Big Data Engineering

The Hitchhikers Guide

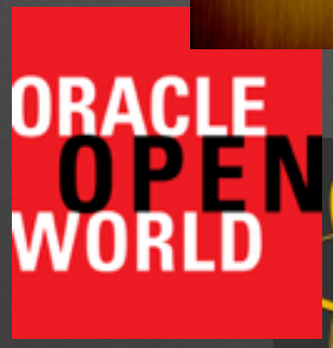
Abdelmonaim Remani

@PolymathicCoder

PolymathicCoder@gmail.com

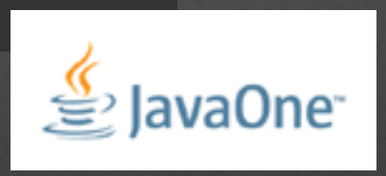
@PolymathicCoder

JavaZone



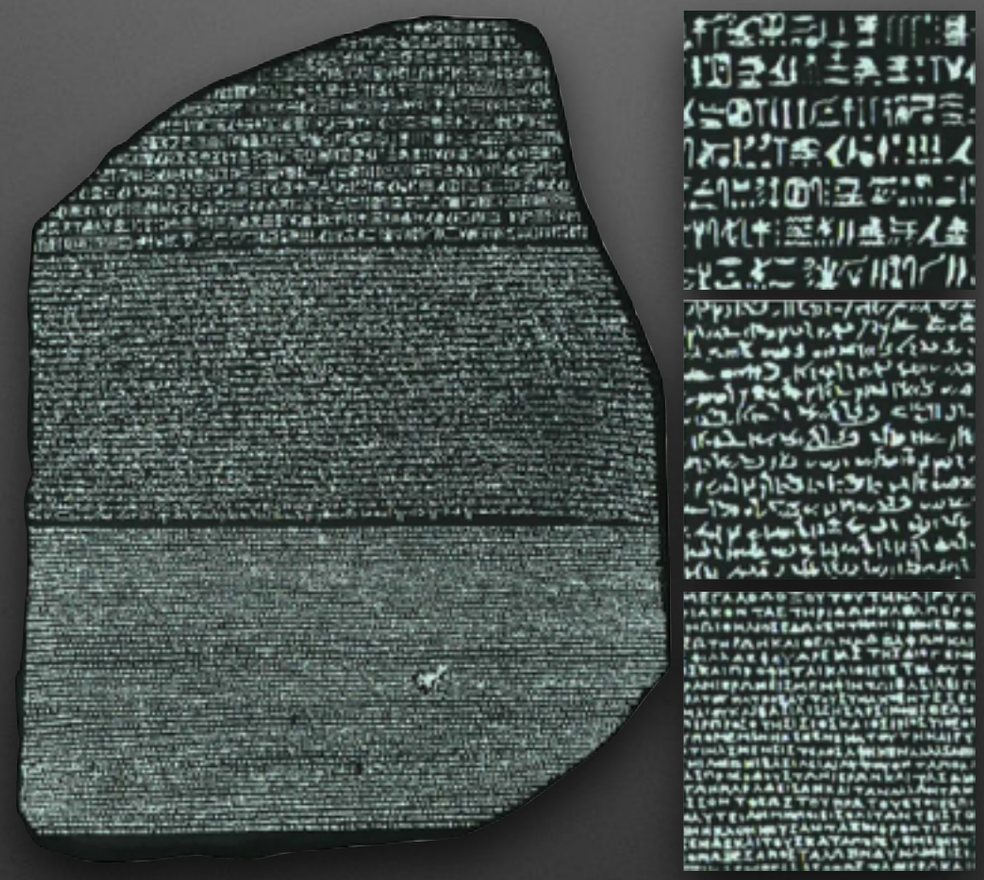
ÜBERCONF

ULTIMATE JAVA / AGILITY EVENT



Database?

A system allowing for storing data and consistently reading back it in a later time



The file system is the simplest database / data store

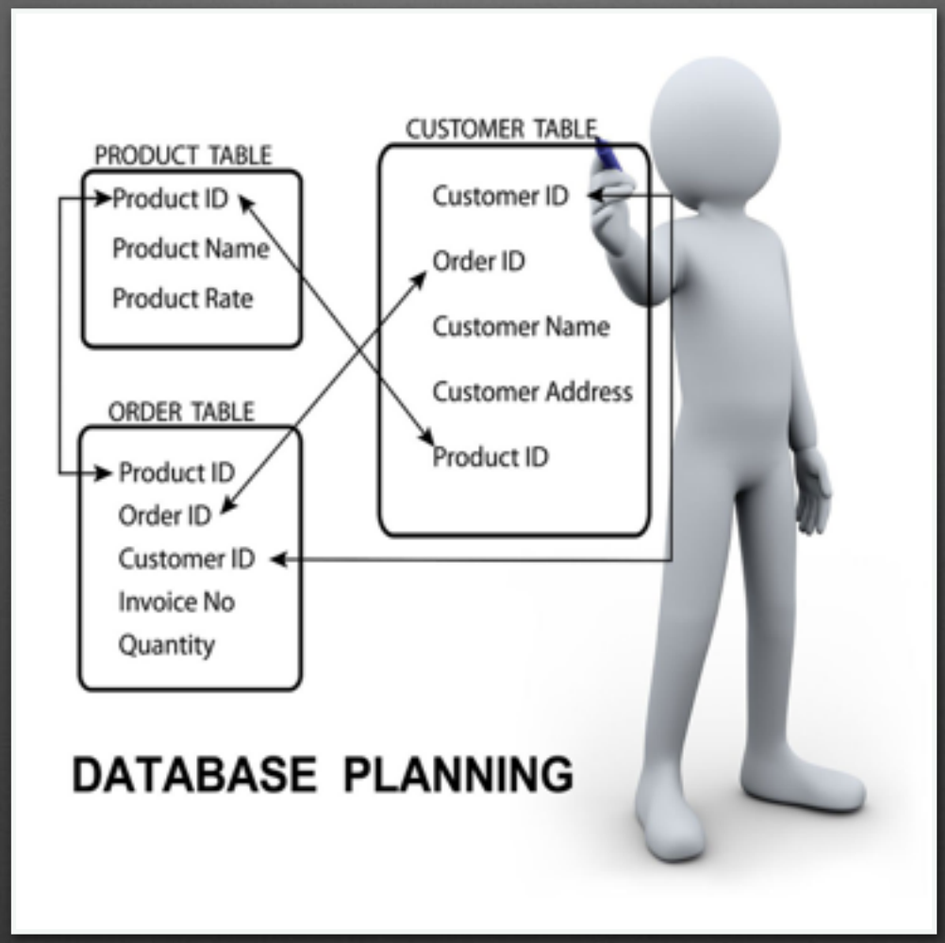
Flat-File Databases

Consistency ▶ No Concurrent Access

No Concurrent Access
▶ Data Not Always Available!

Relational Databases

Consistency and Availability ▶ No Redundancy



No Redundancy ▶ Normalization

Relational Databases

- Increased Application Complexity
  - OOP
  - O/R Mismatch
    - ORM Frameworks
- More Data & Complex Queries
  - Decreased Performance
    - OLAP/OLTP Schism
    - Vertical Scaling

Relational Databases

- Even More Data & More Complex Queries
  - Decreased Performance
  - Horizontal Scaling

(Master/Slave & Data Sharding)

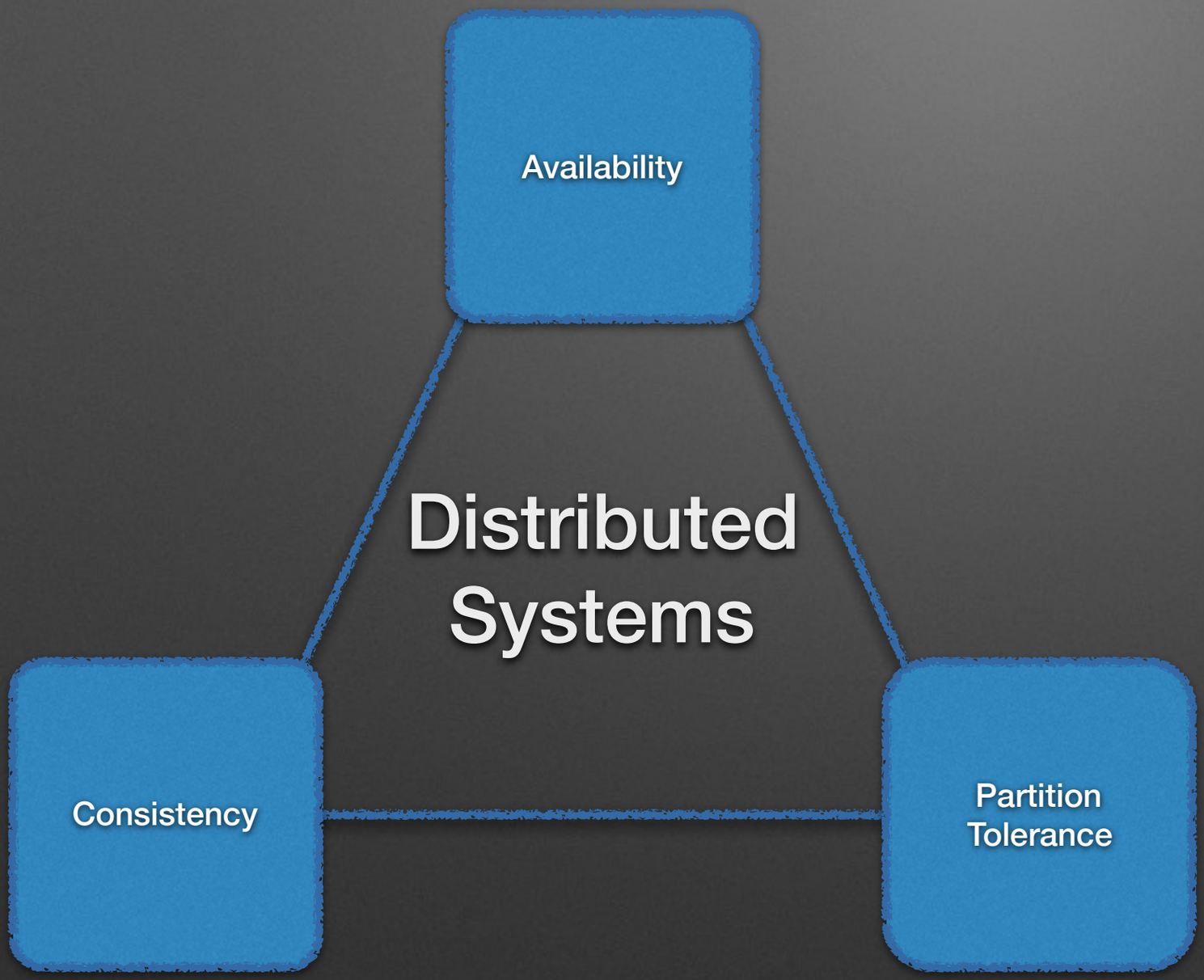
RDBMS as a Distributed System

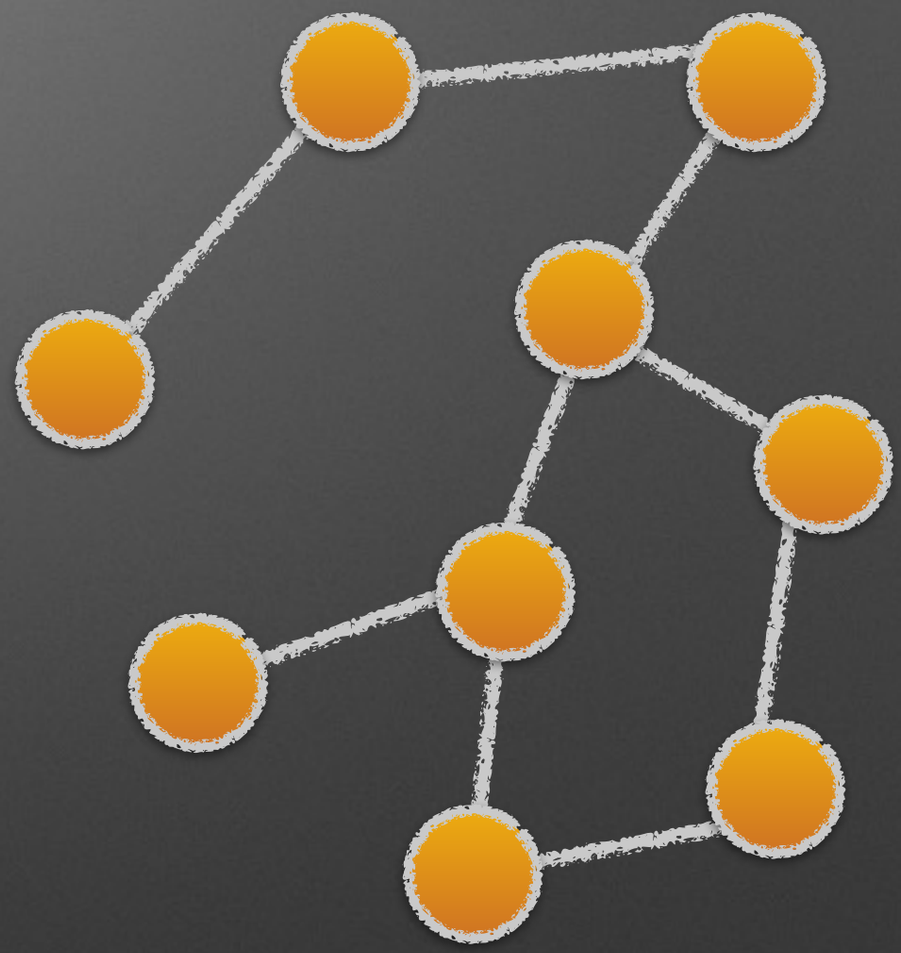
- No Consistency and No Availability

CAP Theorem

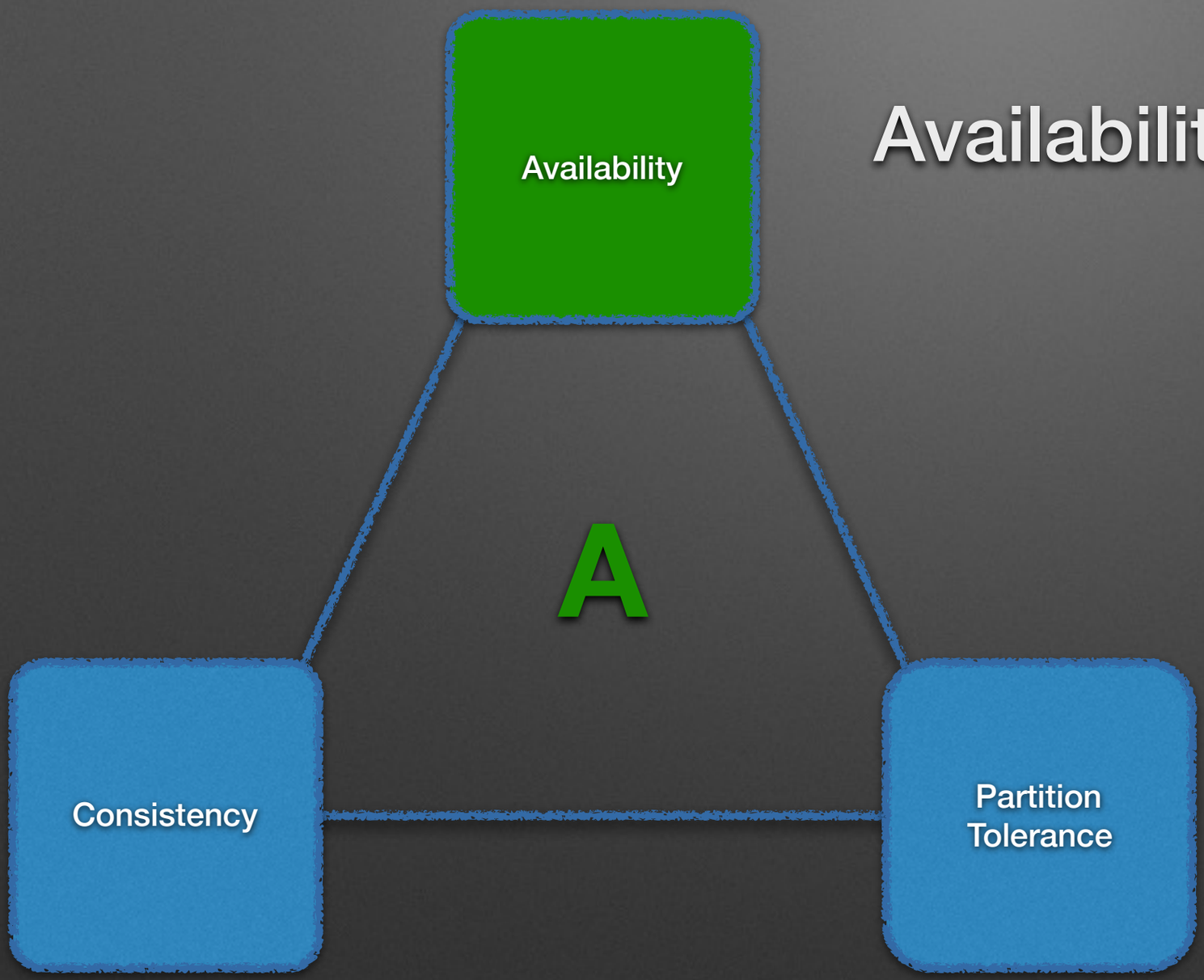
Eric A. Brewer

CAP Theorem

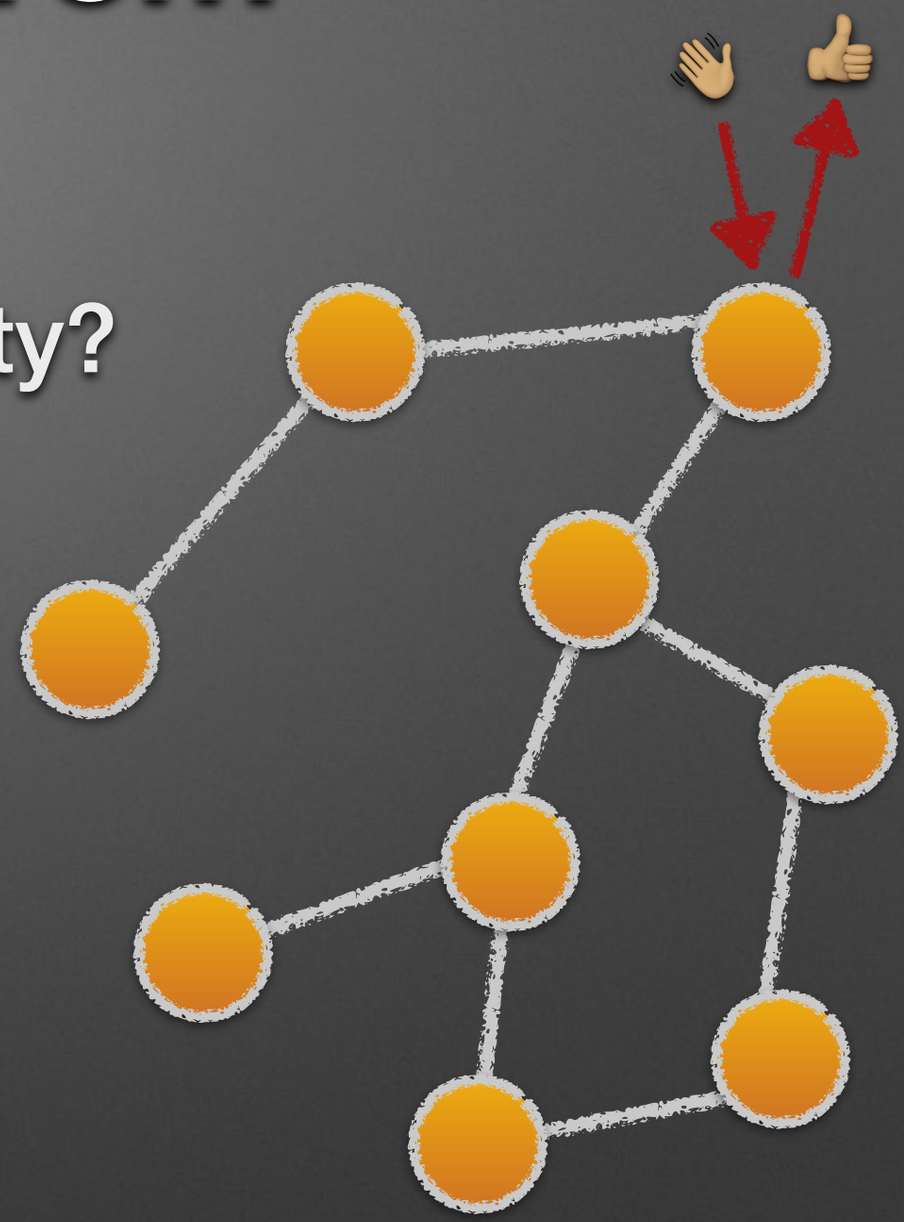

Availability
Distributed Systems
Consistency
Partition Tolerance



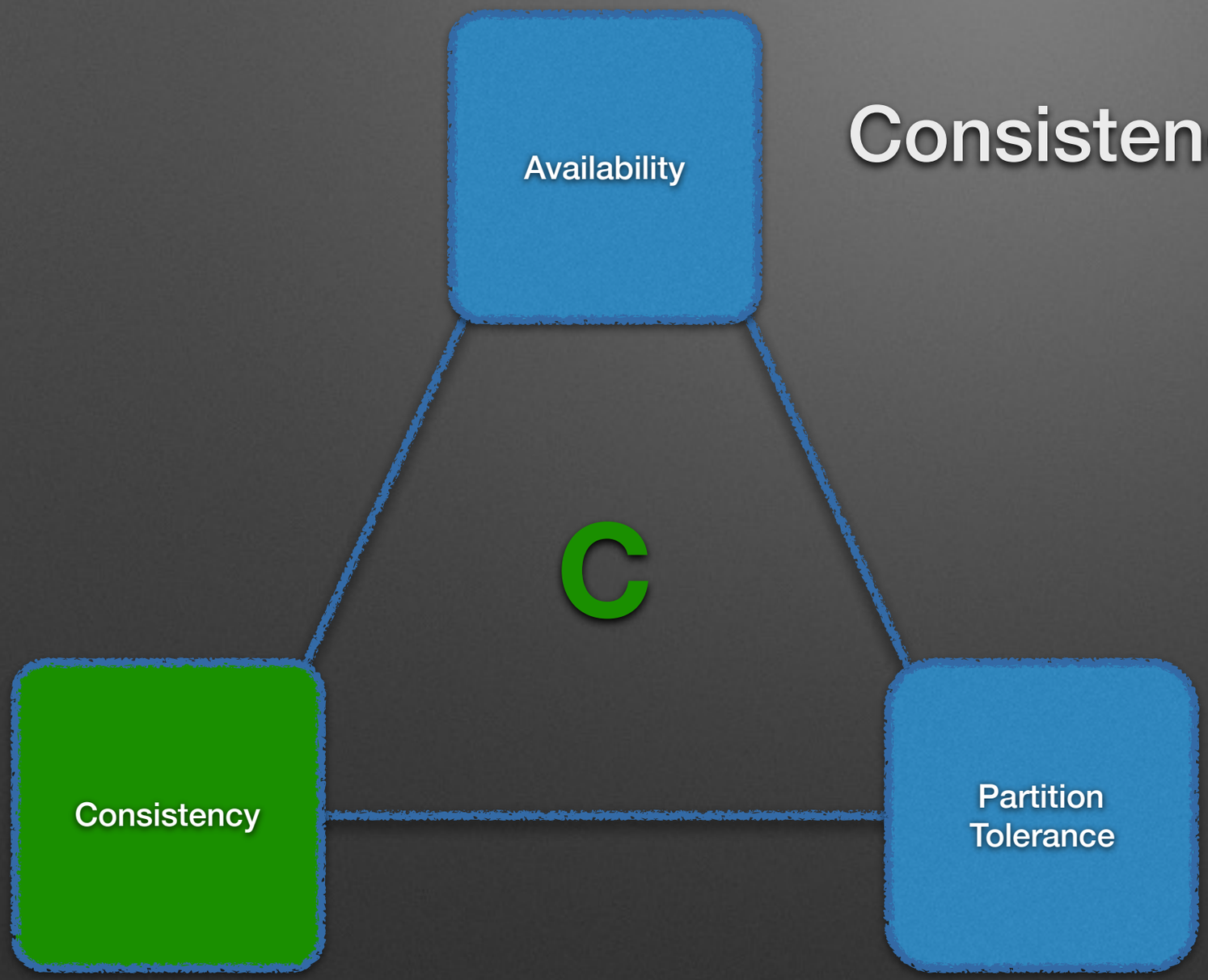
CAP Theorem



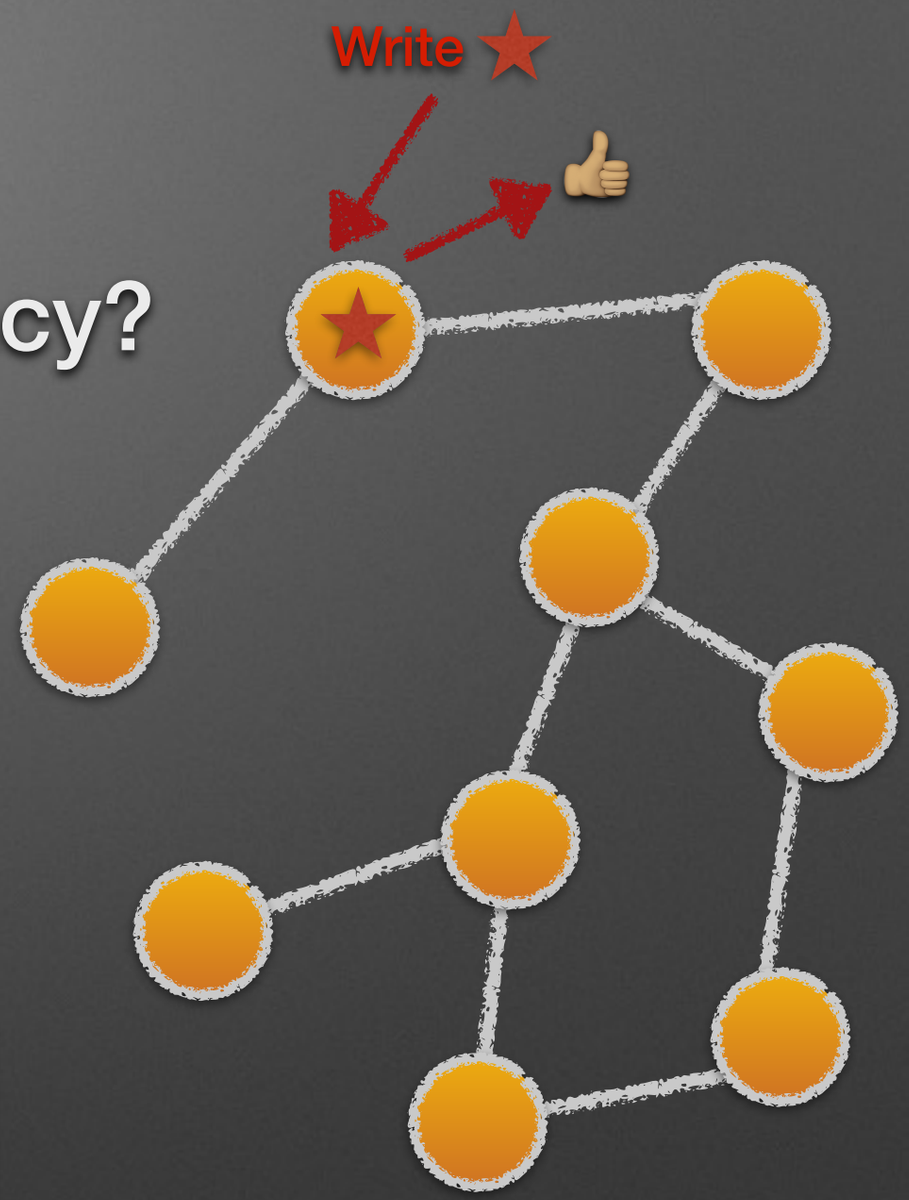
Availability?



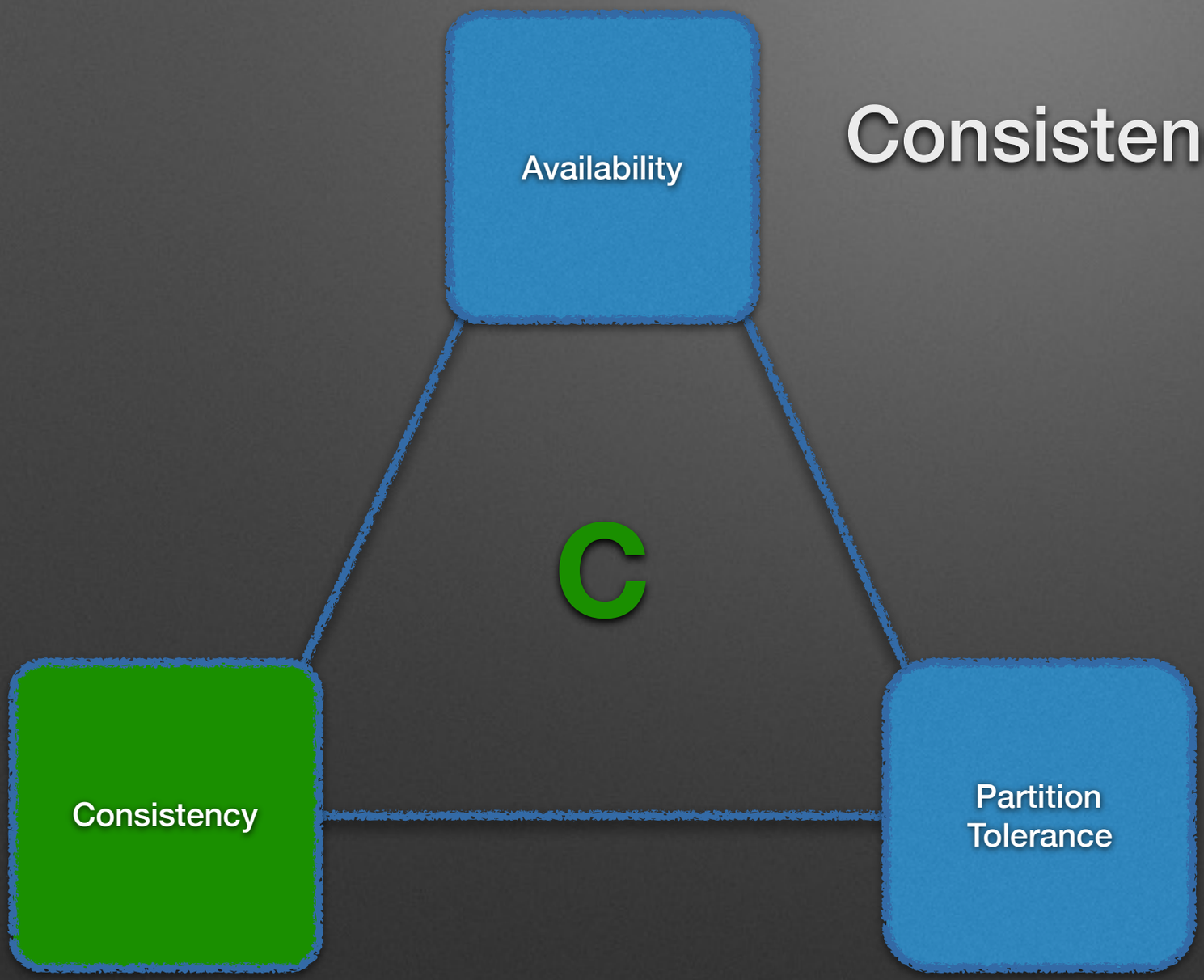
CAP Theorem



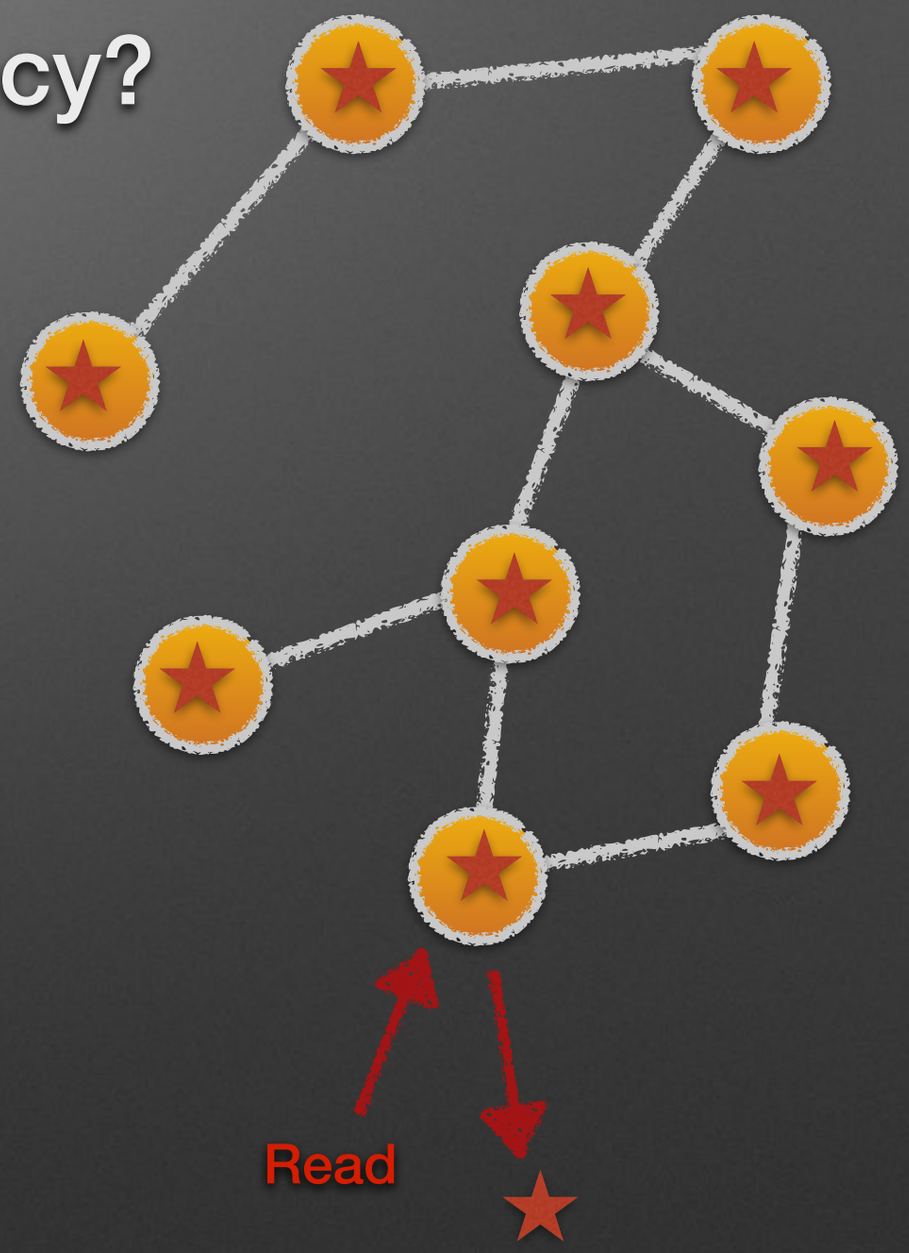
Consistency?



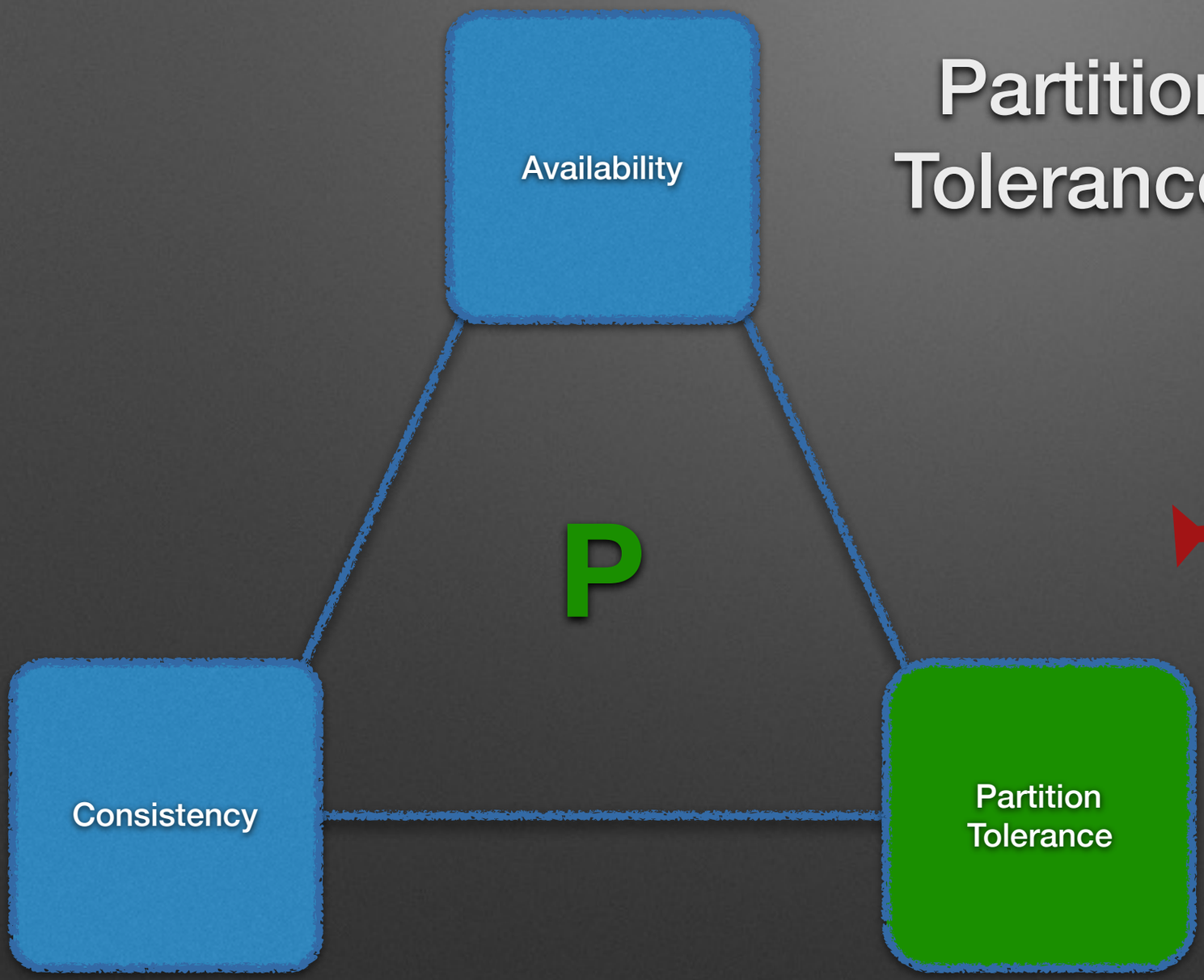
CAP Theorem



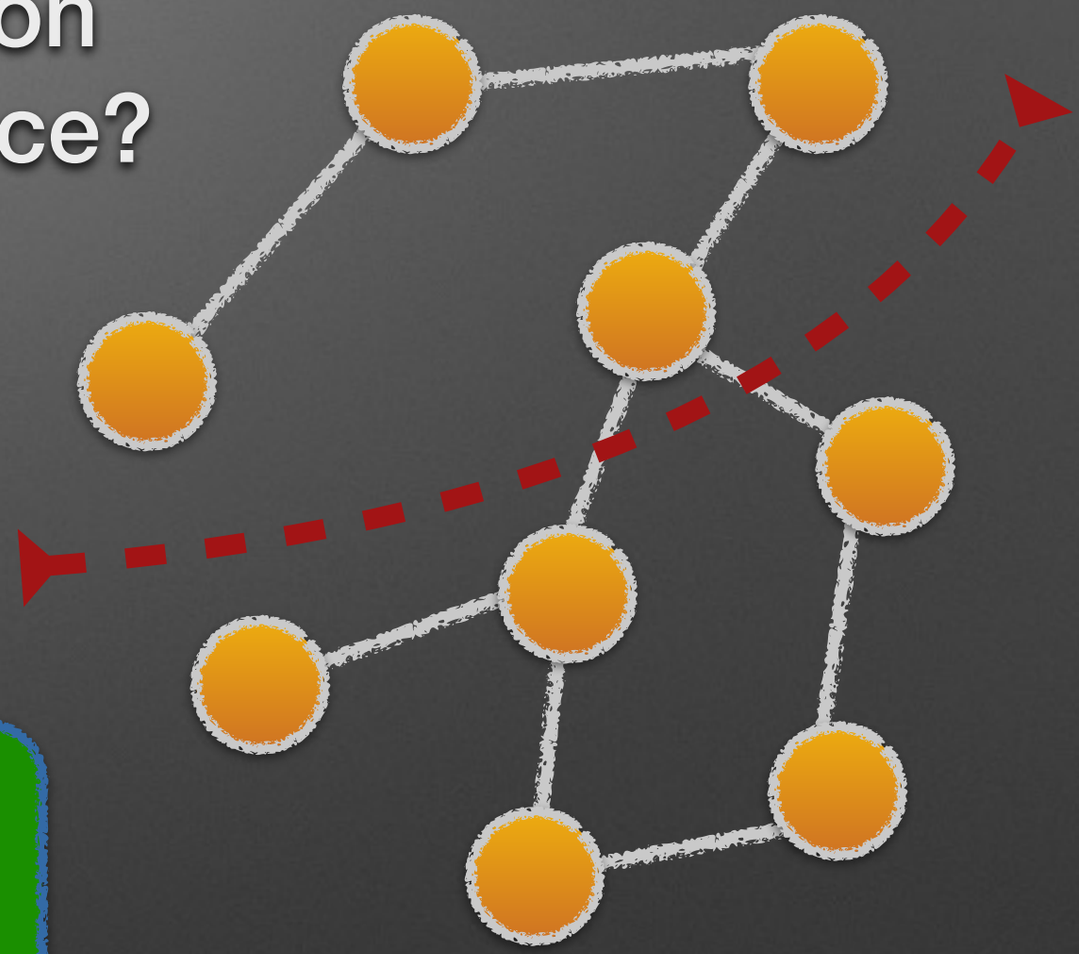
Consistency?



CAP Theorem



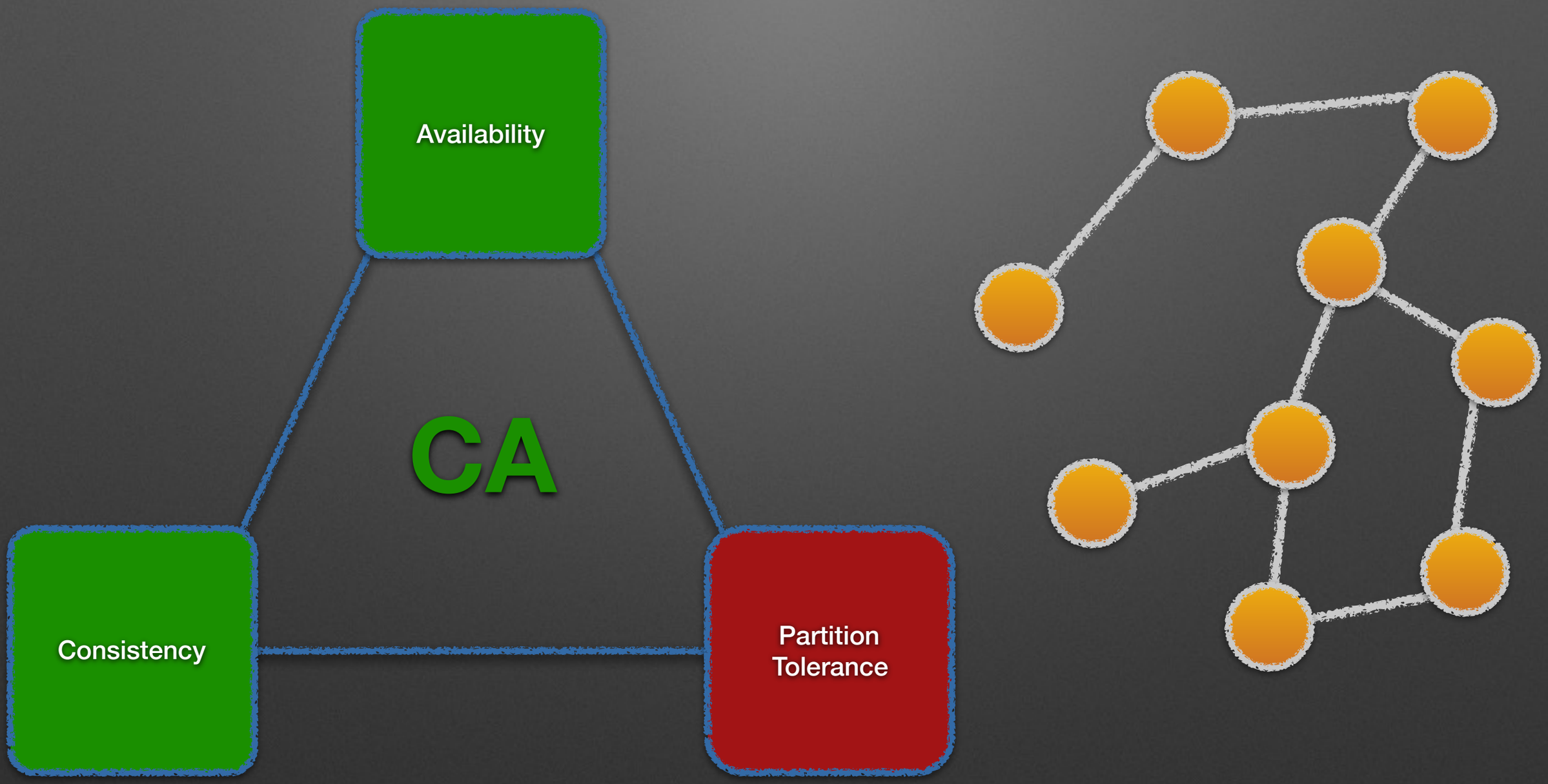
Partition
Tolerance?



CAP Theorem

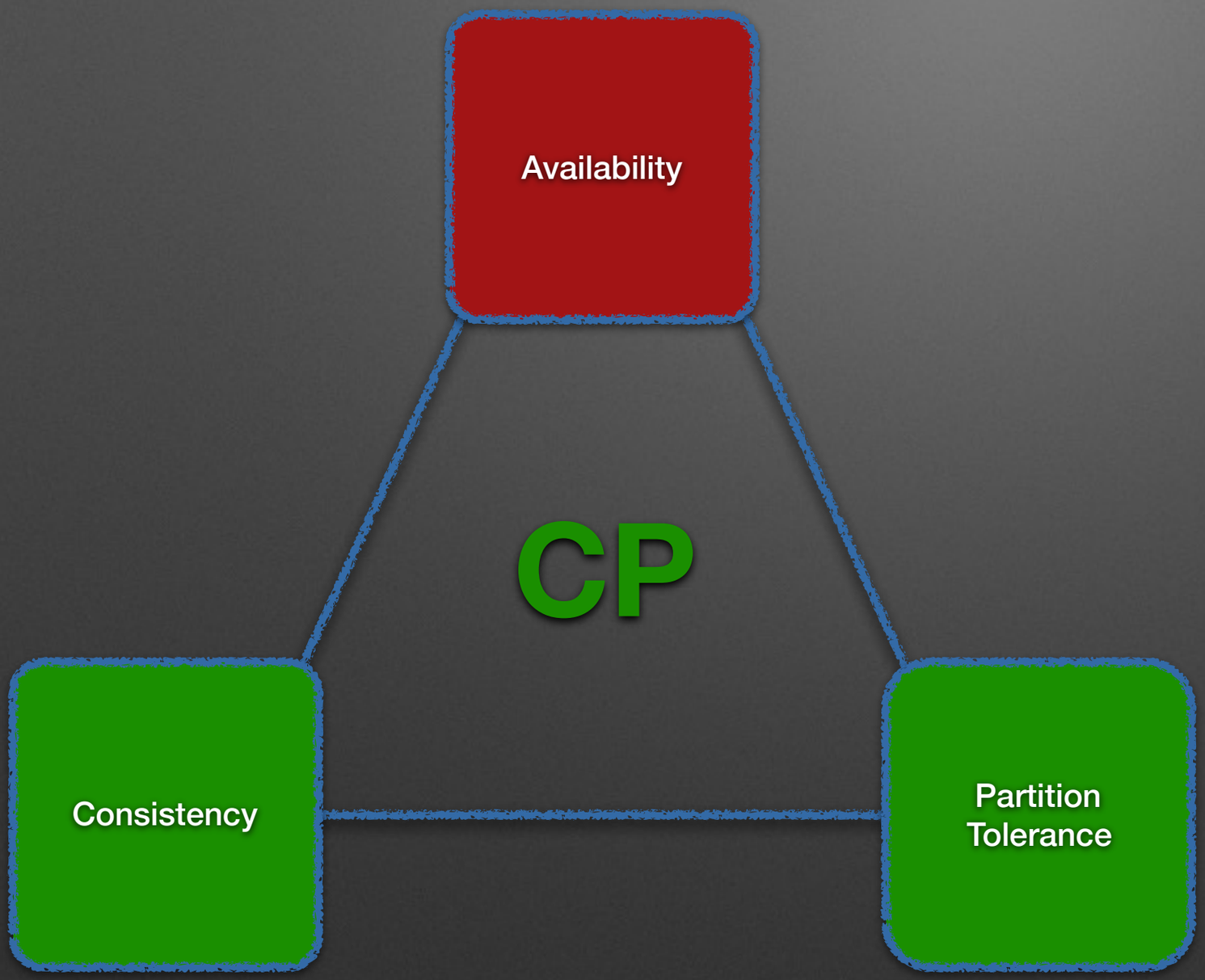

We offer three kinds of service:
GOOD - CHEAP - FAST
You can pick any two
GOOD service CHEAP won't be FAST
GOOD service FAST won't be CHEAP
FAST service CHEAP won't be GOOD

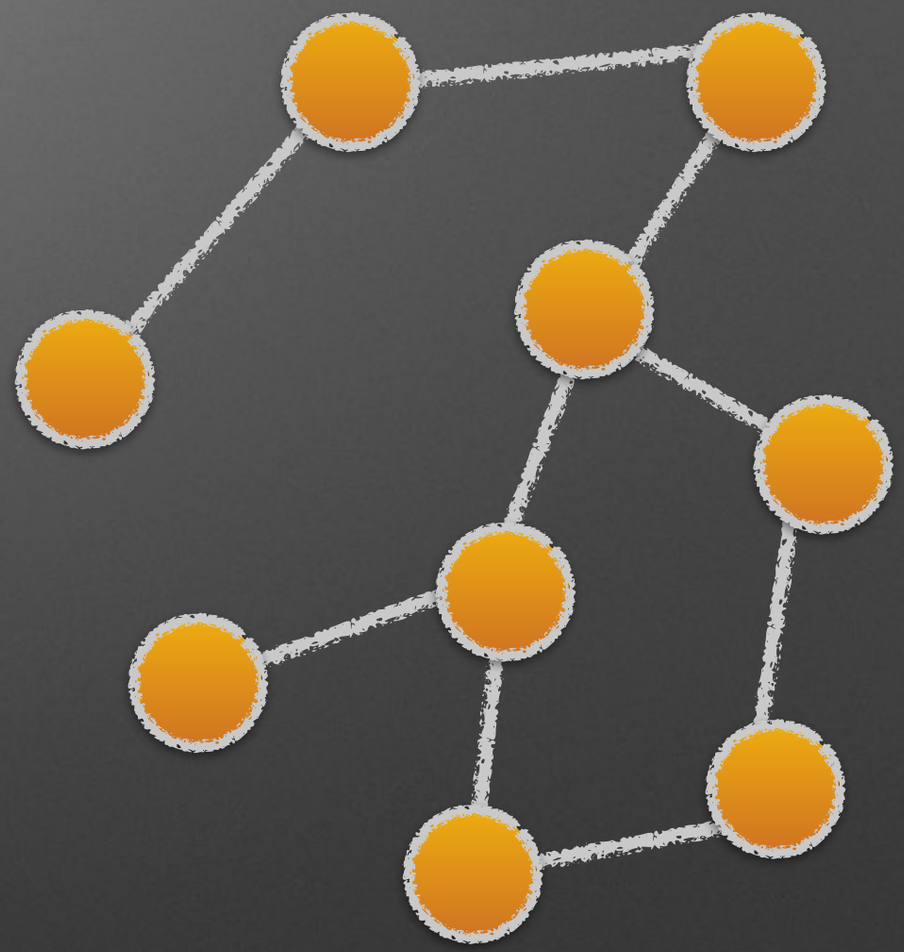
CAP Theorem



Sounds like Relational Databases!

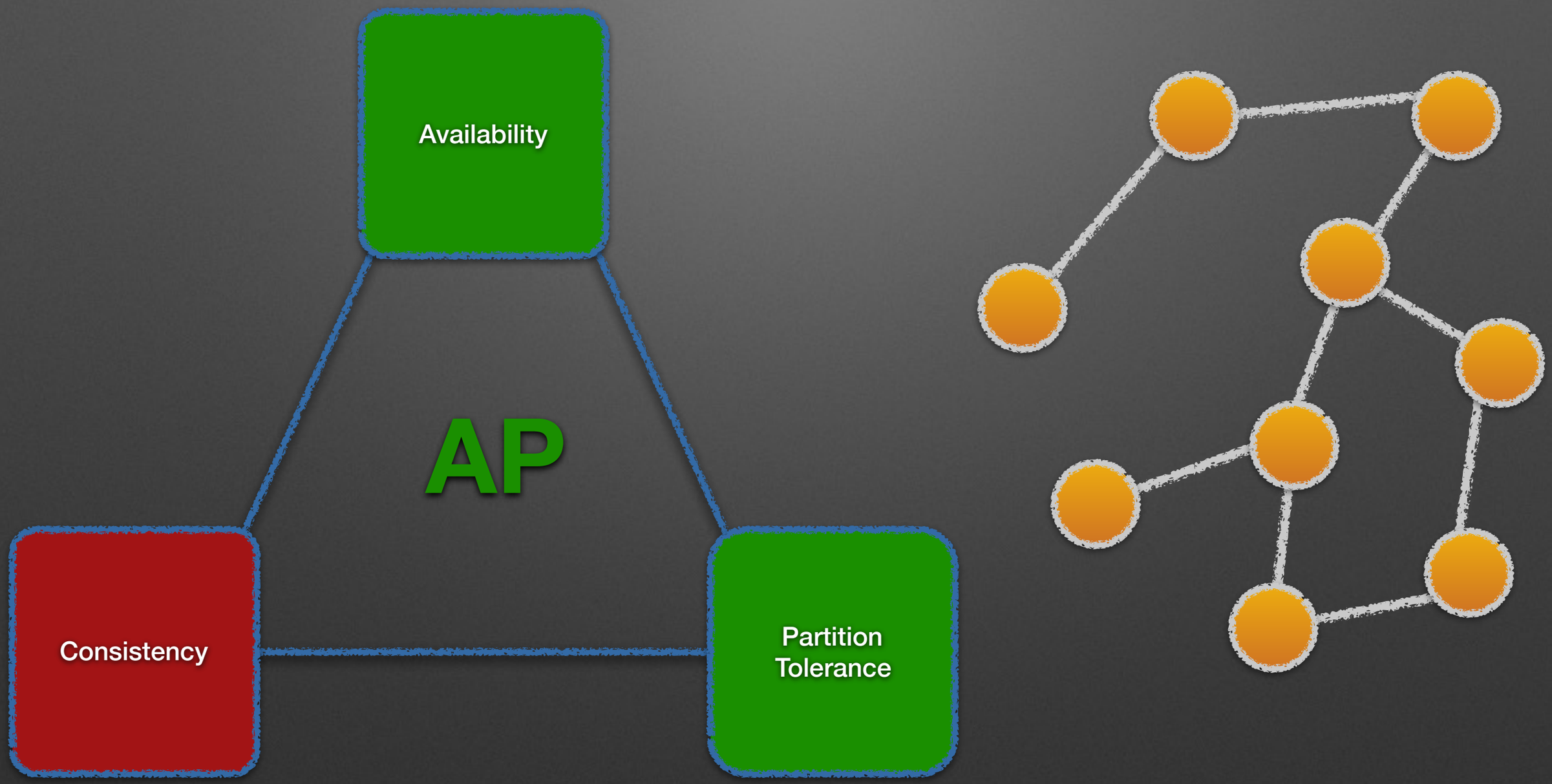
CAP Theorem





Is that what flat file databases were?

CAP Theorem



Is an AP datastore even possible?

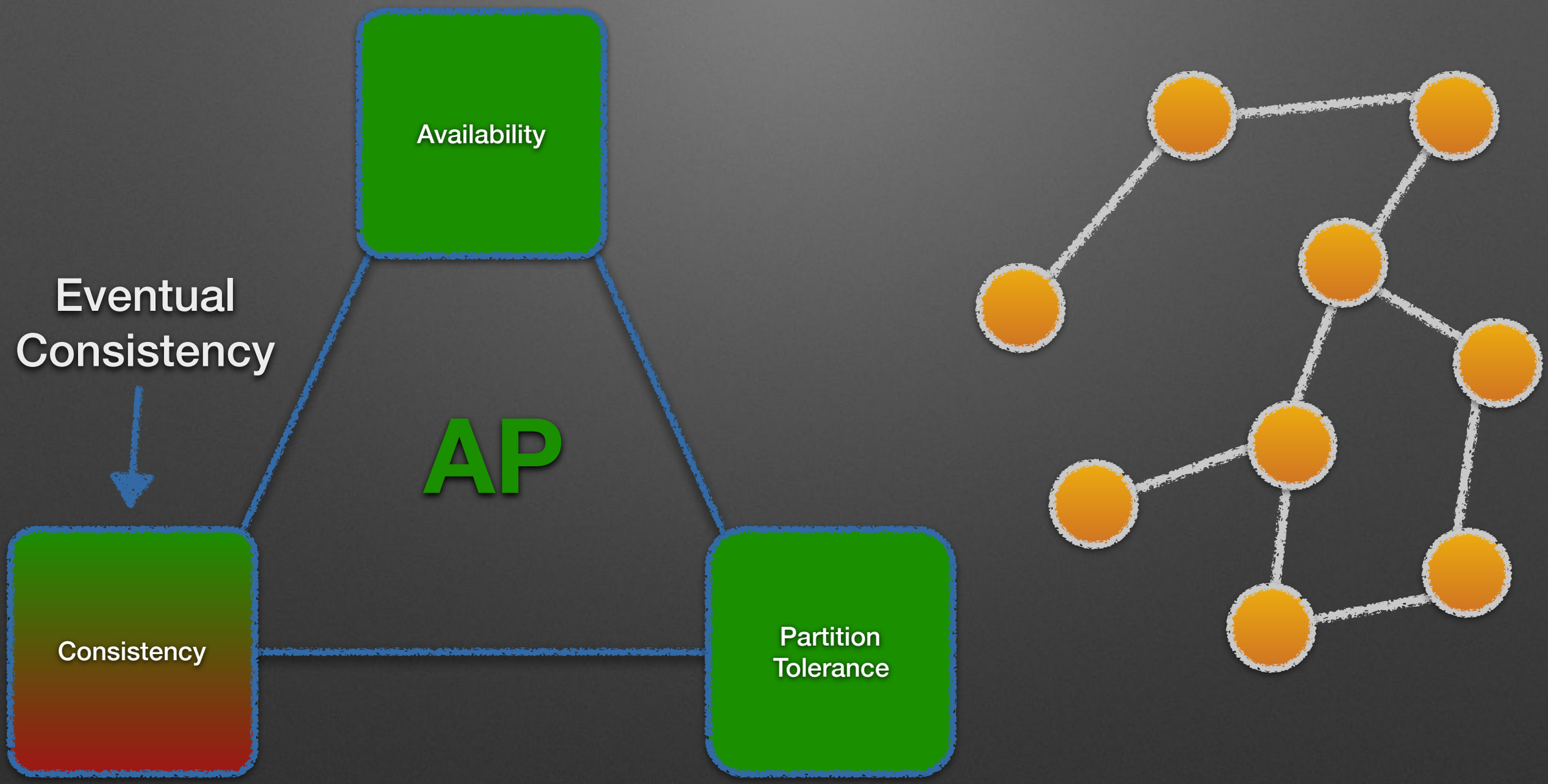
CAP Theorem

"Because partitions are rare, there is little reason to forfeit C or A when the system is not partitioned. Second, the choice between C and A can occur many times within the same system at very fine granularity; not only can subsystems make different choices, but the choice can change according to the operation or even the specific data or user involved. Finally, all three properties are more continuous than binary. Availability is obviously continuous from 0 to 100 percent, but there are also many levels of consistency, and even partitions have nuances, including disagreement within the system about whether a partition exists."

- Eric A. Brewer

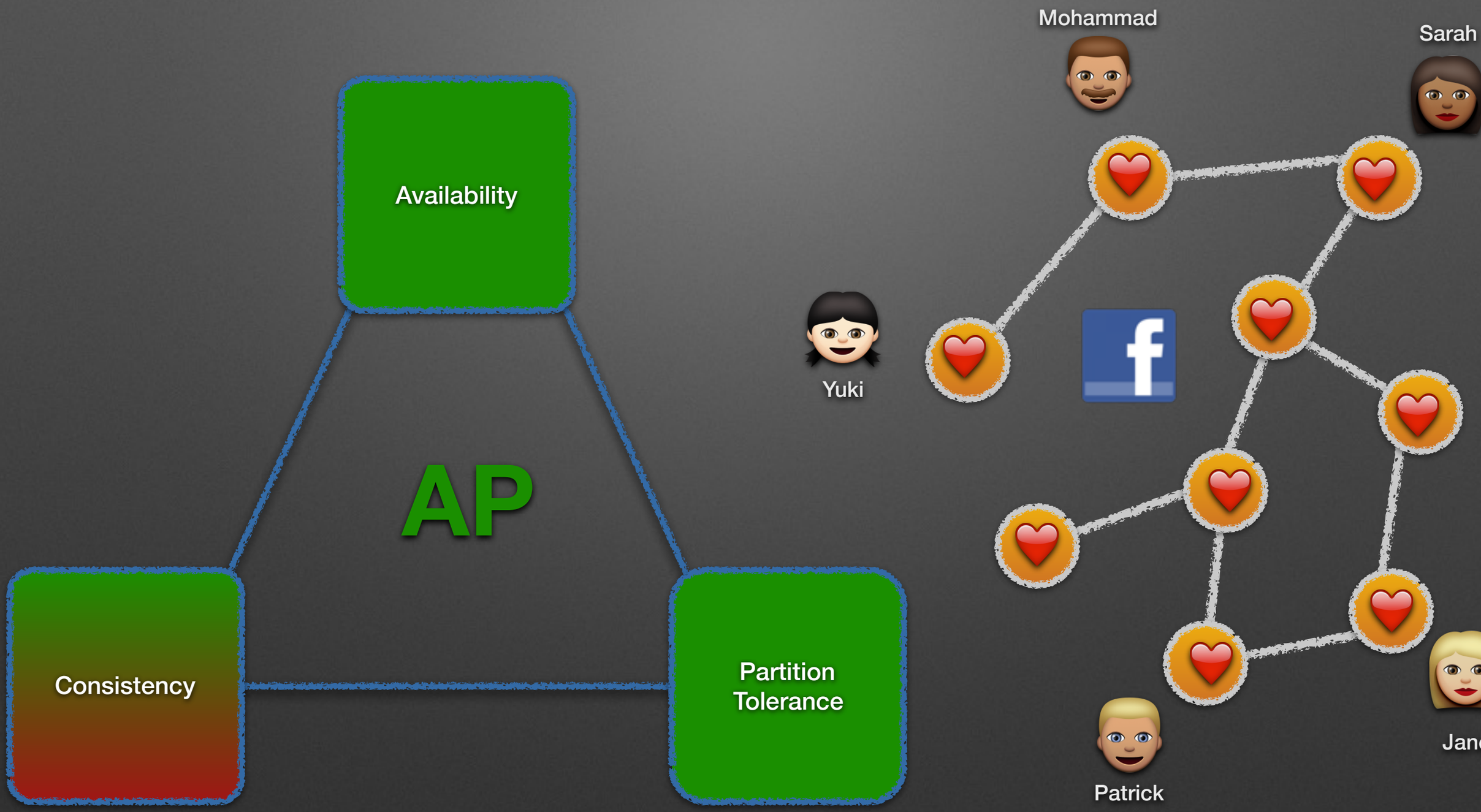
Is an AP datastore even possible?

CAP Theorem



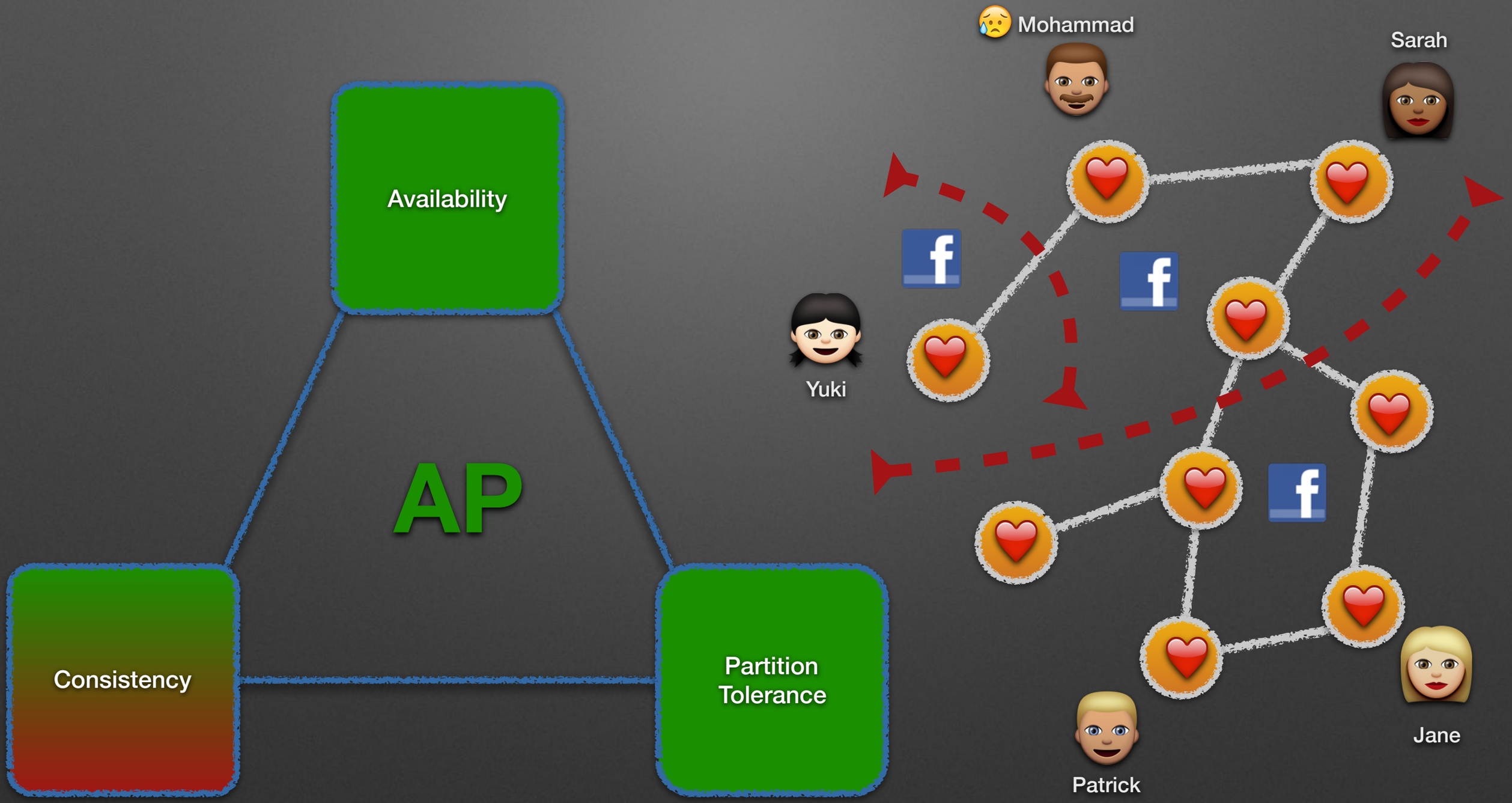
Is an AP datastore even possible?

CAP Theorem



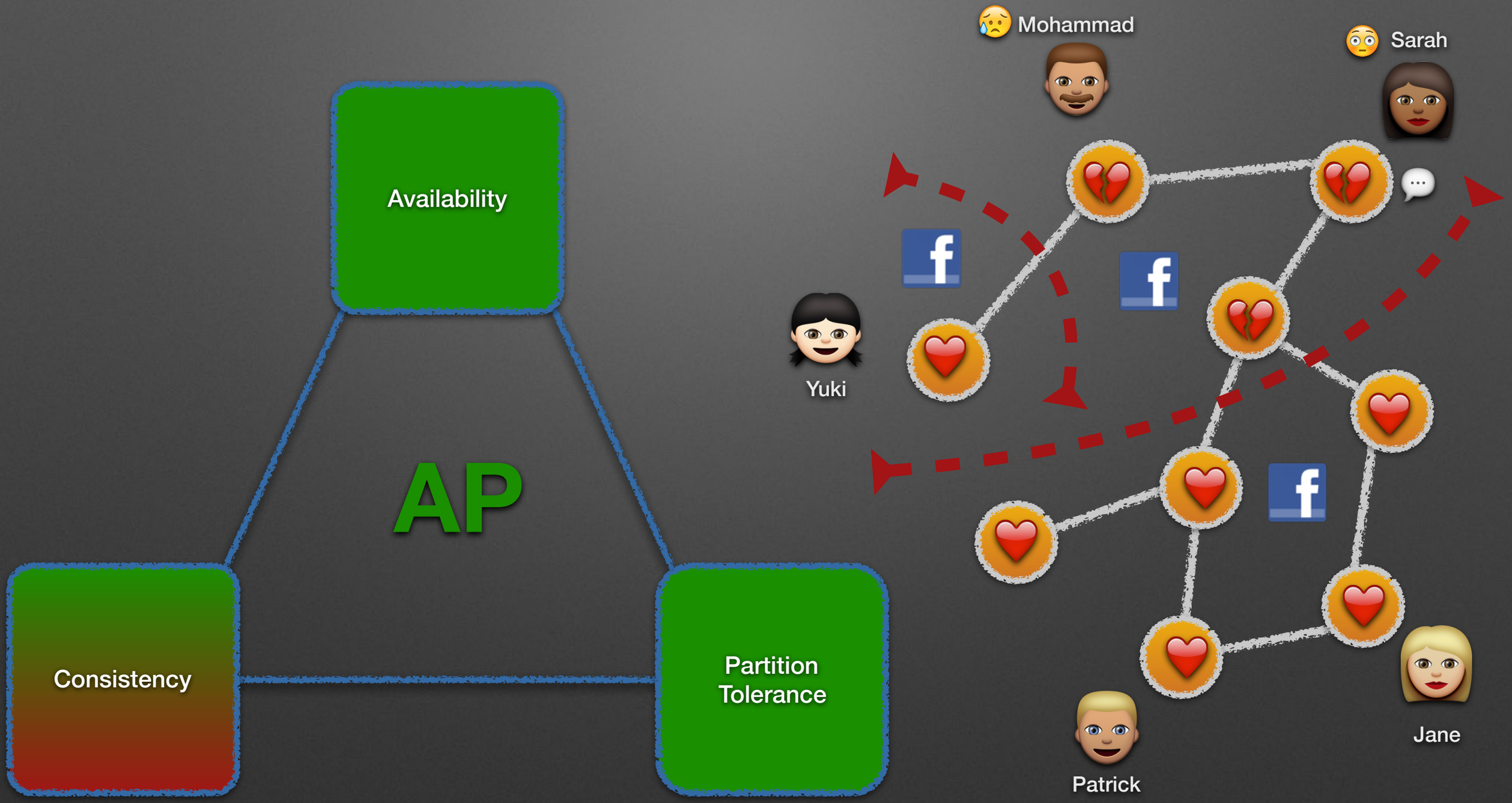
Is an AP datastore even possible?

CAP Theorem



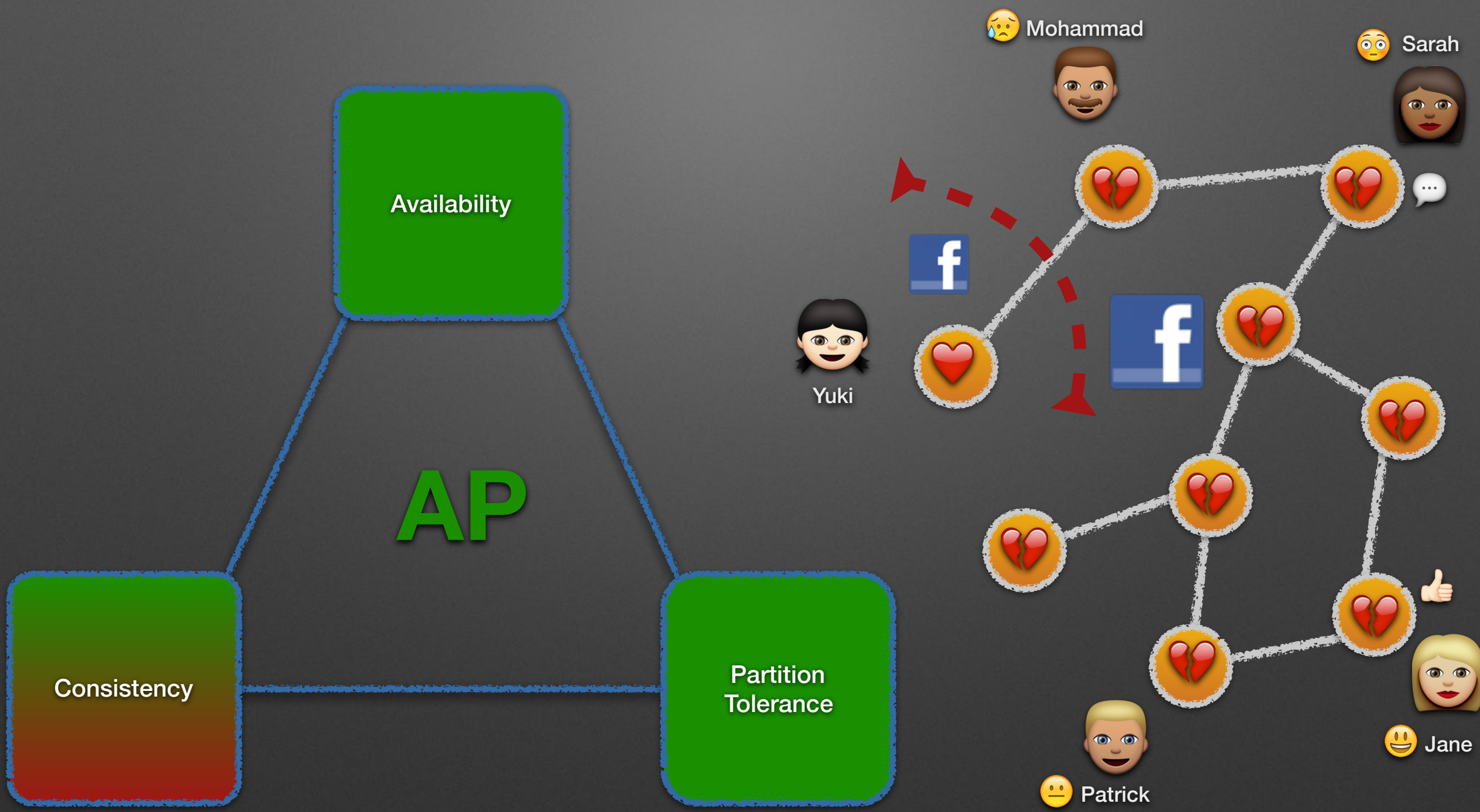
Is an AP datastore even possible?

CAP Theorem



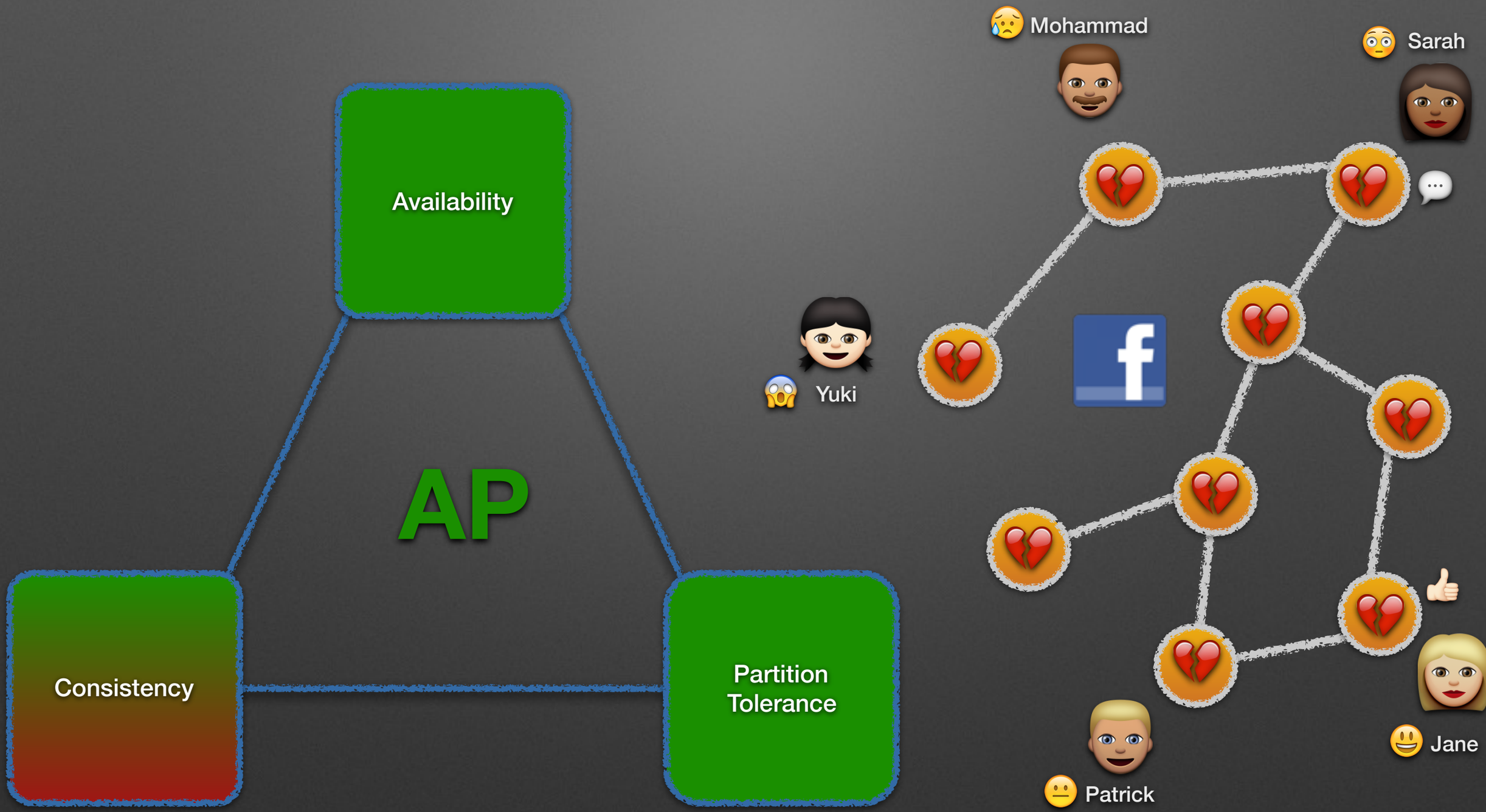
Is an AP datastore even possible?

CAP Theorem



Is an AP datastore even possible?

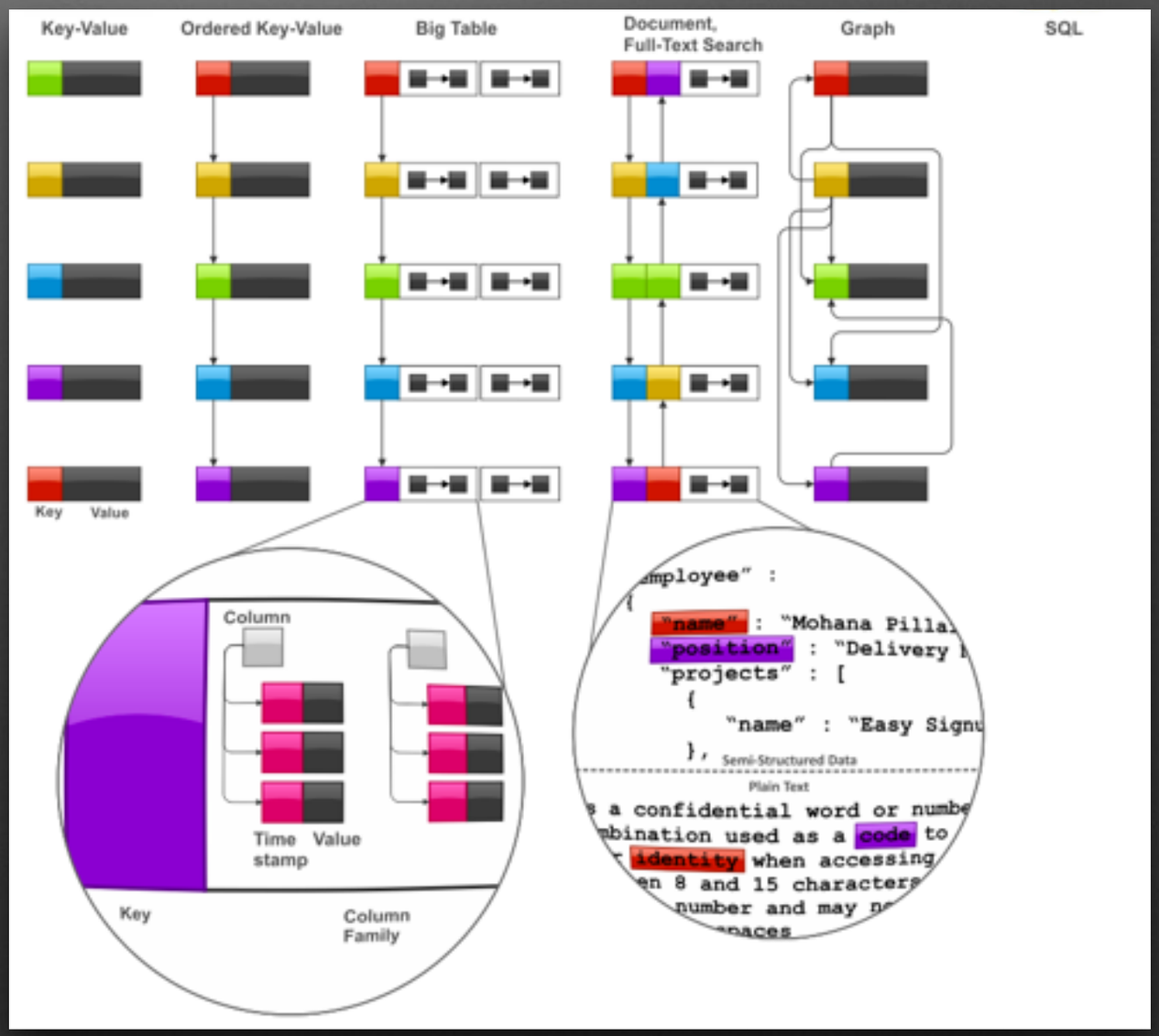
CAP Theorem



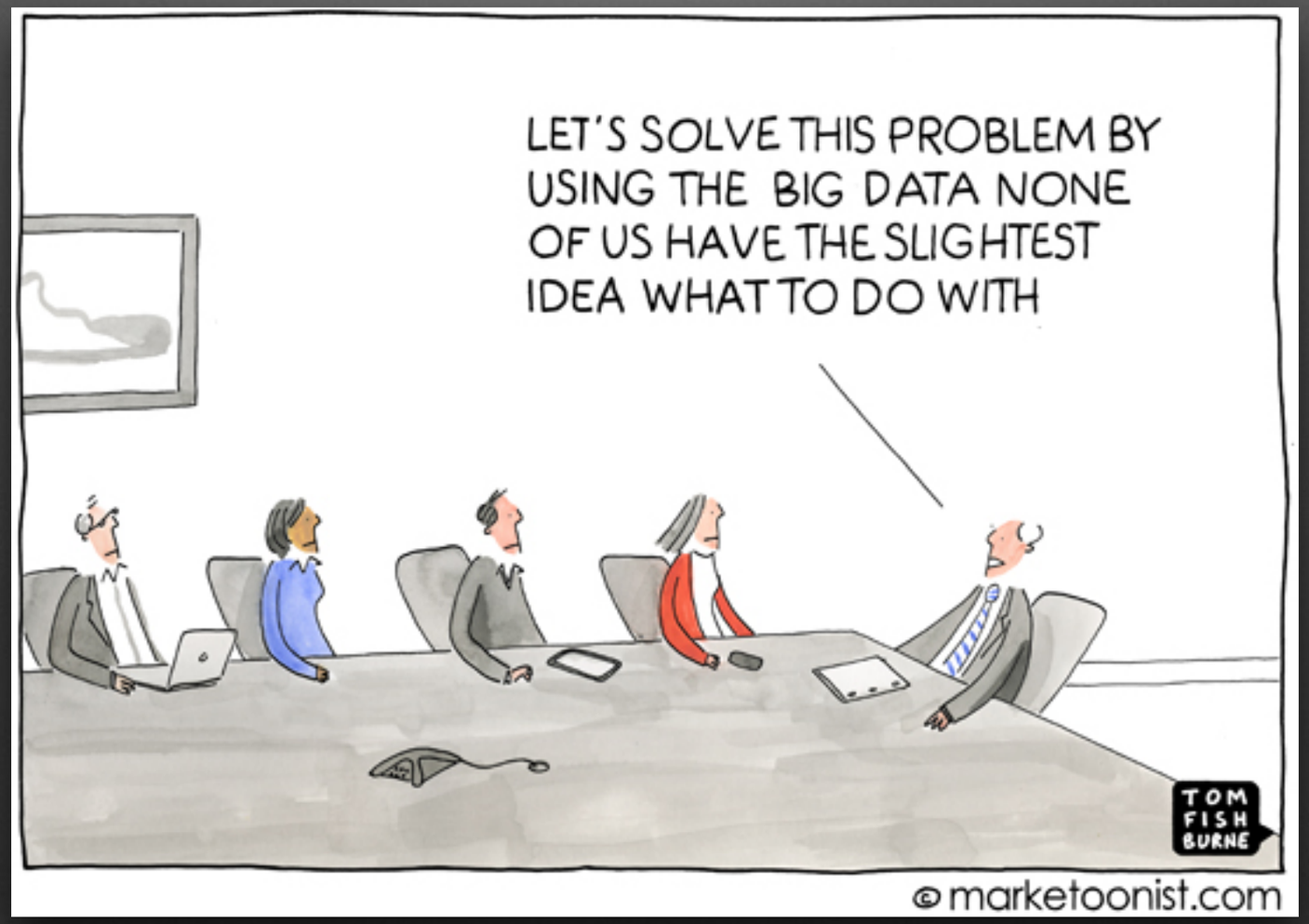
Is an AP datastore even possible?

NoSQL

- Key-Value Stores
- Document Stores
- Columnar Stores
- Graphs Stores



Big Data?


LET'S SOLVE THIS PROBLEM BY USING THE BIG DATA NONE OF US HAVE THE SLIGHTEST IDEA WHAT TO DO WITH
TOM FISH BURNE
© marketoonist.com

Big Data?

- A broad term
- Datasets that are
  - Too large

    OR

  - Too complex
- For traditional data processing applications

Big Data Characteristics

- “Big Data is high volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization.”

“The Importance of Big Data: A Definition”
By Mark Beyer and Douglas Laney of Gartner Inc. (2012)
https://www.gartner.com/doc/2057415/importance-big-data-definition

Big Data Characteristics

- Three characteristics known as V3
  - Volume
    - How much data?
  - Velocity
    - How fast is the data generated?
    - How fast does it need to be processed?
  - Variety
    - How many different types, formats, or any other criterion it may be categorized under?

Big Data Characteristics

- Variability
  - How consistent or inconsistent is the data?
- Veracity
  - How accurate is the data?
- Complexity
  - Where it is from (Multitude of sources r a single source?)
  - How does it relate to other data?
  - Is it easy to make sense of?
  - Etc…

Problem

- Too much data
- Complex queries

Solution

Apache Hadoop

https://hadoop.apache.org/

History

- 2004
  - “MapReduce: Simplified Data Processing on Large Clusters”
    - Published by Jeffrey Dean and Sanjay Ghemawat of Google
      - http://research.google.com/archive/mapreduce.html
- 2005
  - Apache Hadoop (Includes a Java Implementation of MapReduce)
  - Implemented by Doug Cutting and Mike Cafarella of Yahoo while working on Apache Nutch (https://nutch.apache.org/)

Overview

Data Storage Model
HDFS
(Hadoop Distributed File System)

- Reliable data storage model on cheap and unreliable commodity hardware
    - Reliability through Redundancy
- High-throughput and optimized for large dataset
    - Large black sizes (64MB or 128 MB)

Overview

Data Processing Model
MapReduce

- Data transfer overhead
  - Data sampling Not good-enough or possible
- Data Proximity
  - Taking processing to data rather than bringing data to processing

Overview

Data Processing Model
MapReduce

- MapReduce Algorithm
  - Map
    - Queries are split and distributed across parallel nodes and processed in parallel
  - Reduce
    - Results of queries are gathered and delivered

Overview

The path of the righteous man is beset on all sides by the inequities of the selfish and the tyranny of evil men.

Blessed is he who, in the name of charity and good will, shepherds the weak through the valley of the darkness.

For he is truly his brother's keeper and the finder of lost children. And I will strike down upon thee with great vengeance and furious anger those who attempt to poison and destroy my brothers. And you will know I am the Lord when I lay my vengeance upon you.

Data Processing Model
MapReduce

Map	Reduce

The path of the righteous man is beset on all sides by the inequities of the selfish and the tyranny of evil men.

Blessed is he who, in the name of charity and good will, shepherds the weak through the valley of the darkness.

For he is truly his brother's keeper and the finder of lost children. And I will strike down upon thee with great vengeance and furious anger those who attempt to poison and destroy my brothers. And you will know I am the Lord when I lay my vengeance upon you.

the 5
is 1

the 4
is 1

the 2
is 1
you 2

the 5
the 4
the 2

the 11

is 1
is 1
is 1
you 2

is 3
you 2

the 11
is 3
you 2

Architecture

HDFS

MapReduce

Master

Name Node

Secondary Name Node

Job Tracker

Slave

Data Node

Data Node

Data Node

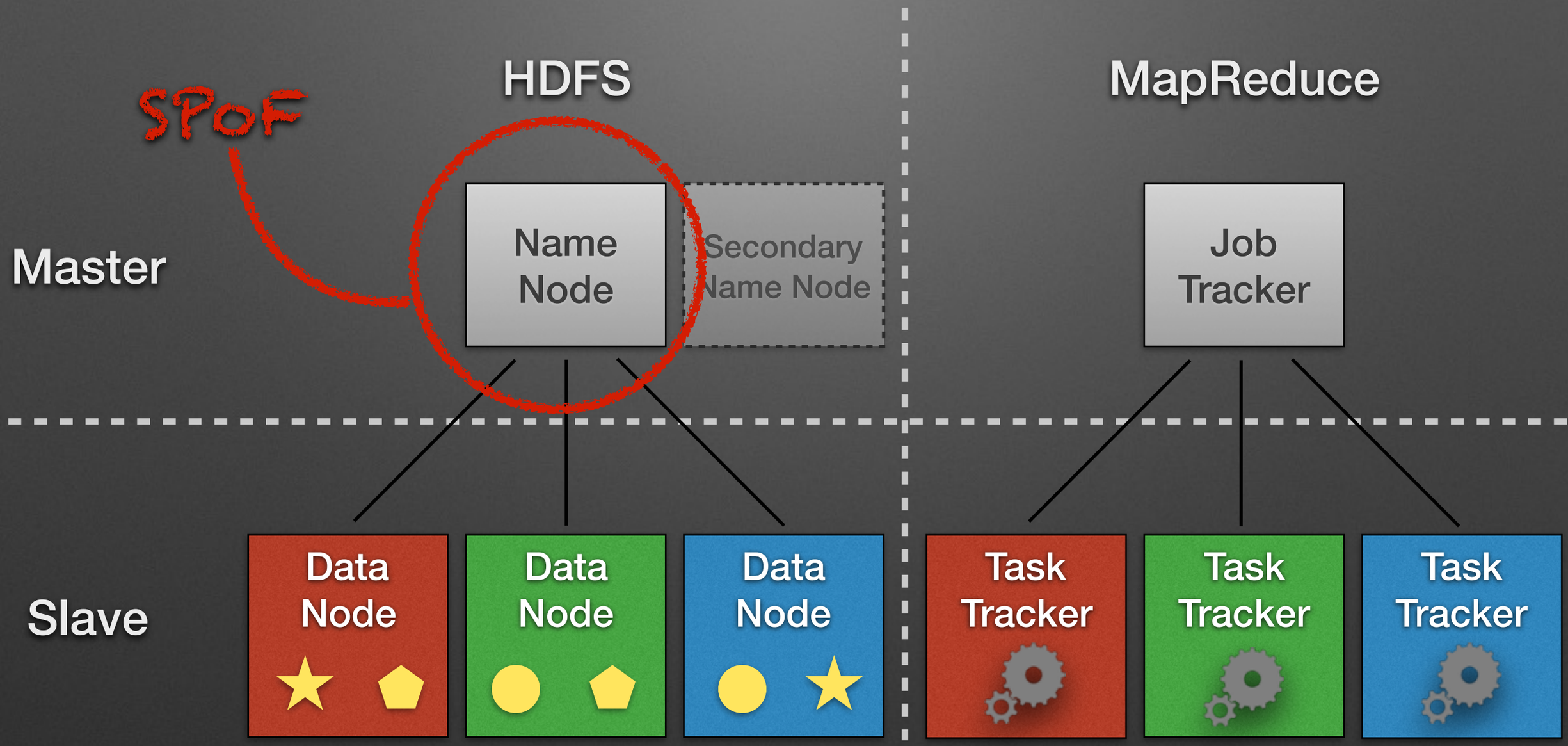
Task Tracker

Task Tracker

Task Tracker

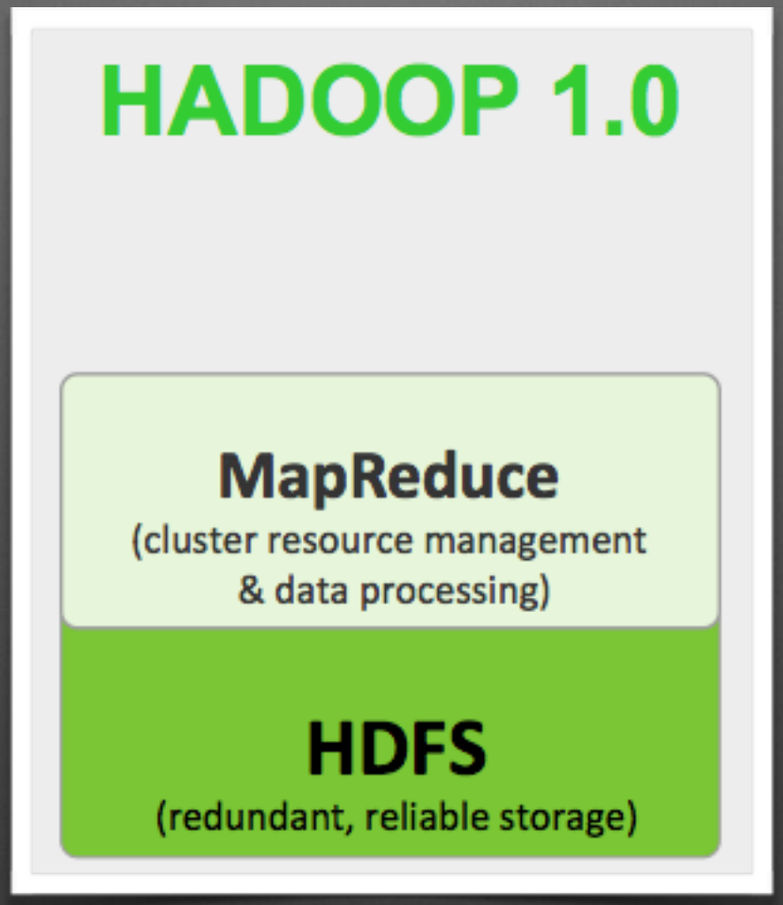
Hadoop 1.0

Architecture



Hadoop 1.0

Architecture

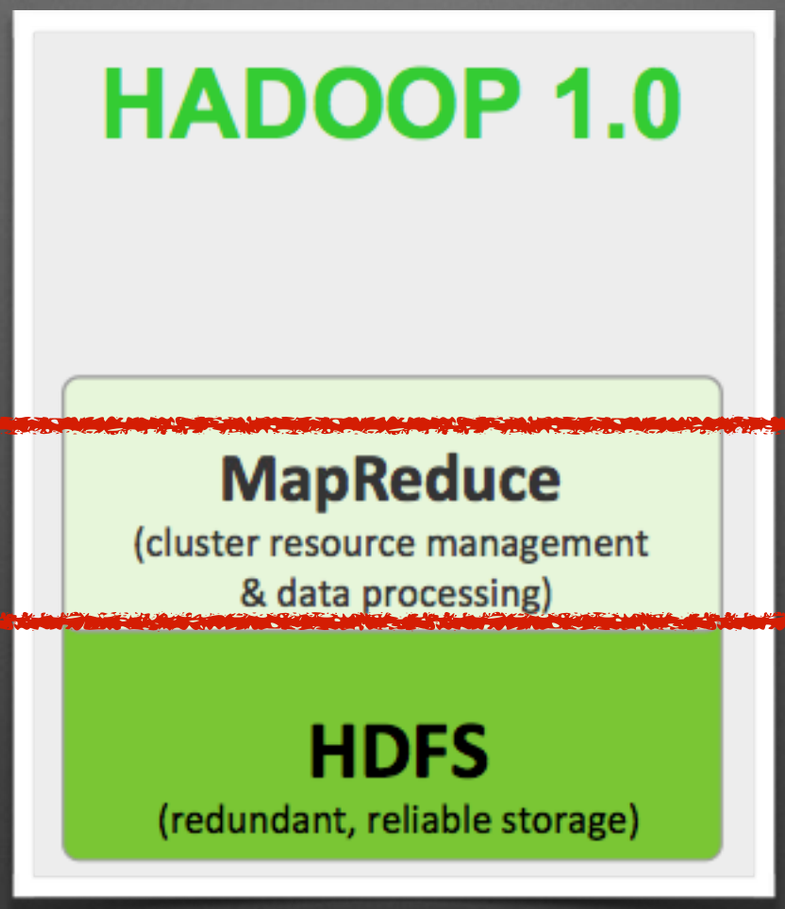


Courtesy of Hortonworks

Hadoop 1.0

Architecture

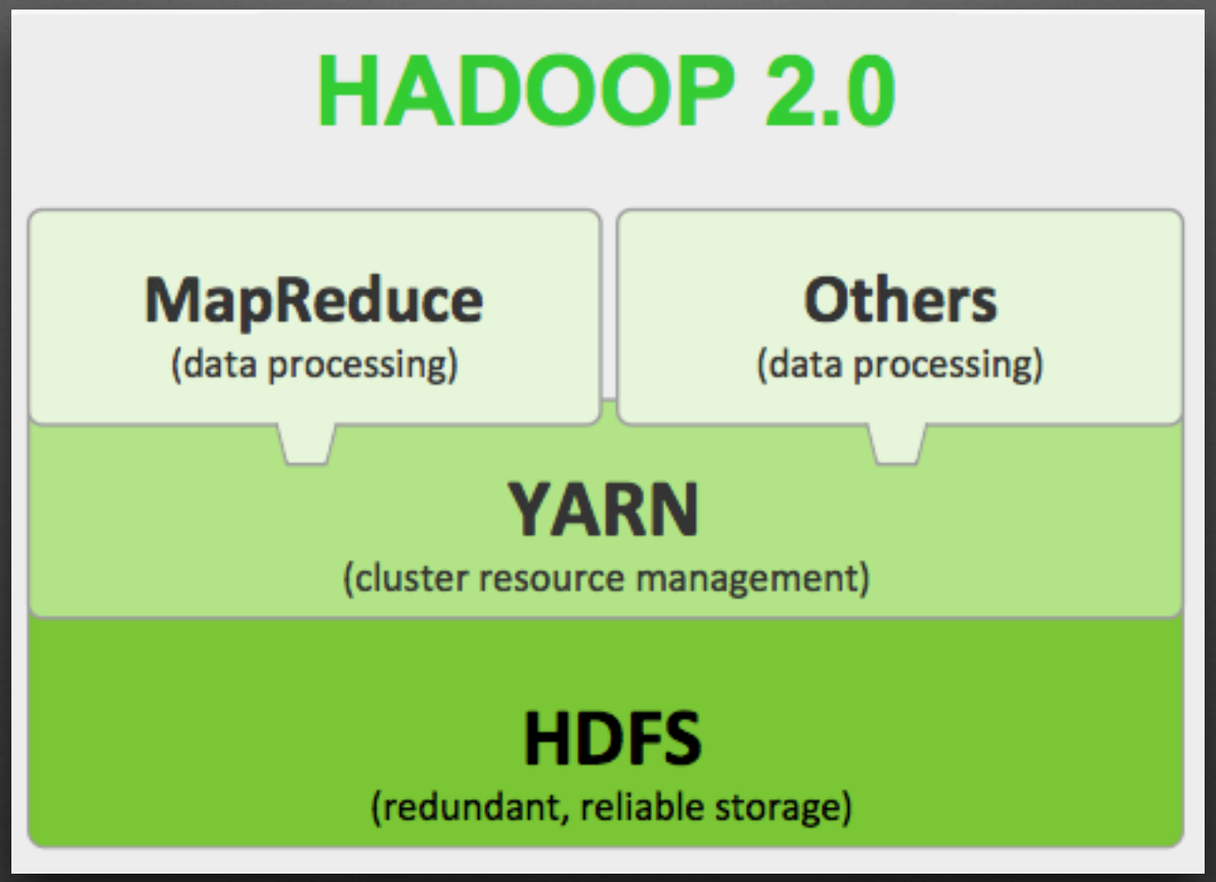
Coupling



Courtesy of Hortonworks

Hadoop 1.0

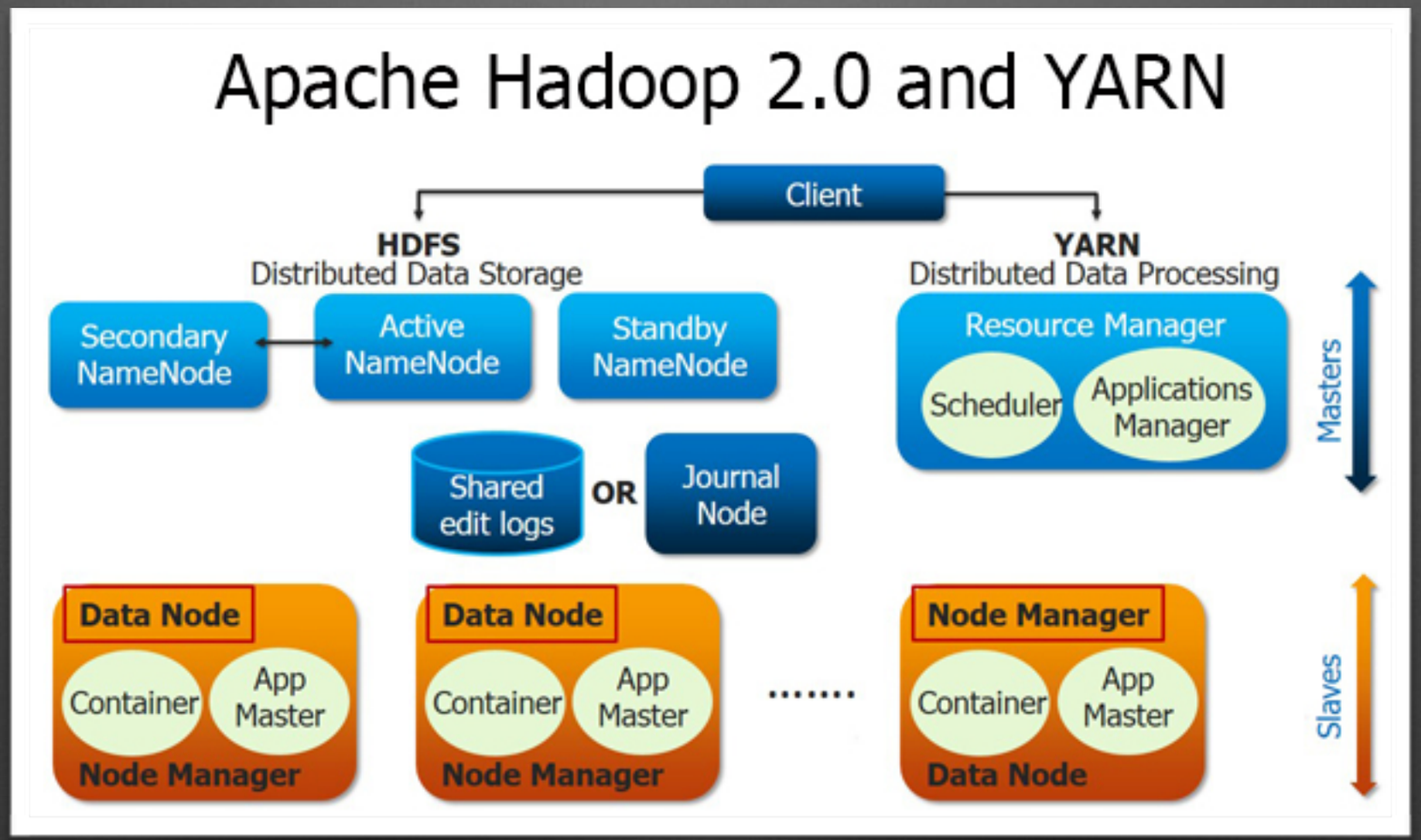
Architecture



Courtesy of Hortonworks

Hadoop 2.0

Architecture



Courtesy of edureka!

Interface

- Java Framework / API
- CLI (Command Line Interface)
- Apache Thrift API (Support for many languages)
- Streaming API (Support for many languages)

Problem

- Writing Java MapReduce sucks!
  - Simple parametrized queries
  - ETL (Extract, Transform, and Load)
  - Sequenced Data Transformations (Filtering, grouping, partitioning, joining, etc…)

Solution

- A higher-level language to hide the complexity of MapReduce Java Code
- An ad-hoc way to create and execute MapReduce jobs

Solution

Apache Pig

https://pig.apache.org/

History

- 2006
  - Developed Apache Pig at Yahoo
- 2007
  - Donated to the Apache Software Foundation

Overview

- Dataflow Programming
- “A paradigm that models a program as a directed graph of the data flowing between operations” - Wikipedia
- Some features of functional languages

Overview

- Pig Latin
    - A scripting language
        - Declare schema for data (complex data types, collections, etc…)
        - Projections (foreach … generate)
        - Declare execution plans
            - DAG (Directed Acyclic Flow) data pipelines as opposed to sequential pipelines (Supports splits)

Overview

- Generates machine-optimized MapReduce jobs (Lazy evaluations, etc…)
- Perfect for ETL jobs
- Extendable
  - UDFs (User-Defined Functions)
  - DataFu a UDF library by LinkedIn http://data.linkedin.com/opensource/datafu

Interface

- Java API

- grunt shell (CLI - Command Line Interface)

Solution

Apache Hive

https://hive.apache.org/

History

- Donated to the Apache Pig at Facebook
- 2008
  - Donated to the Apache Software Foundation

Overview

- Hive can own and manage storage files on HDFS
- Metastore
  - Metadata to manage files in a directory structure
  - A relational DB (Apache Derby by default, MySQL, etc…)

Overview

- HiveQL
  - Declarative
  - Similar syntax to SQL
  - DDL for defining data schema
    - Operations on databases, internal tables, external tables, partitions, etc…
    - Complex data types and collections
  - DML for manipulating data
    - No support for insert, update, or delete
    - Query result can be inserted into tables

Overview

- Hive driver compiles queries, optimizes them, and executes them by invoking MapReduce jobs
- ETL (Extract, Transform, and Load) jobs require temporary tables or nested queries
- Extendable
  - UDFs (User-Defined Functions)
  - Custom SerDes (Serializer/Deserializer)

Interface

- Java API
- Hive CLI (Command Line Interface)
- Web Interface
- Apache Thrift API (Support for many languages)
- JDBC/ODBC facades

VS.

- ETL with Pig
- Query with Hive
- Learning Curve
    - Pig Latin > HiveQL

Problem

- Workflow & scheduling can be a nightmare
    - Building data pipelines / Job coordination and execution
        - On a regular basis
        - As data becomes available
        - Depending on the outcome of another job
- Shell scripts & cron
    - Seriously!!!

Solution

Apache Oozie

https://oozie.apache.org/

Apache Azkaban

http://azkaban.github.io/azkaban

History

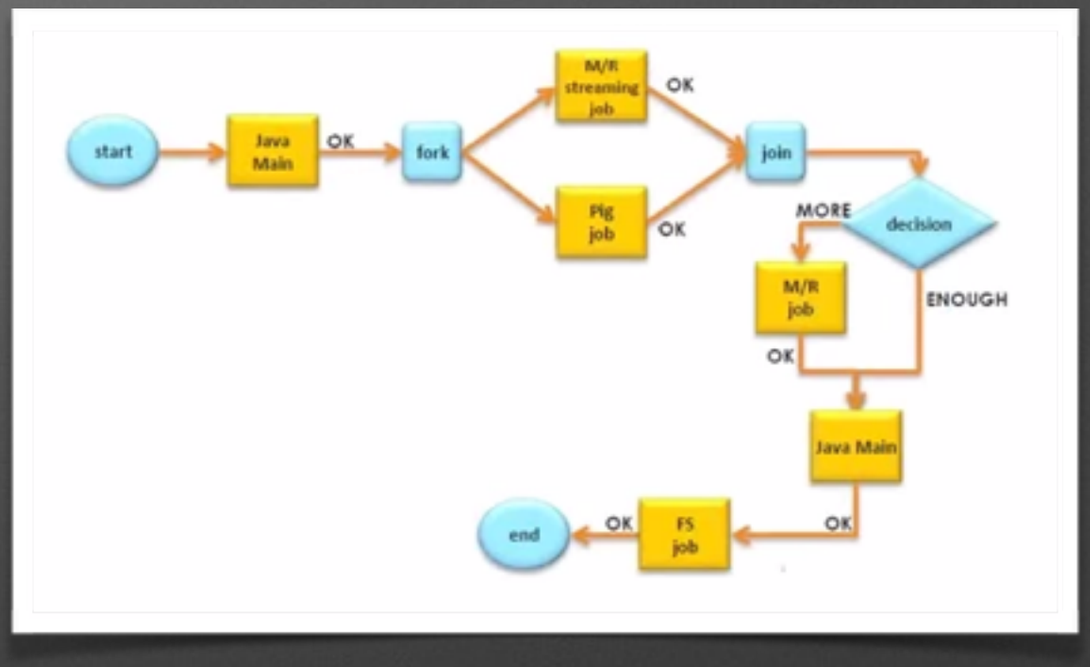
- Apache Oozie
- 2008
  - Donated to the Apache Software Foundation

Overview

- Apache Oozie
  - A workflow scheduling system to manage Hadoop jobs
  - Scaleable
  - Reliable
  - Extendable

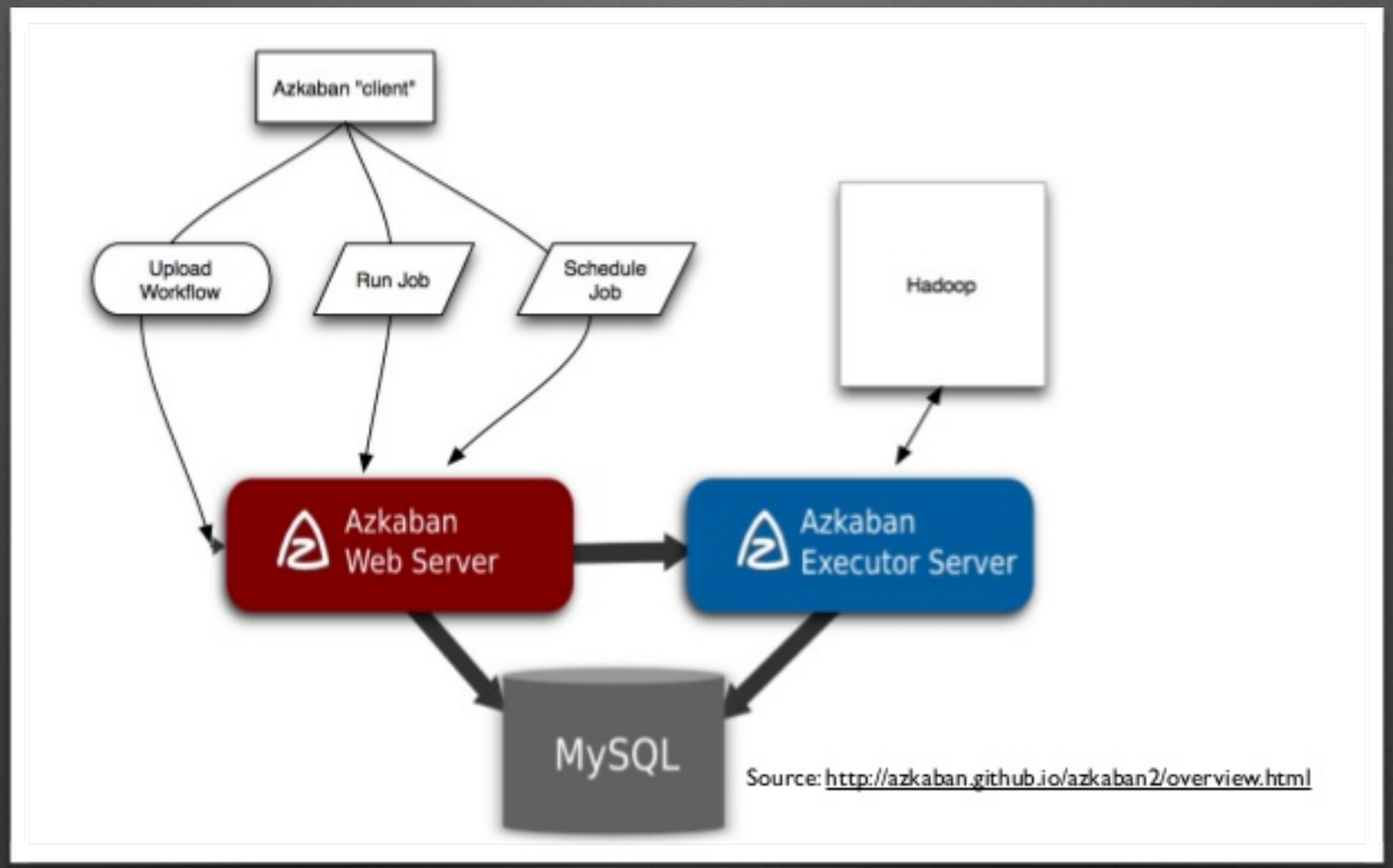
Architecture

- Apache Oozie
  - Oozie Workflow
    - DAGs (Directed Acyclic Graph of actions)
  - Oozie Coordinator
    - Triggers workflow by time and data availability



Architecture

- Azkaban



Interface

- Apache Oozie
  - Java API
  - Declarative XML Configuration
- Azkaban
  - Azkaban CLI (Command Line Interface)
  - Declarative Configuration (Property files and pre-defined directory structure)

Problem

- Batch Processing is high-latency
- Need real time solutions for real time problems
  - An architecture based on queues and workers (Pub/Sub) sucks
    - Complex and tedious to deal with (Routing, serialization, partitioning, etc…)
    - Message brokers are slow and have scalability issue
- Producers and downstream services are coupled

Solution

Apache Storm

https://storm.apache.org/

History

- 2011
- Implemented by Nathan Marz and team at BlackType (Acquired by Twitter)

Overview

- A higher level of abstraction than message passing
- Written in Clojure and Java
- The most watched JVM project on GitHub (At some point)

Overview

- A distributed real-time computation system
- Ideal for stream processing and continuous computation
    - Processing unbound streams of data at real-time
- Fast, Scalable, Reliable, and Fault-tolerant

Architecture

- Nimbus (Analogous to Hadoop's Job Tracker)
- Zookeeper (Cluster coordination)
- Supervisor (Analogous to Hadoop's Task Tracker)

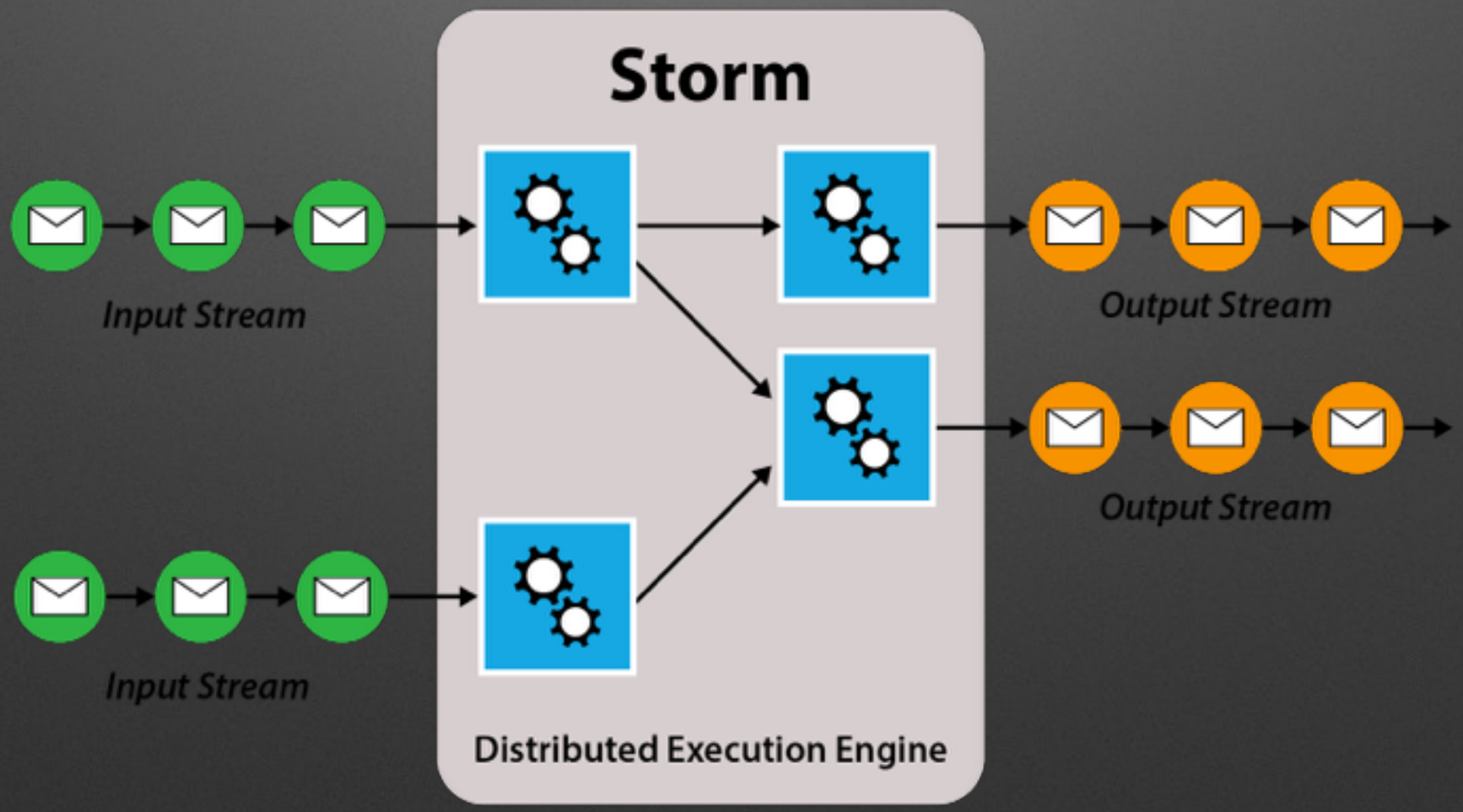
Architecture

- Topology
    - Runs forever
    - A DAG (Directed Acyclic Graph) representing a network of streams, spouts, and bolts
- Stream
    - An unbounded stream of tuples
- Spout
    - Taps into a data source
    - Feeds data
- Bolt
    - A Computation unit (func, filter, aggregation, joins, machine learning, etc…)
    - Process input streams and produces output streams

Architecture

STORM

- Spouts and bolts are inherently parallelized and communicate though message passing



Courtesy of Ran.ga.na.than Balashanmugam, ThoughtWorks!

Interface

- Java API
- Clojure API
- CLI (Command Line Interface)

Problem

- Batch Processing is high-latency
- Need real time solutions for real time problems
  - An architecture based on queues and workers (Pub/Sub) sucks
    - Complex and tedious to deal with (Routing, serialization, partitioning, etc…)
    - Message brokers are slow and have scalability issue
- Producers and downstream services are coupled

Solution

- Decoupling producers and data pipelines to downstream service
- Producers to send all to one system

Solution

Apache Kafka

https://kafka.apache.org/

History

- 2011
  - Open-sourced by LinkedIn

Overview

- Decoupling producers and data pipelines to downstream service
- Producers to send all to one system
- Pub/Sub messaging implemented as a distributed commit log
- Distributed, partitioned, replicated, and scalable

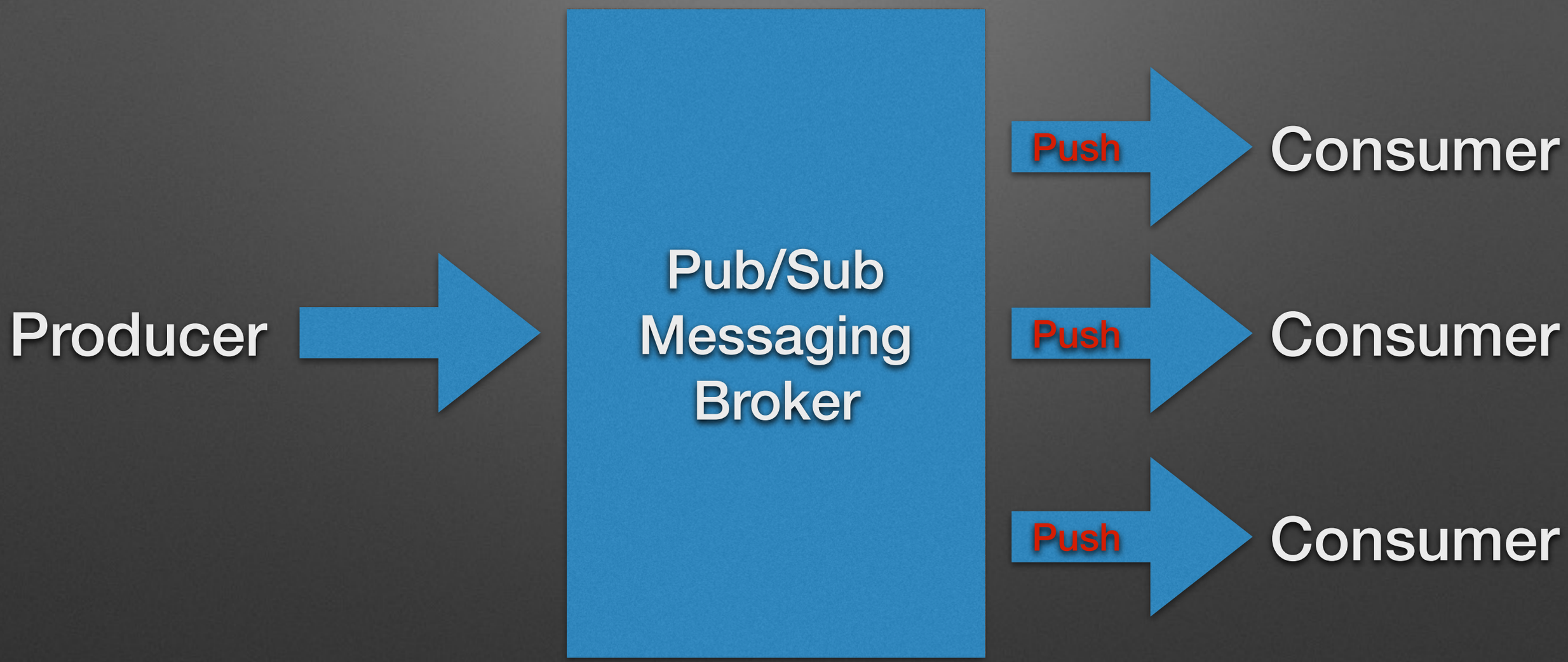
Architecture

- Broker
    - A cluster of one or more servers
- Producer
    - Publishes data to topics
- Consumer
    - Reads messages under a topic

Architecture

- Topic
    - The category under which the message is stored and published
    - Partition
        - Multiple partitions per topic
        - Each contains messages replicated across multiple servers
            - Uniquely identified by an offset within the partition
        - There is one leader per partition to handle read/write request and passively replicate to the followers
        - One a leader dies, another is elected

Pub.Sub vs. kafka



Pub.Sub vs. kafka

Scaling out consumers without impaction the broker

Interface

- Java API
- CLI (Command Line Interface)

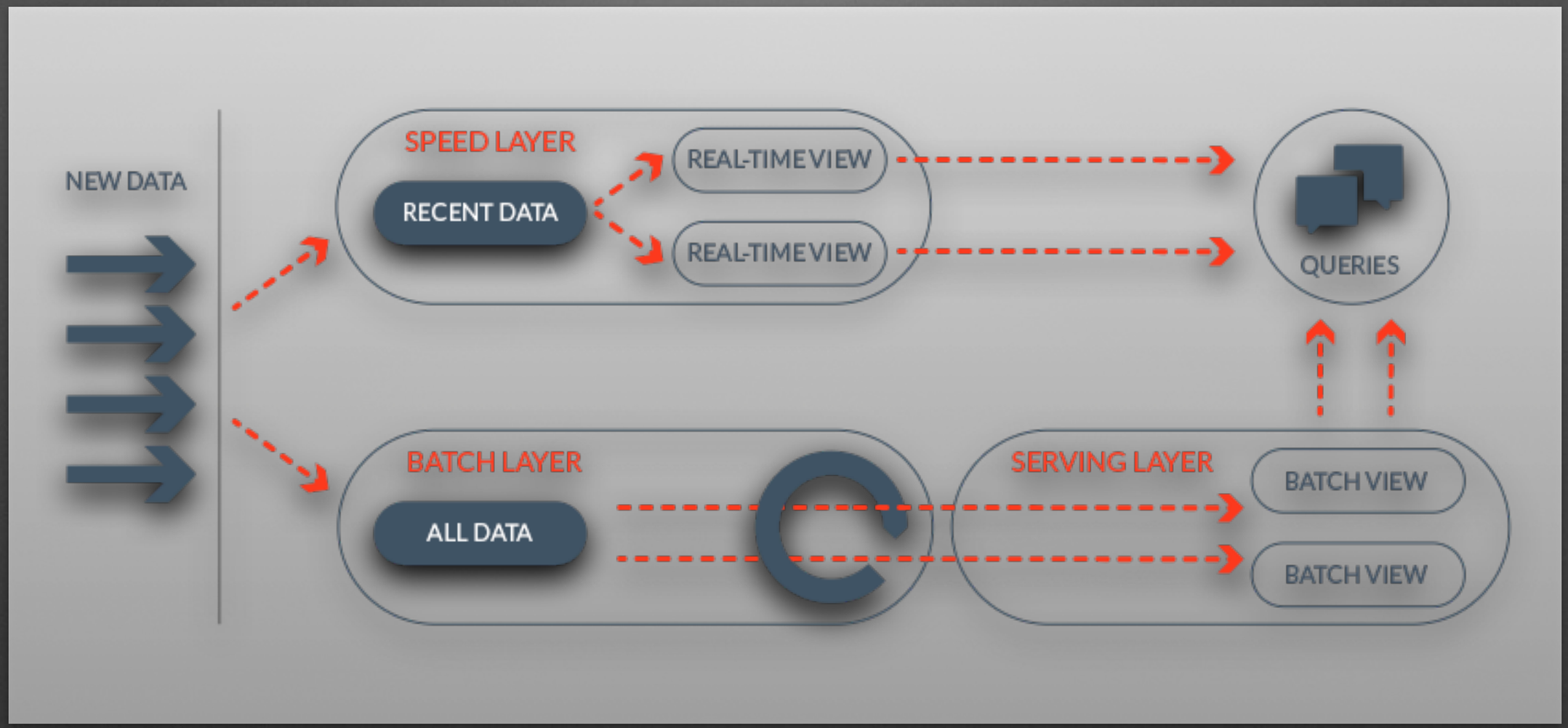
λ

Lambda Architecture

Lambda Architecture

- The term was coined by Nathan Marz
  - http://www.datasalt.com/2014/01/lambda-architecture-a-state-of-the-art/
- Decoupling batch processing and real-time processing

Lambda Architecture



Lambda Architecture

- Data is dispatched to either
    - The Batch Layer
        - Managing the master dataset (Immutable append-only set of raw data)
        - Pre-compute batch view
    - The Serving Layer
        - Indexes the batch view to support low-latency and ad-hoc queries
    - The Speed Layer
        - Deals with recent data only

Problem

- This really sucks, especially if you are a data scientist with no specific queries
  - Write code, compile, build, and package
  - Deploy
  - Run
  - Wait …
  - Output
  - Error/Wrong Query
  - Start over

Solution

Apache Spark

https://spark.apache.org/

History

- 2010
  - “Spark: Cluster Computing with Working Sets”
    - Published by Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica at UC, Berkeley
    - http://www.cs.berkeley.edu/~matei/papers/2010/hotcloud_spark.pdf

Overview

- Written in Scala
- Interactive/Real-time processing

Overview

- A Data analytics cluster computing framework
- In-memory cluster computing
    - Allows applications to load data into the memory of a cluster and query it repeatedly
- Built atop of HDFS but not tied to MapReduce
- Faster than MapReduce
    - 100x in-memory
    - 10x on disk

Overview

- A unified platform on top of Spark execution engine
    - Shark (Full Hive support)
    - Spark Streaming
    - GraphX
    - MLib (Machine Learning Library out of the box)

Overview

- Upcoming
  - Java 8 support
  - Spark SQL (Not just Hive)
  - BlinkDB for approximations (Approximate queries)
  - Spark R (R wrapper for Spark)
- Spork (Porting Pig Scripts to Spark)

STORM vs.

- Spark is
- Much higher throughput
- Much faster

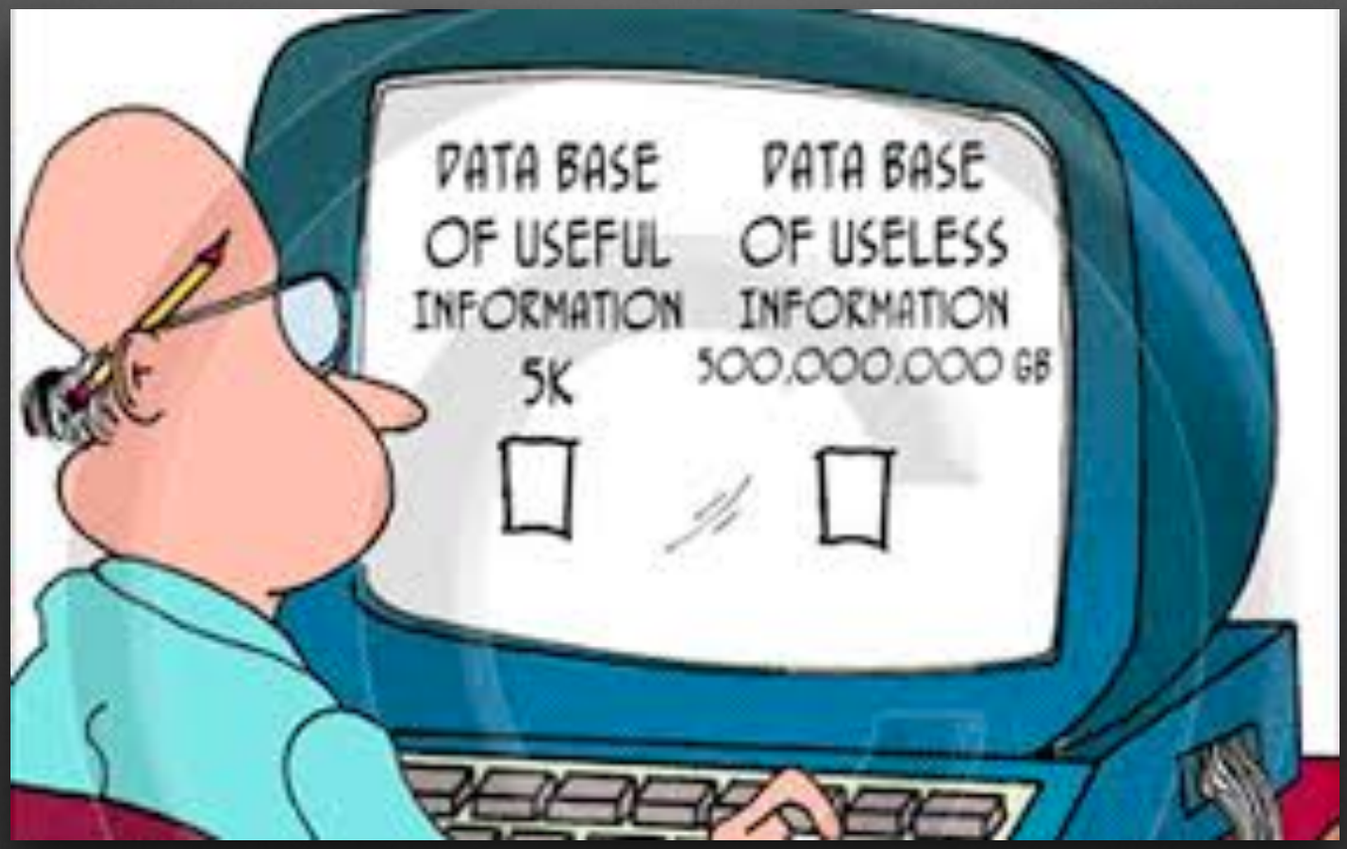
Architecture

- RDD (Resilient Distributed Datasets)
    - Immutable, In-memory, and Fault-tolerant
    - Operations on parallelized collections of element
        - Transformations
            - map, filter, union, sample, groupBy, join, etc…
        - Actions
            - collect, count, first, takeSample, forEach, etc…
        - Persist/Caching
            - Multiple storage levels (memory, disk, both)
- DAG (Directed Acyclic Graph) execution engine

Interface

- Interactive shell
  - spark-shell (Scala)
  - pyspark (Python)

Enterprise Data Lake



Enterprise Data Lake

- All data is stored in a centralized Hadoop repository
- All data is raw
    - Schema on-read as opposed to schema on-write
    - No upfront data modeling
- All data is accessible
- All other data processing systems are downstream including the data warehouse

File System

Flat-File Database

Relational Database

Analytics

Transactions

Data Warehouse
(OLAP)

Relational Database
(OLTP)

Big Data

NoSQL

Big Data
Analytics

Big Data
Transactions

Interactive

Real-Time
Streaming

Batch

Key-Value

Document

Columnar

Graph

Thank You!

@PolymathicCoder