

FOSSCOM 2016

Big Data Streaming processing using Apache Storm

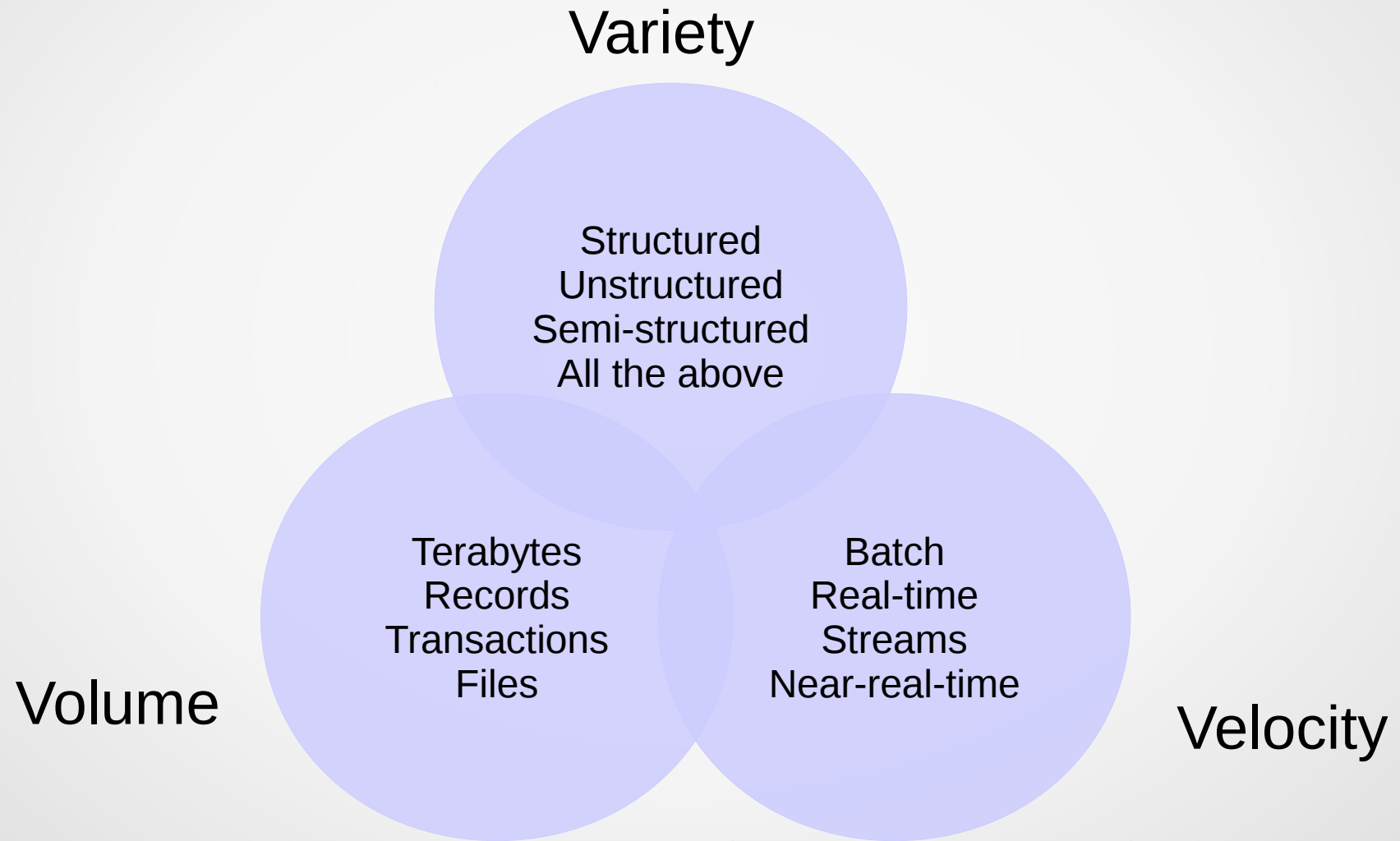
Agenda

- Big Data concepts
- Batch & Streaming processing
- NoSQL persistence
- Apache Storm and Apache Kafka
- Streaming application demo
- Considerations on Big Data applications

Who we are?

- Adrianos Dadis (@qiozas )
- Patroclos Christou
- Eleftheria Chavelia
- Sofia Nomikou

Big Data characteristics



Use cases

- 360-degree customer view
- Internet of Things
- Data warehouse optimization
- Information security
- Sentiment analysis
- Urban Planning (MIT)

Big Data Pros

- Better user experience
- Predictive analytics
- Answer complex business questions

Big Data Cons

- Privacy
- False truths
- Strict personalized content

Databases

- Persistency & Concurrency
- Relational Databases provide strong benefits
- Relational Databases pitfalls:
 - Application development productivity
 - Large-scale data
 - Capture more data
 - Process data faster
 - Costs

NoSQL

- Schema agnostic
- Nonrelational
- Commodity hardware
- Highly distributable
- NOT a silver bullet

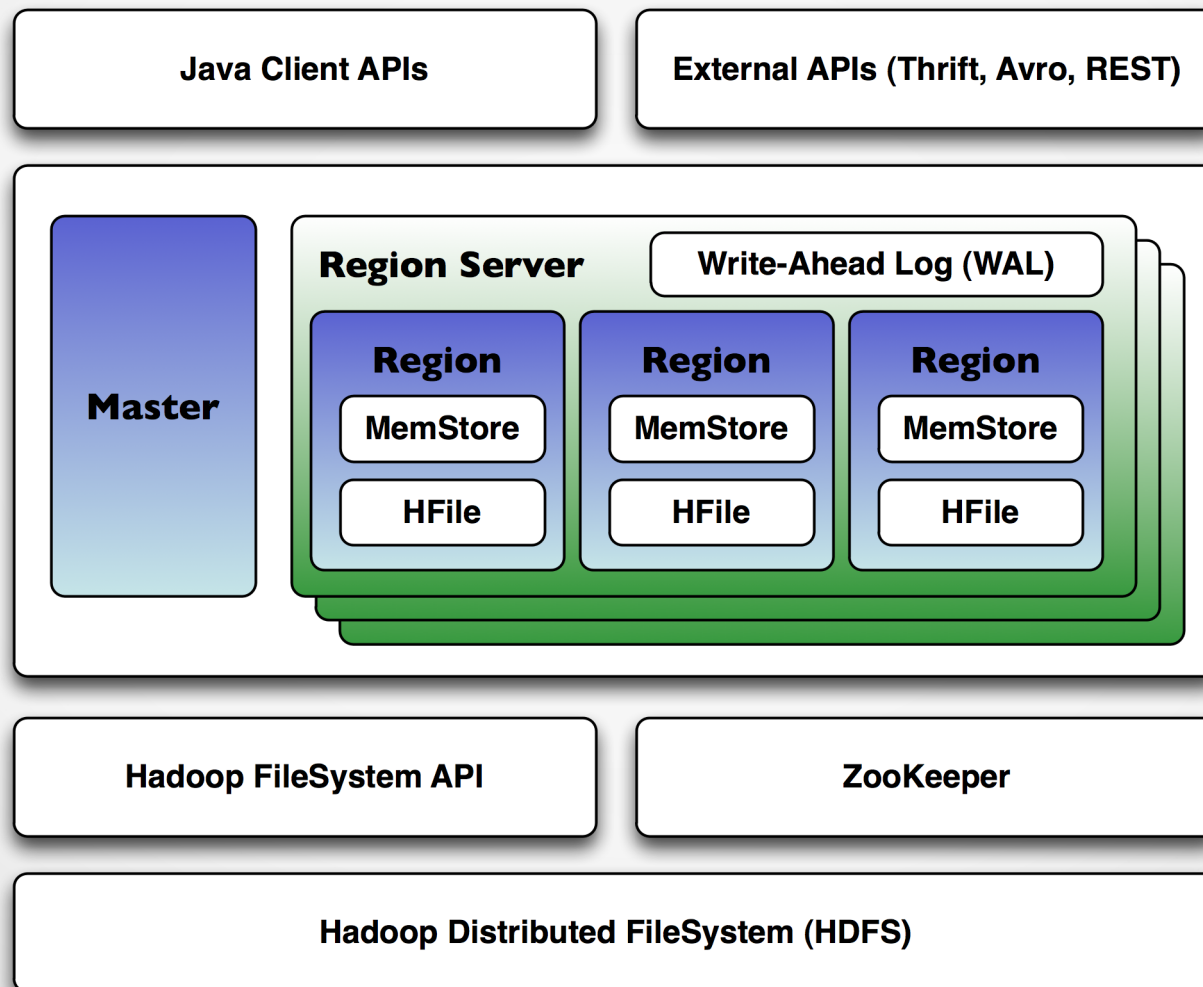
NoSQL & CAP theorem

- Consistency
 - all nodes see the same data at the same time
- Availability
 - a guarantee that every request receives a response about whether it succeeded or failed
- Partition tolerance
 - the system continues to operate despite arbitrary partitioning due to network failures
- CP or AP

NoSQL Data Models

- Key Value
 - Riak, Redis, Aerospike
- Document
 - MongoDB, Elasticsearch
- Column Family
 - Cassandra, HBase
- Graph
 - Neo4J, ArangoDB, OrientDB

Apache HBase



Big Data processing modes

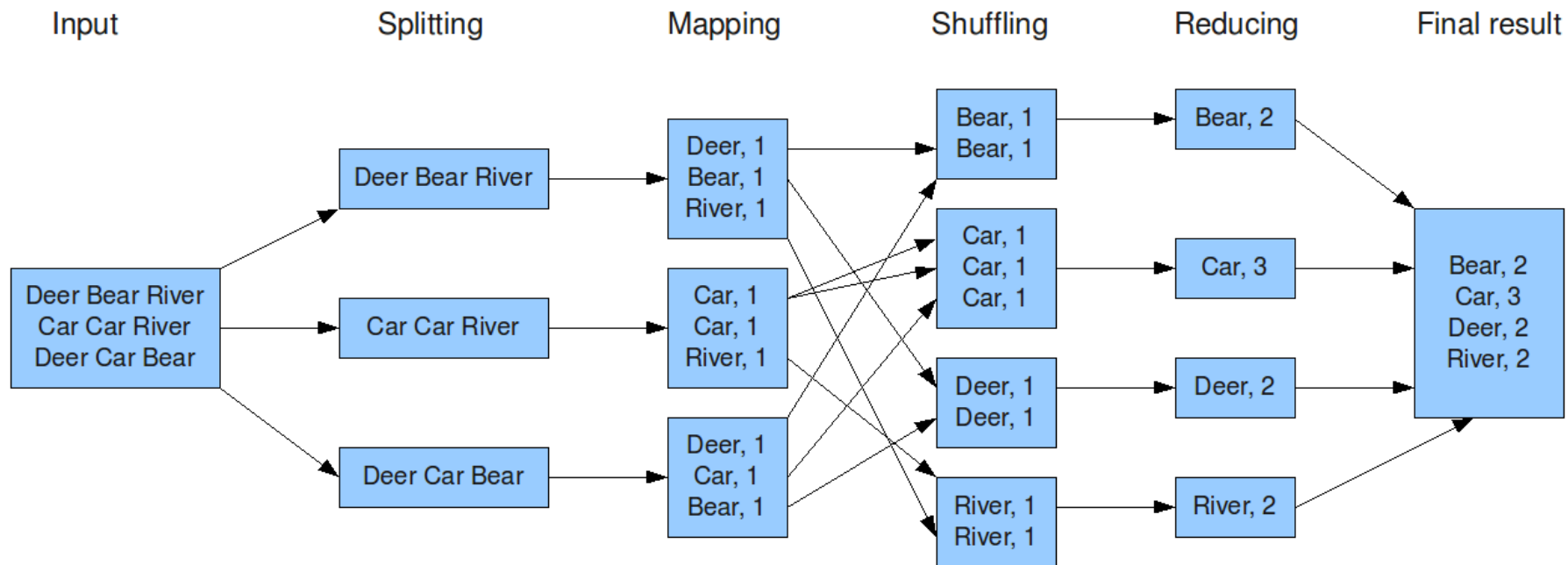
- Batch
 - Long running jobs
- Streaming processing
 - One-at-a-time
 - Micro batch

Apache Hadoop - MapReduce

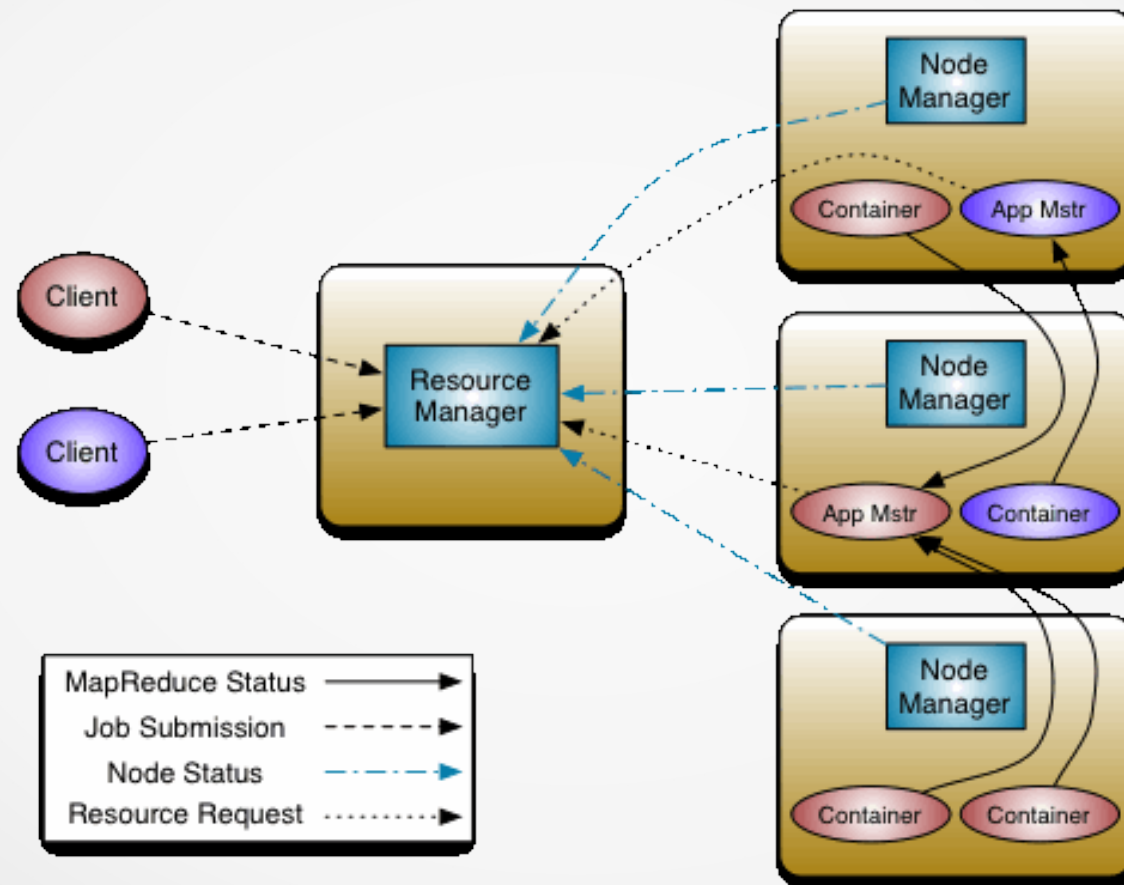
- Programming/Processing model
- HDFS
- Steps:
 - **Map**
 - **Shuffle**
 - **Reduce**
- Apache Hadoop ecosystem
 - Pig
 - Hive
 - *more*

Map Reduce – word count

The overall MapReduce word count process



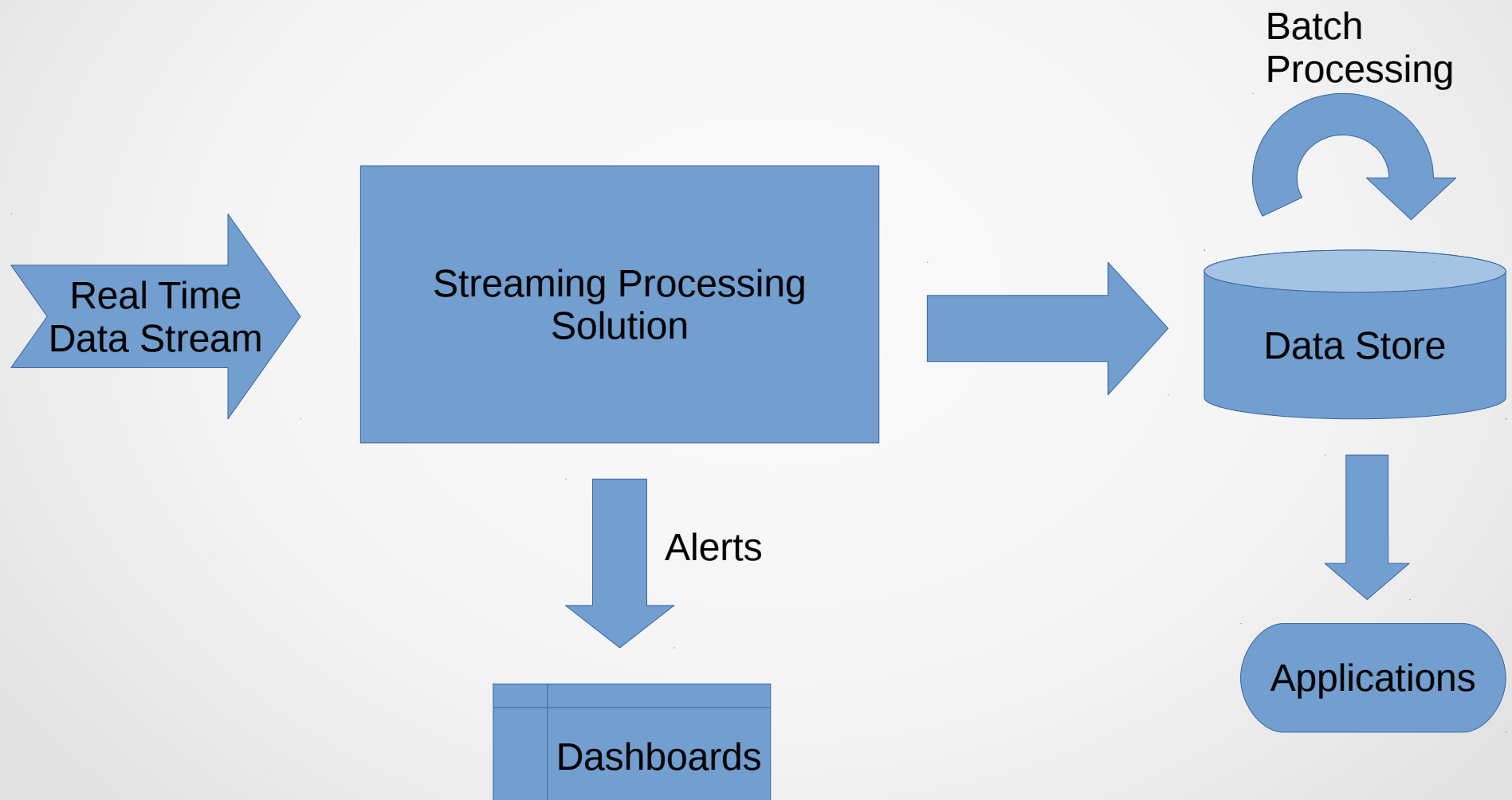
YARN



Apache Spark (batch processing)

- Similar to MapReduce but faster
- Resilient Distributed Dataset
- Ecosystem
 - Spark SQL
 - MLlib
 - GraphX
 - Streaming
 - Scala / R / Python / Java / SQL

Streaming processing



Streaming processing Use Cases

- Prevent security fraud
- Optimize order routing
- Predictive maintenance
- Optimize personalized content
- Optimize prices or offers
- Prevent stock outs
- Optimize bandwidth allocation
- Prevent application failures

Streaming processing frameworks

- Apache Storm
- Apache Spark Streaming
- Apache Flink
- Apache Samza

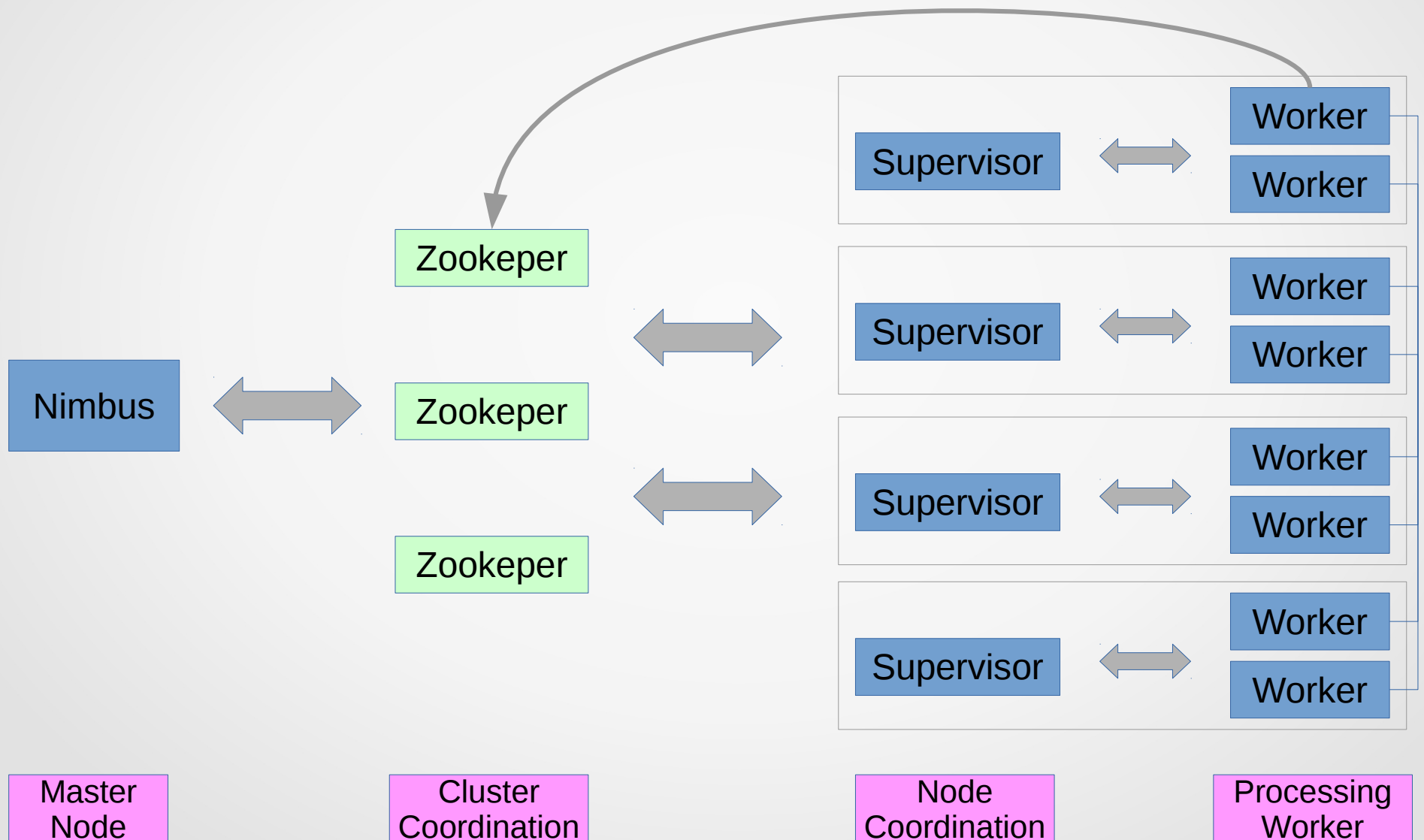
Apache Storm

- Creator: Nathan Marz
- Distributed real-time computation system for processing large volumes of high-velocity data
- Most popular engine in industry
- Characteristics:
 - Fast
 - Scalable
 - Fault-tolerant
 - Reliable
 - Easy to operate
 - Easy to develop

Storm modes

- One-at-a-time processing (pure Storm)
 - Very low latency
 - Very simple development model
 - At-Most-Once and At-Least-Once semantics
- Micro batch processing (Storm Trident)
 - Increased latency on event
 - Better throughput for large rates (it depends)
 - More complex development
 - Exactly-Once semantics (per batch)

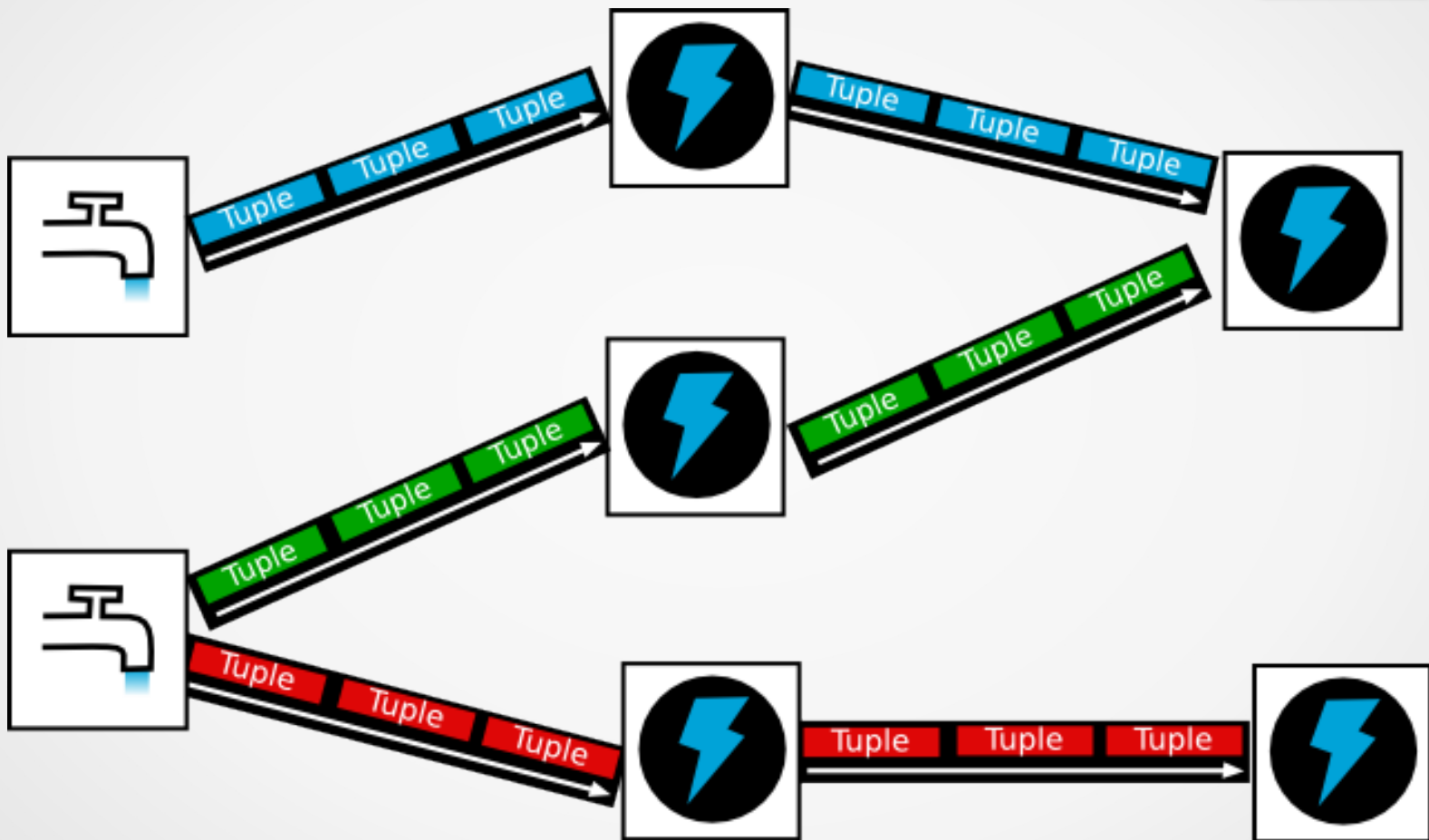
Storm Architecture



Storm concepts

- **Topology** : A graph of computation
- **Tuple** : Storm uses tuples as its data model
- **Stream** : An unbounded sequence of tuples
- **Spout** : A source of streams in a topology
- **Bolt** : All processing in topologies is done in bolts
- **Stream Grouping** : Defines how that stream should be partitioned among the bolt's tasks.

Storm topology



Storm topology example

```
Config conf = new Config();
conf.setNumWorkers(2); // use two worker processes

topologyBuilder.setSpout("blue-spout", new BlueSpout(), 2); // set parallelism hint to 2

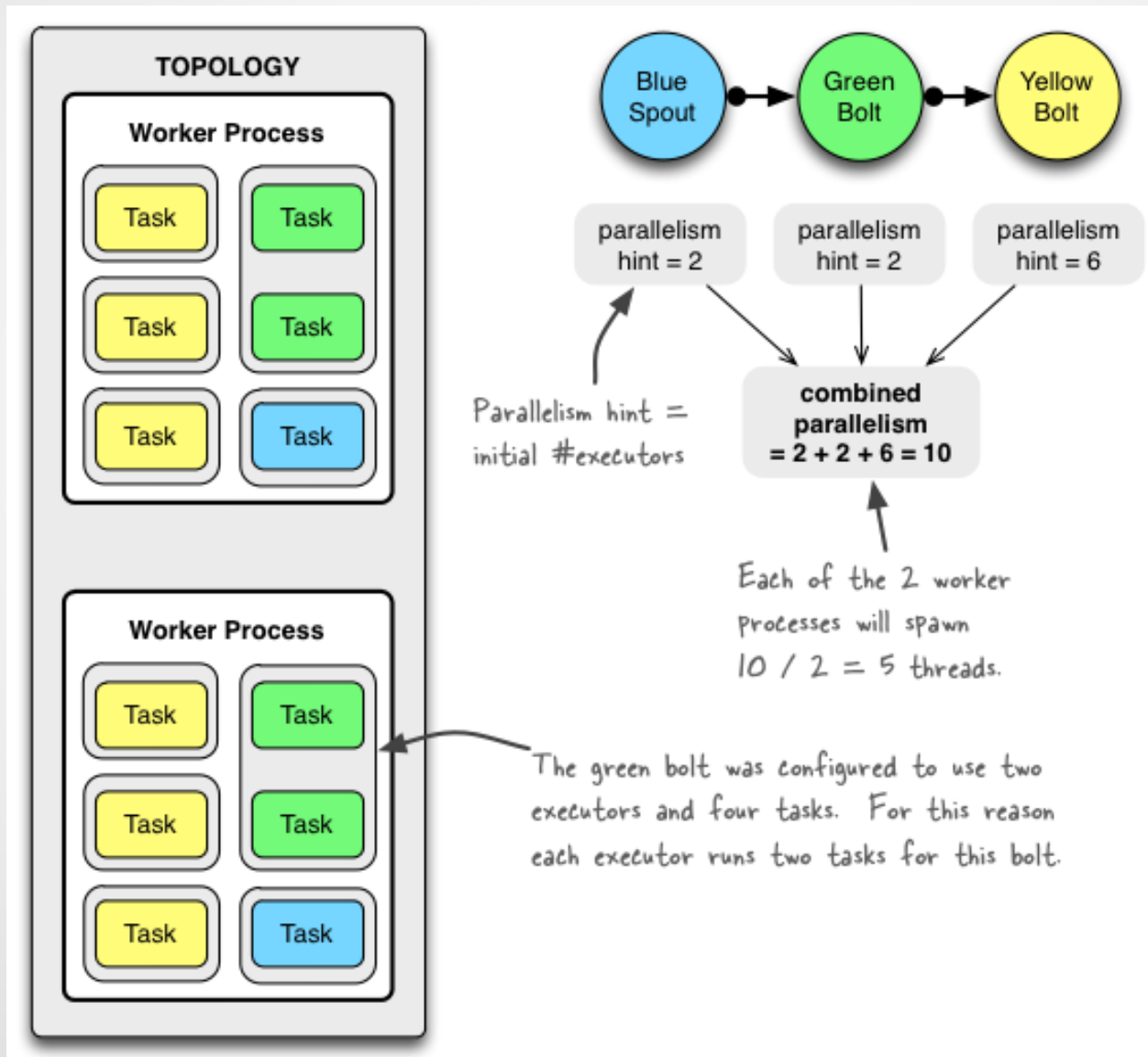
topologyBuilder.setBolt("green-bolt", new GreenBolt(), 2)
    .setNumTasks(4)
    .shuffleGrouping("blue-spout");

topologyBuilder.setBolt("yellow-bolt", new YellowBolt(), 6)
    .shuffleGrouping("green-bolt");

StormSubmitter.submitTopology(
    "mytopology",
    conf,
    topologyBuilder.createTopology()
);
```



Storm topology parallelism



Storm Trident

the cow jumped over the moon
the man went to the store and bought some candy
four score and seven years ago
how many apples can you eat
the cow jumped over the moon
the man went to the store and bought some candy
four score and seven years ago
how many apples can you eat
the cow jumped over the moon
the man went to the store and bought some candy



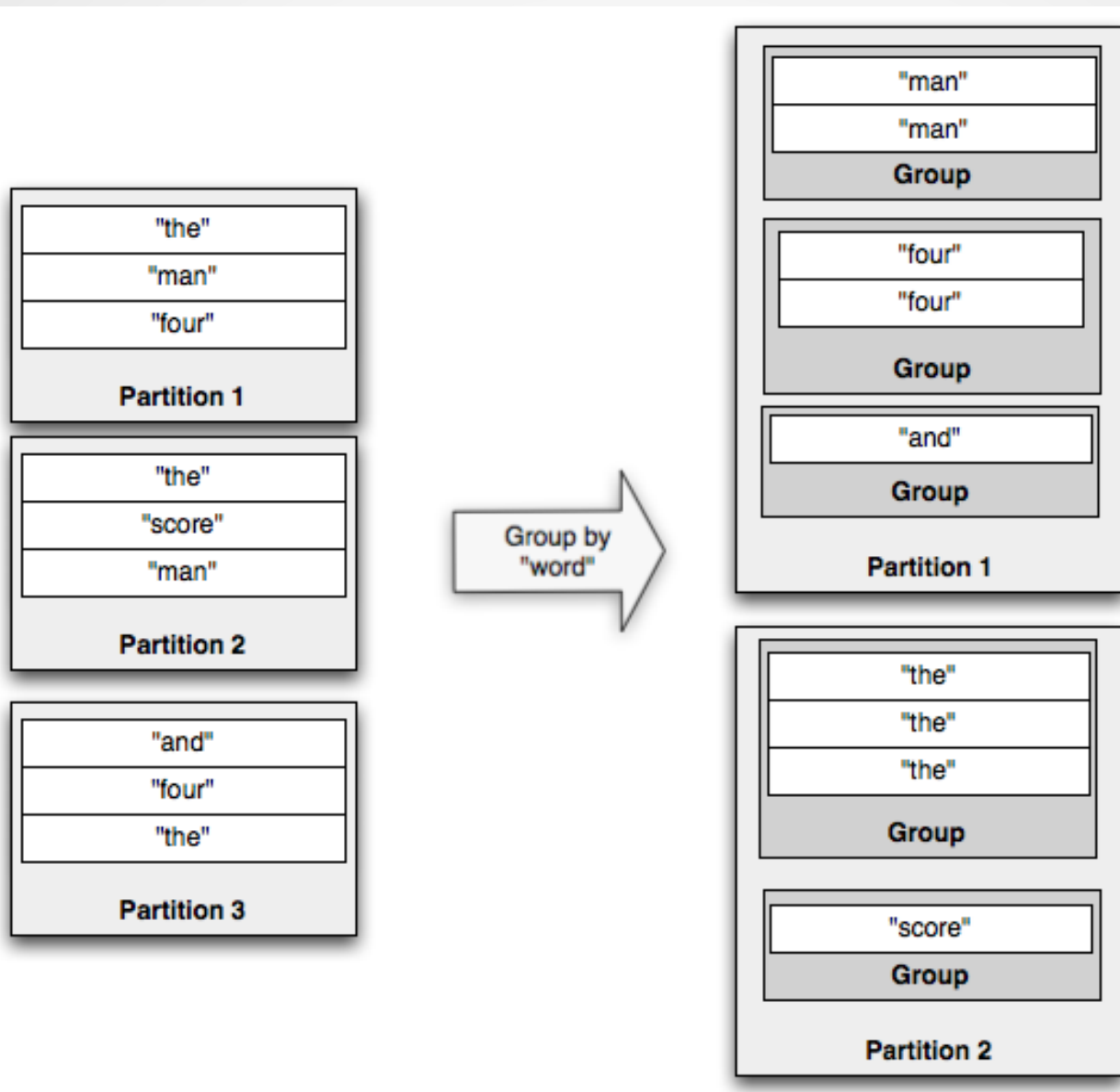
the cow jumped over the moon
the man went to the store and bought some candy
four score and seven years ago
Batch 1

how many apples can you eat
the cow jumped over the moon
the man went to the store and bought some candy
four score and seven years ago
how many apples can you eat
Batch 2

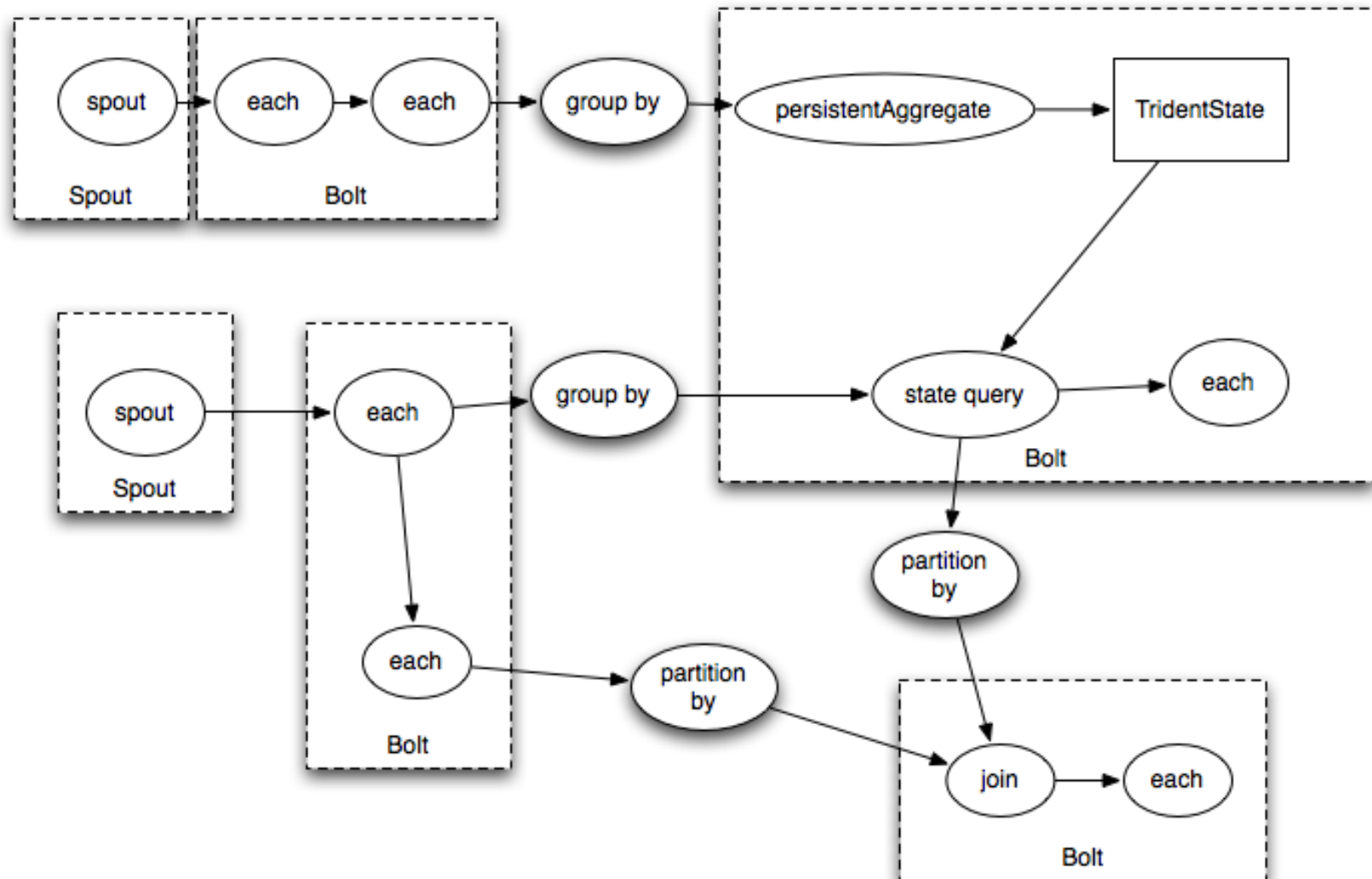
the cow jumped over the moon
the man went to the store and bought some candy
Batch 3

```
TridentTopology topology = new TridentTopology();
TridentState wordCounts =
    topology.newStream("spout1", spout)
        .each(new Fields("sentence"), new Split(), new Fields("word"))
        .groupBy(new Fields("word"))
        .persistentAggregate(new MemoryMapState.Factory(), new Count(), new Fields("count"))
        .parallelismHint(6);
```

Storm Trident partitioning



Storm Trident

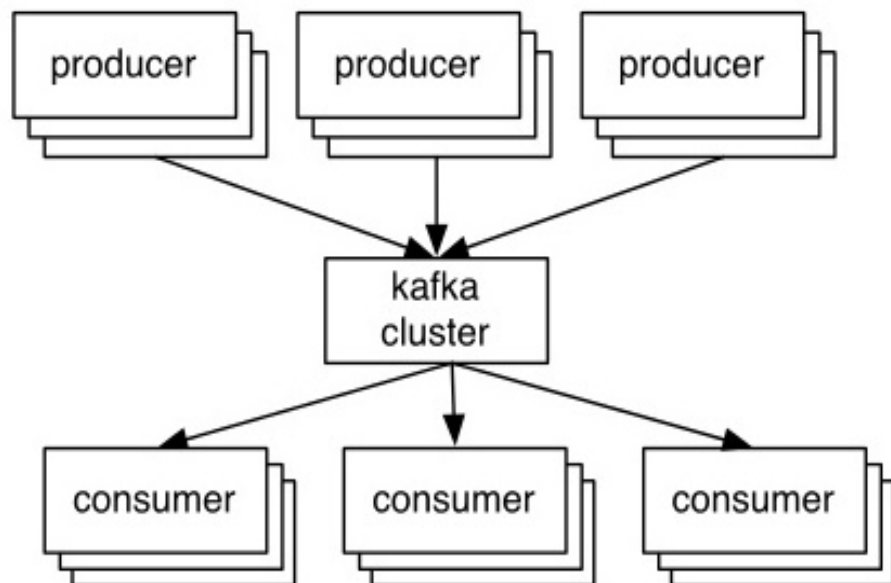


Messaging Systems

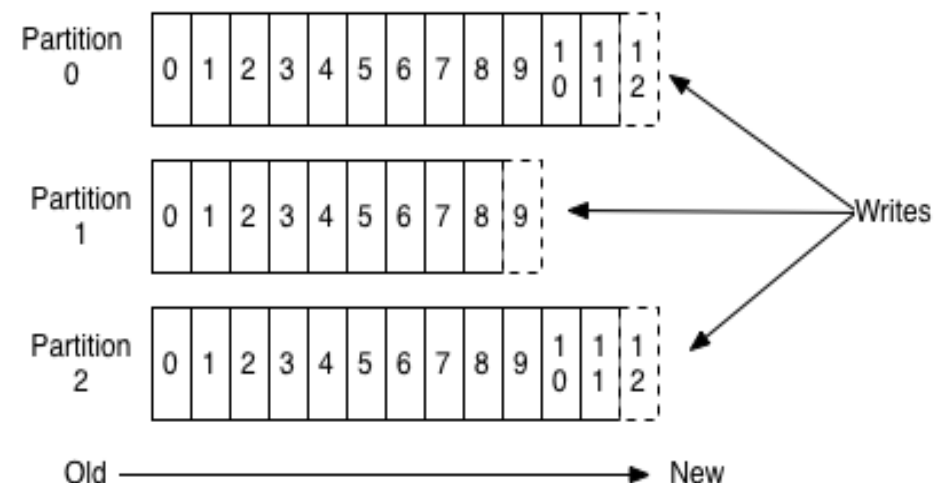
- Decouple processing from data producers
- Buffer unprocessed messages
- Models: queuing & publish-subscribe
- Frameworks
 - Kafka
 - RabbitMQ
 - ActiveMQ

Apache Kafka

- Distributed, partitioned, replicated commit log service
- Maintains feeds of messages in Topics
- Publish-Subscribe model

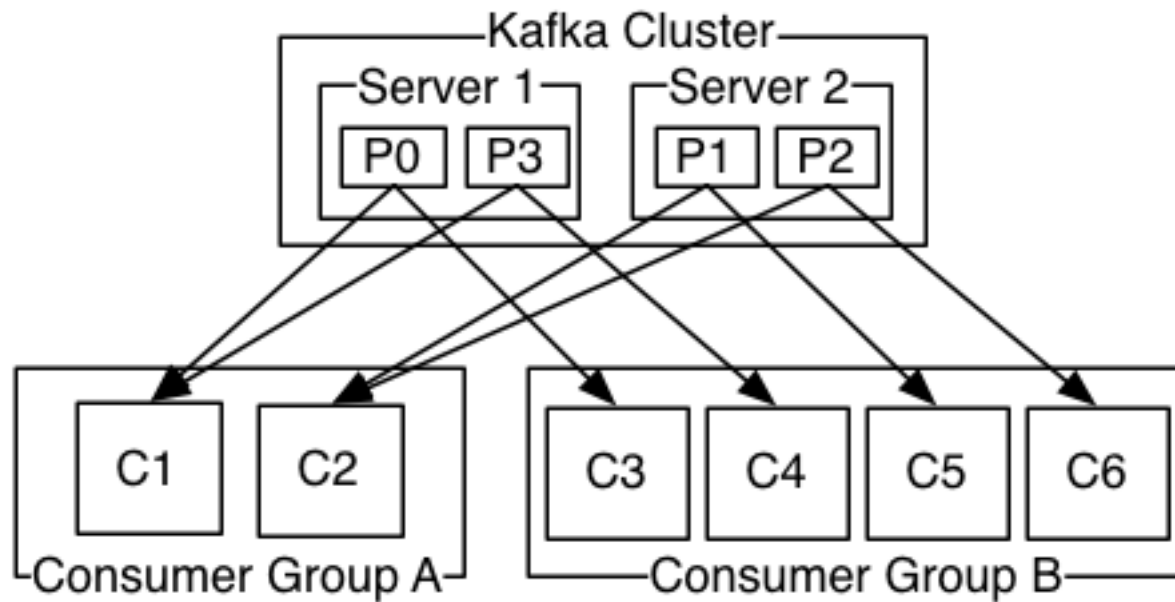


Anatomy of a Topic



Apache Kafka

- Consumers coordinate what to read
- Consumer & Consumer Group
- Brokers



Big Data Apps common problems

- Integrate many input sources
- Performance & Scalability
- Complex infrastructure
- Hard to debug
- Difficult to select correct frameworks

Big Data Apps hints

- Failure (HW or data) is a normal case
- Design for scalability from day one
- Queries drive schema design
- Continuous Integration
- Metrics from day one
- Performance tests (real data)
- Monitor
- Appropriate people

Demo Architecture

- TODOs

Thank YOU :-)

Questions ???