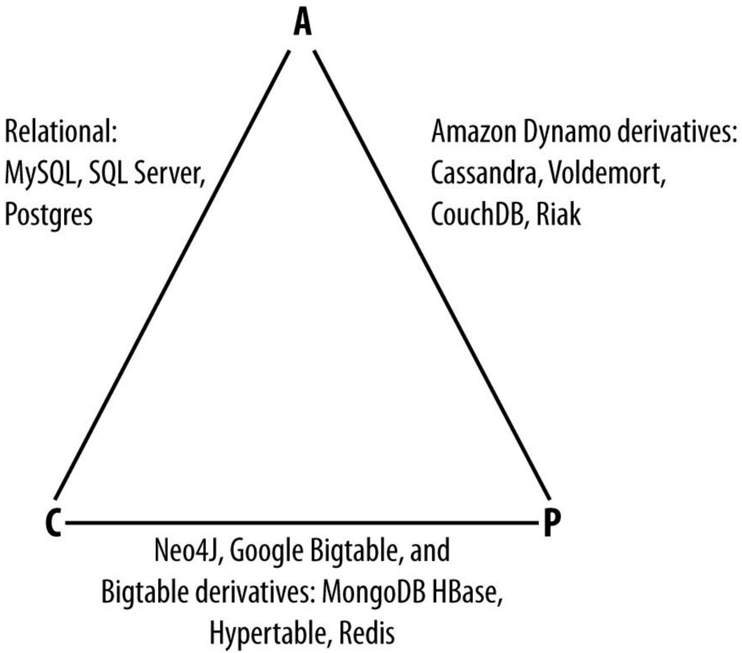
- Runs every second
- Choose a random node to gossip with
- Use accrual failure detection to determine if a node is down
  - For example, increase the convict threshold on cloud services

Tunable Consistency

- Consistency means whether read always return the most recently written value
- In other systems, the consistency level is defined by the protocol

CAP Theorem





Relational:
MySQL, SQL Server,
Postgres

Amazon Dynamo derivatives:
Cassandra, Voldemort,
CouchDB, Riak

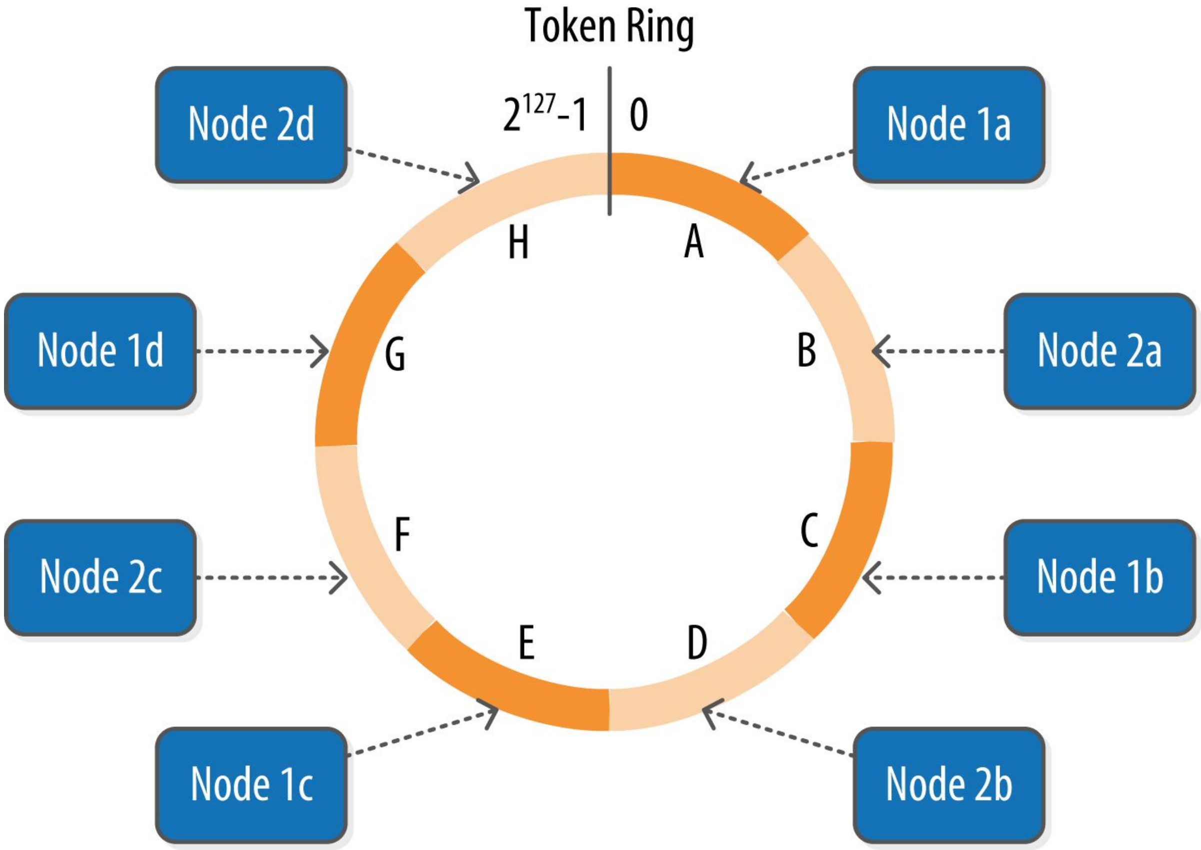
Neo4J, Google Bigtable, and
Bigtable derivatives: MongoDB HBase,
Hypertable, Redis

Architecture

- Design Strategy
- **Concepts**
- Internal

Ring

- In a Cassandra cluster, data is assigned to nodes as if they form a ring of tokens



Partitioner

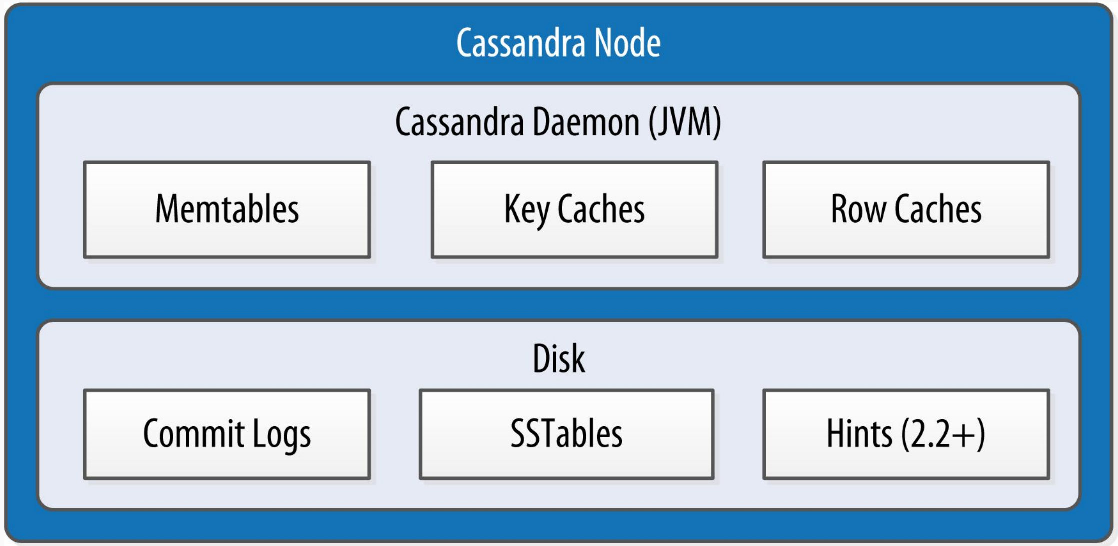
- A hashing algorithm to determine how data is distributed across the cluster
- By default murmur3 hashing algorithm is being used

Architecture

- Design Strategy
- Concepts
- **Internal**

Internal
Data Structure

- Commit log
- memtable
- SSTable



Commit log

- Crash-recovery mechanism
- All write operation is immediately written to commit log
- Will not count if no commit log is written

memtable

- Value will be added to memtable after commit log
- In-memory store to speed up operations

SSTables

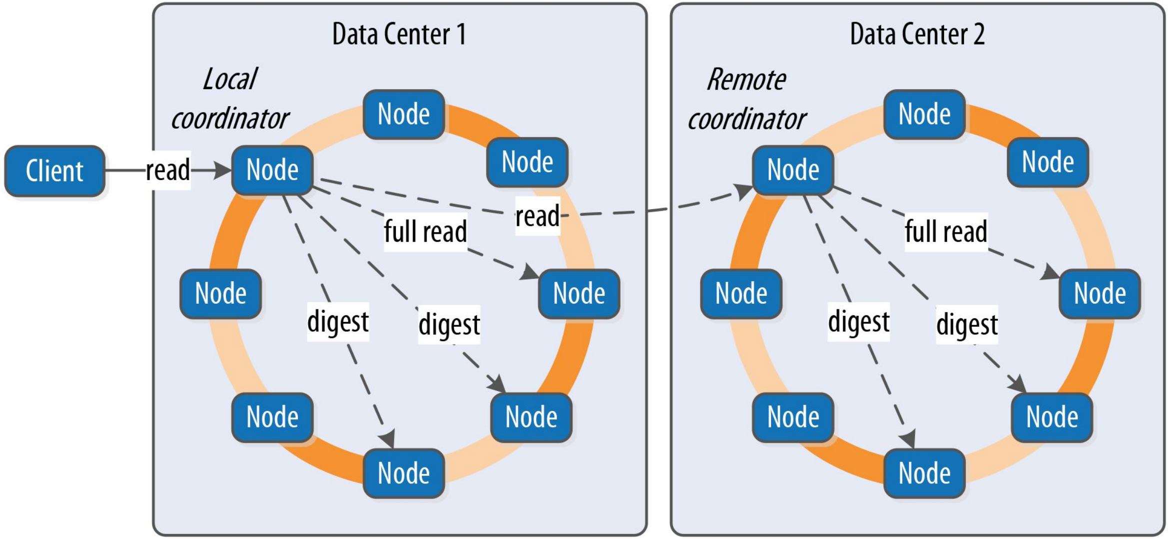
- Content of memtable gets written to SSTable after memtable is full
- Immutable cannot be changed
- Changes are appended
- Sequential write to disk

Compaction

- Merge of SSTables
- New merged data is sorted as well
- Reduce number of seeks

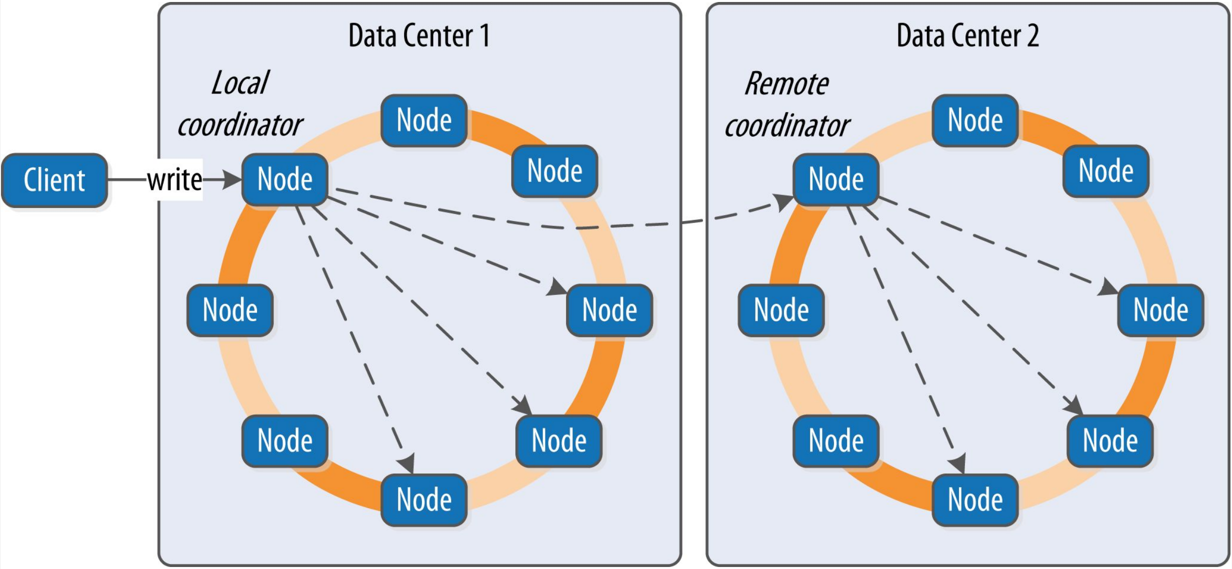
Read Operations inside Cassandra

- Client can contact any node to read



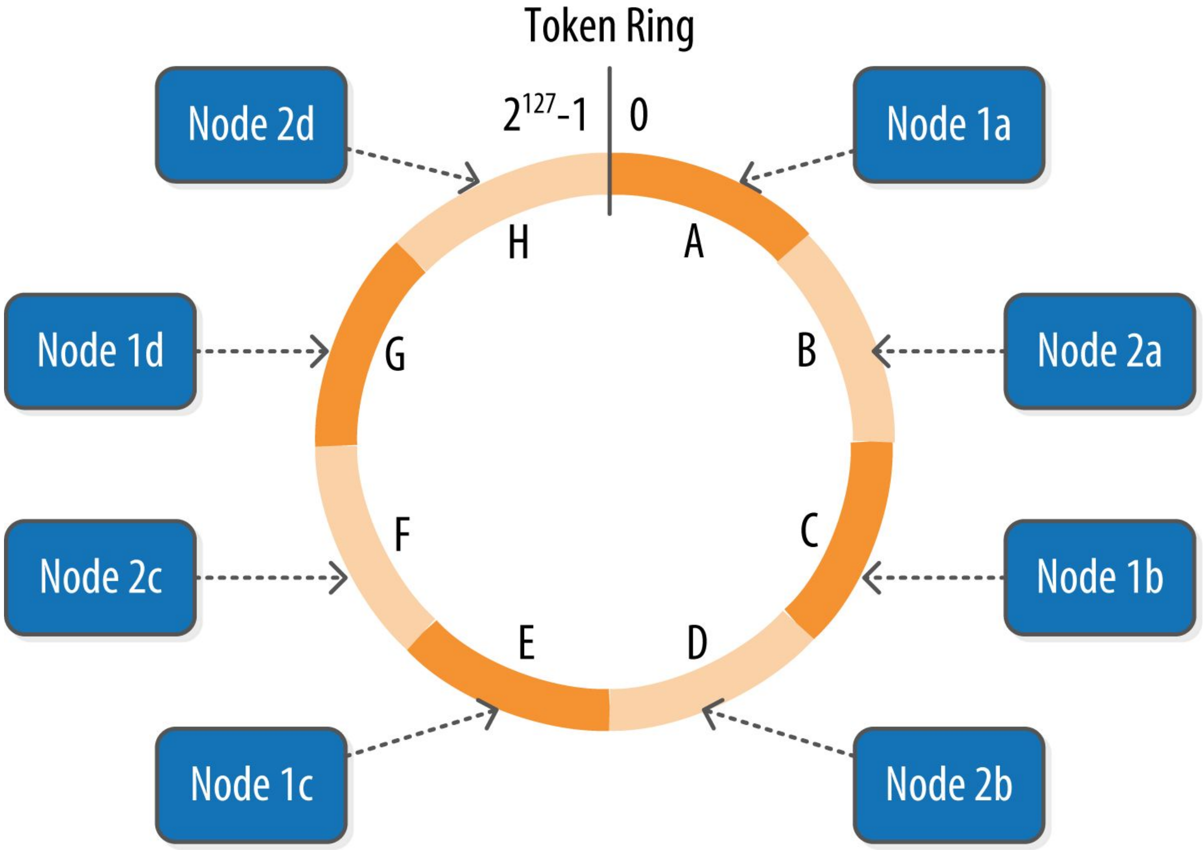
Write Operations inside Cassandra

- Client can contact any node to Write



Data Replication Inside Cassandra

- A node can serve as replica for other ranges
- If nodes goes down replica will serve the request



Agenda

- Use Cases
- What is Cassandra
- Architecture
- **Cassandra Usage**

Cassandra Usage

- General Data Storage
- Time-series Data Storage
- TTL Data Storage

General Data Storage

- You can use Cassandra cluster as your primary data persistence layer
- Easy to operate

Time-series Data Storage

- Data is sorted and written sequentially to disk
- Perfect for retrieving data and filter range
- Fast access due to small disk seeks

TTL Data Storage

- Some data can be discarded after some time
- With Cassandra TTL on data, this feature is easy to implement

Similar Systems

- HBase (https://hbase.apache.org/)
  - Java, inspired by Google BigTable
- MongoDB (https://www.mongodb.com/)
  - C++, C, JavaScript, document based database
- CouchDB (http://couchdb.apache.org/)
  - Erlang, document based database

Q&A