saltmarch MEDIA™

GREAT INDIAN DEVELOPER SUMMIT™

April 25-28, 2017. Bangalore

Spark, Mesos, Akka, Cassandra and Kafka (SMACK) Stack Real Time Big Data

Rohit Bhardwaj
Principal Cloud Engineer

SMACK AGENDA

- SMACK stack overview
- Bigdata characteristics
- Cassandra NoSQL database
- Spark Streaming
- Kafka and Akka event sourcing
- Mesos Cluster management
- Mesos Scheduling and execution

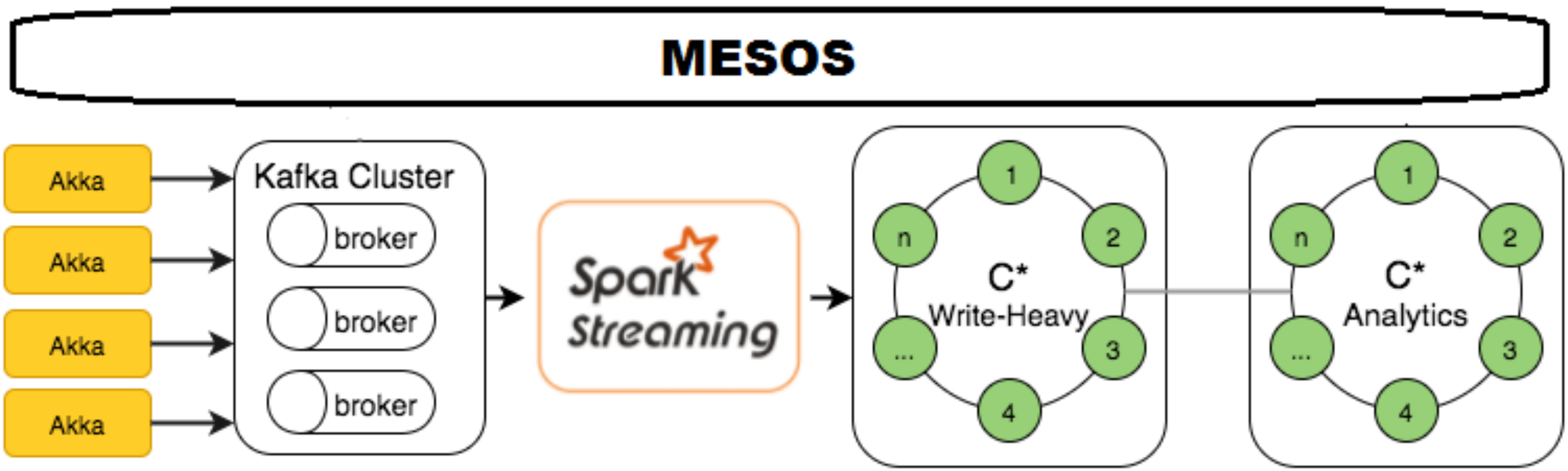
Spark for Micro-Batch Processing

Mesos for Cluster Management

Akka for Event Processing

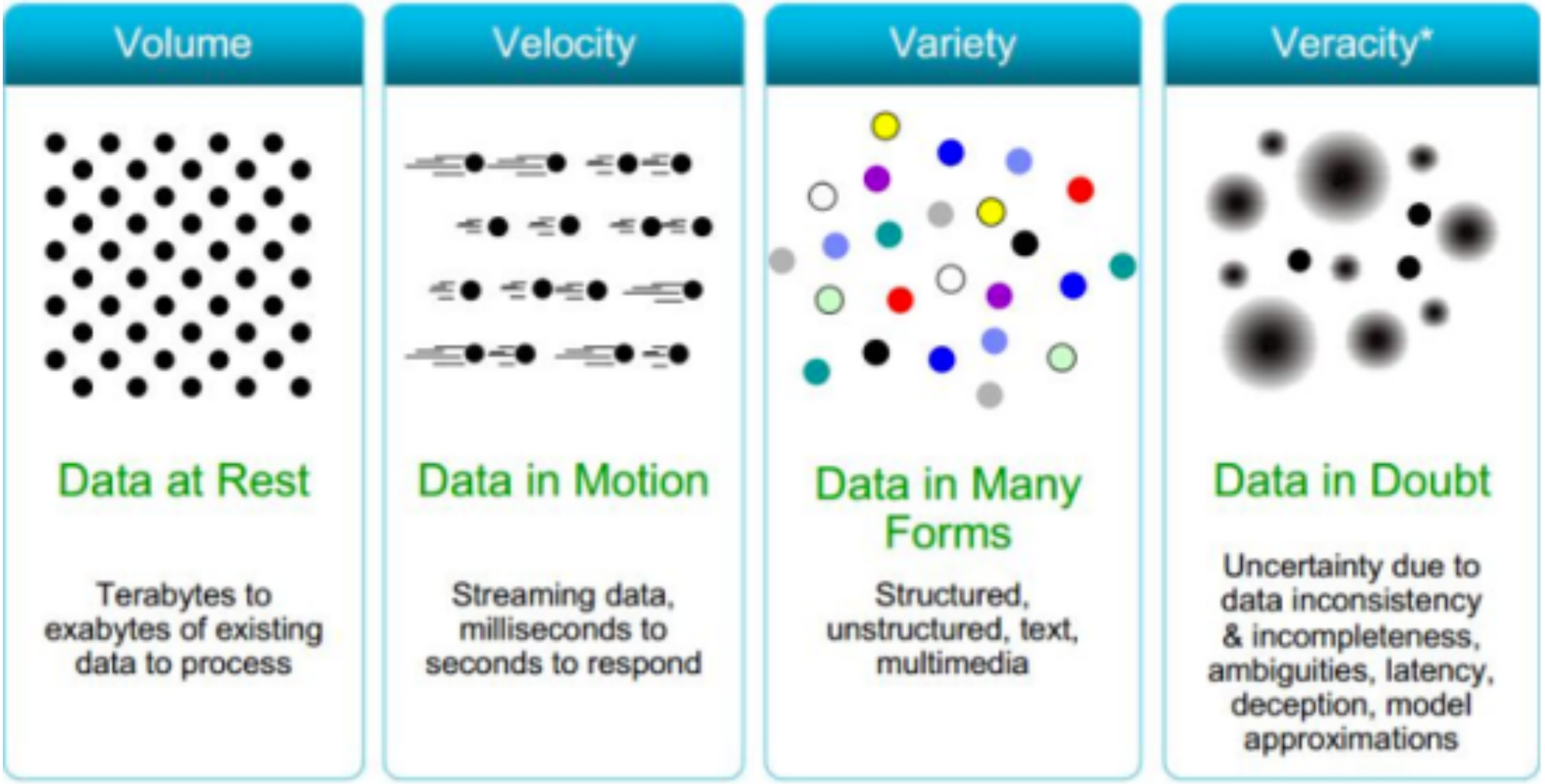
Cassandra for Persistence

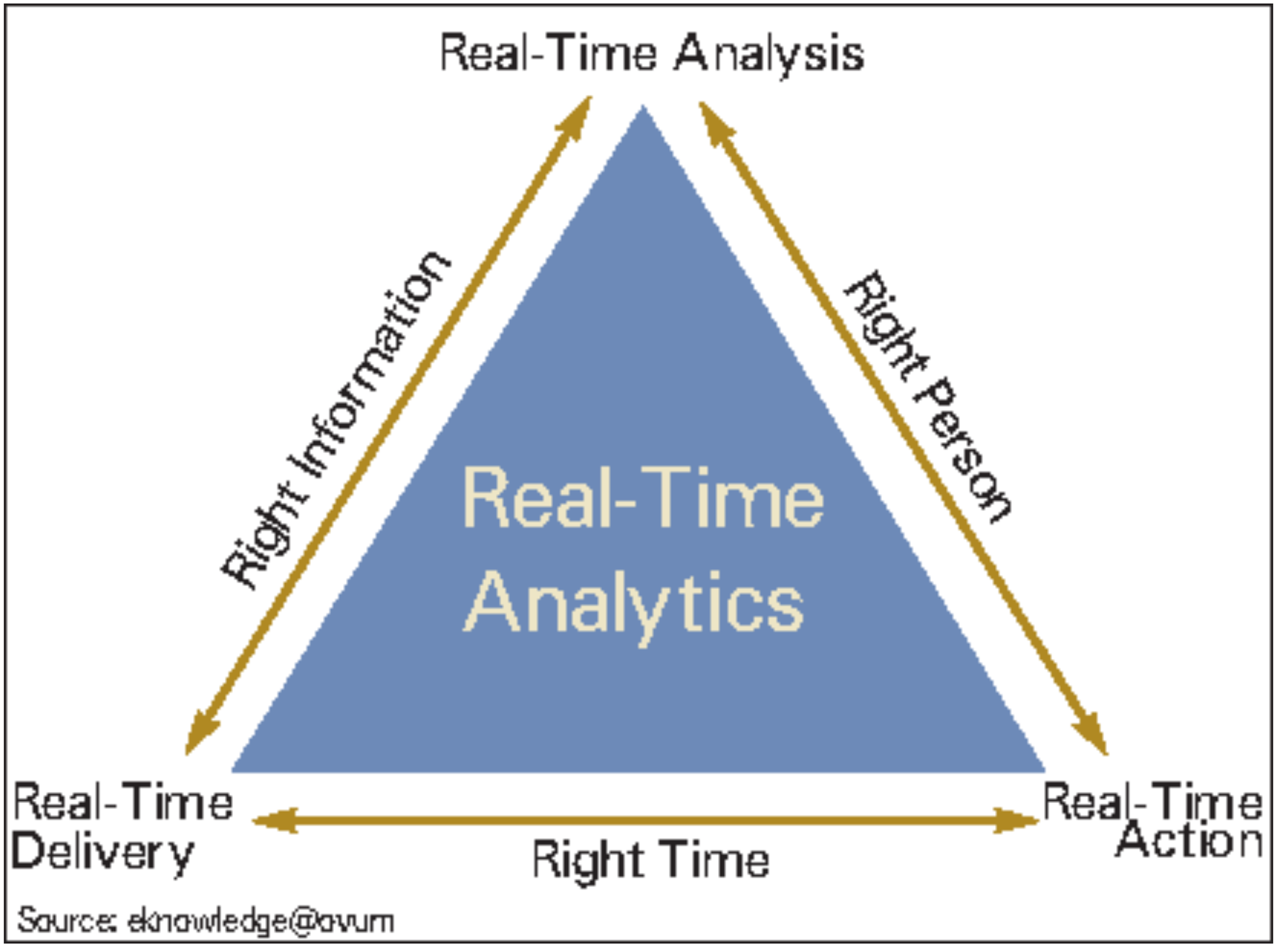
Kafka for Event Transport

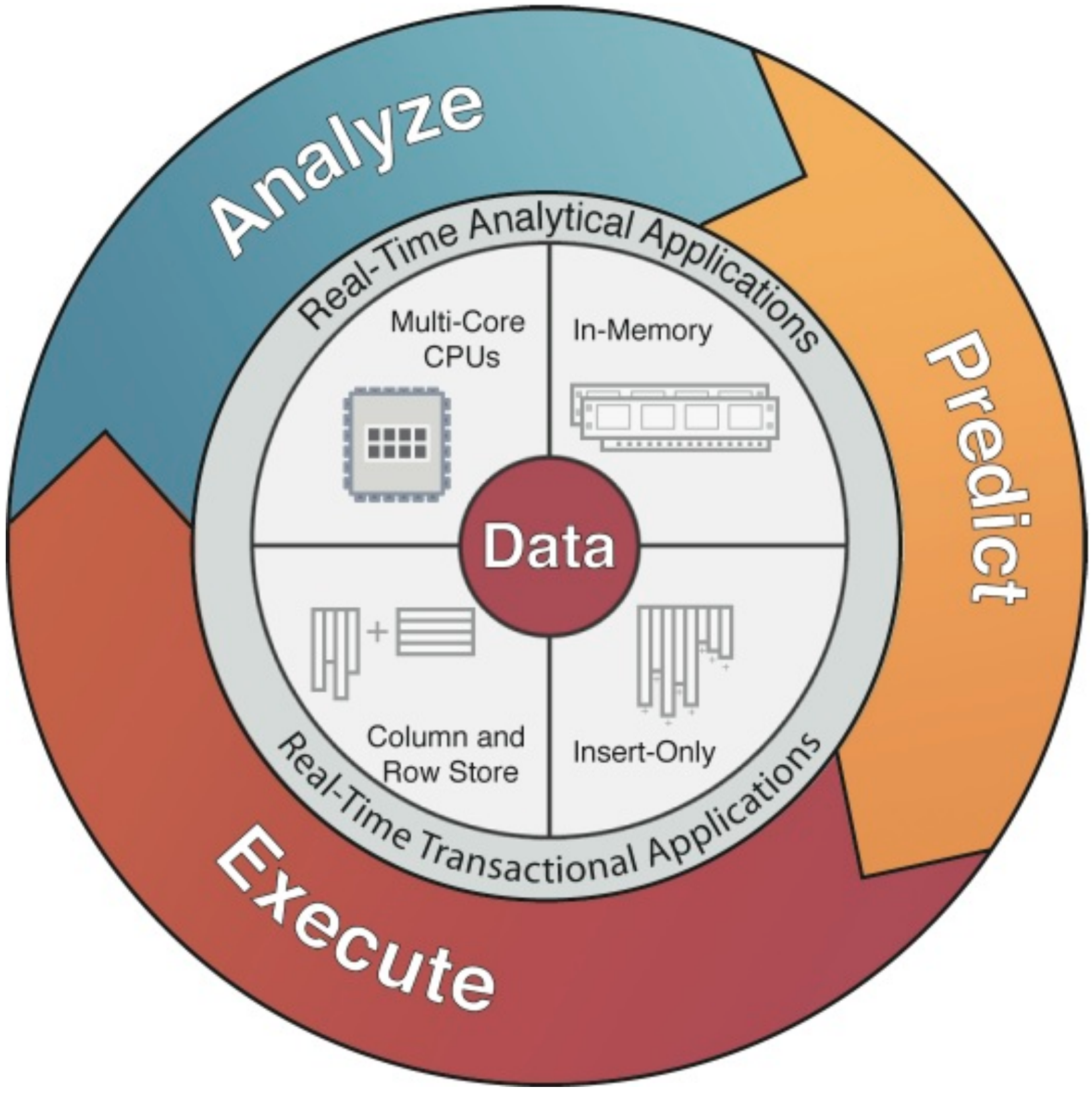


http://preview.tinyurl.com/kcqyjsl

BIG DATA CHARACTERISTICS




Real-Time Analysis
Right Information
Right Person
Real-Time Analytics
Real-Time Delivery
Right Time
Real-Time Action
Source: eknowledge@ovum

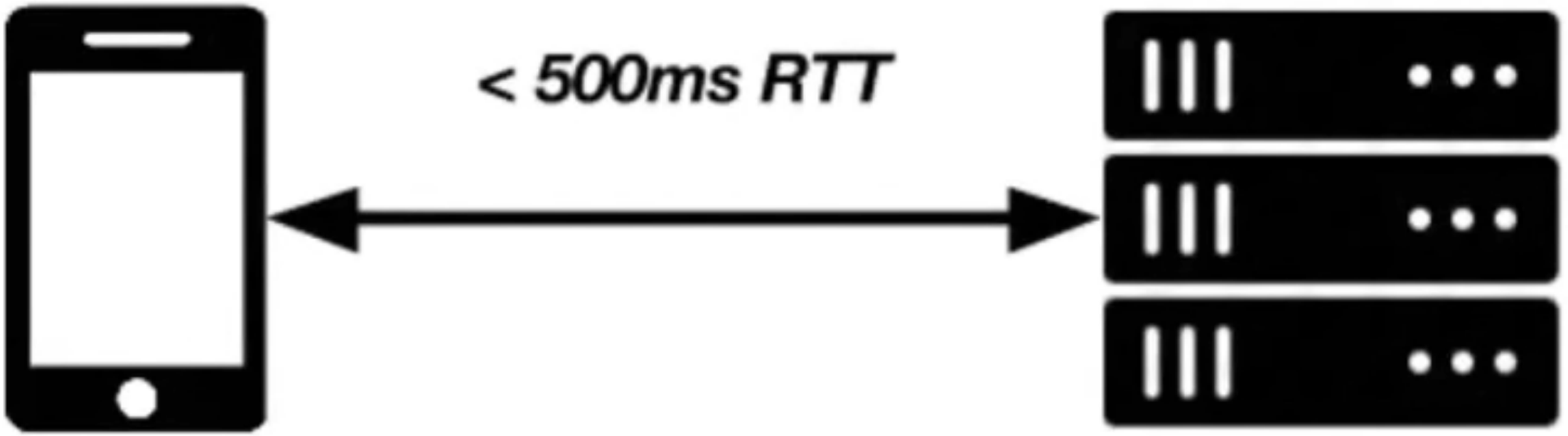


https://www.qubole.com/resources/solution/best-use-cases-for-big-data-analytics/

99.99% Uptime

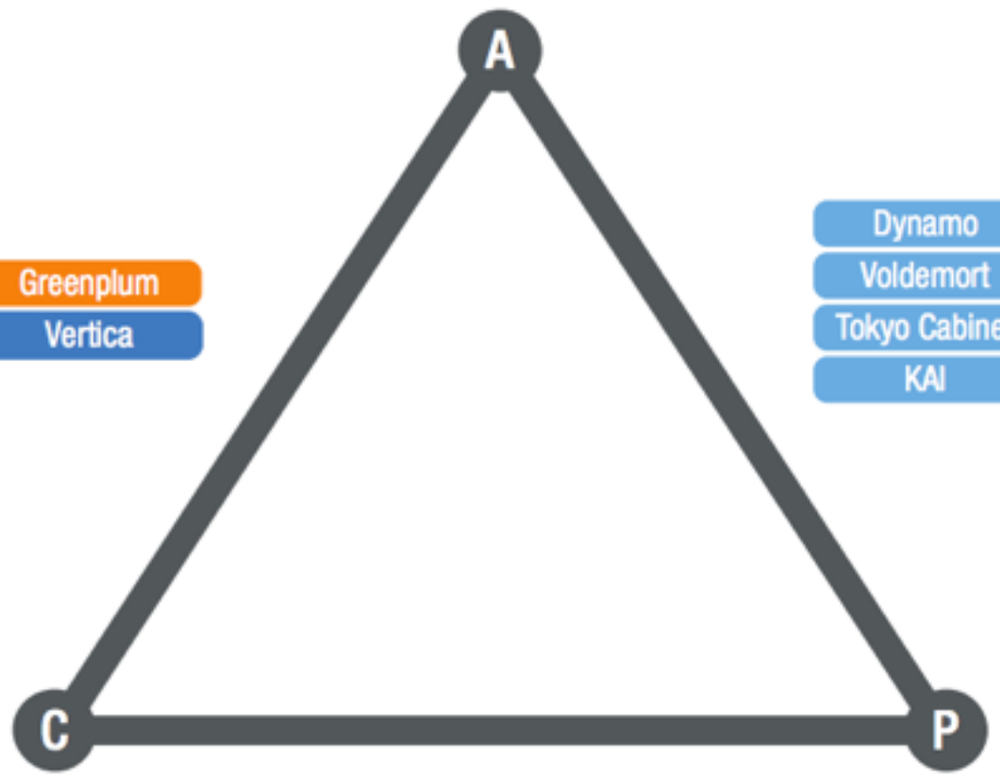
AVAILABILITY LEVEL[11]	ESTIMATED DOWNTIME PER YEAR
99.999%	5 minutes
99.99%	52 minutes
99.9%	8.5 hours
99%	3.5 days or about 87 hours

Subsecond End-User Latency



UBER

Availability:
Each client can always read and write



CA

RDBMSs	Greenplum
Aster Data	Vertica

AP

Dynamo	Cassandra
Voldemort	SimpleDB
Tokyo Cabinet	CouchDB
KAI	Riak

CP

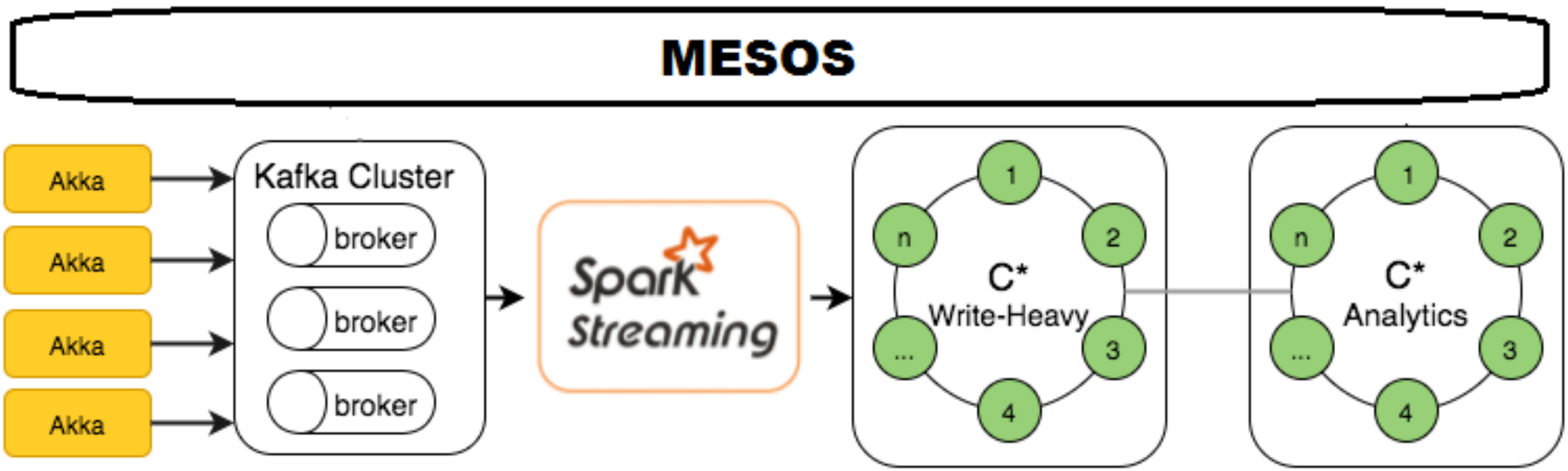
Big Table	Mongo DB	Berkeley DB
Hypertable	Terrastore	Memcache DB
Hbase	Scalaris	Redis

Consistency:
All clients always have the same view of the data

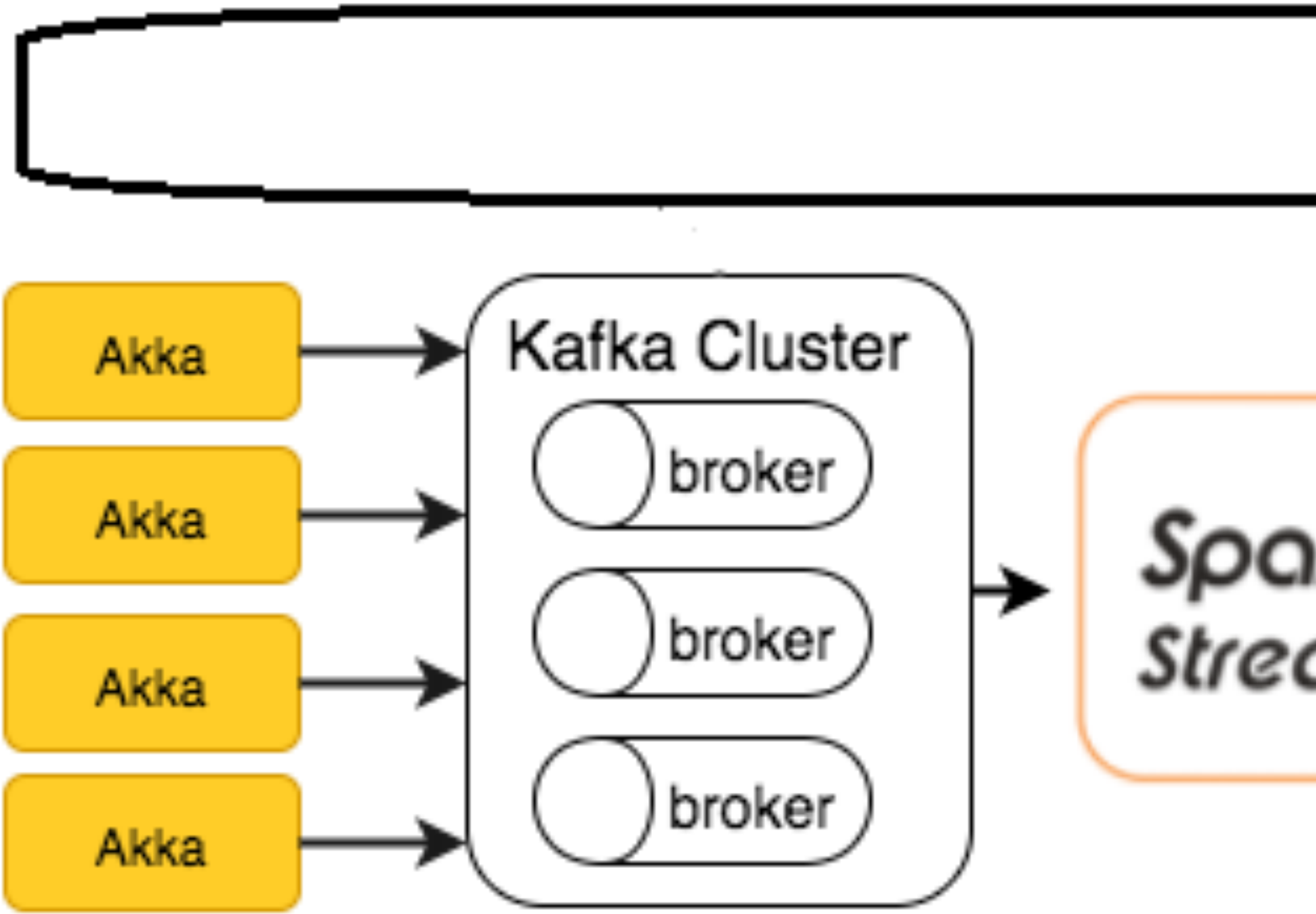
Partition Tolerance:
The system works well despite physical network partitions

Data Models

- Relational (Comparison)
- Key-Value
- Column-Oriented/Tabular
- Document-Oriented

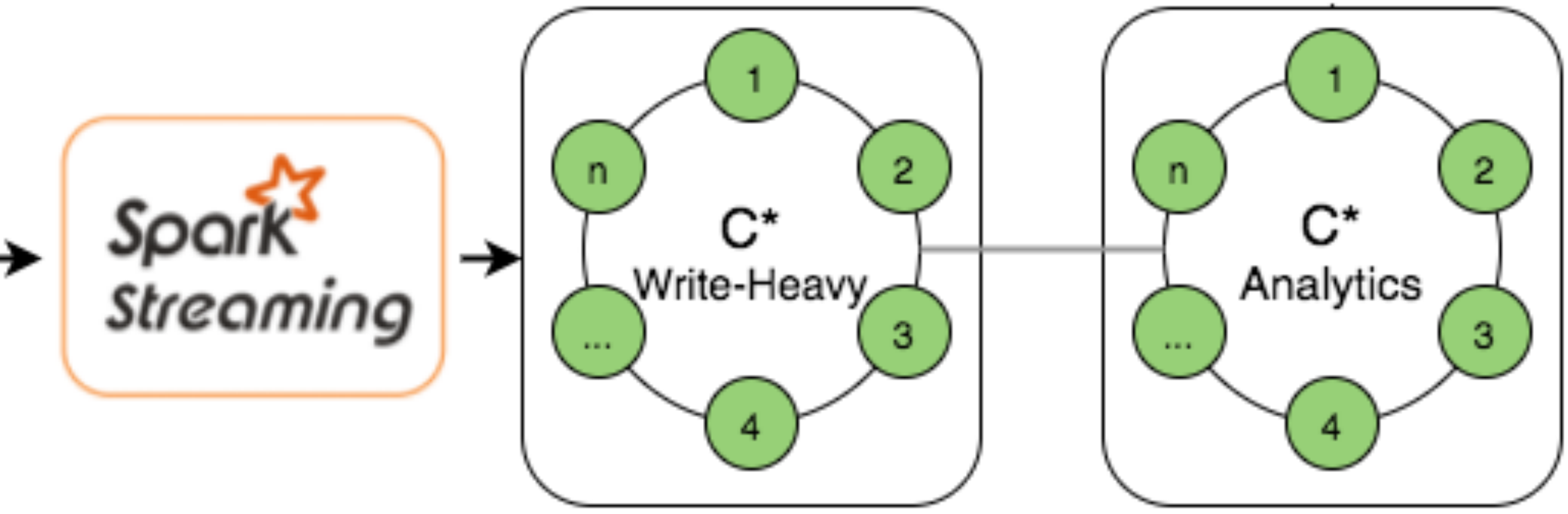


http://preview.tinyurl.com/kcqyjsl



http://preview.tinyurl.com/kcqyjsl

MESOS



http://preview.tinyurl.com/kcqyjsl

Cassandra

A highly scalable, eventually consistent, distributed, structured key-value store.

Bigtable

Google

Dynamo

amazon.com

facebook

Cassandra

cassandra.apache.org

Cassandra

Home | Download | Getting Started | Planet Cassandra | Contribute

Welcome | Video | Slides

Welcome to Apache Cassandra ™

The Apache Cassandra database is the right choice when you need scalability and high availability without compromising performance. Linear scalability and proven fault-tolerance on commodity hardware or cloud infrastructure make it the perfect platform for mission-critical data. Cassandra's support for replicating across multiple datacenters is best-in-class, providing lower latency for your users and the peace of mind of knowing that you can survive regional outages.

Cassandra's data model offers the convenience of column indexes with the performance of log-structured updates, strong support for denormalization and materialized views, and powerful built-in caching.

Download

Tick-Tock release **3.4** (Changes)
3.0.x release **3.0.4** (Changes)
2.2.x release **2.2.5** (Changes)

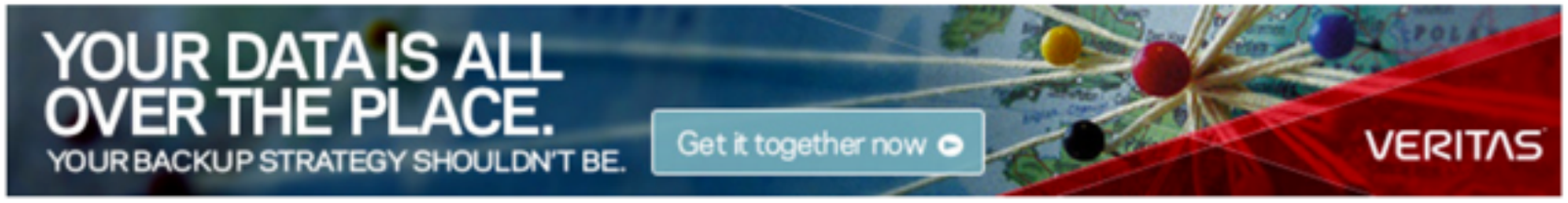
Download options

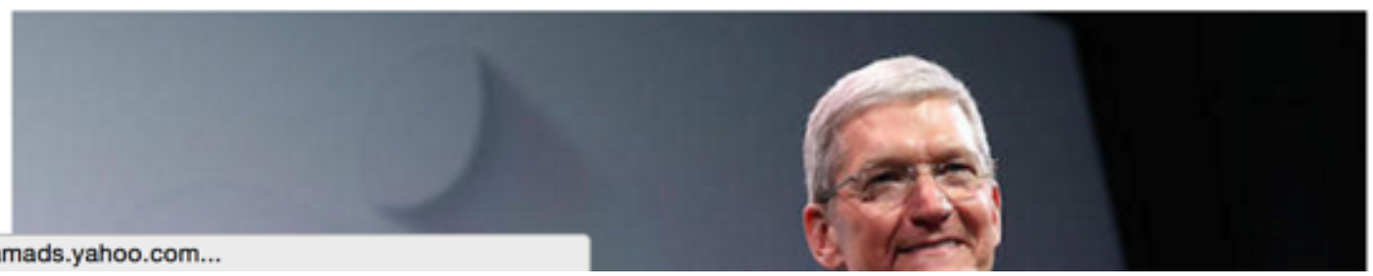
SOFTWARE

Apple's secret NoSQL sauce includes a hefty dose of Cassandra

If you want to scale like Apple, you might need to consider Cassandra. Matt Asay explains.

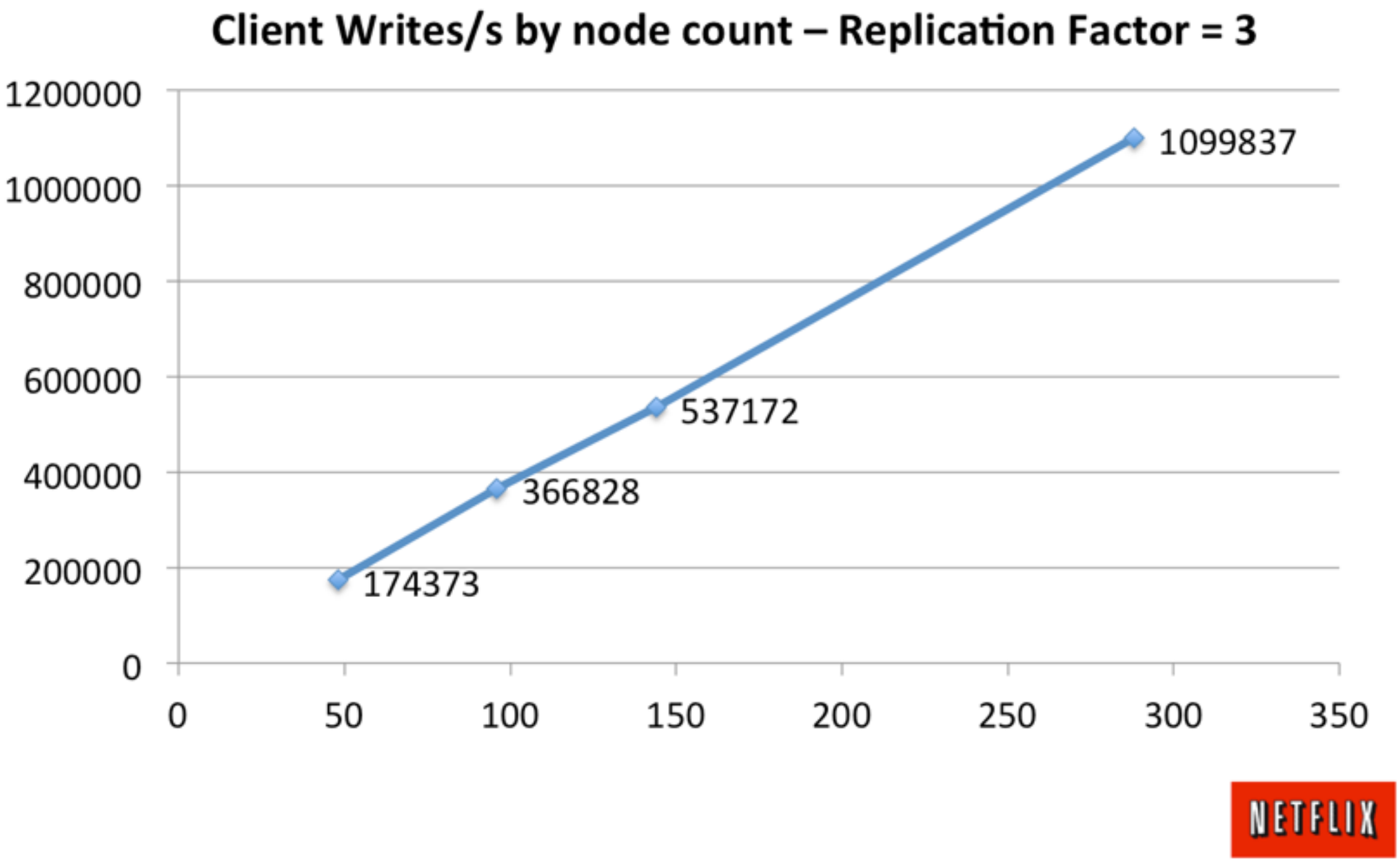
By Matt Asay | September 16, 2015, 6:49 AM PST





for syndication.streamads.yahoo.com...

Scale-Up Linearity



http://image.slidesharecdn.com/cassandraeu1-120330040410-phpapp01/95/cassandra-eu-2012-netflixs-cassandra-architecture-and-open-source-efforts-5-728.jpg?cb=1333080333

NETFLIX

intuit.

CALL OF DUTY

twitter

ebay

Expedia

NASA

CISCO

for more, see PlanetCassandra.org/companies/

Netflix case study

Consumer Electronics

AWS Cloud Services

CDN Edge Locations

Customer Device (PC, PS3, TV…)

Web Site or Discovery API

Streaming API

OpenConnect CDN Boxes

User Data

Personalization

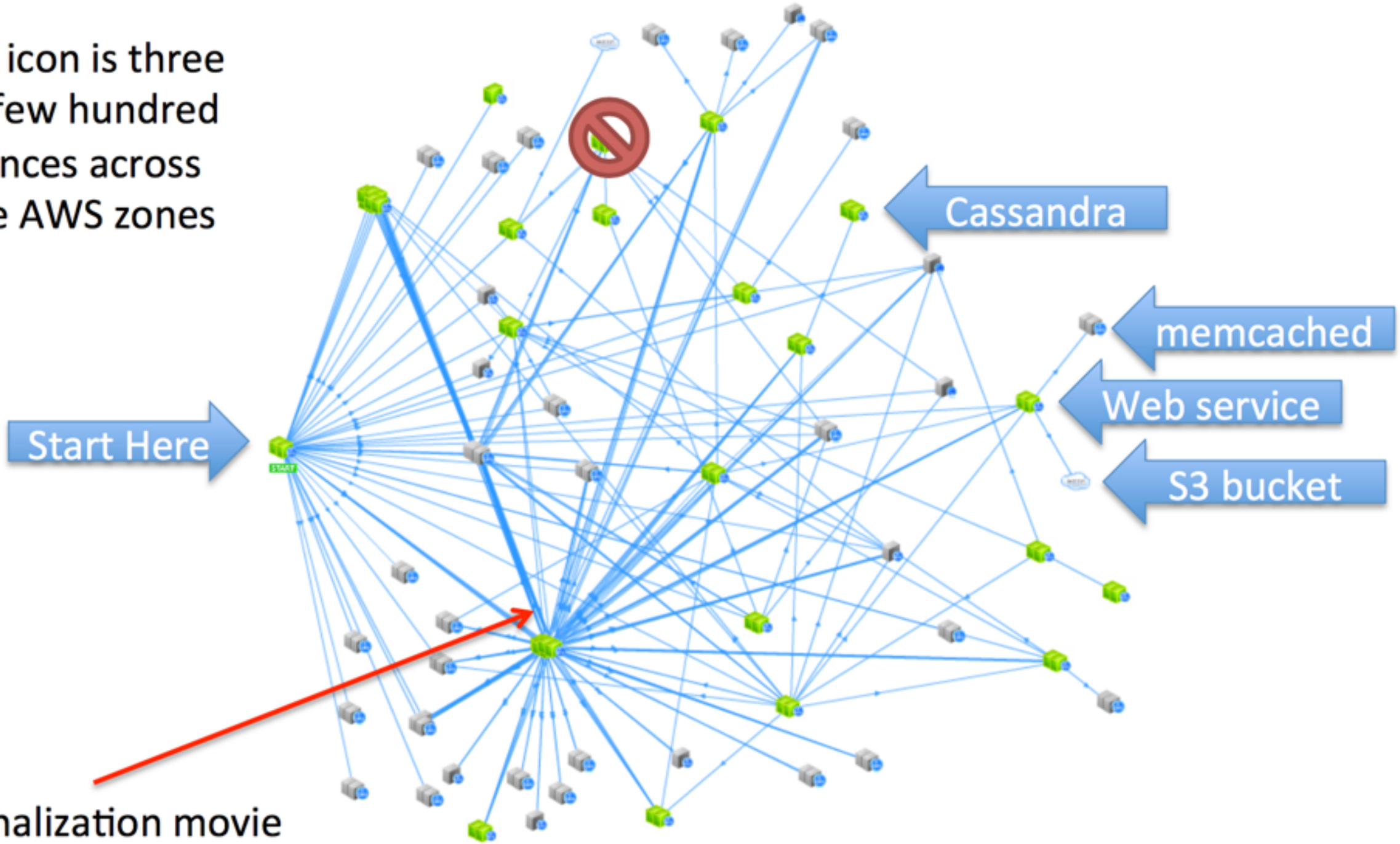
DRM

QoS Logging

CDN Management and Steering

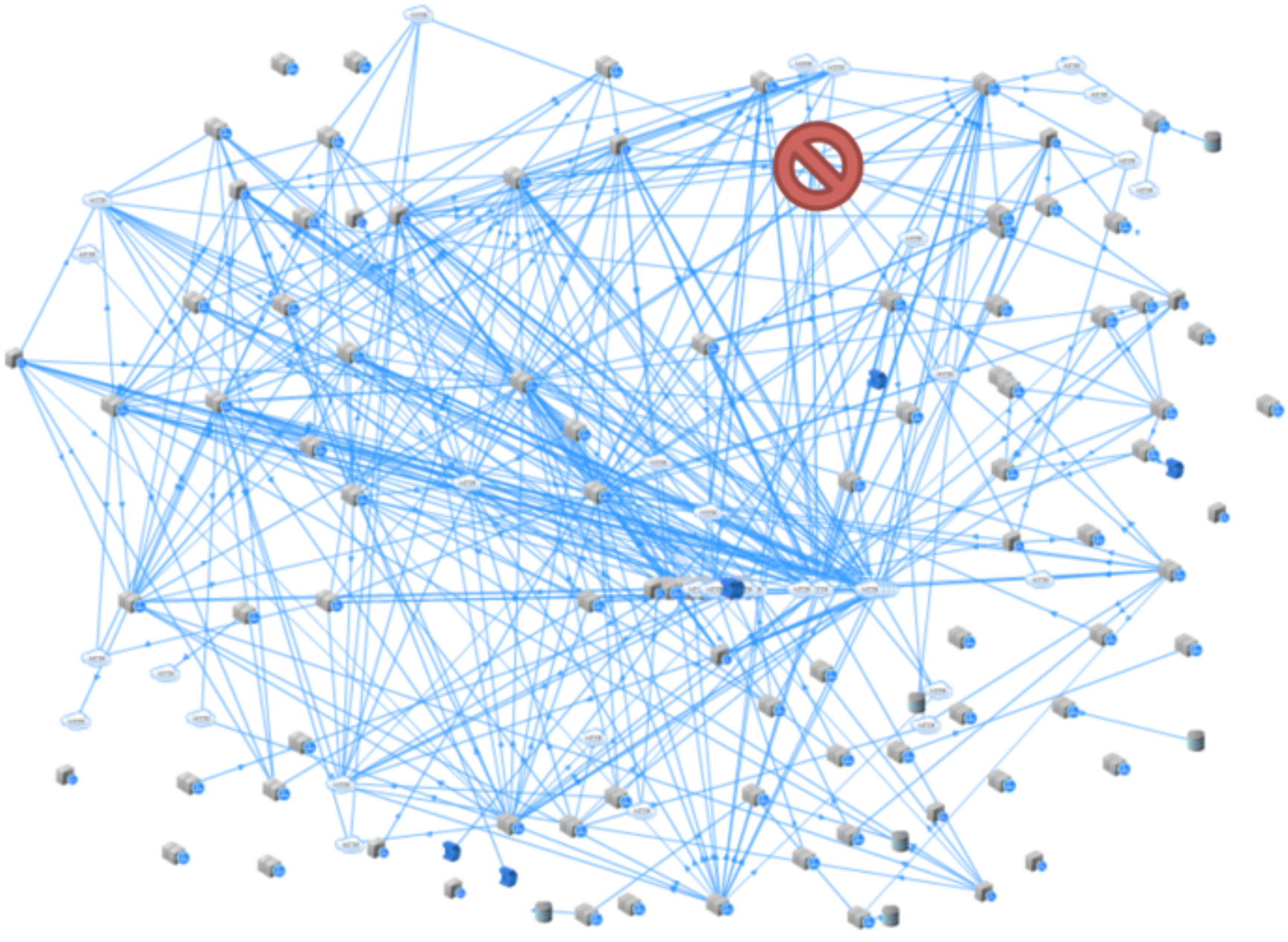
Content Encoding

Each icon is three to a few hundred instances across three AWS zones



Personalization movie group chooser

Component microservices



ABCallServiceInternal

191,390 | 0 | 0.0 %
0 | 5 | 1

Host: **32.9/s**
Cluster: **19,139.6/s**

Circuit Closed

Hosts	**581**	90th	**17ms**
Median	**6ms**	99th	**64ms**
Mean	**10ms**	99.5th	**114ms**

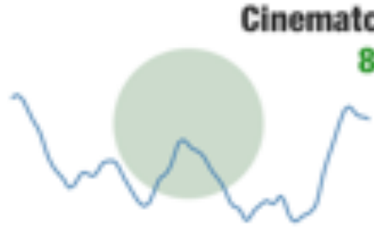
IdentityCookieAuth

190,804 | 0 | 0.0 %
0 | 50 | 0

Host: **32.8/s**
Cluster: **19,085.4/s**

Circuit Closed

Hosts	**581**	90th	**1ms**
Median	**0ms**	99th	**34ms**
Mean	**1ms**	99.5th	**49ms**



CinematchGetPredictions

80,105 | 5 | 0.0 %
0 | 0 | 0

Host: **13.8/s**
Cluster: **8,011.0/s**

Circuit Closed

Hosts	**581**	90th	**22ms**
Median	**3ms**	99th	**122ms**
Mean	**11ms**	99.5th	**312ms**

CryptexDecipher

74,718 | 92 | 0.0 %
0 | 0 | 58

Host: **12.9/s**
Cluster: **7,486.8/s**

Circuit Closed

Hosts	**581**	90th	**11ms**
Median	**3ms**	99th	**67ms**
Mean	**10ms**	99.5th	**405ms**

CinematchGetMovieRatings

60,672 | 2 | 0.0 %
0 | 0 | 0

Host: **10.4/s**
Cluster: **6,067.4/s**

Circuit Closed

Hosts	**581**	90th	**41ms**
Median	**14ms**	99th	**143ms**
Mean	**21ms**	99.5th	**215ms**

VideoHistoryGetBookmarks

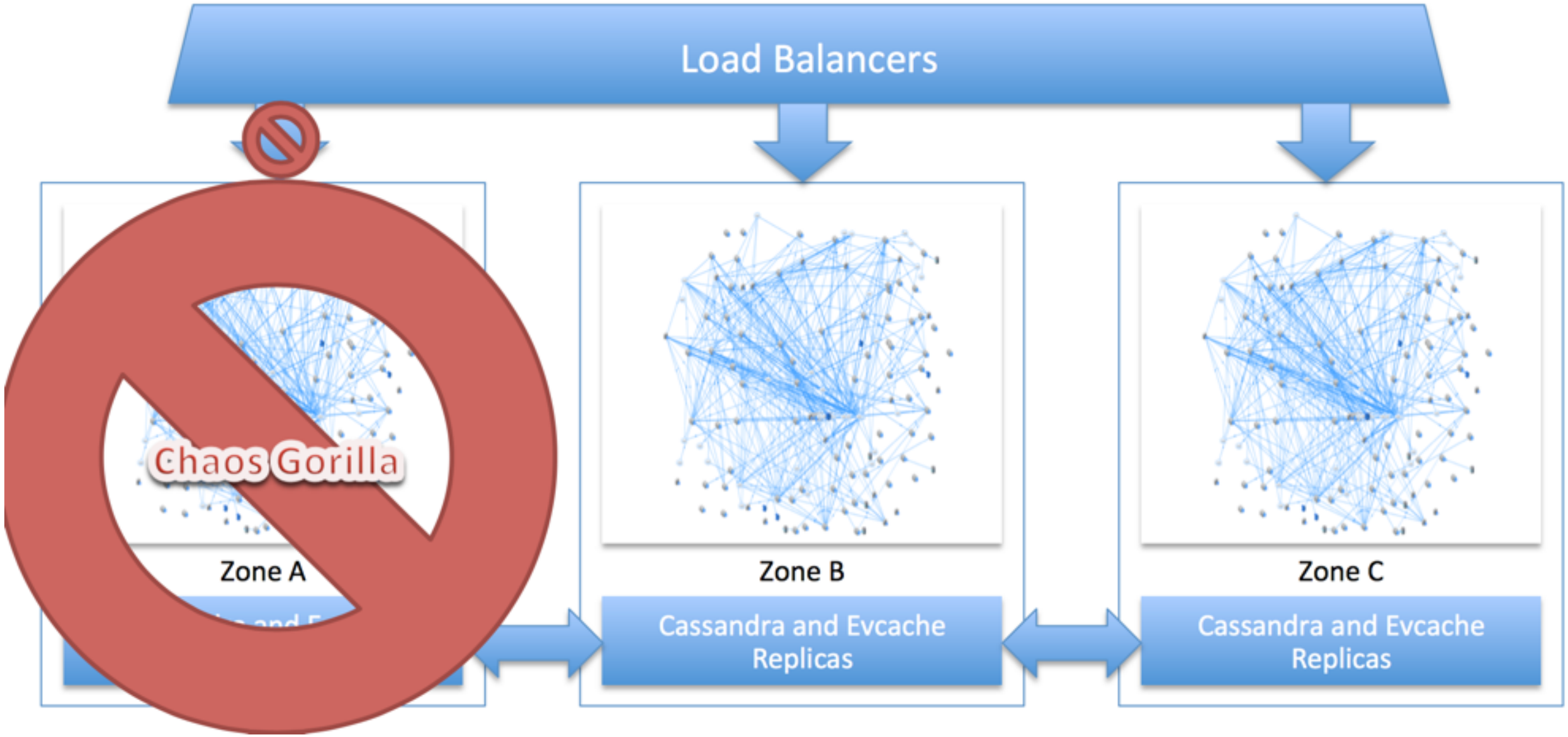
48,281 | 60 | 0.1 %
0 | 0 | 0

Host: **8.3/s**
Cluster: **4,834.1/s**

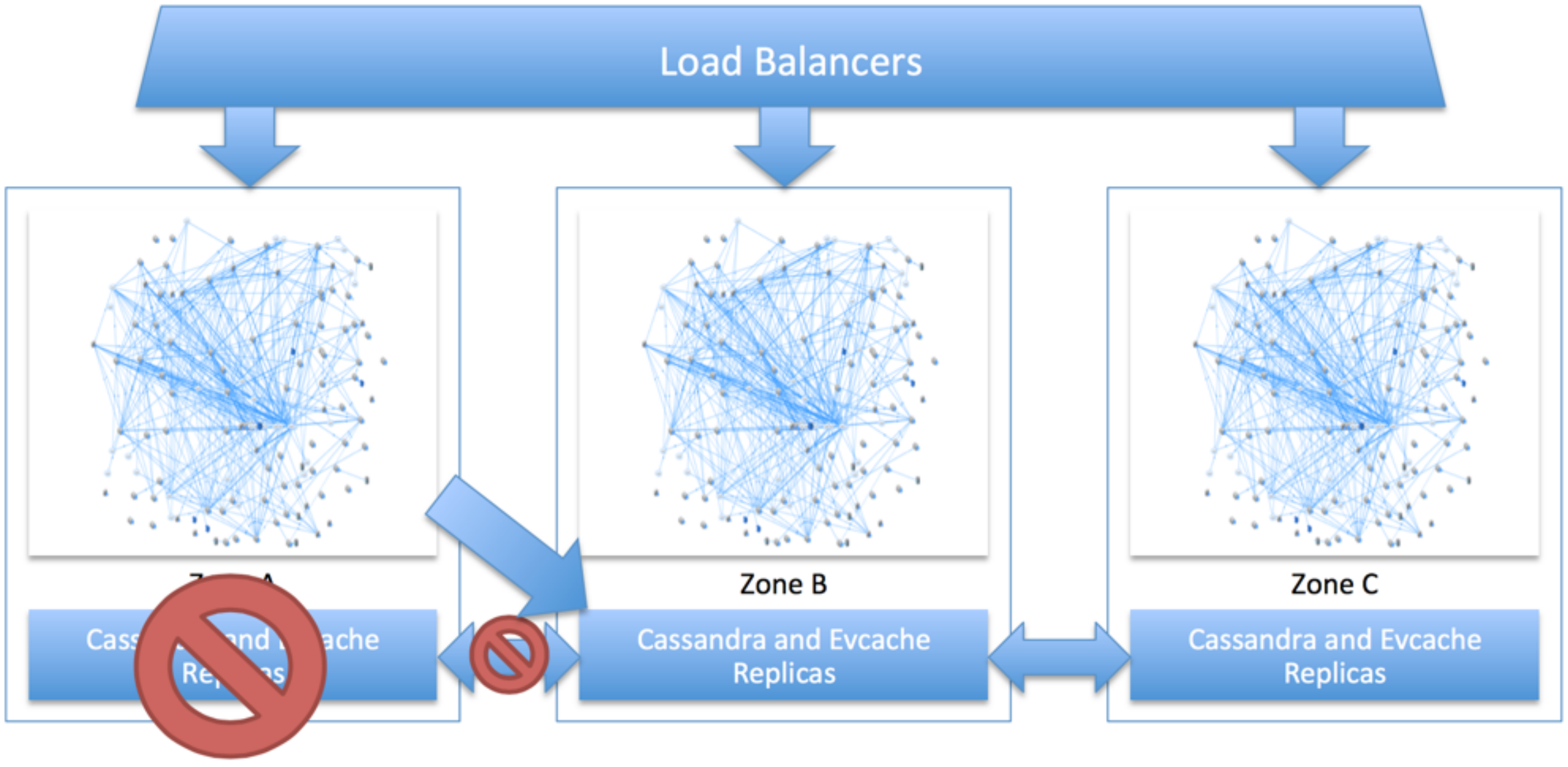
Circuit Closed

Hosts	**581**	90th	**26ms**
Median	**8ms**	99th	**104ms**
Mean	**13ms**	99.5th	**158ms**

Chaos Gorilla



Cassandra maintenance

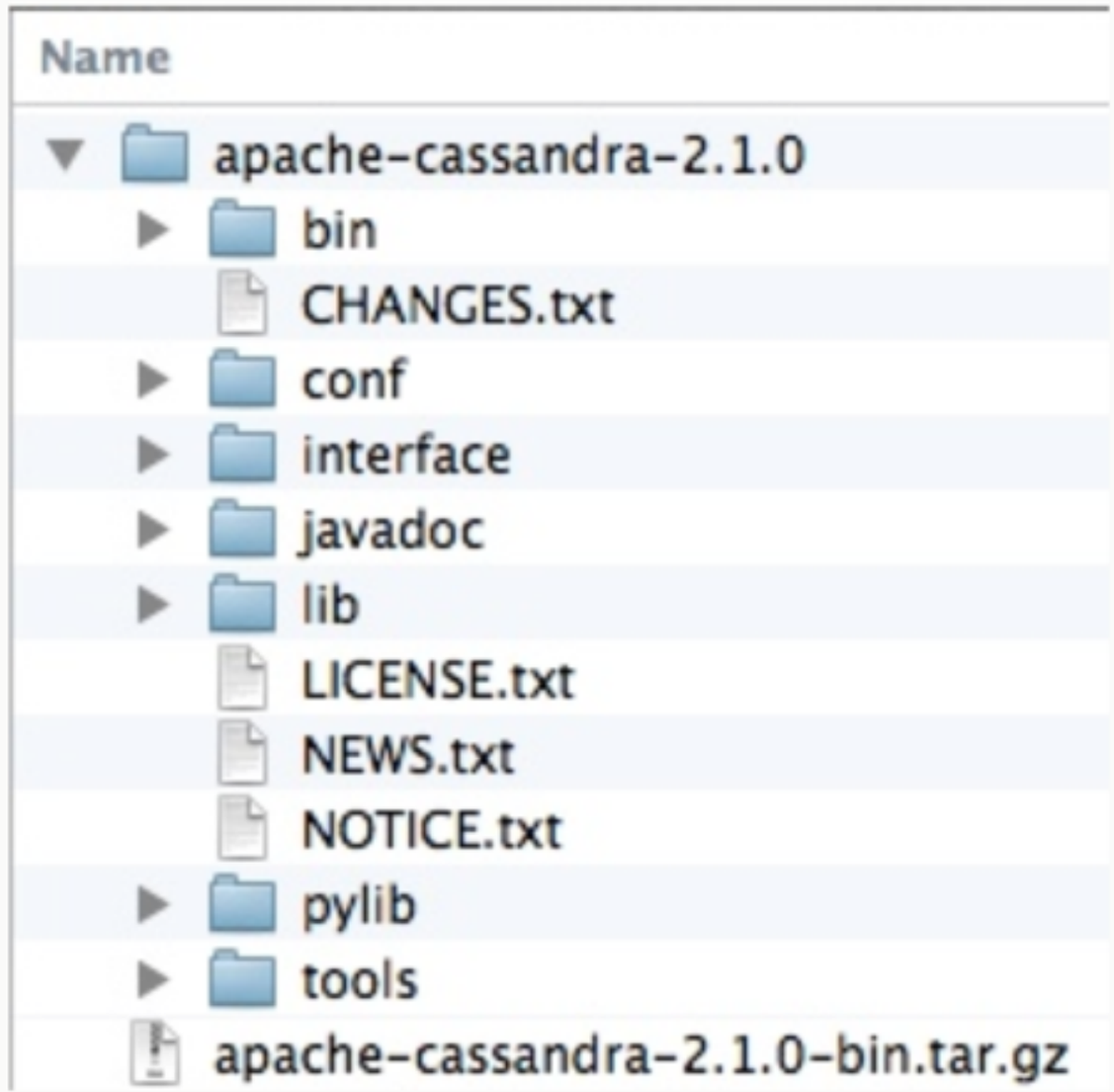


Cassandra

- 10x more read throughput
- 8x faster read latency (up to 100x faster)
- 8x more write throughput
- 10x slower write latency (with the default configuration; that is, no write durability for HBase)
- 8x faster scan latency
- 4x more scan throughput

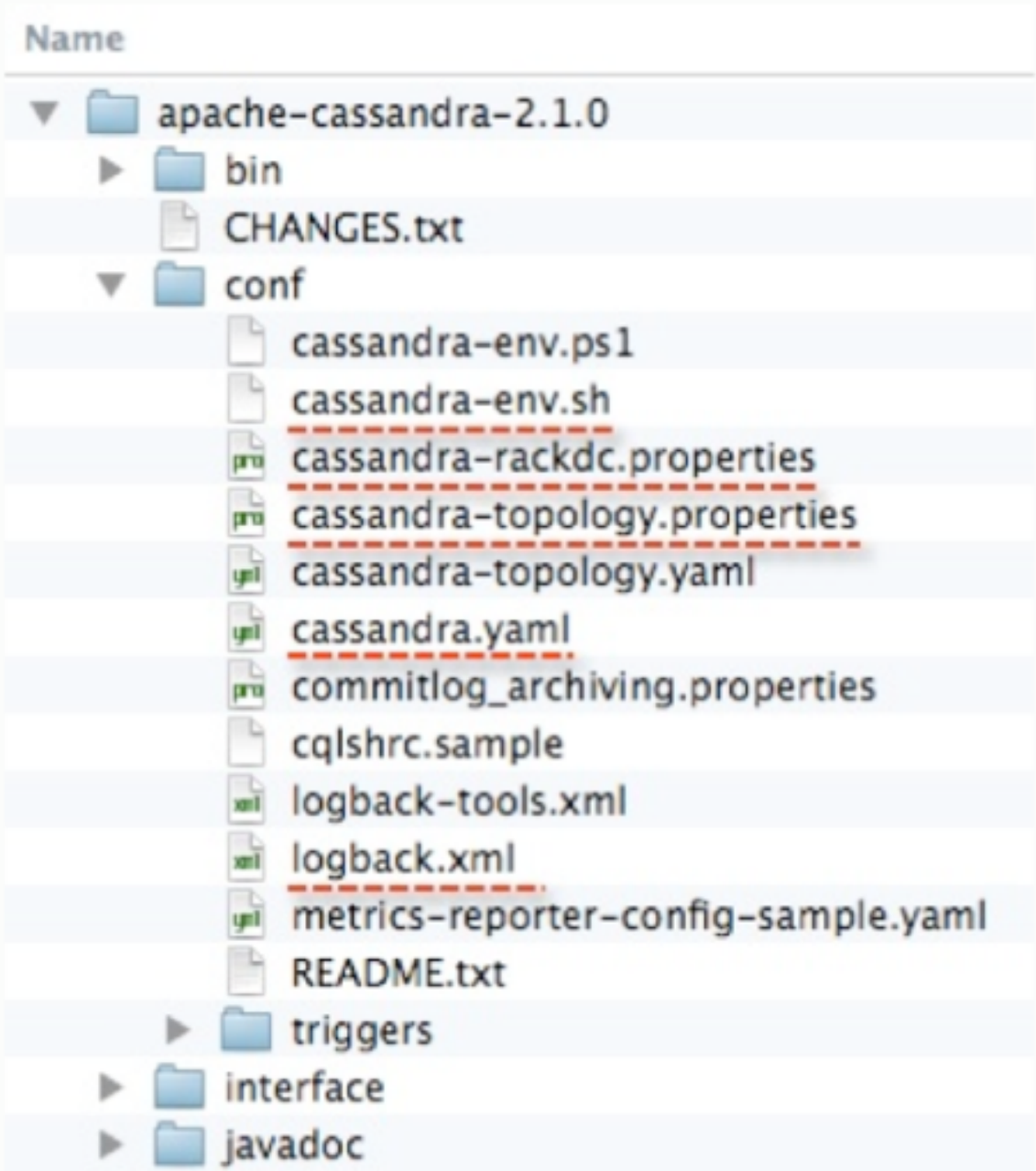
INSTALLATION

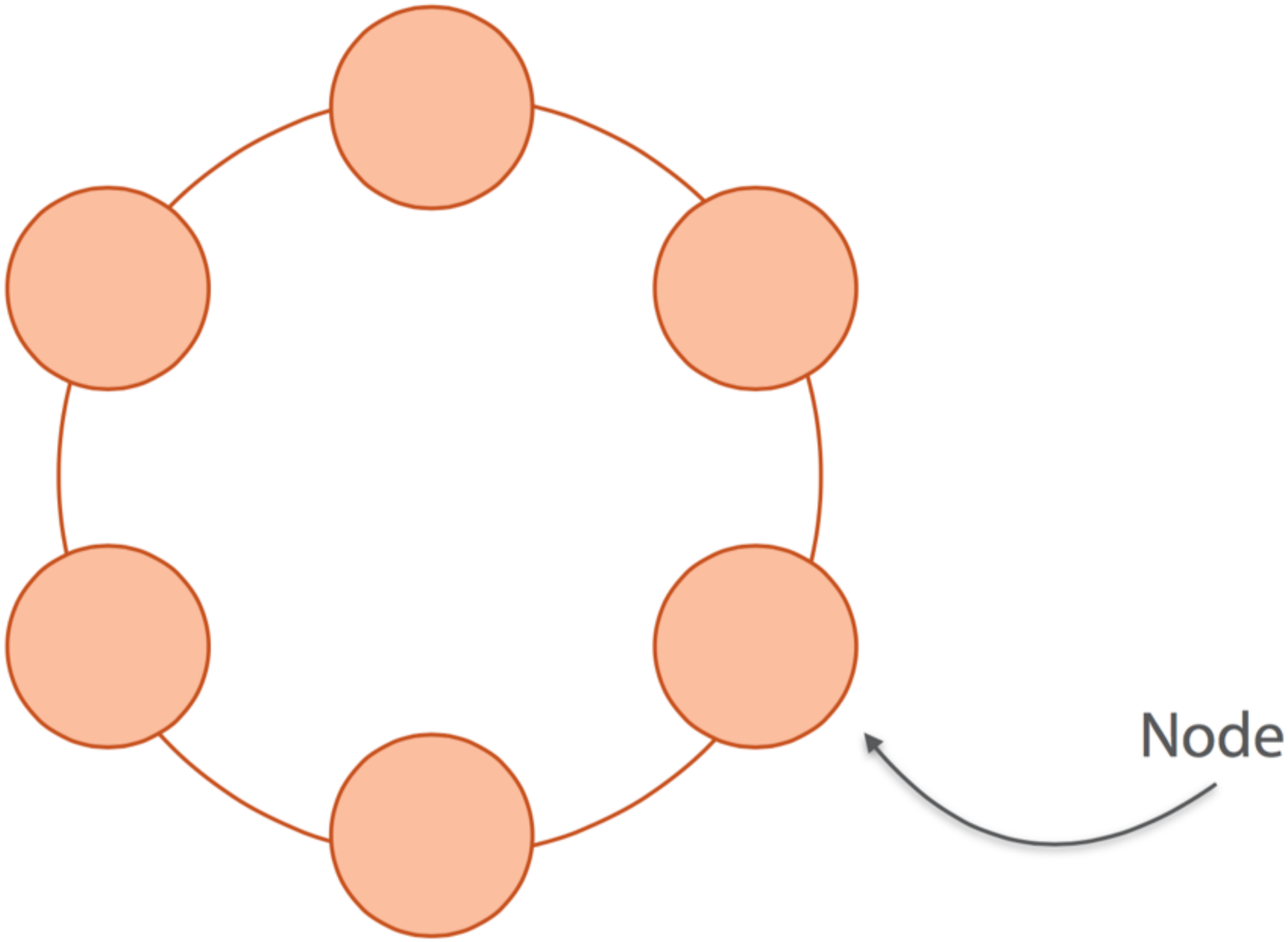
- Tarball installations create these folders in a single location:
  - bin: executables (*cassandra*, *cqlsh*, *nodetool*, etc)
  - conf: Configuration files - *cassandra.yaml*
  - javadoc: C* source documentation
  - lib: Library dependencies (jars)
  - pylib: Python libraries (e.g. for *cqlsh*, which is written in Python)
  - tools: Additional tools (e.g. *cassandra-stress* which stresses a C* cluster)



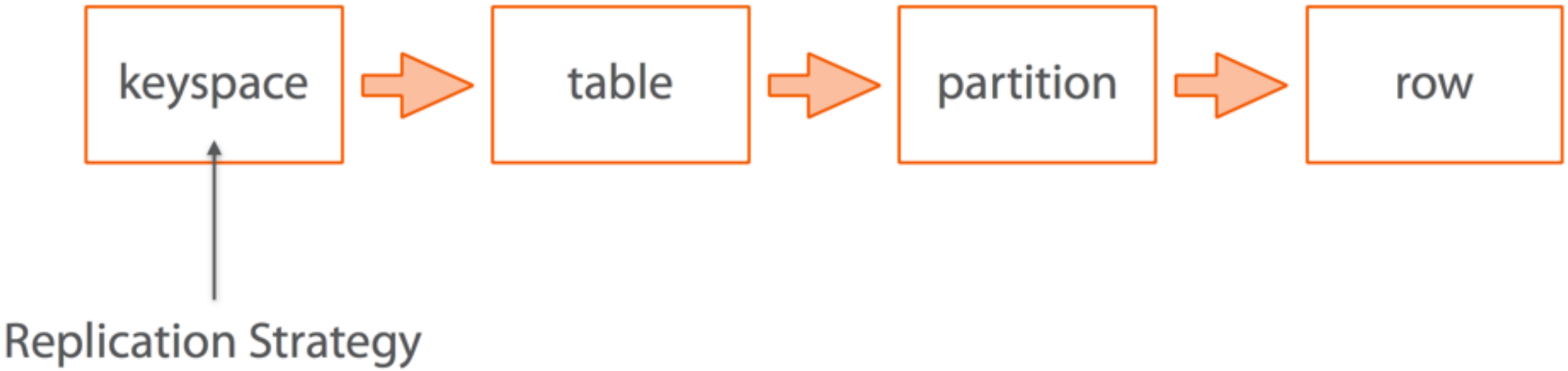
configuration files

- cassandra.yaml
  - One file per node--must agree with other node's files
  - Parameters defined throughout
- cassandra-env.sh
  - Memory settings
  - JMX settings
- logback.xml
  - System log location
  - Logging level



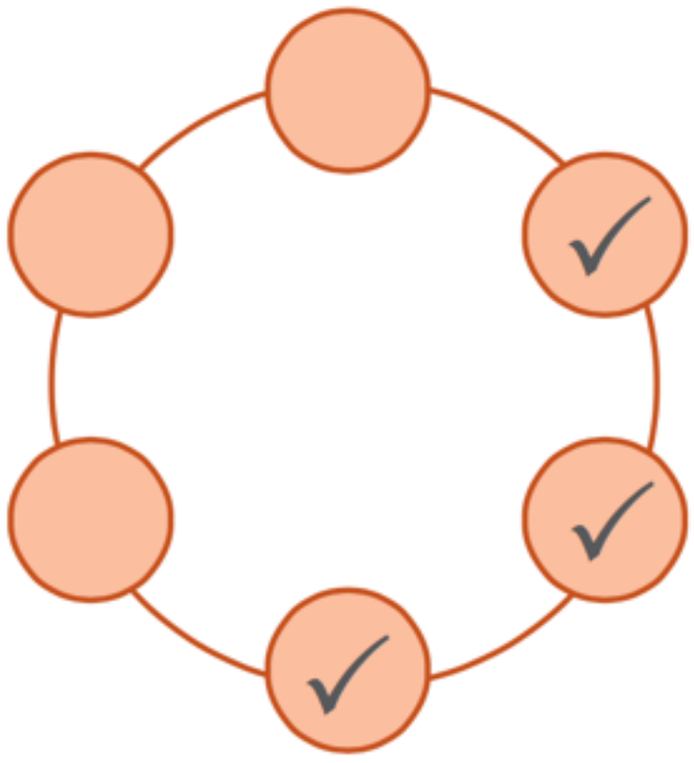

Node

cassandra terminology



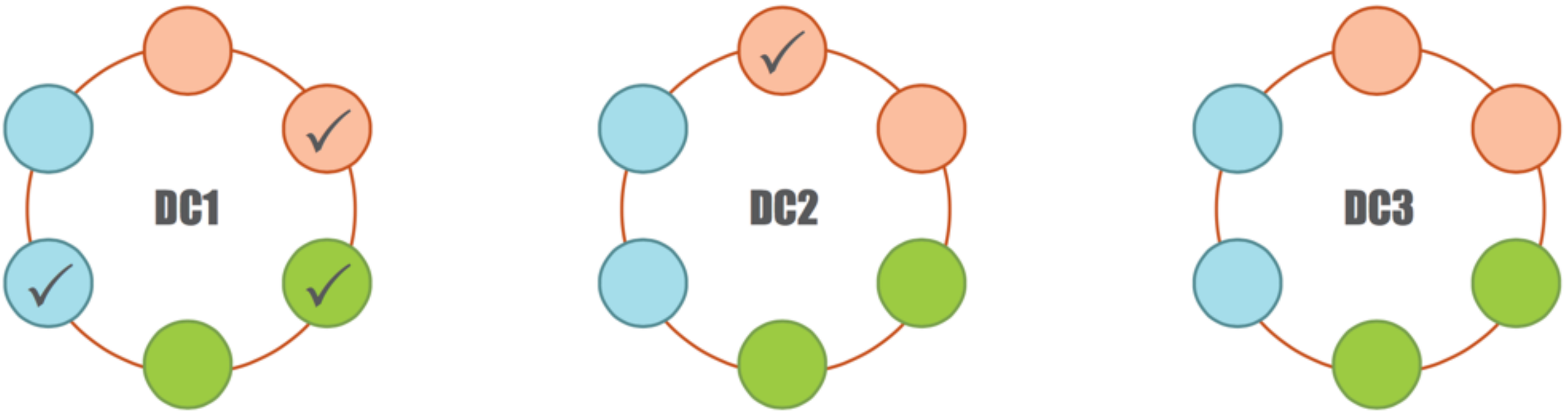
simple strategy

- CREATE KEYSPACE nfjs WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 3 };

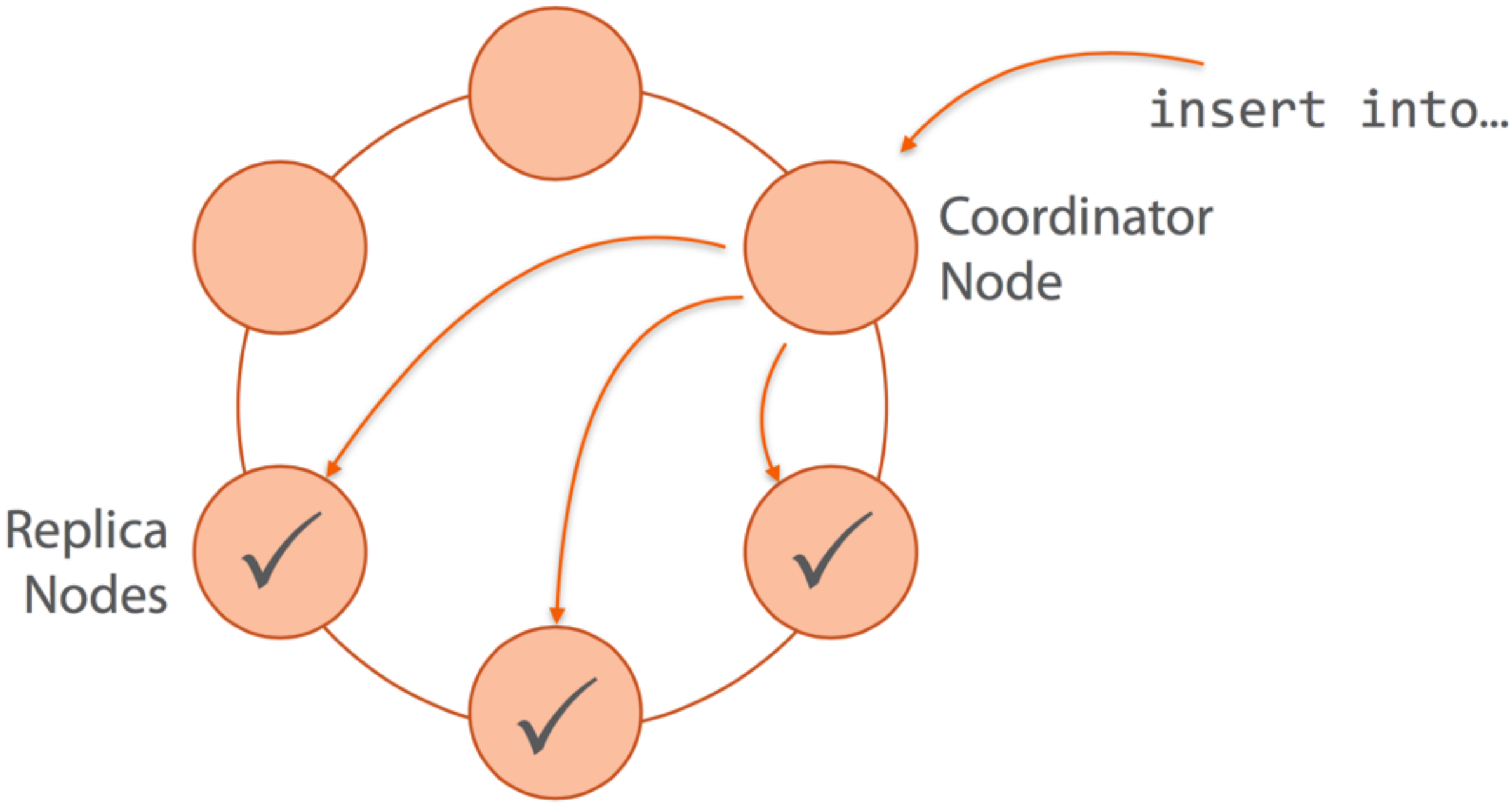


network topology strategy

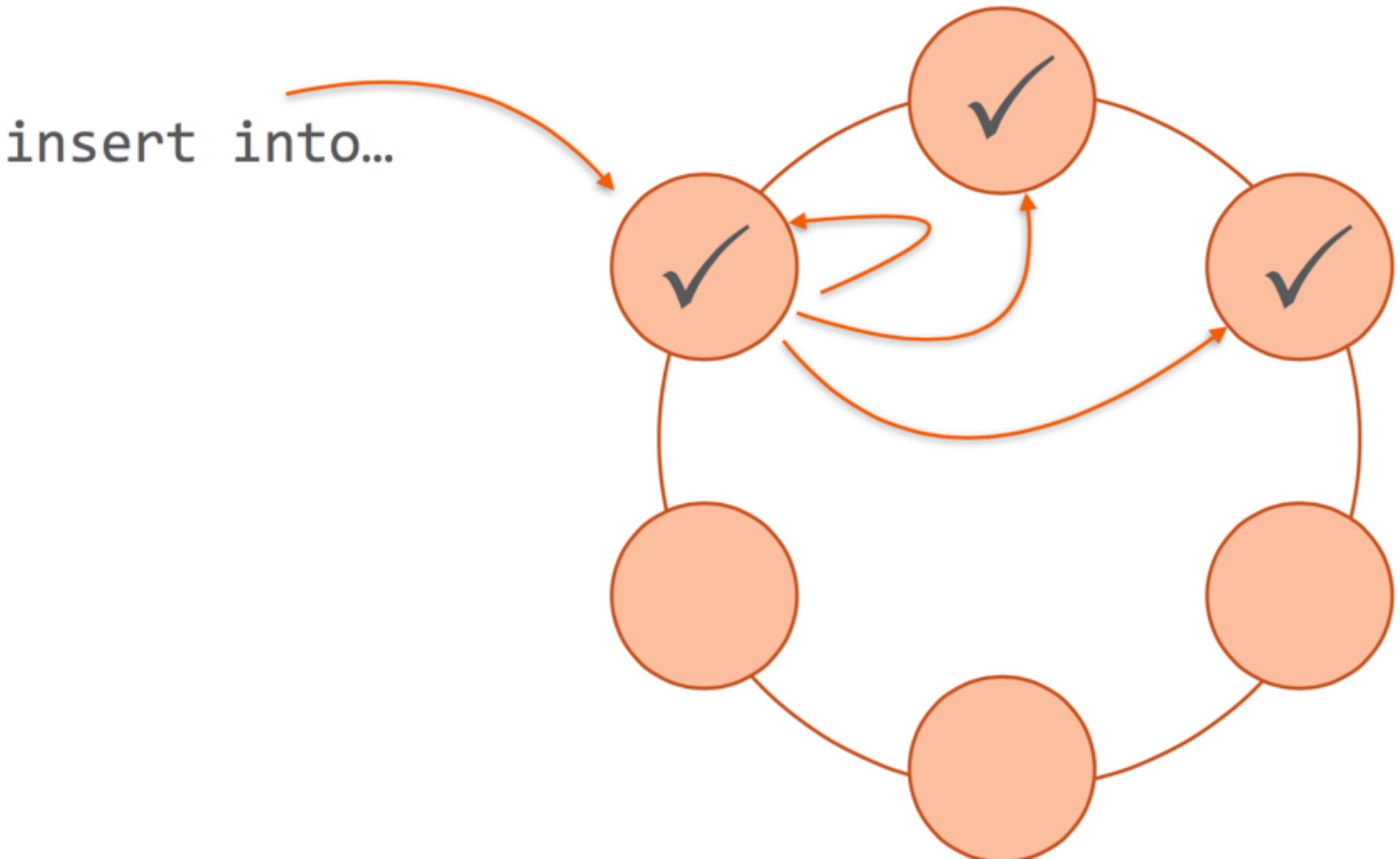
- create KEYSPACE nfjs WITH REPLICATION = {'class': 'NetworkTopologyStrategy', 'DC1': 3, 'DC2', 1};



Reads and Writes in Cassandra



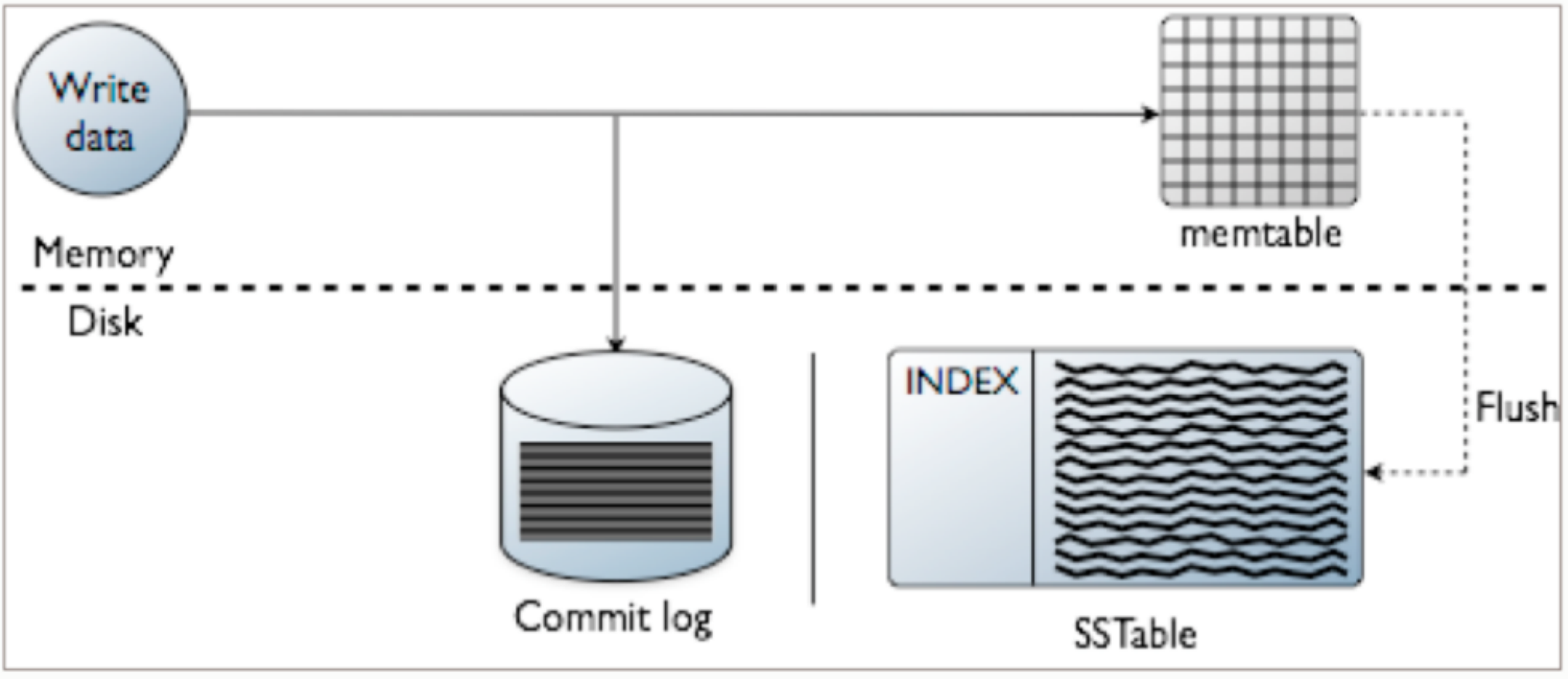
Reads and Writes in Cassandra



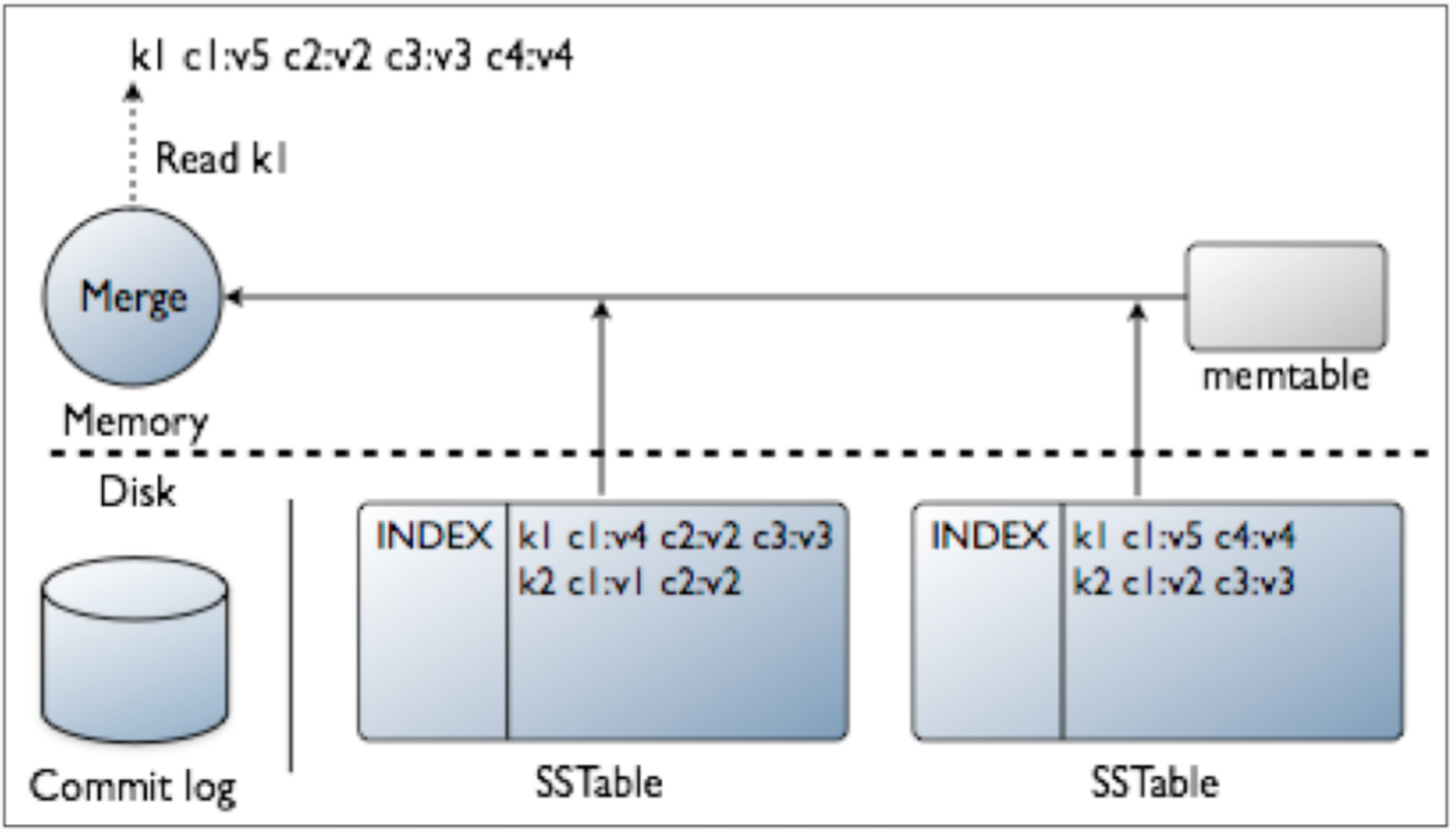
creating tables

Writing the CQL statements

```
CREATE TABLE comments_by_user (
        user_id UUID,
        posted_timestamp TIMESTAMP,
        video_id TIMEUUID,
        comment TEXT,
        title TEXT,
        type TEXT,
        tags SET<TEXT>,
        preview_thumbnails MAP<INT, BLOB>,
        PRIMARY KEY ((user_id), posted_timestamp, video_id)
) WITH CLUSTERING ORDER BY (posted_timestamp DESC, video_id ASC);
```


Write data
memtable
Memory
Disk
Flush
INDEX
Commit log
SSTable

read processing in node



AFFILIATE LIST

Country | State | City | Commercial | CLEAR RESULTS

Partition keys

- CREATE TABLE crossfit_gyms (
- gym_name text,
- city text,
- state_province text,
- country_code text,
- PRIMARY KEY (gym_name)
-);

http://datascale.io/cassandra-partitioning-and-clustering-keys-explained/

compound keys

```
CREATE TABLE crossfit_gyms (
    gym_name text,
    city text,
    state_province text,
    country_code text,
    PRIMARY KEY (gym_name)
);
```

Clustering keys

```
CREATE TABLE crossfit_gyms_by_location (
    country_code text,
    state_province text,
    city text,
    gym_name text,
    PRIMARY KEY (country_code, state_province, city,
gym_name)
);
```

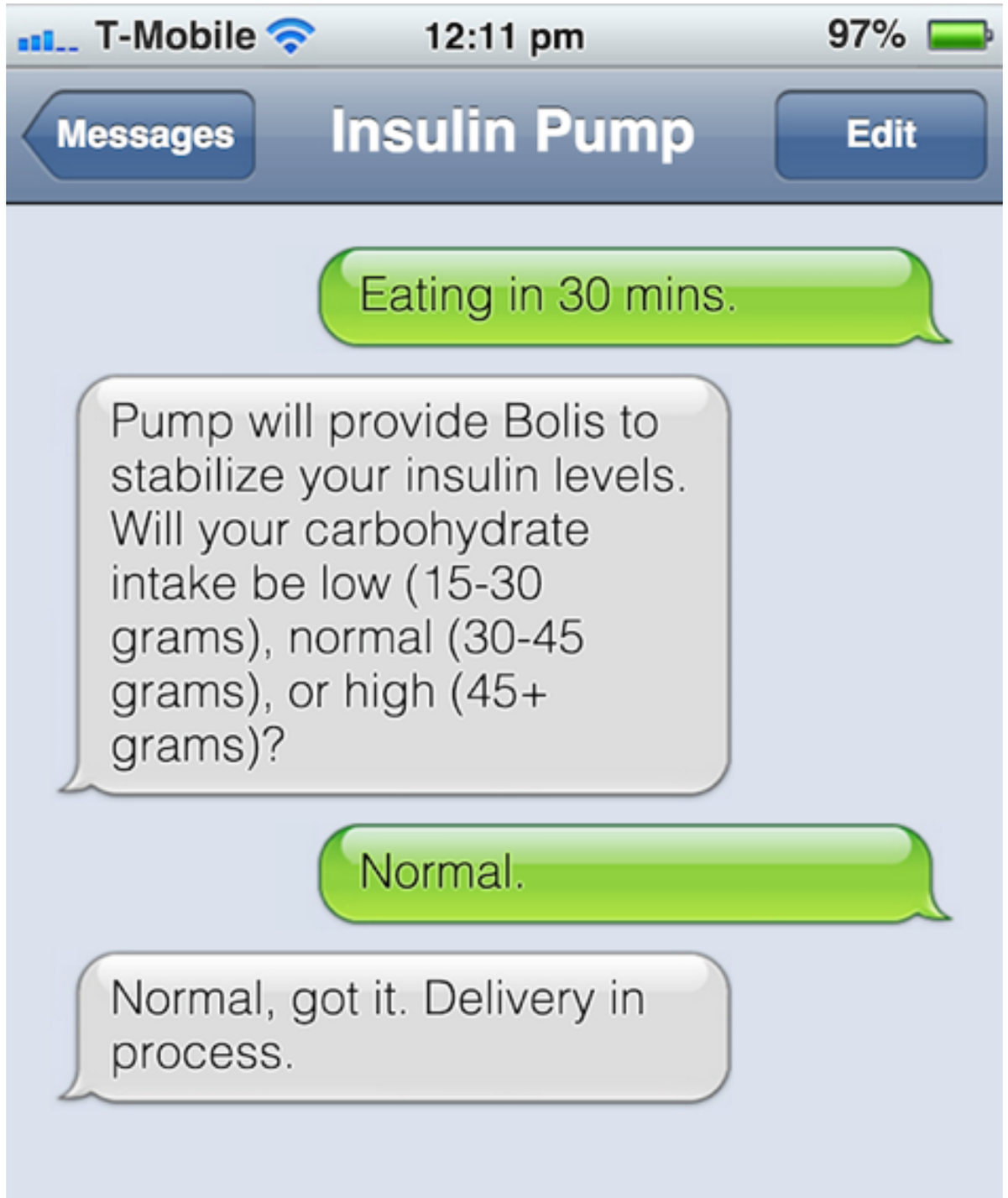
order by

```
CREATE TABLE crossfit_gyms_by_location (
    country_code text,
    state_province text,
    city text,
    gym_name text,
    PRIMARY KEY (country_code, state_province,
city, gym_name)
) WITH CLUSTERING ORDER BY (state_province DESC,
city ASC, gym_name ASC);
```

hashed values

- **country_code l state_province l city l gym_name**
- -------------+---------------+--------------+------------------------
- CAN | ON | Toronto | CrossFit Leslieville
- CAN | ON | Toronto | CrossFit Toronto
- CAN | BC | Vancouver | CrossFit BC
- CAN | BC | Vancouver | CrossFit Vancouver
- USA | NY | New York | CrossFit Metropolis
- USA | NY | New York | CrossFit NYC
- USA | NV | Las Vegas | CrossFit Las Vegas
- USA | NV | Las Vegas | Kaizen CrossFit
- USA | CA | San Francisco | LaLanne Fitness CrossFit
- USA | CA | San Francisco | San Francisco CrossFit

Medical Devices as IoT



http://dtrapezoid.com/time-series-data-modeling-for-medical-devices.html

insulin

- CREATE KEYSPACE insulin_pump
- WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 3 }
- USE insulin_pump;

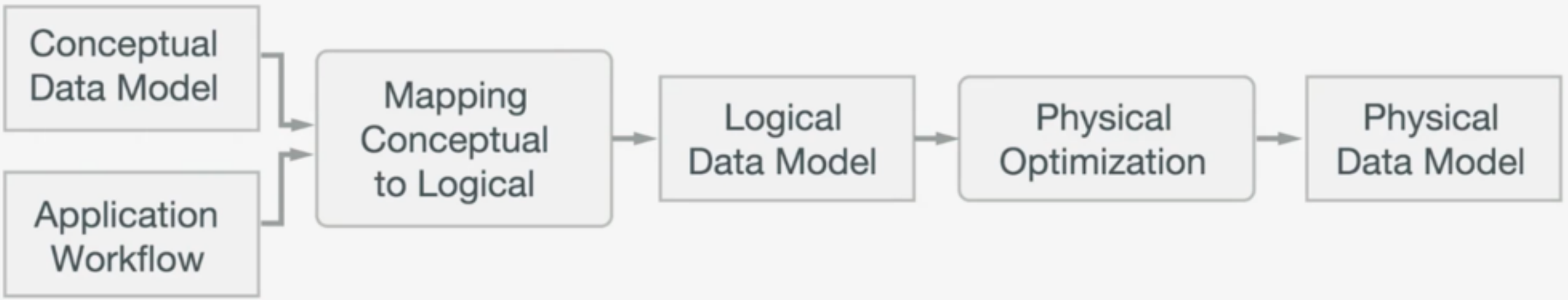
insulin_level table

- CREATE TABLE insulin_level (
- insulin_pump_id text,
- event_time timestamp,
- insulin_level_mcU_ml int,
- **PRIMARY KEY (insulin_pump_id,event_time)**
-);

insulin_level_by_day

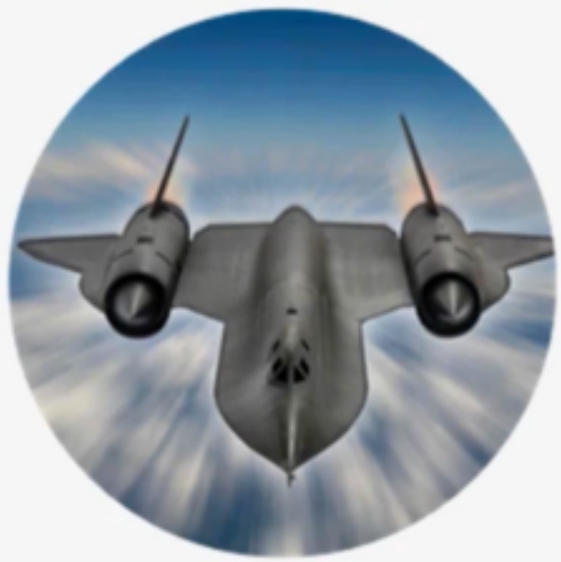
- CREATE TABLE insulin_level_by_day (
- insulin_pump_id text,
- date text,
- event_time timestamp,
- insulin_level_mcU_ml int,
- PRIMARY KEY ((insulin_pump_id,date),event_time)
-);

Cassandra data model

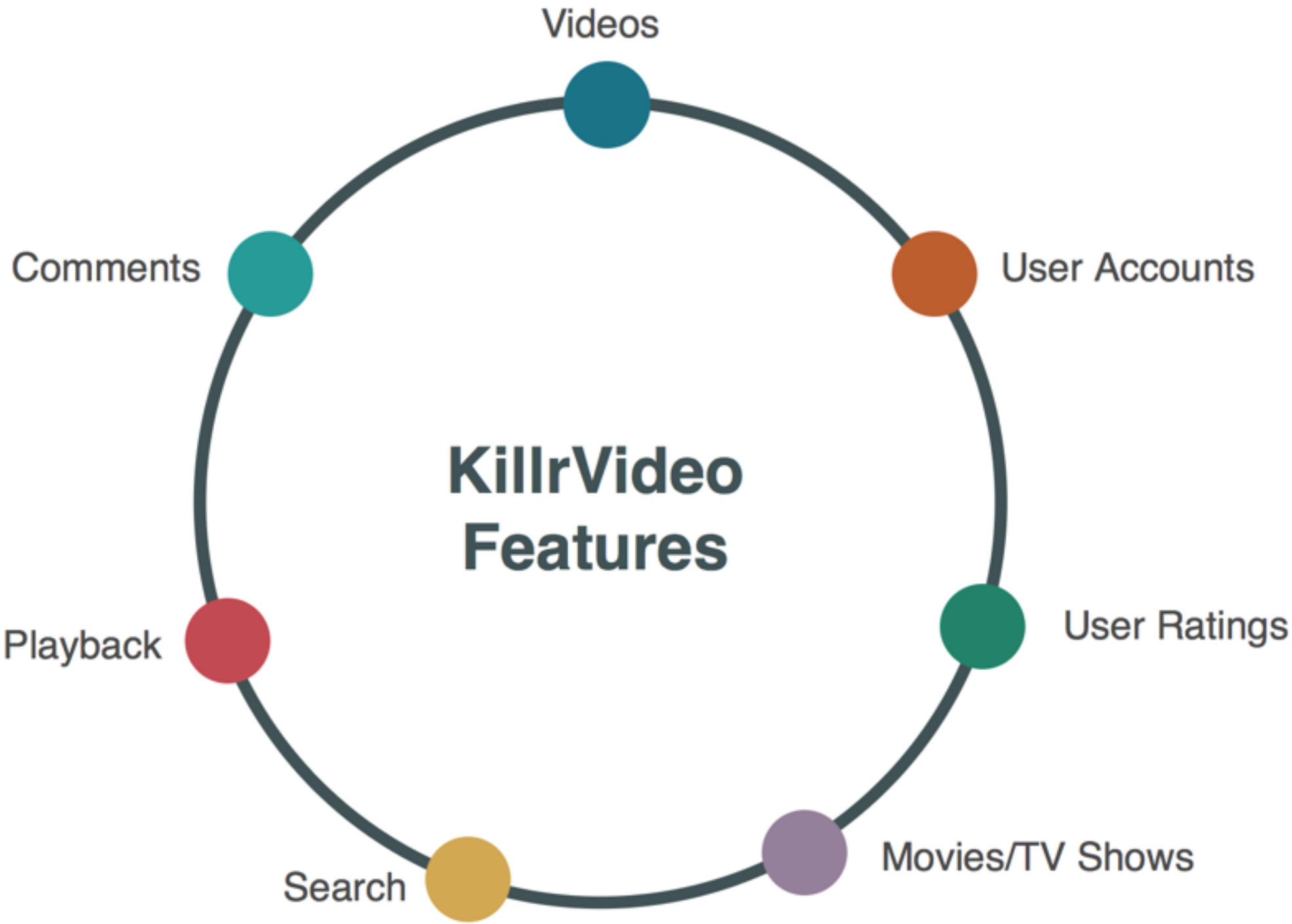


queries

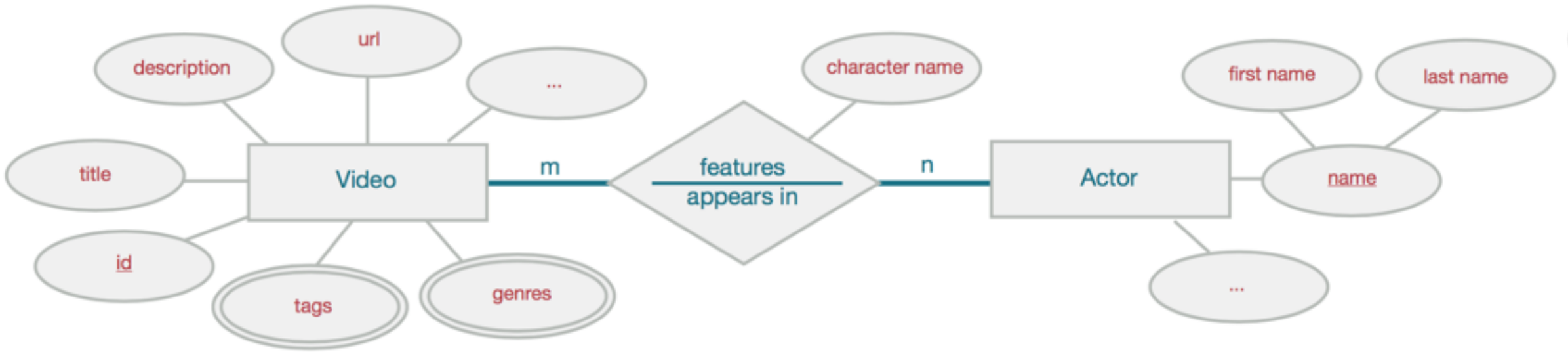


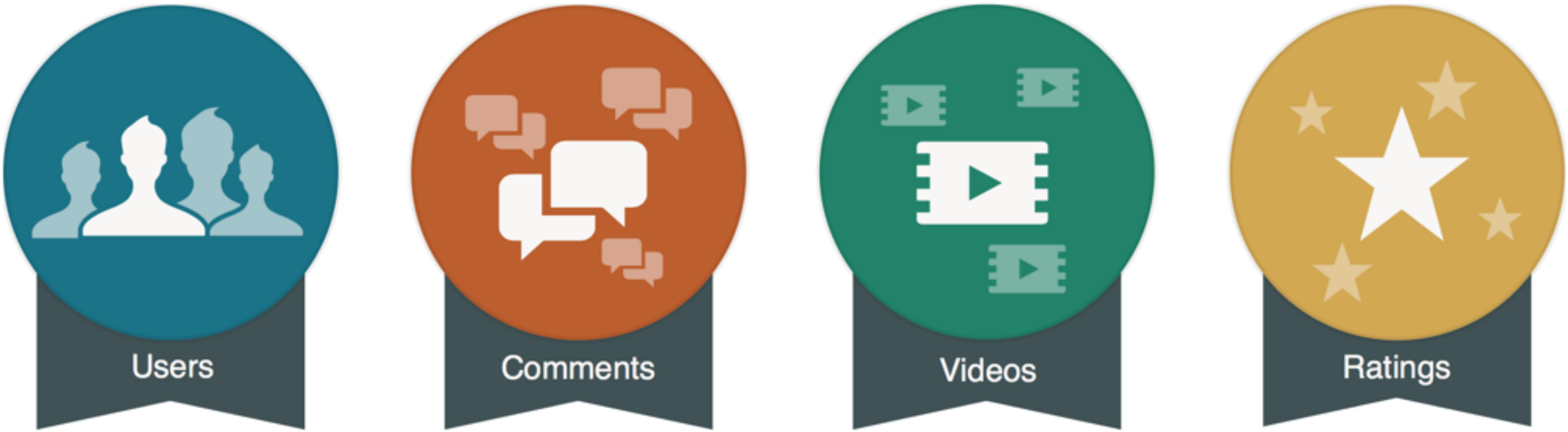
killrVideo.com

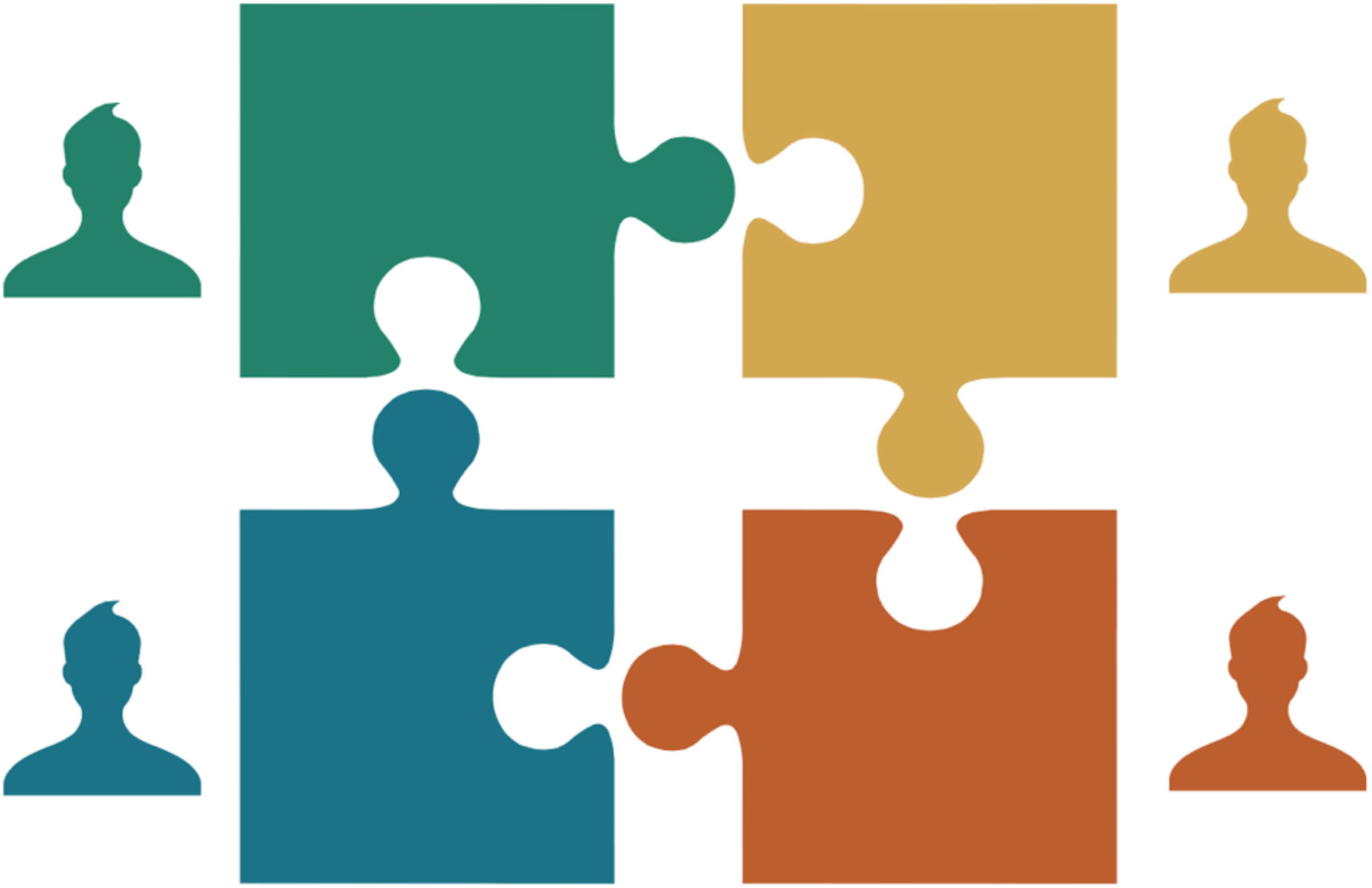


Conceptual Data Modeling



purpose

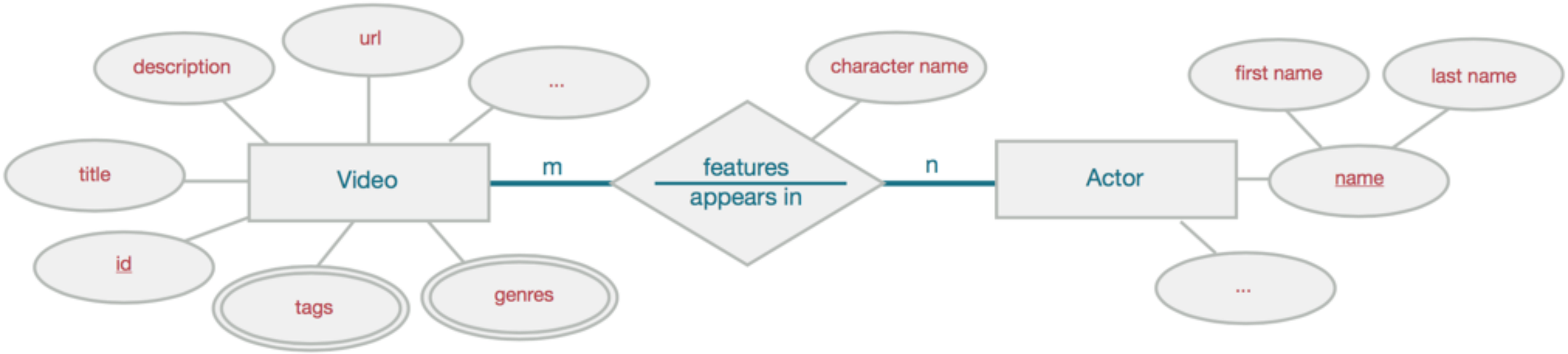




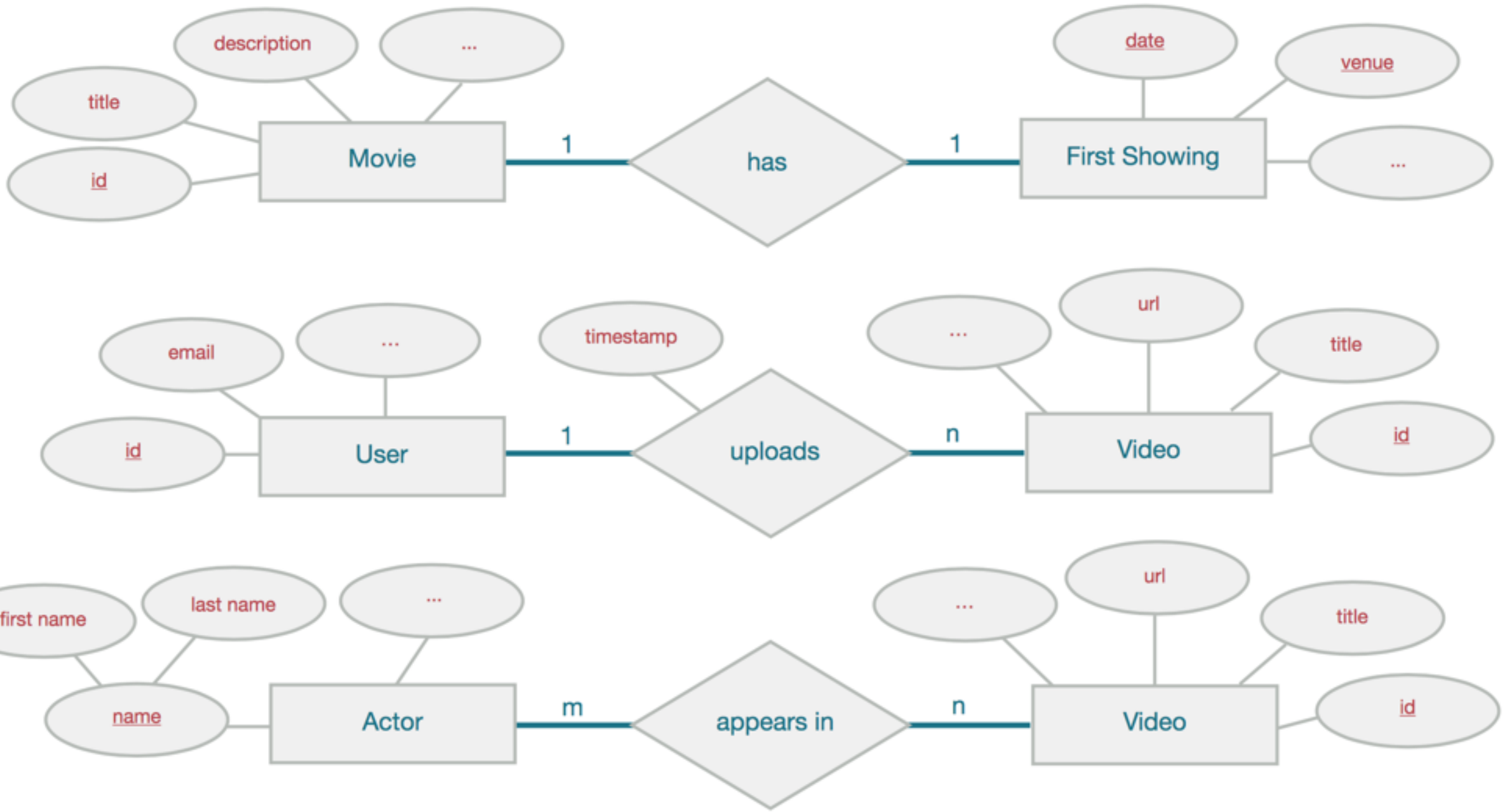
abstraction

abstraction

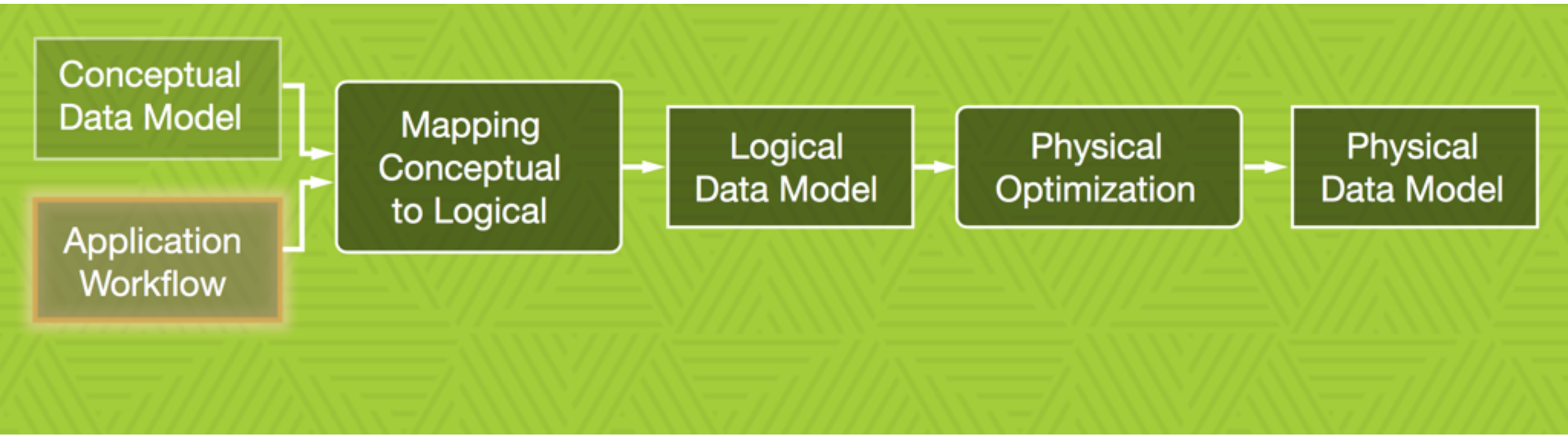
Entity-Relationship (ER) Model



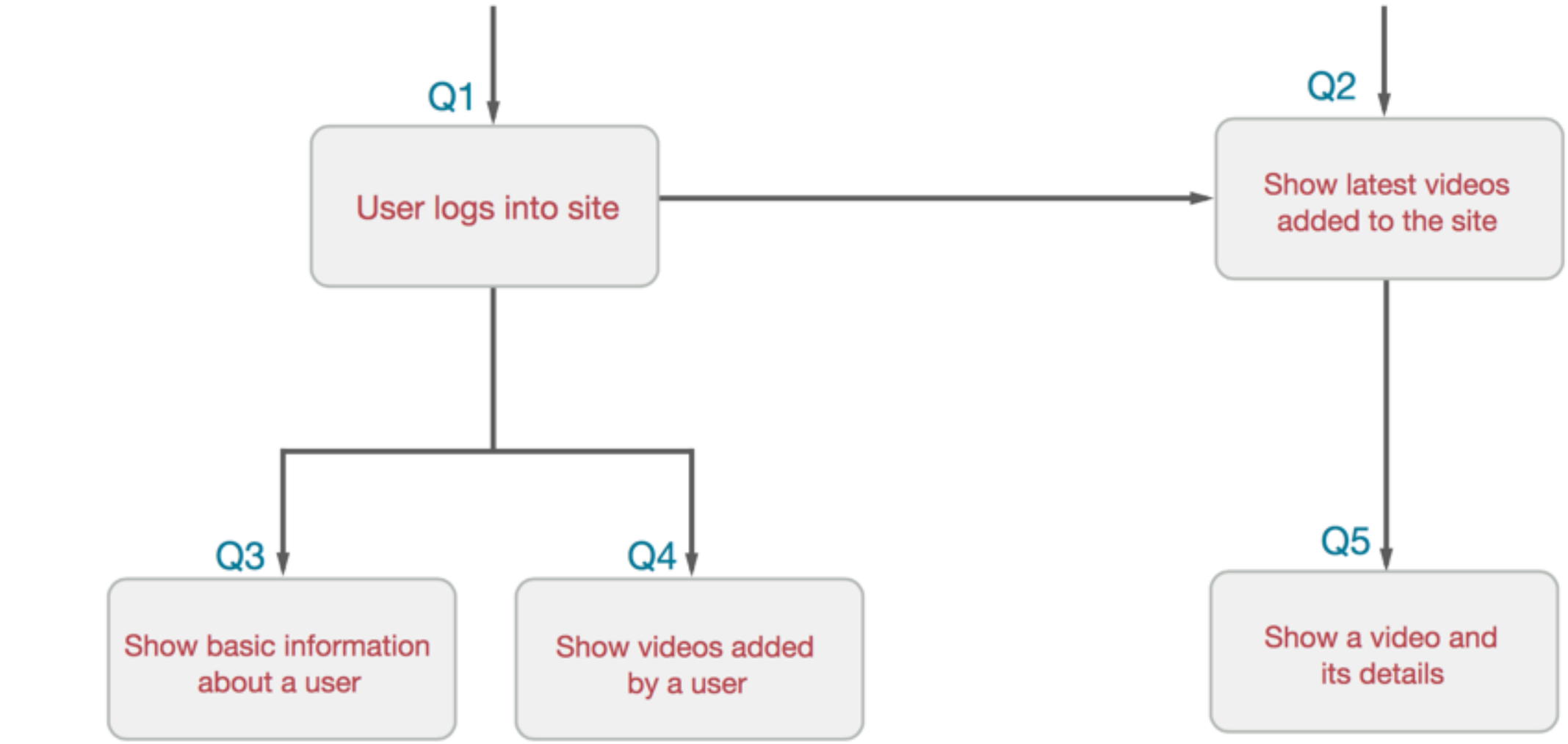
relationship keys



application workflow model



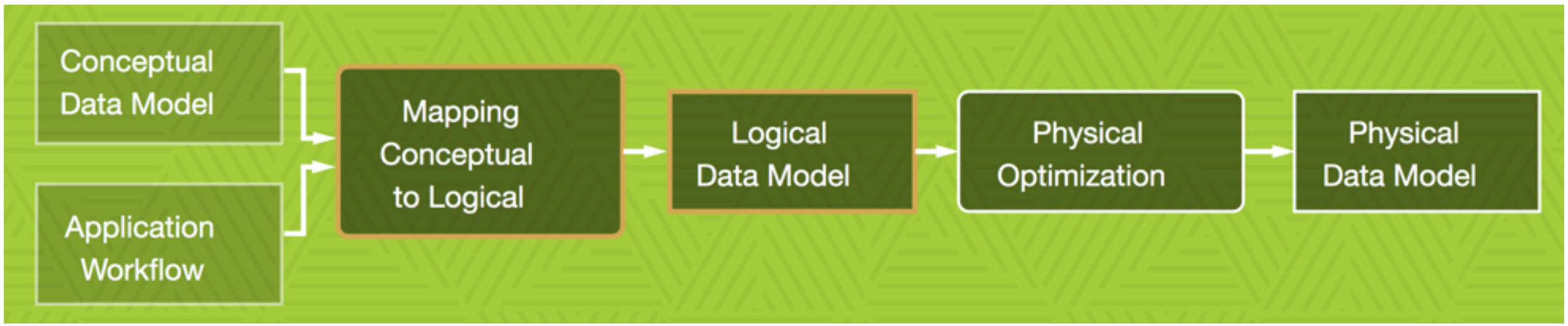
Task: Find a video that has a particular video id



ACCESS PATTERNS

Q1: Find a user with a **specified email**
Q2: Find most recently uploaded **videos**
Q3: Find a **user with a specified id**
Q4: Find videos uploaded by a **user with a known id**(show most recently uploaded videos first)
Q5: Find a video with a **specified video id**

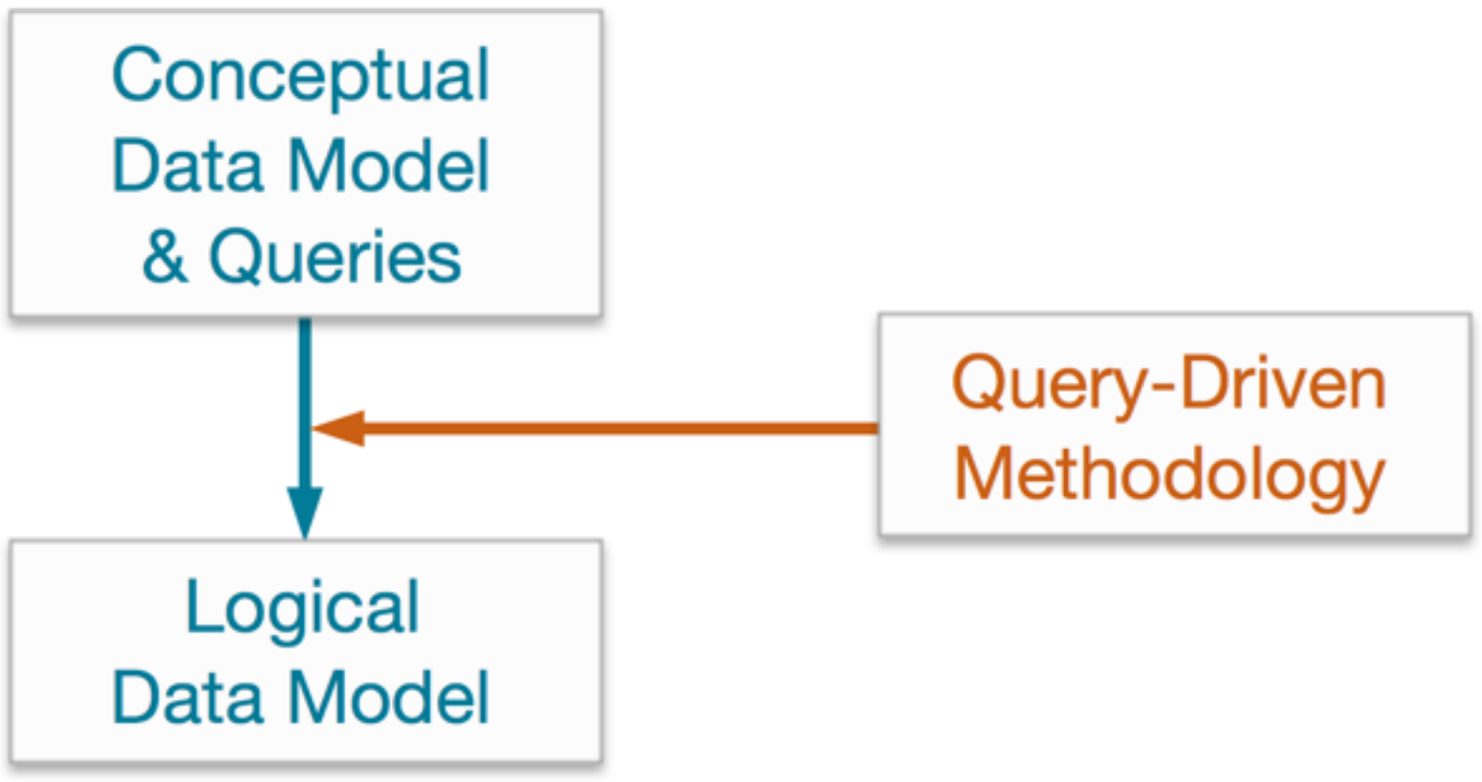
Mapping Conceptual to Logical



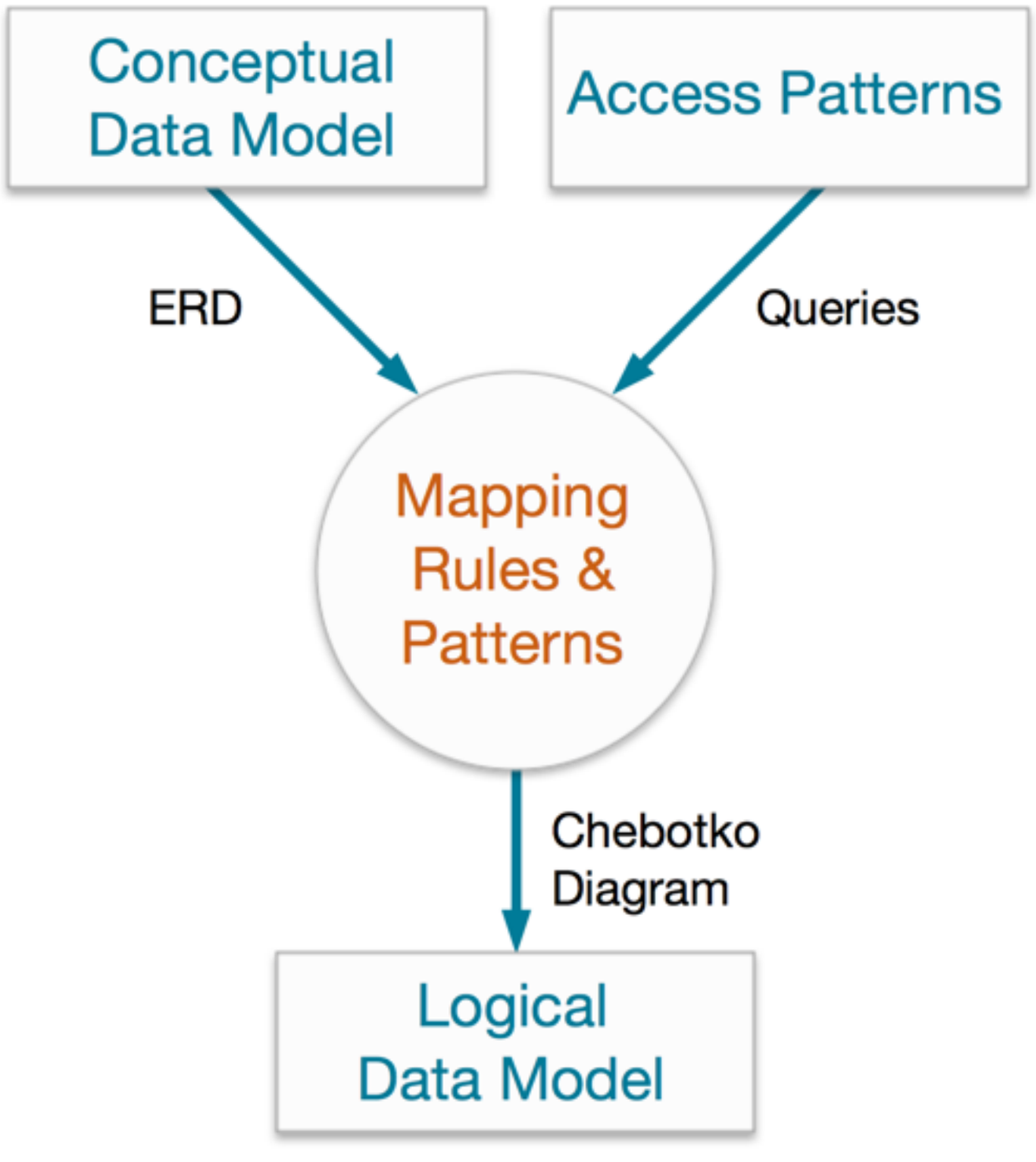
Data Modeling Methodology

Process to design a logical model

- Uses a top-down approach
- Can be algorithmically defined
- Effective in the long run—vs. dark art

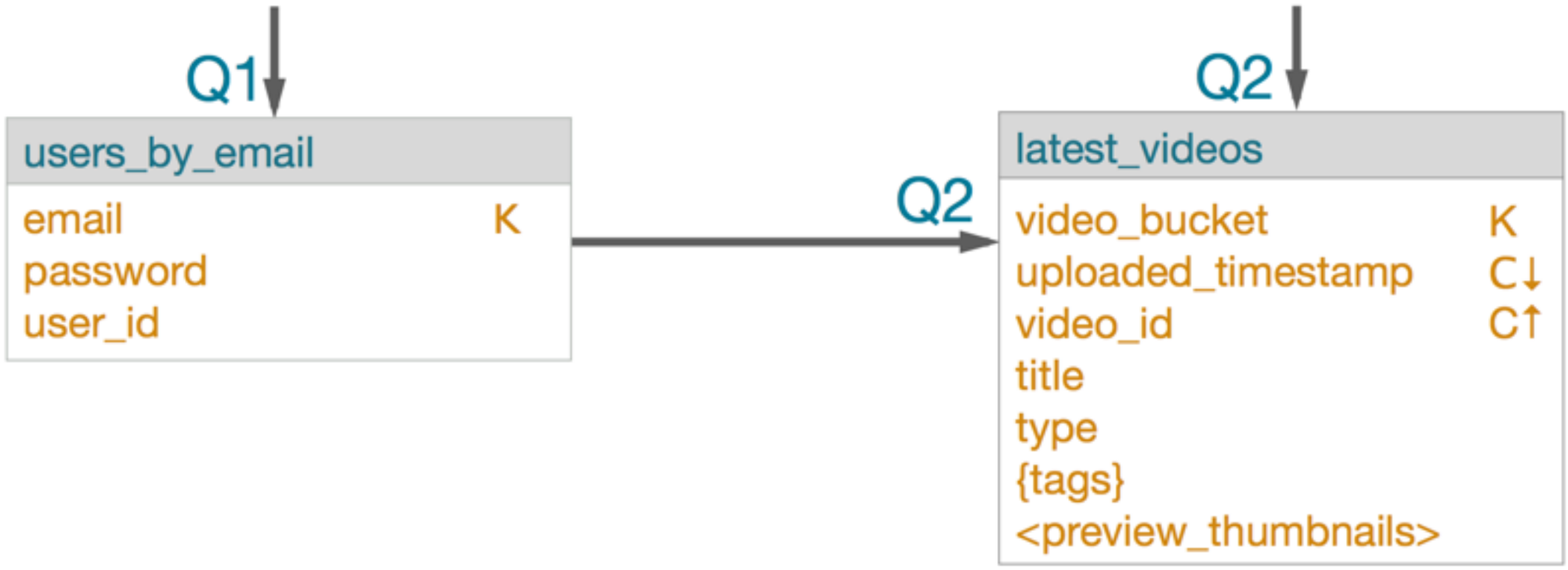


Query-Driven Data Modeling



Chebotko Diagram

- Visual diagram for Cassandra tables and access patterns



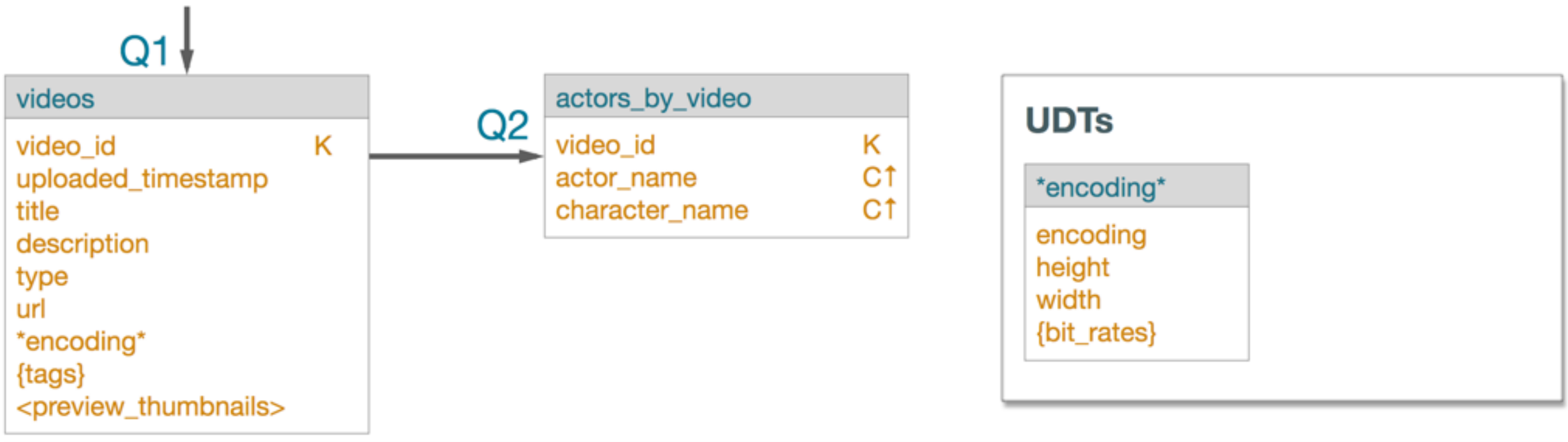
ACCESS PATTERNS

Q1: Find a user with a **specified email**

Q2: Find most recently uploaded videos

Chebotko Diagrams

- Graphical representation of Cassandra database schema design
- Documents the logical and physical data model

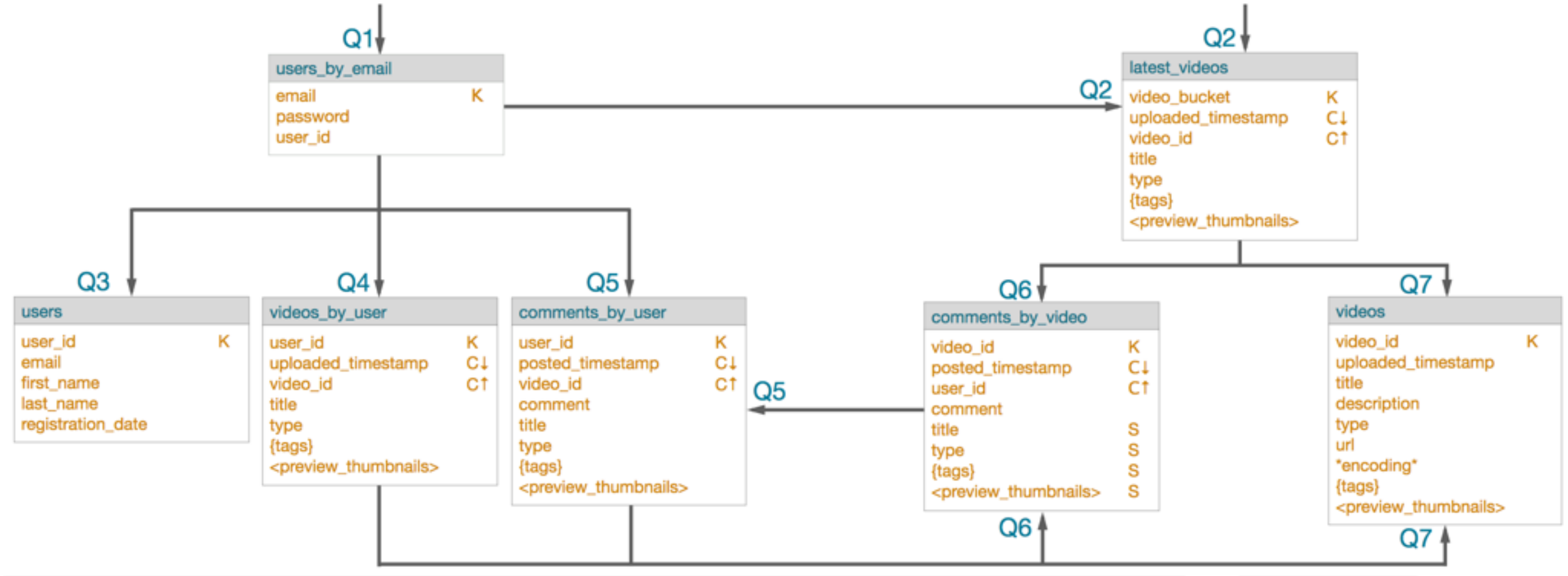


ACCESS PATTERNS

Q1: Find a video with a **specified video id**

Q2: Find actors for a **video with a known id** (show actor names in ascending order)

Example Chebotko Diagram



ACCESS PATTERNS

Q1: Find a user with a **specified email**
Q2: Find most recently uploaded videos
Q3: Find a **user with a specified id**
Q4: Find videos uploaded by a **user with a known id** (show most recently uploaded videos first)
Q5: Find comments posted by a **user with a known id** (show most recently posted comments first)
Q6: Find comments posted for a **video with a known id** (show most recent comments first)
Q7: Find a video with a **specified video id**

UDTs

encoding
encoding
height
width
{bit_rates}

Duplicate Data

Better to duplicate than to join data

- Partition per query and data nesting may result in data duplication
  - Query results are pre-computed and materialized
  - Data can be duplicated across tables, partitions, and / or rows

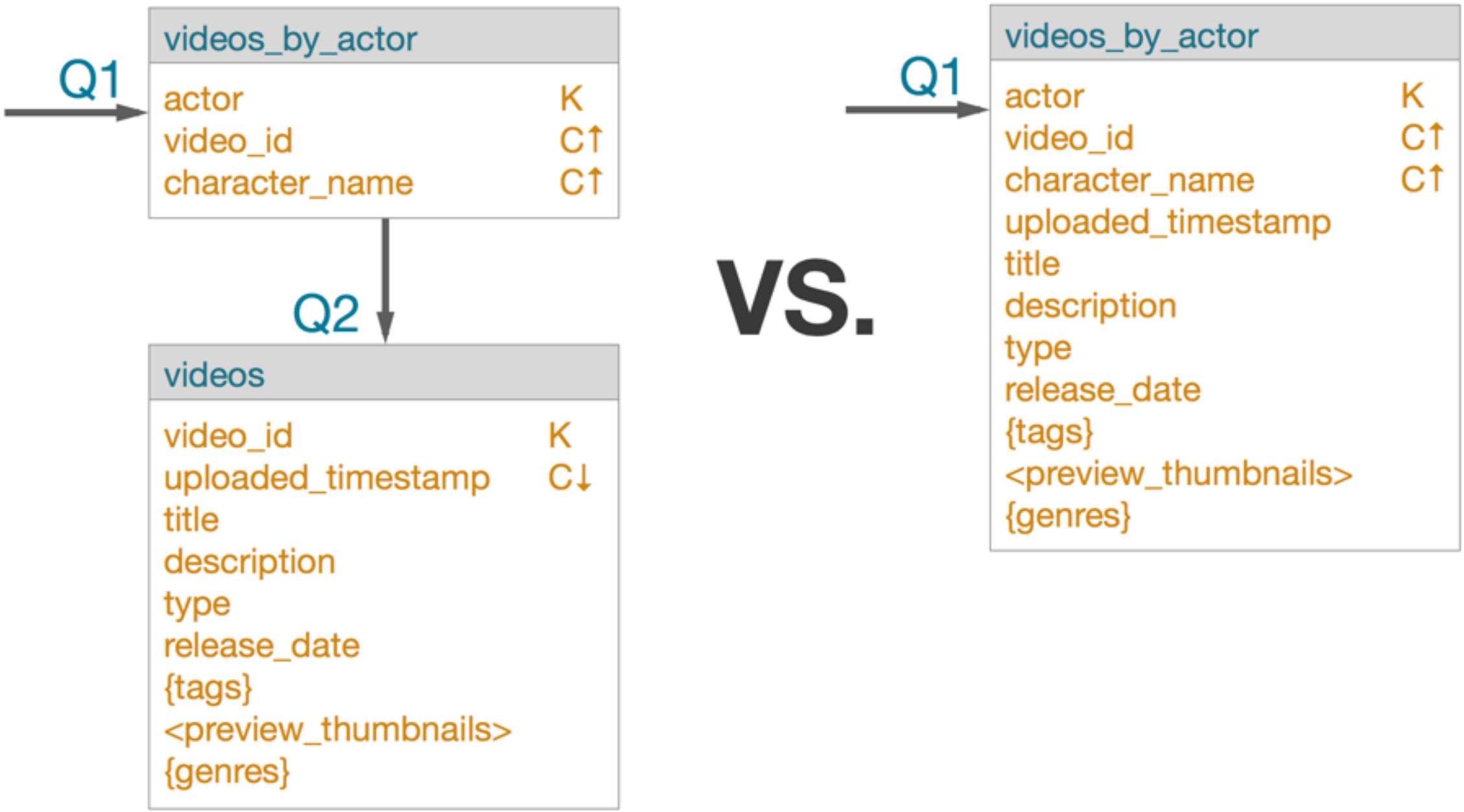
videos_by_actor	
actor	K
release_date	C↓
video_id	C↑
title	
type	
{tags}	
<preview_thumbnails>	

videos_by_genre	
genre	K
release_date	C↓
video_id	C↑
title	
type	
{tags}	
<preview_thumbnails>	

videos_by_tag	
tag	K
release_date	C↓
video_id	C↑
title	
type	
{tags}	
<preview_thumbnails>	

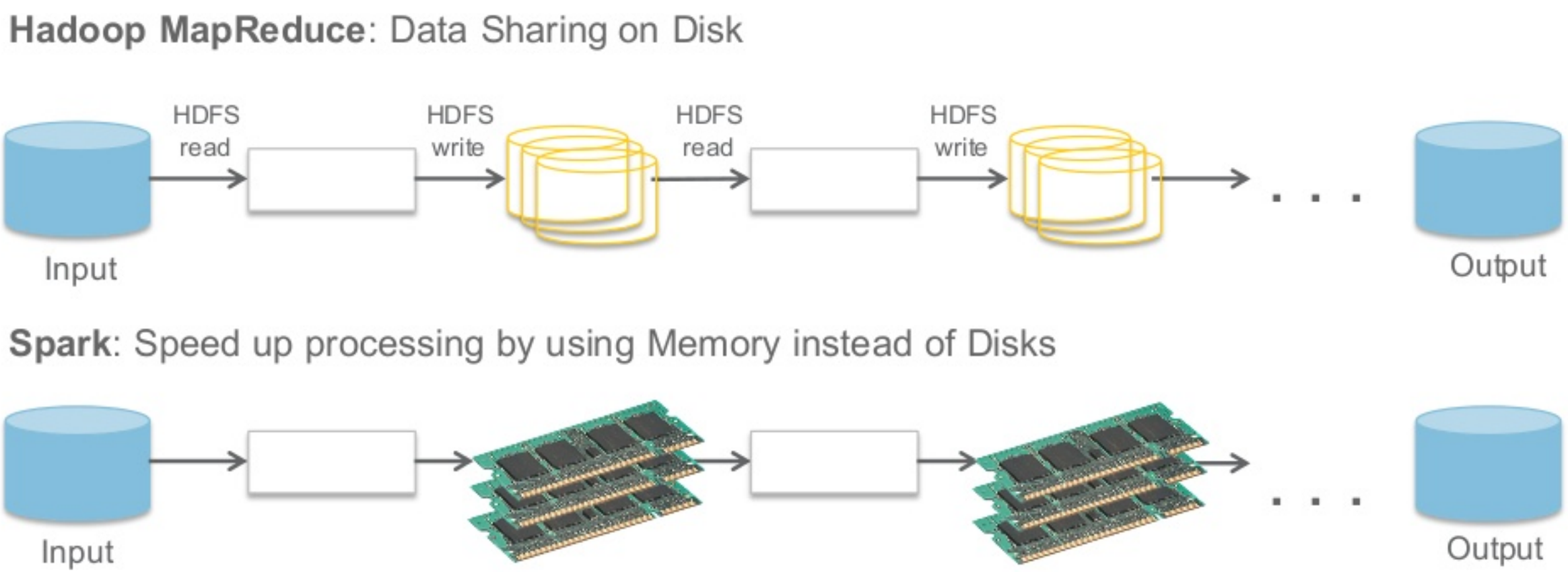
Duplicate Data

Data duplication can scale, joins cannot

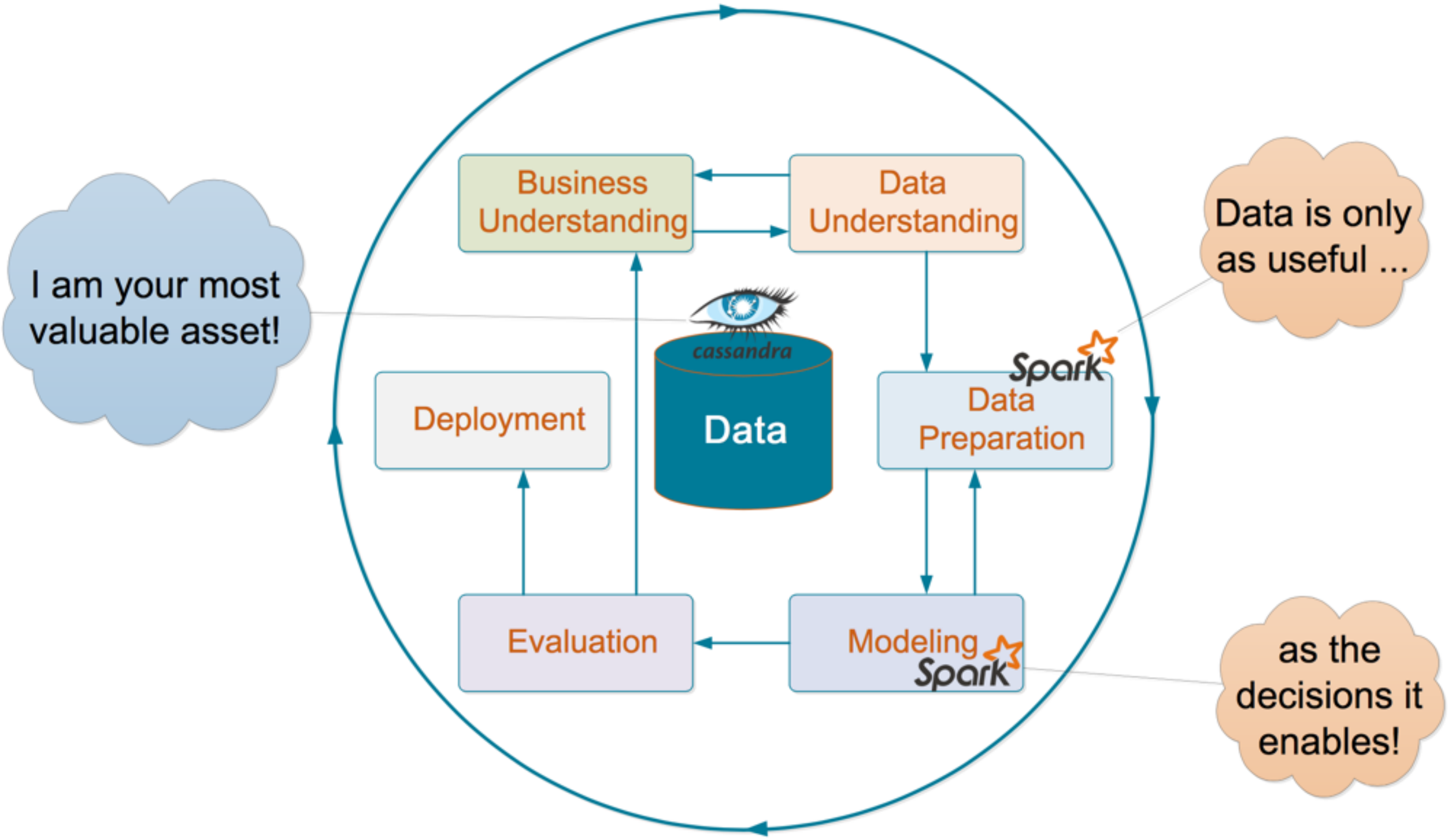



Spark

Apache Spark for data in motion



spark / cassandra role



Distributed computation engine designed for big data and in-memory processing

- Interactive and batch analytics
- Up to 100x faster than Hadoop
- 5-10x less code than Hadoop

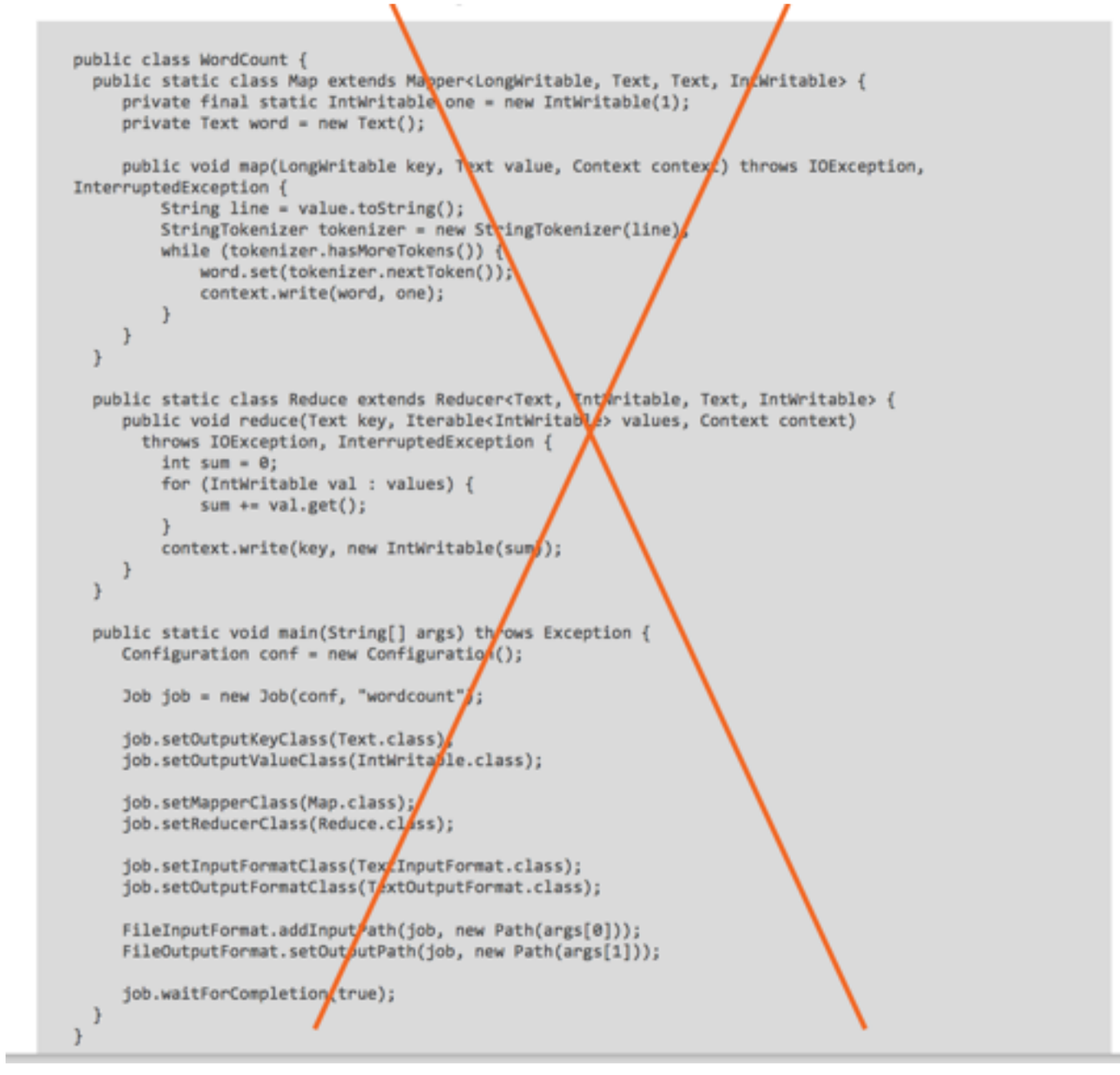
Databricks, Inc. [US] https://databricks.com/blog/2014/11/05/spark-officially-sets-a-new-record-in-large-scale-sorting.html

Spark officially sets a new record in large-scale sorting

November 5, 2014 | by Reynold Xin

	Hadoop MR Record	**Spark Record**	**Spark 1 PB**
Data Size	102.5 TB	100 TB	1000 TB
Elapsed Time	72 mins	23 mins	234 mins
# Nodes	2100	206	190
# Cores	50400 physical	6592 virtualized	6080 virtualized
Cluster disk throughput	3150 GB/s (est.)	618 GB/s	570 GB/s
Sort Benchmark Daytona Rules	Yes	Yes	No
Network	dedicated data center, 10Gbps	virtualized (EC2) 10Gbps network	virtualized (EC2) 10Gbps network
Sort rate	**1.42 TB/min**	**4.27 TB/min**	**4.27 TB/min**
Sort rate/node	**0.67 GB/min**	**20.7 GB/min**	**22.5 GB/min**

big data



```
public class WordCount {
  public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
     private final static IntWritable one = new IntWritable(1);
     private Text word = new Text();

     public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
         String line = value.toString();
         StringTokenizer tokenizer = new StringTokenizer(line);
         while (tokenizer.hasMoreTokens()) {
             word.set(tokenizer.nextToken());
             context.write(word, one);
         }
     }
  }

  public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
     public void reduce(Text key, Iterable<IntWritable> values, Context context)
       throws IOException, InterruptedException {
         int sum = 0;
         for (IntWritable val : values) {
             sum += val.get();
         }
         context.write(key, new IntWritable(sum));
     }
  }

  public static void main(String[] args) throws Exception {
     Configuration conf = new Configuration();

     Job job = new Job(conf, "wordcount");

     job.setOutputKeyClass(Text.class);
     job.setOutputValueClass(IntWritable.class);

     job.setMapperClass(Map.class);
     job.setReducerClass(Reduce.class);

     job.setInputFormatClass(TextInputFormat.class);
     job.setOutputFormatClass(TextOutputFormat.class);

     FileInputFormat.addInputPath(job, new Path(args[0]));
     FileOutputFormat.setOutputPath(job, new Path(args[1]));

     job.waitForCompletion(true);
  }
}
```

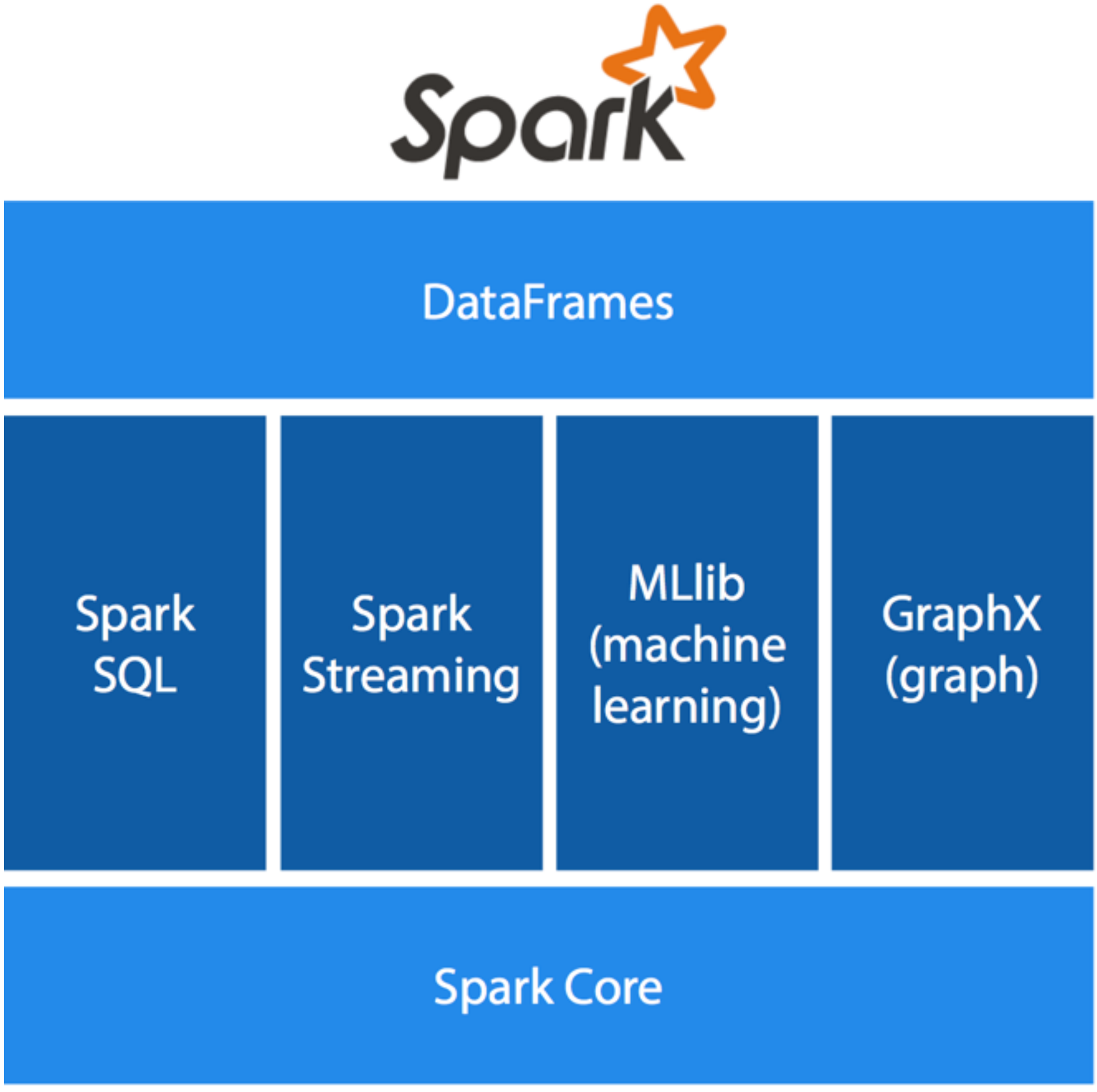
big data

```
public class WordCount {
  public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
     private final static IntWritable one = new IntWritable(1);
     private Text word = new Text();

     public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
         String line = value.toString();
         StringTokenizer tokenizer = new StringTokenizer(line);
         while (tokenizer.hasMoreTokens()) {
             word.set(tokenizer.nextToken());
             context.write(word, one);
         }
     }
  }

  public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
     public void reduce(Text key, Iterable<IntWritable> values, Context context)
       throws IOException, InterruptedException {
         int sum = 0;
         for (IntWritable val : values) {
             sum += val.get();
         }
         context.write(key, new IntWritable(sum));
     }
  }

  public static void main(String[] args) throws Exception {
     Configuration conf = new Configuration();

     Job job = new Job(conf, "wordcount");

     job.setOutputKeyClass(Text.class);
     job.setOutputValueClass(IntWritable.class);

     job.setMapperClass(Map.class);
     job.setReducerClass(Reduce.class);

     job.setInputFormatClass(TextInputFormat.class);
     job.setOutputFormatClass(TextOutputFormat.class);

     FileInputFormat.addInputPath(job, new Path(args[0]));
     FileOutputFormat.setOutputPath(job, new Path(args[1]));

     job.waitForCompletion(true);
  }
}
```

```
object WordCount{
   def main(def main(args: Array[String])){
     val sparkConf = new SparkConf()
                         .setAppName("wordcount")
     val sc = new SparkContext(sparkConf)
     sc.textFile(args(0))
       .flatMap(_.split(" "))
       .countByValue
       .saveAsTextFile(args(1))
   }
}
```

Why spark?

- Readability
- Expressiveness
- Fast
- Testability
- Interactive
- Fault Tolerant
- Unify Big Data


Spark
DataFrames
Spark SQL
Spark Streaming
MLlib (machine learning)
GraphX (graph)
Spark Core

Who is using spark?

OOYALA

Yahoo!

CONVIVA

NETFLIX

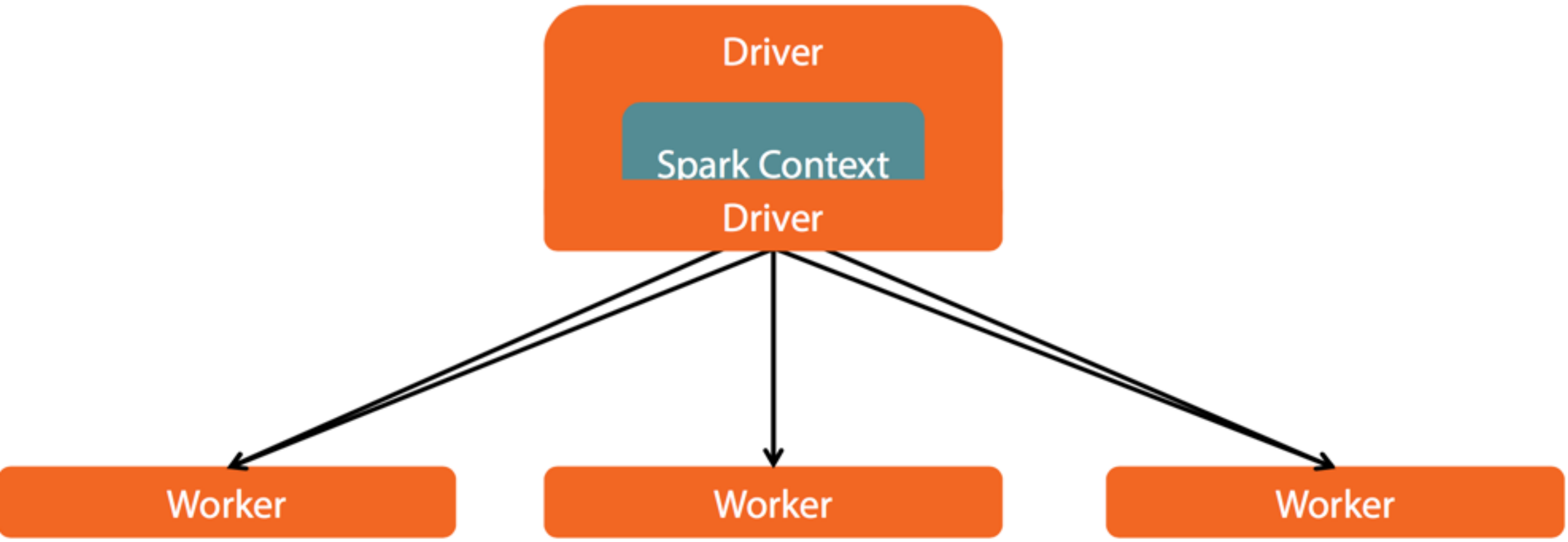
PANDORA

Goldman Sachs

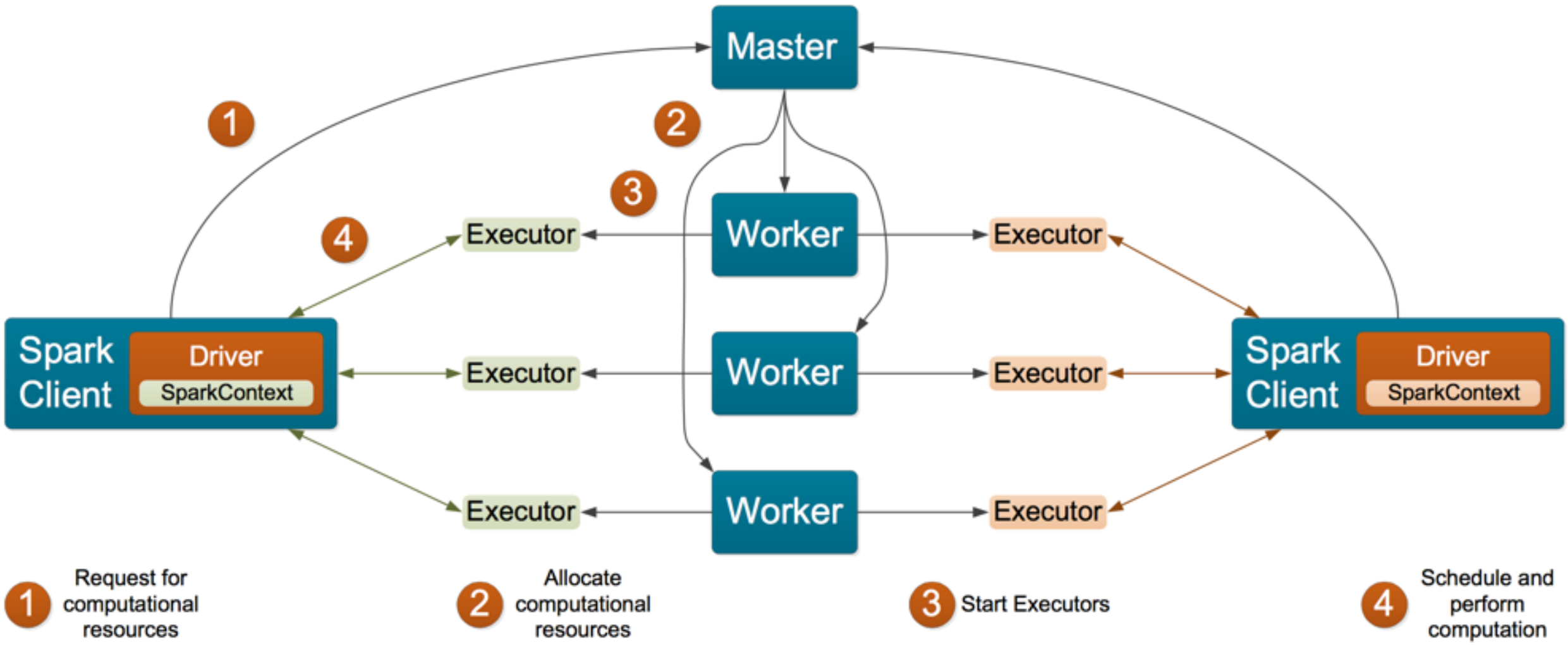
comcast

ebay

hhmi | janelia Research Campus


Driver
Spark Context
Driver
Worker
Worker
Worker

spark architecture



Spark download

Download Apache Spark™

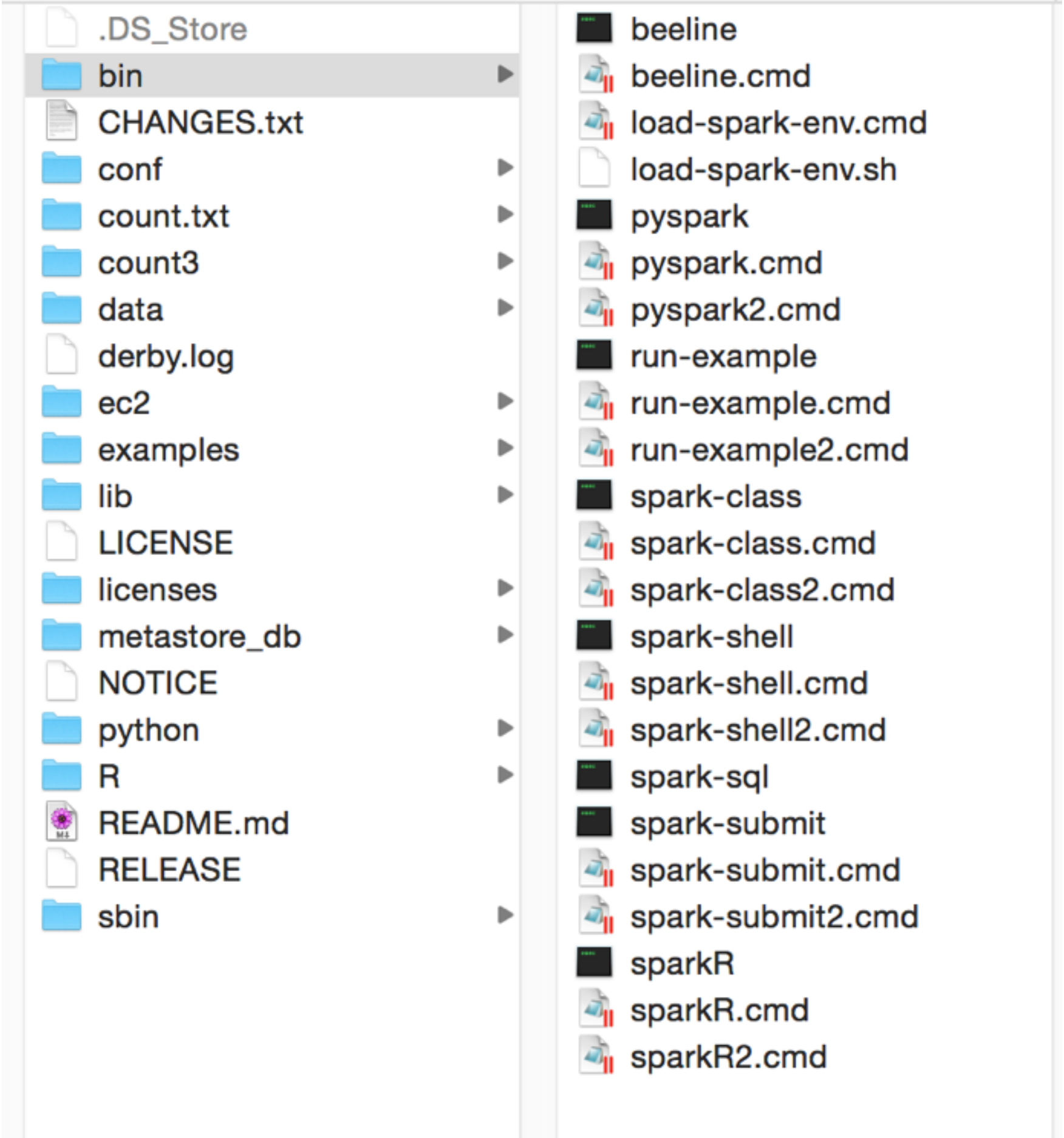
Our latest stable version is Apache Spark 1.6.2, released on June 25, 2016 (release notes) (git tag)

1. Choose a Spark release: 1.6.2 (Jun 25 2016)
2. Choose a package type: Pre-built for Hadoop 2.6
3. Choose a download type: Select Apache Mirror
4. Download Spark: spark-1.6.2-bin-hadoop2.6.tgz
5. Verify this release using the 1.6.2 signatures and checksums and project release KEYS.

Note: Scala 2.11 users should download the Spark source package and build with Scala 2.11 support.

http://www-us.apache.org/dist/spark/spark-1.6.2/spark-1.6.2-bin-hadoop2.6.tgz

Add export SPARK_LOCAL_IP="127.0.0.1" to load-spark-env.sh


.DS_Store
bin
CHANGES.txt
conf
count.txt
count3
data
derby.log
ec2
examples
lib
LICENSE
licenses
metastore_db
NOTICE
python
R
README.md
RELEASE
sbin
beeline
beeline.cmd
load-spark-env.cmd
load-spark-env.sh
pyspark
pyspark.cmd
pyspark2.cmd
run-example
run-example.cmd
run-example2.cmd
spark-class
spark-class.cmd
spark-class2.cmd
spark-shell
spark-shell.cmd
spark-shell2.cmd
spark-sql
spark-submit
spark-submit.cmd
spark-submit2.cmd
sparkR
sparkR.cmd
sparkR2.cmd

Spark shell

- ./bin/spark-shell --master local[2]
- Interactive Scala-REPL-based Spark client

Hello spark

- scala> println("Hello NFJS Spark!")
- Hello NFJS Spark!
- scala> val seven = 2 + 5
- seven: Int = 7

spark context : sc

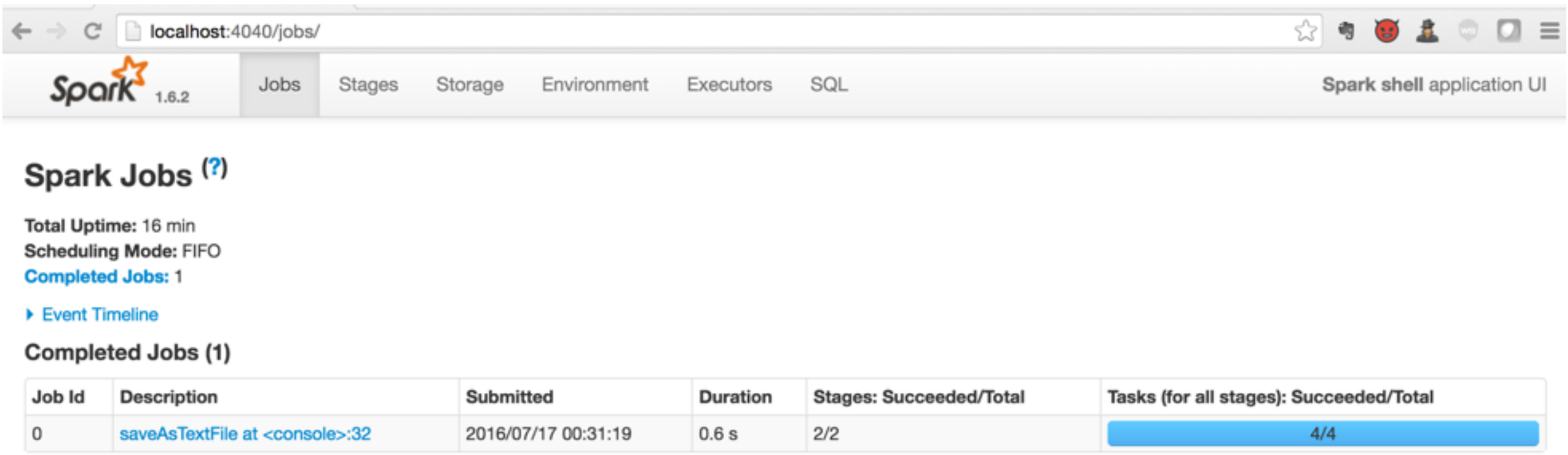
```
scala> sc.
accumulable                    accumulableCollection
accumulator                    addFile
addJar                         addSparkListener
appName                        applicationAttemptId
applicationId                  asInstanceOf
binaryFiles                    binaryRecords
broadcast                      cancelAllJobs
cancelJobGroup                 clearCallSite
clearFiles                     clearJars
clearJobGroup                  defaultMinPartitions
defaultMinSplits               defaultParallelism
emptyRDD                       externalBlockStoreFolderName
files                          getAllPools
getCheckpointDir               getConf
getExecutorMemoryStatus        getExecutorStorageStatus
getLocalProperty               getPersistentRDDs
getPoolForName                 getRDDStorageInfo
getSchedulingMode              hadoopConfiguration
hadoopFile                     hadoopRDD
initLocalProperties            isInstanceOf
isLocal                        isStopped
jars                           killExecutor
killExecutors                  makeRDD
master                         metricsSystem
newAPIHadoopFile               newAPIHadoopRDD
objectFile                     parallelize
range                          requestExecutors
runApproximateJob              runJob
sequenceFile                   setCallSite
setCheckpointDir               setJobDescription
setJobGroup                    setLocalProperty
setLogLevel                    sparkUser
startTime                      statusTracker
stop                           submitJob
tachyonFolderName              textFile
toString                       union
version                        wholeTextFiles
```

Word Count - to directory

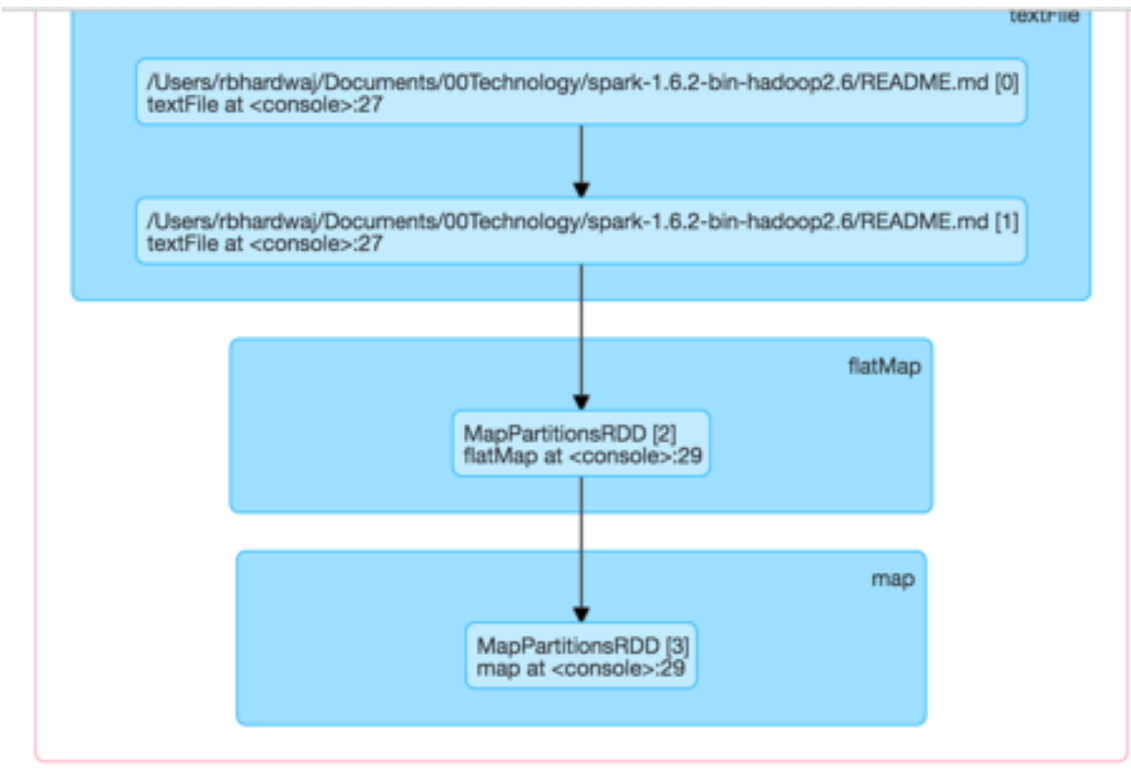
- :paste
- // Entering paste mode (ctrl-D to finish)

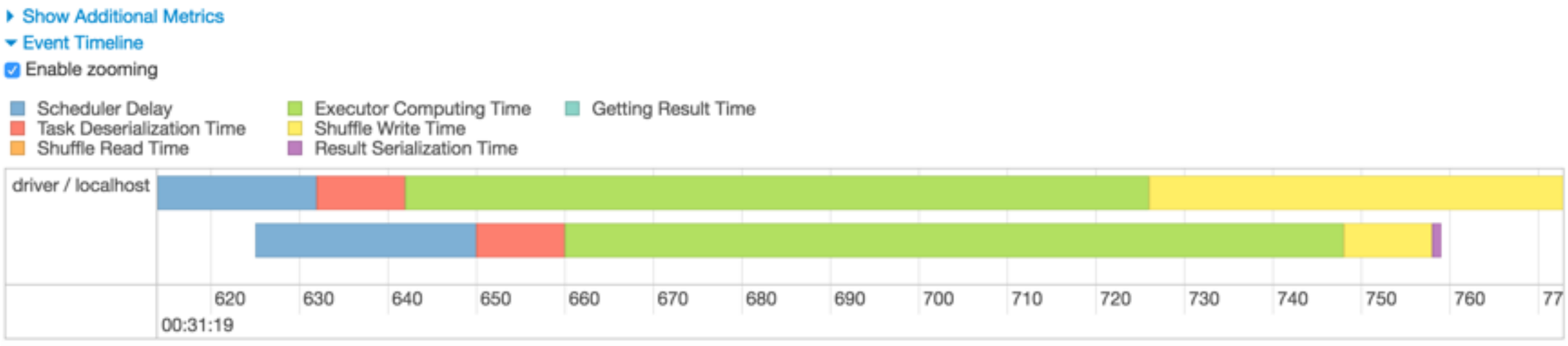
- val textFile = sc.textFile("README.md")
- val counts = textFile
- .flatMap(line => line.split(" "))
- .map(word => (word, 1))
- .reduceByKey(_ + _)
- counts.saveAsTextFile("count5")
- // Exiting paste mode, now interpreting.

Application web ui



stages


textFile
/Users/rbhardwaj/Documents/00Technology/spark-1.6.2-bin-hadoop2.6/README.md [0]
textFile at <console>:27
/Users/rbhardwaj/Documents/00Technology/spark-1.6.2-bin-hadoop2.6/README.md [1]
textFile at <console>:27
flatMap
MapPartitionsRDD [2]
flatMap at <console>:29
map
MapPartitionsRDD [3]
map at <console>:29


Show Additional Metrics
Event Timeline
Enable zooming
Scheduler Delay
Task Deserialization Time
Shuffle Read Time
Executor Computing Time
Shuffle Write Time
Result Serialization Time
Getting Result Time
driver / localhost
620
630
640
650
660
670
680
690
700
710
720
730
740
750
760
77
00:31:19

Wordcount comma separated

- :paste
- // Entering paste mode (ctrl-D to finish)
- sc.textFile("README.md")
- .flatMap(line => line.split(",").drop(1))
- .map(word => (word,1))
- .reduceByKey{case (x,y) => x + y}
- .collect()
- .foreach(println)
- // Exiting paste mode, now interpreting.

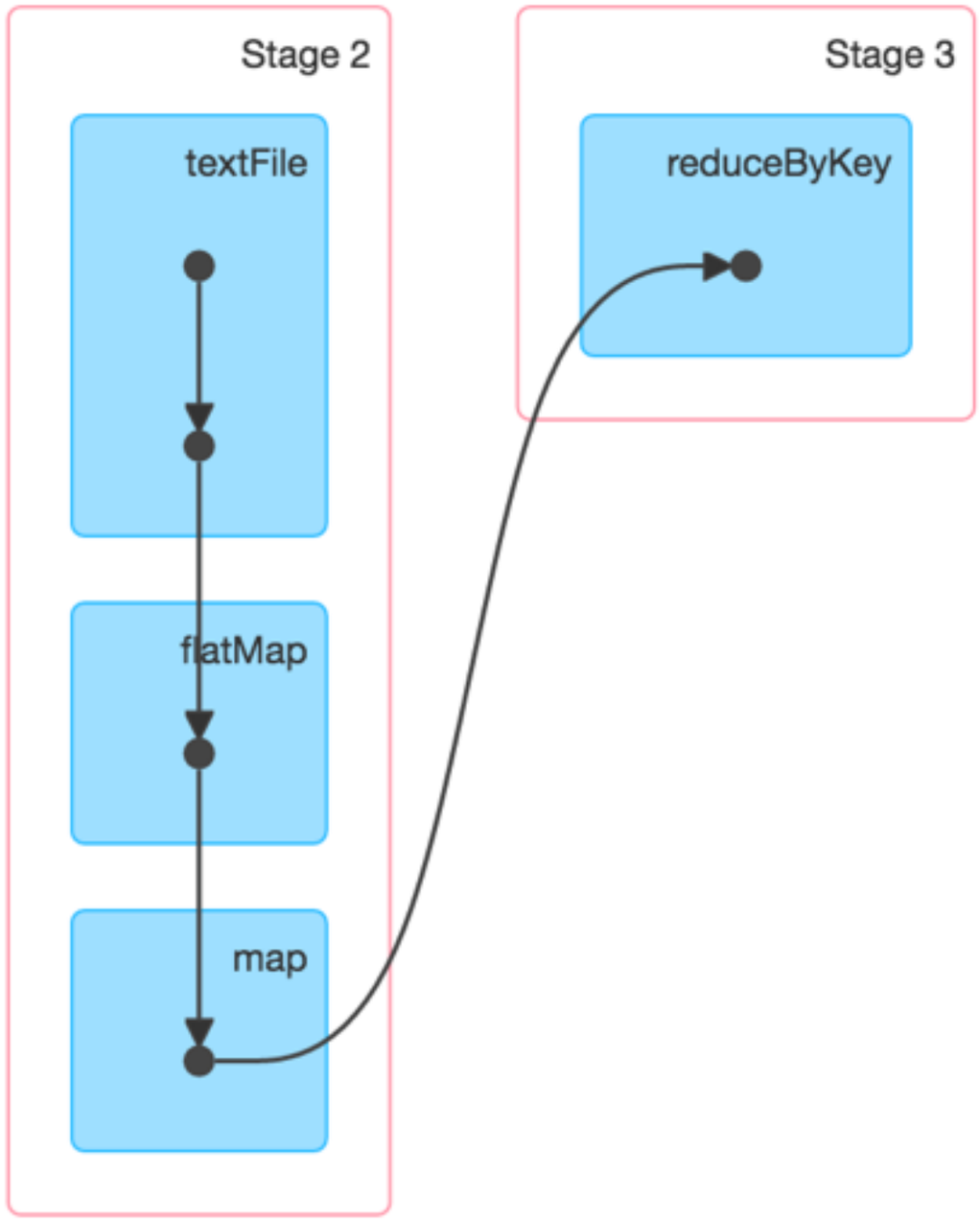
Details for Job 1

Status: SUCCEEDED

Completed Stages: 2

▸ Event Timeline

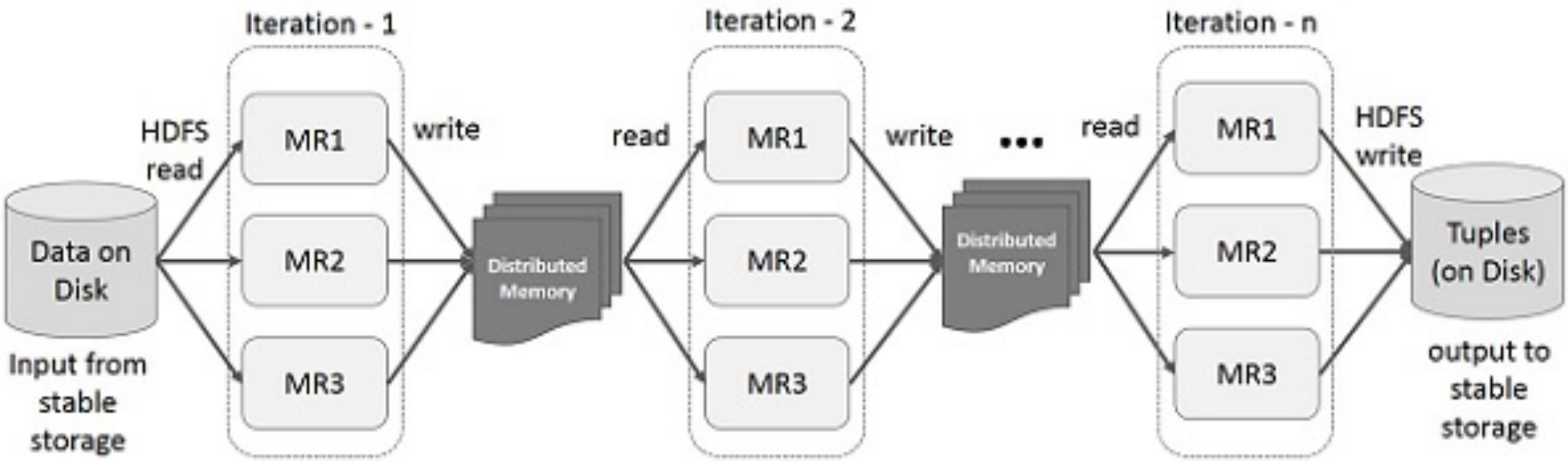
▾ DAG Visualization



Resilient Distributed Datasets (RDD)

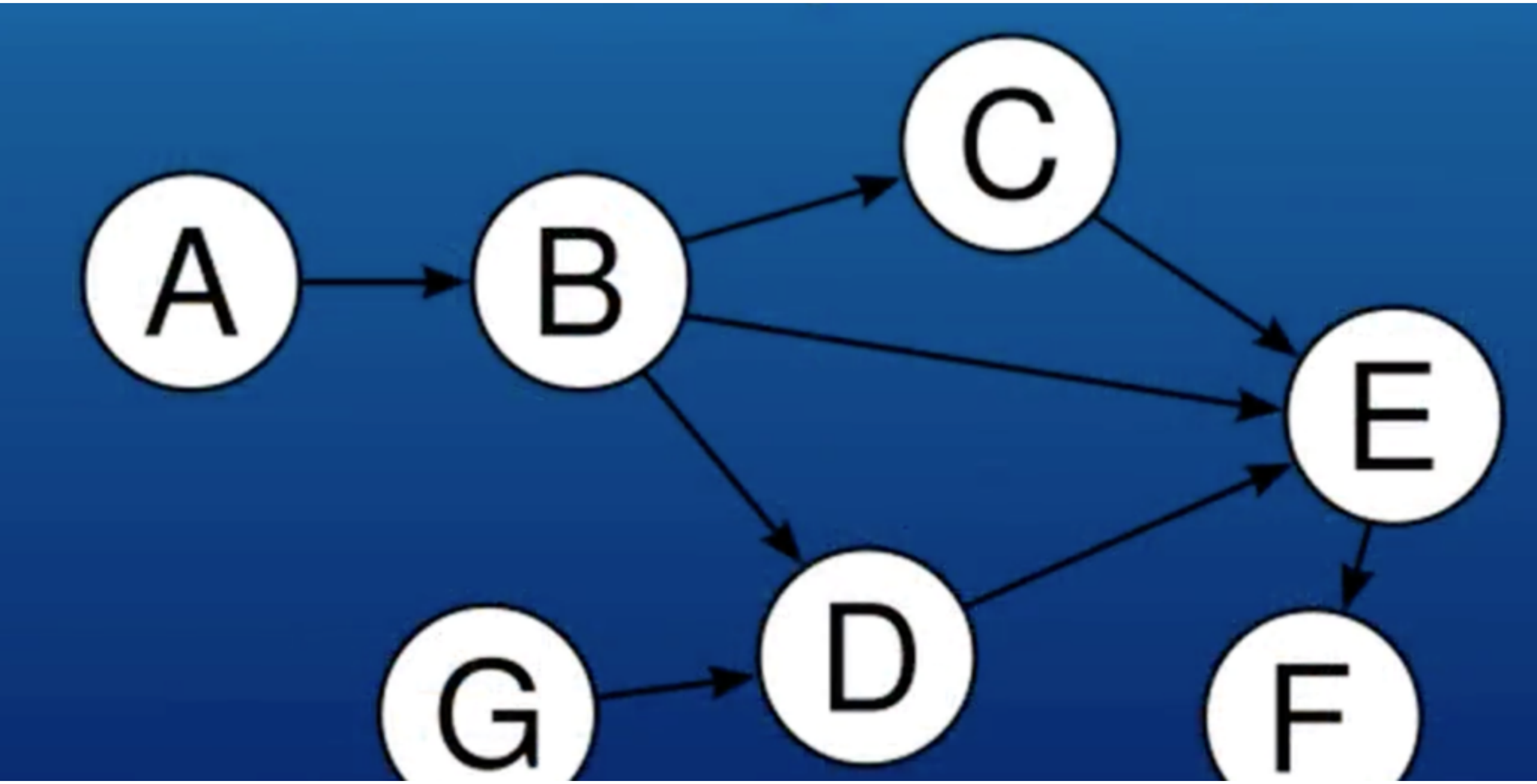
- Immutable
- Re-computable
- Fault tolerant
- Reusable

Resilient Distributed Dataset - RDD

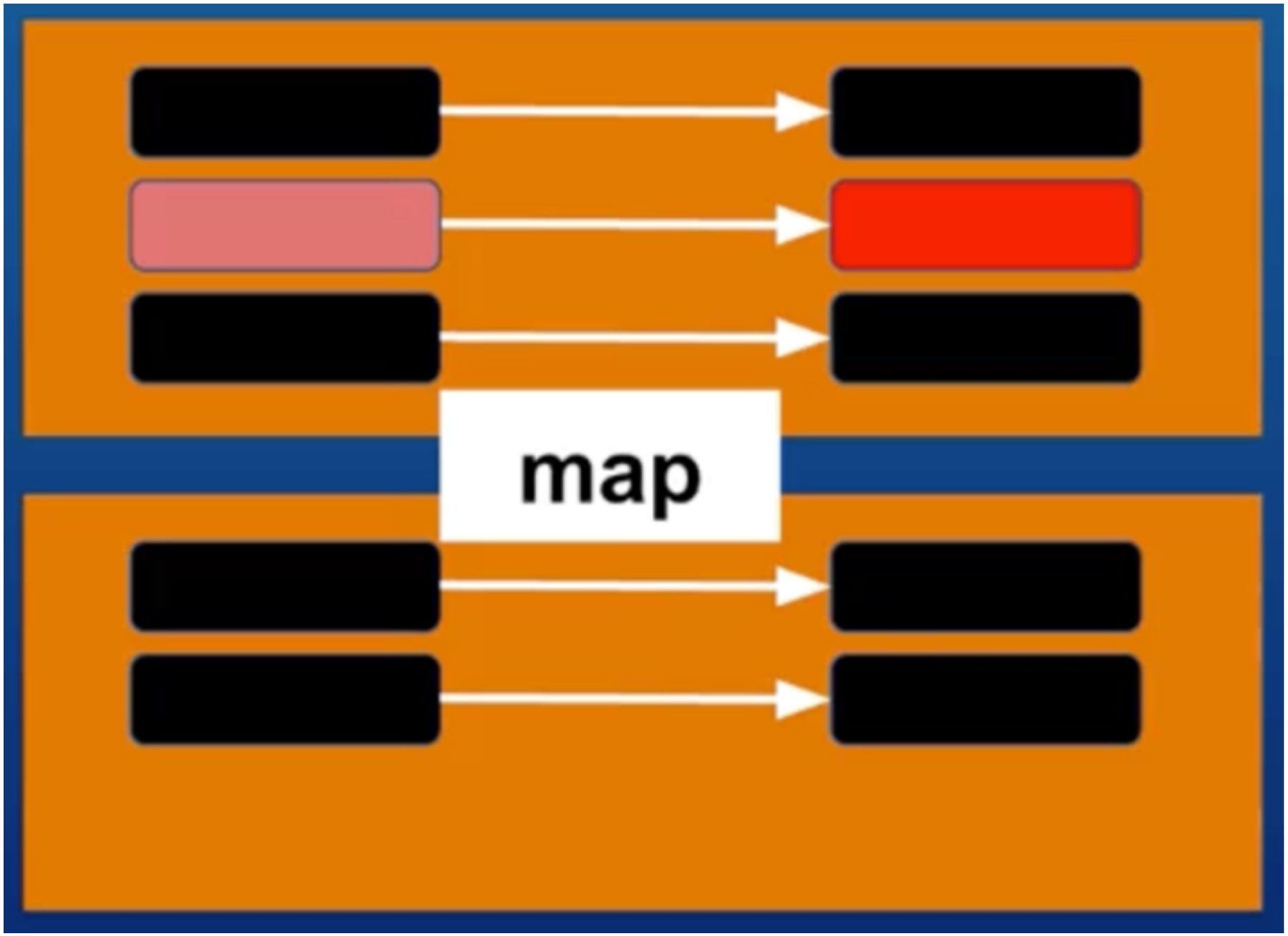


- DAG: Directed Acyclic Graph
- Transformations - map - filter -...
- Actions - collect - collect, count, reduce -...

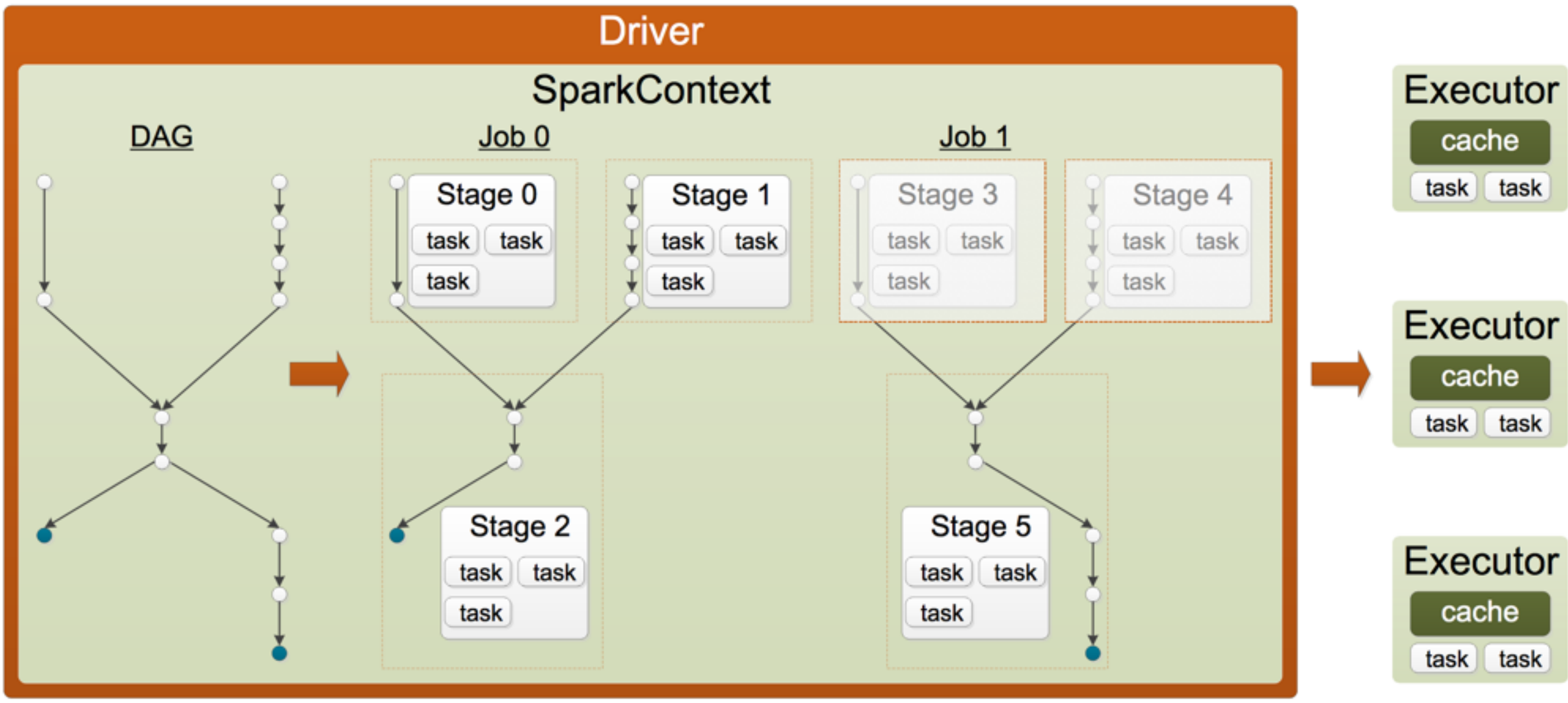
DAG: Directed Acyclic Graph



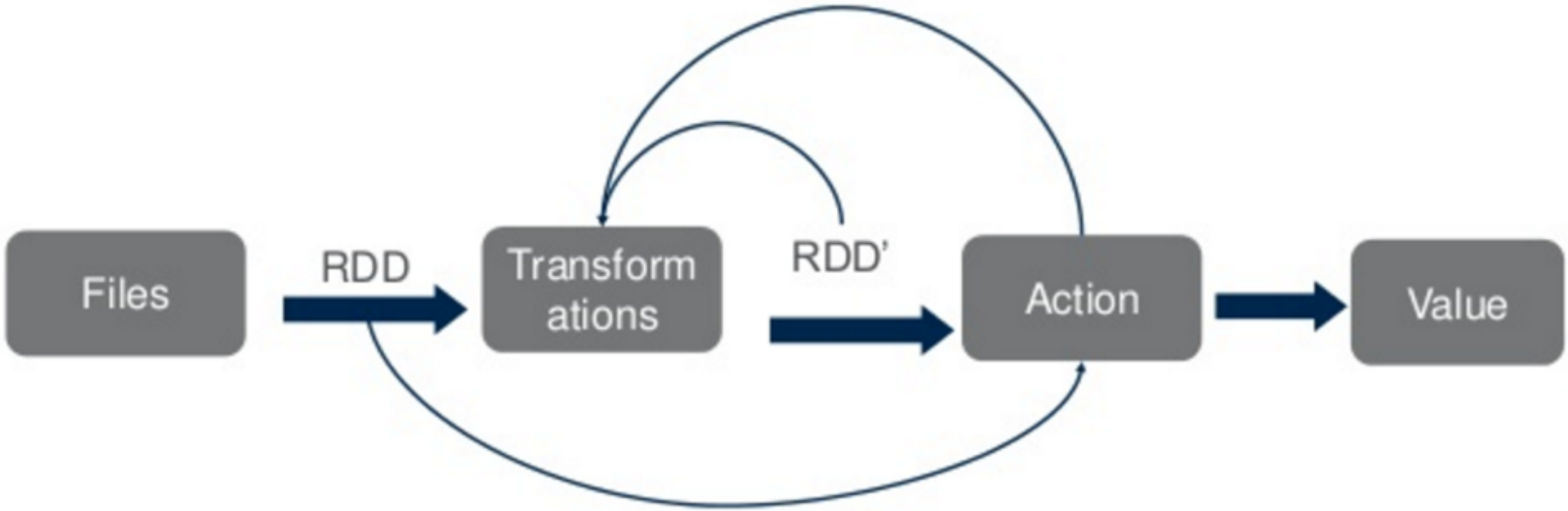
Recovery

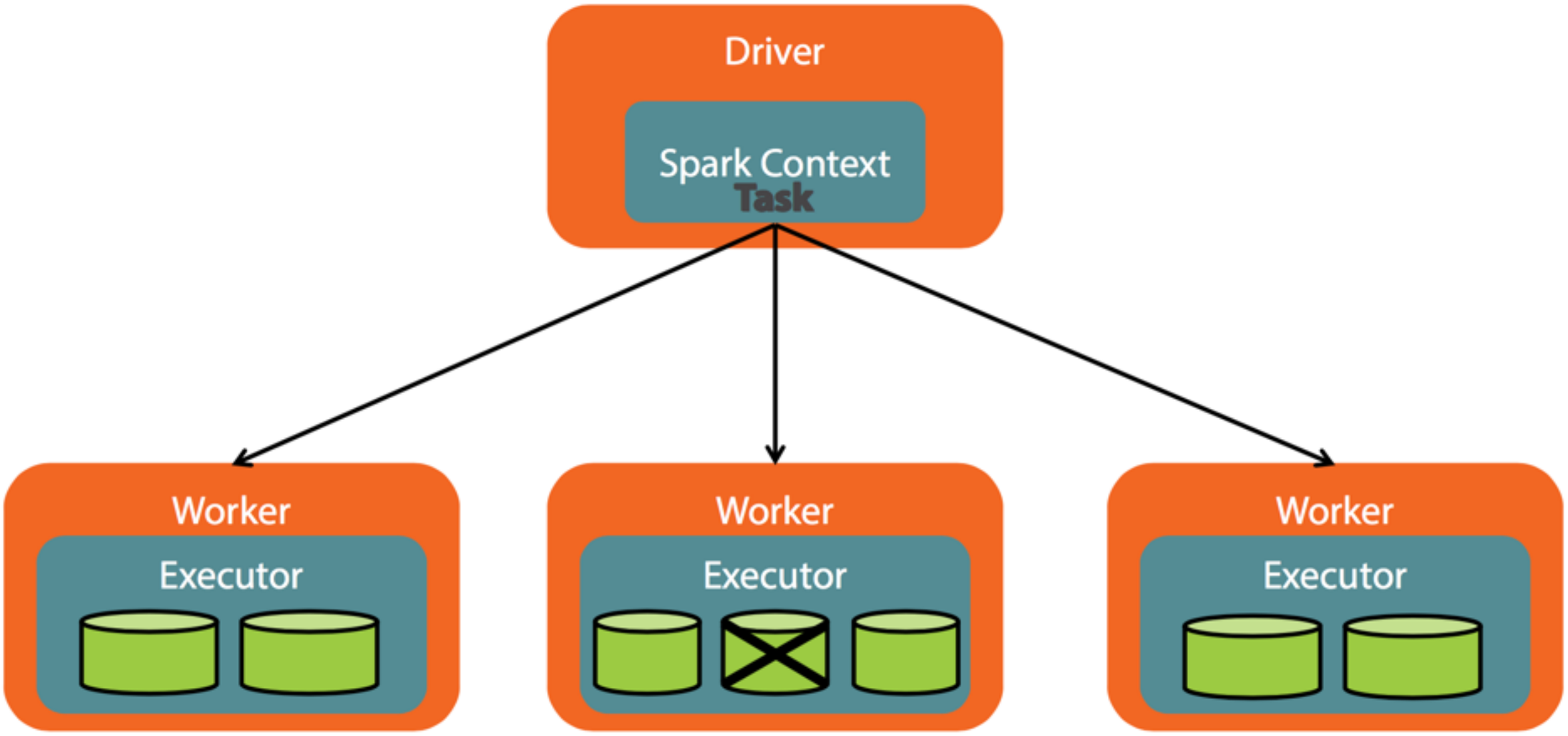


dag and stages

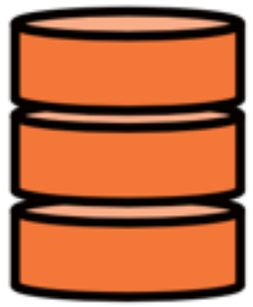


Spark general flow



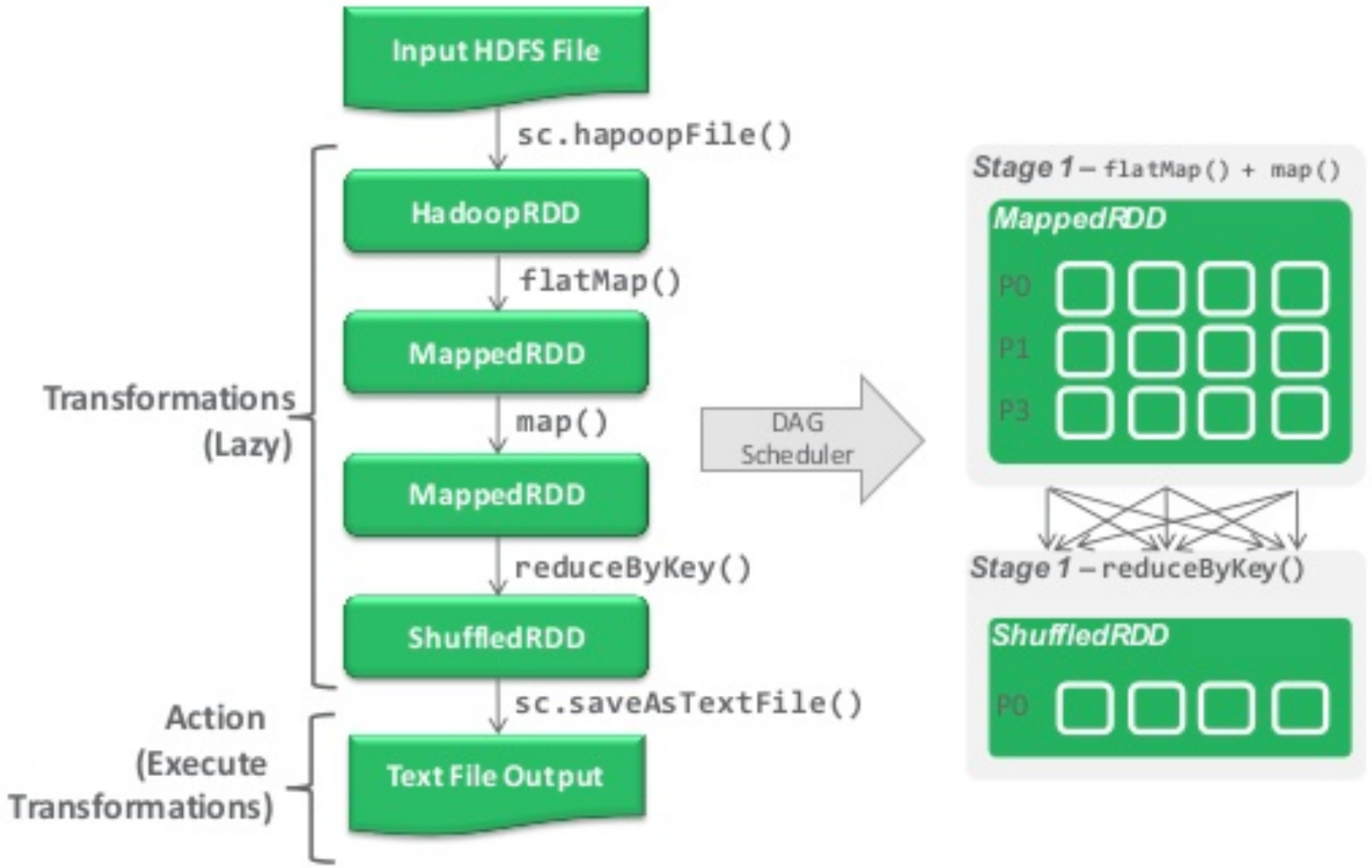

Driver
Spark Context
Task
Worker
Executor
Worker
Executor
Worker
Executor

input



S3

Spark workflow

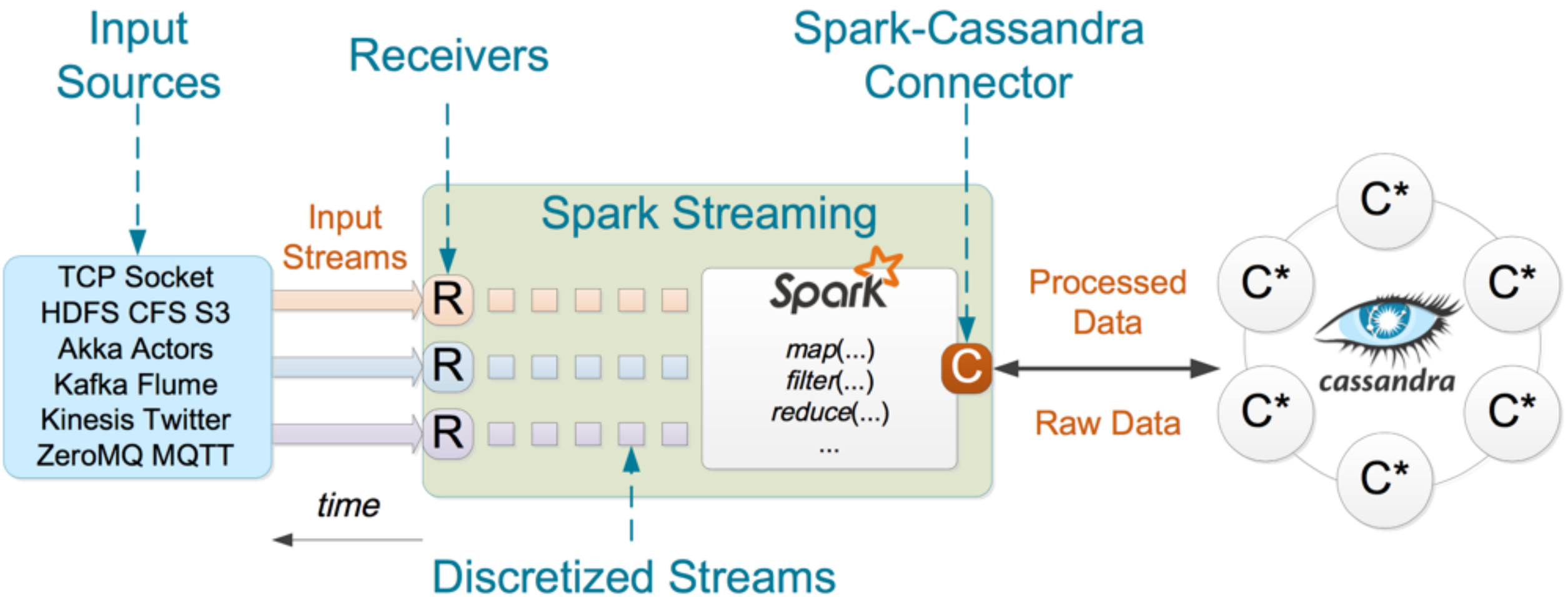


Spark SQL example

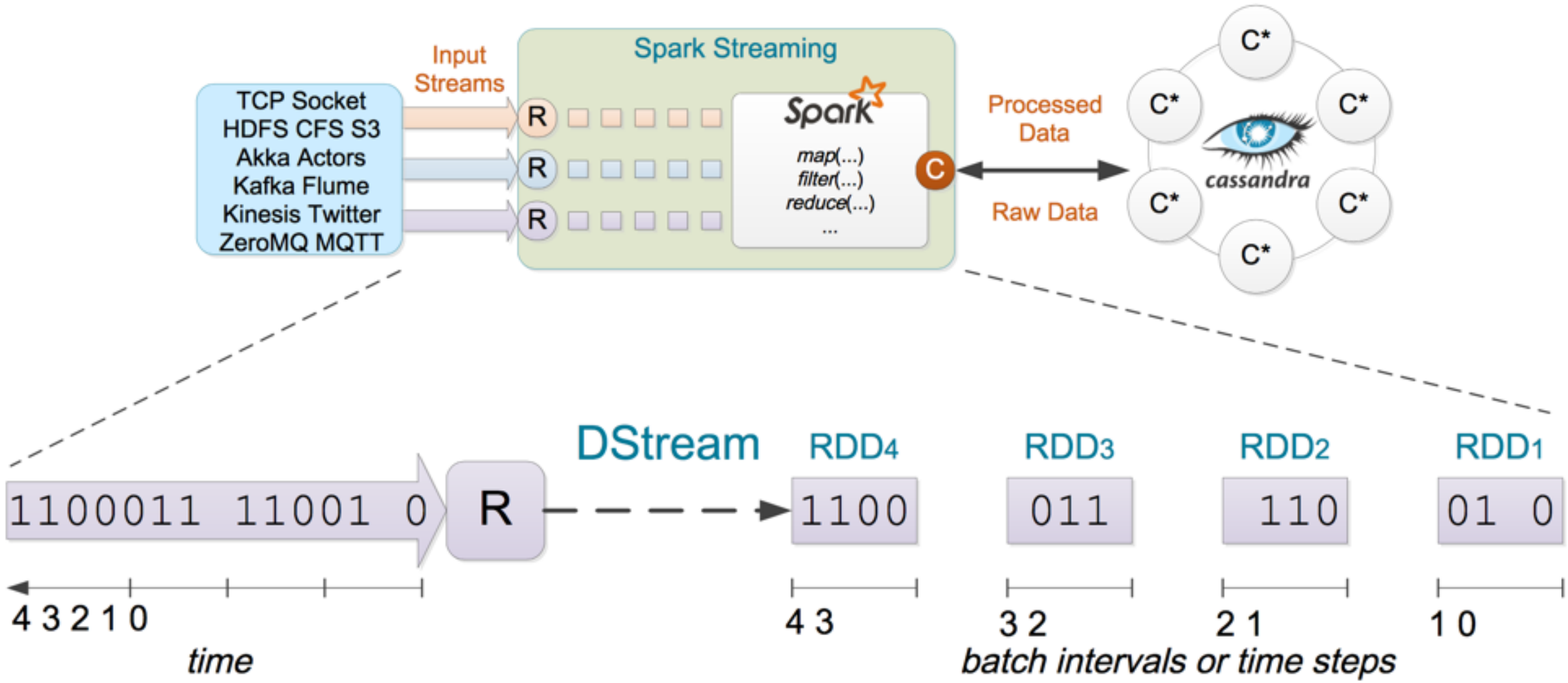
```
import com.datastax.spark.connector._

// Connect to the Spark cluster
val conf = new SparkConf(true)...
val sc = new SparkContext(conf)

// Create Cassandra SQL context
val cc = new CassandraSQLContext(sc)

// Execute SQL query
val rdd = cc.sql("SELECT * FROM keyspace.table WHERE ...")
```

Spark Streaming

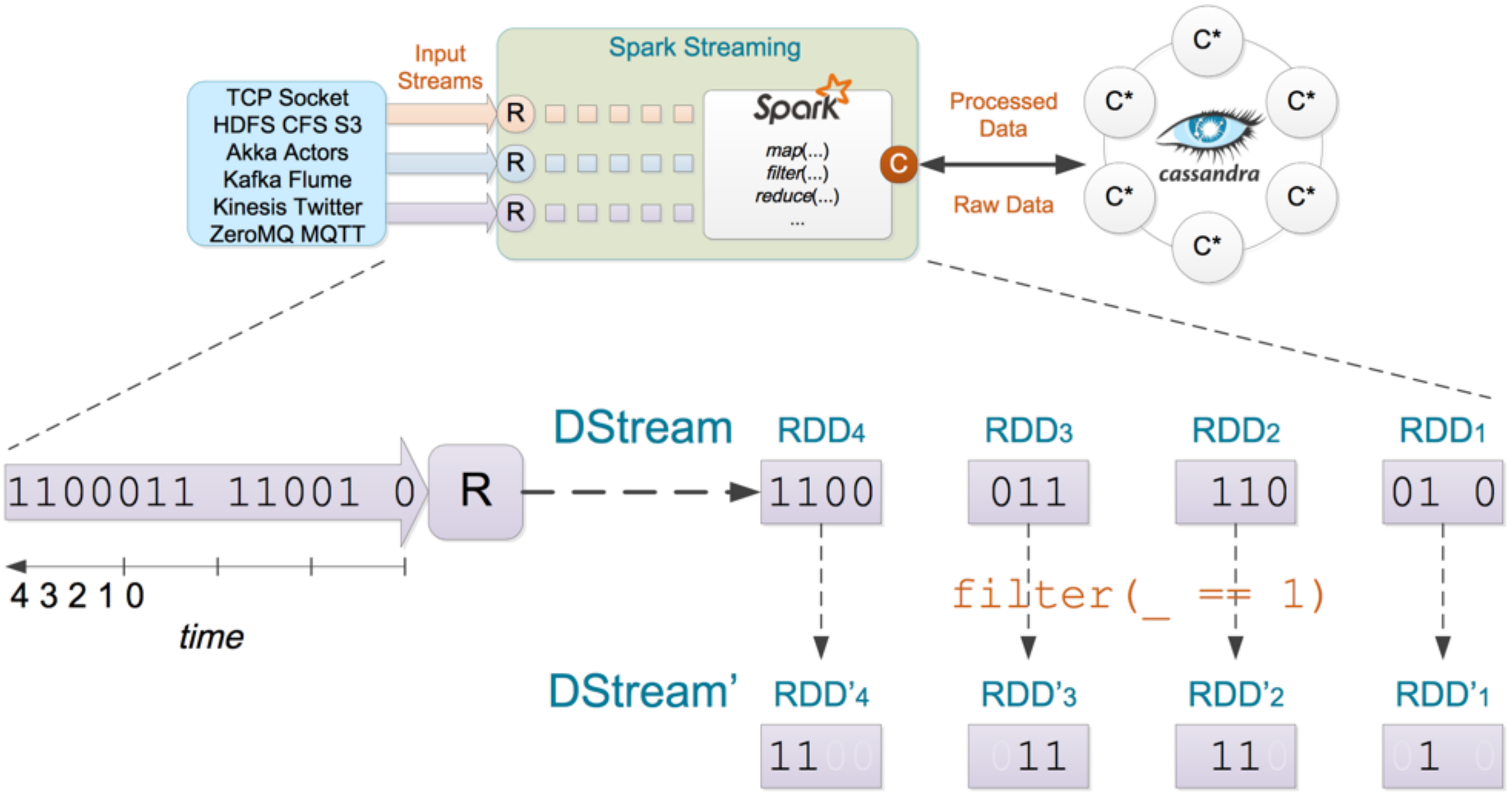
spark streaming



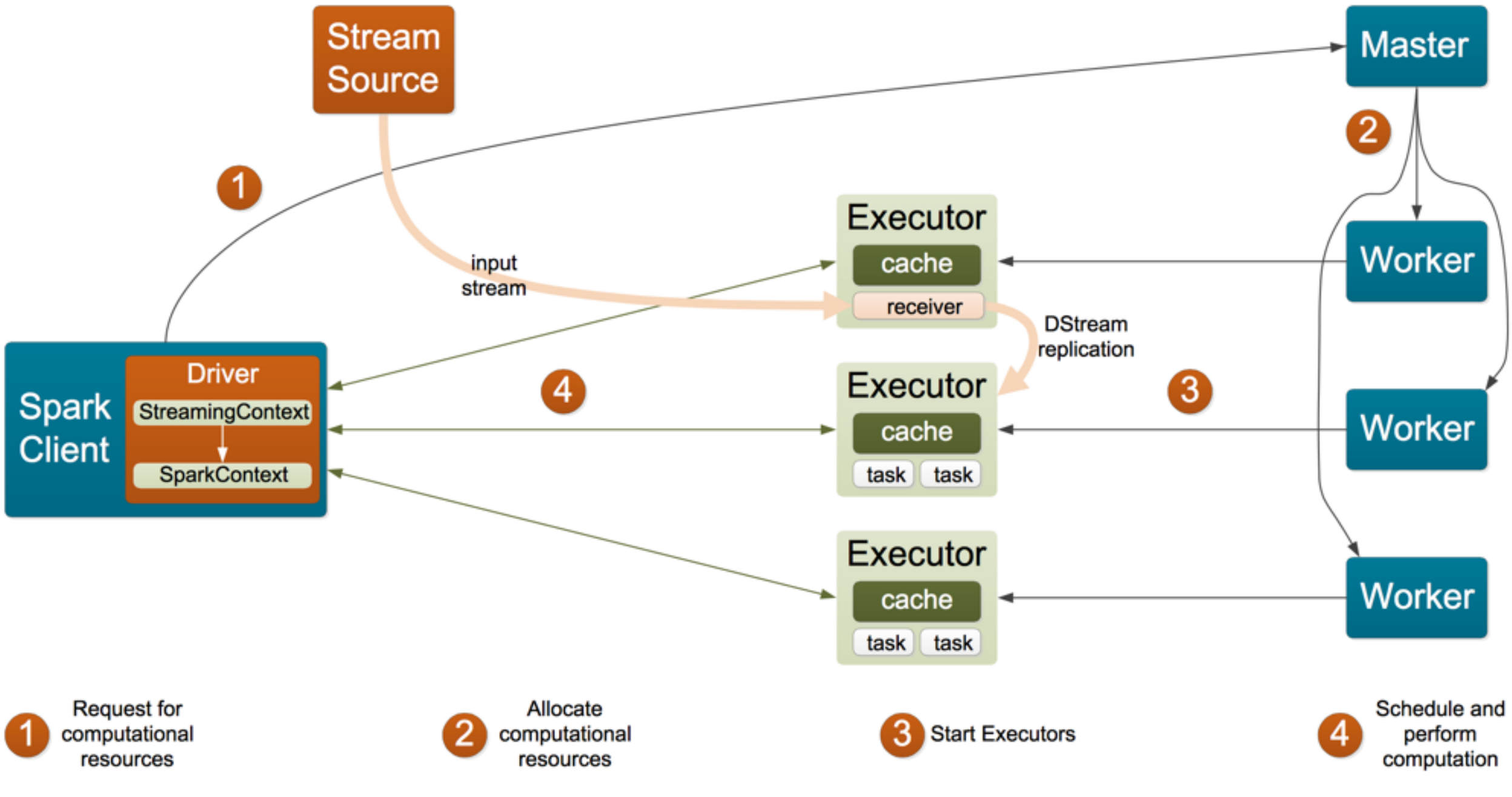
Discretized Stream — DStream



DStream Transformation



Spark Streaming Architecture



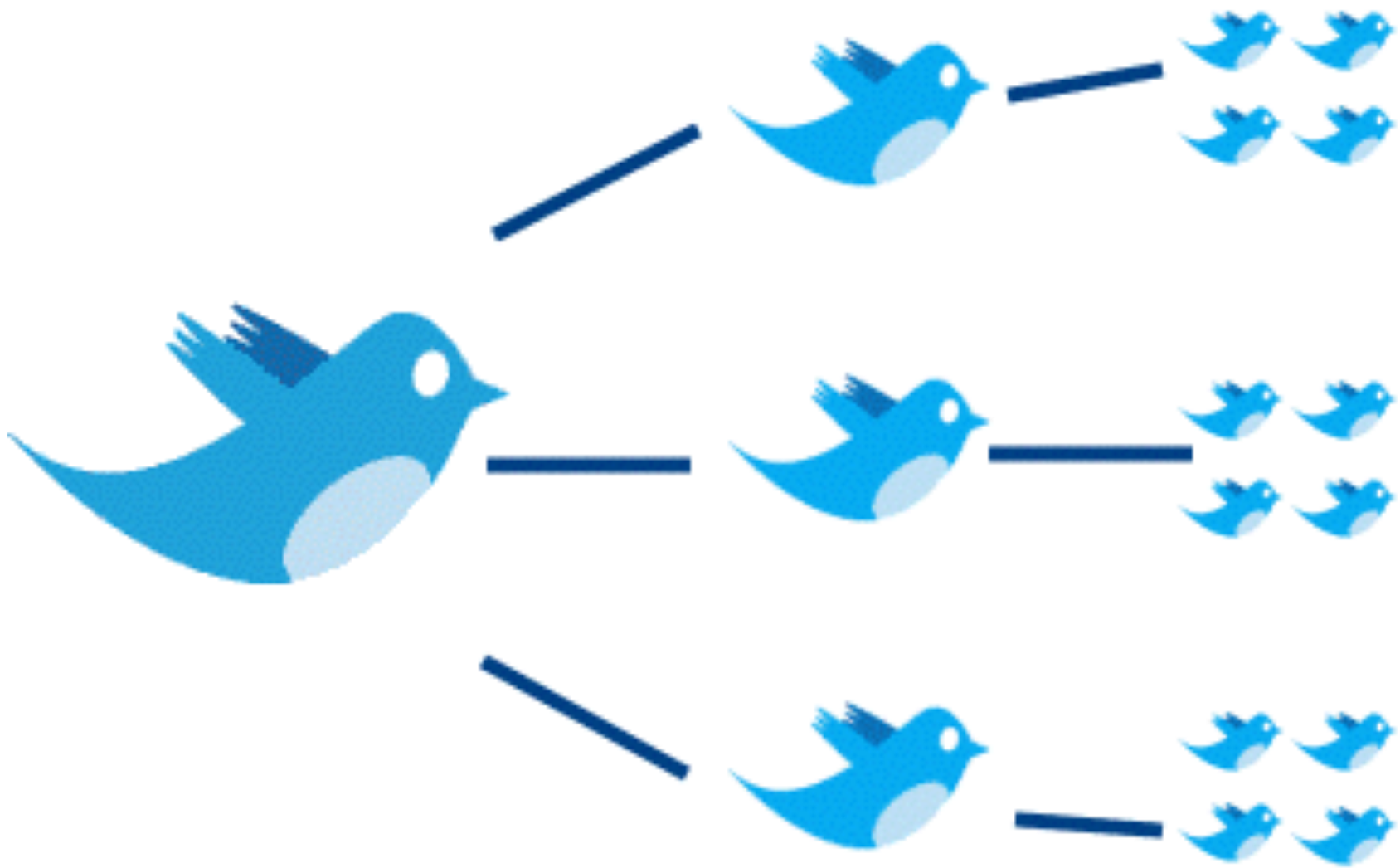
Data Stream Sources

Basic Sources	Advanced Sources	Custom Sources
• File systems • Socket connections • Akka Actors	• Kafka • Flume • Kinesis • Twitter • ZeroMQ • MQTT	• User-defined receivers

QueueStream

```
object QueueStream {

  def main(args: Array[String]) {

    StreamingExamples.setStreamingLogLevels()
    val sparkConf = new SparkConf().setAppName("QueueStream")
    // Create the context
    val ssc = new StreamingContext(sparkConf, Seconds(1))

    // Create the queue through which RDDs can be pushed to
    // a QueueInputDStream
    val rddQueue = new SynchronizedQueue[RDD[Int]]()

    // Create the QueueInputDStream and use it do some processing
    val inputStream = ssc.queueStream(rddQueue)
    val mappedStream = inputStream.map(x => (x % 10, 1))
    val reducedStream = mappedStream.reduceByKey(_ + _)
    reducedStream.print()
    ssc.start()

    // Create and push some RDDs into
    for (i <- 1 to 30) {
      rddQueue += ssc.sparkContext.makeRDD(1 to 1000, 10)
      Thread.sleep(1000)
    }
    ssc.stop()
}
```

twitter stream



streamingexample.scala

```
import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.SparkContext._
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._
import com.datastax.spark.connector._
import com.datastax.spark.connector.streaming._

object StreamingExample {
  def main(args: Array[String]) {

     val conf = new SparkConf(true)
        .setAppName("Streaming Example")
        .setMaster("spark://127.0.0.1:7077")
        .set("spark.cassandra.connection.host", "127.0.0.1")
        .set("spark.cleaner.ttl", "3600")
        .setJars(Array("/…/target/scala-2.10/streaming-example_2.10-0.1.jar"))

     val ssc = new StreamingContext(conf, Seconds(4))

     val stream = ssc.socketTextStream("127.0.0.1", 9999)

     stream.flatMap(record => record.split(" "))
           .map(word => (word,1))
           .reduceByKey(_ + _)
           .print()

     ssc.start()
     ssc.awaitTermination()
  }
}
```

results output

```
-------------------------------------------
Time: 1443548140000 ms
-------------------------------------------
(Drama,22)
(Comedy,12)
(Action,16)
(Romance,9)
(Dance,11)
(Family,18)
(Adventure,7)
(Horror,10)
(Thriller,19)
(Fantasy,11)
...

-------------------------------------------
Time: 1443548144000 ms
-------------------------------------------
(Drama,11)
(Comedy,14)
(Action,13)
(Romance,2)
(Dance,9)
(Family,12)
(Adventure,9)
(Science,2)
(Thriller,3)
(Fantasy,8)
...
```


amazon
web services
KINESIS


akka


TCP/IP


kafka


ØMQ

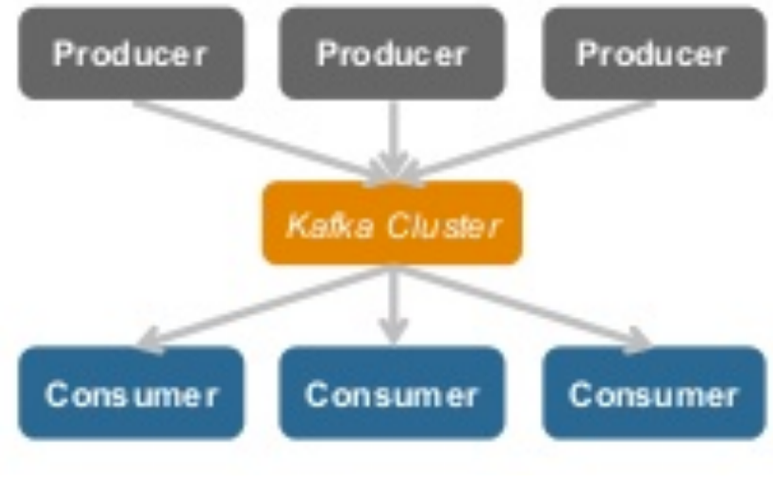

Spark
Streaming

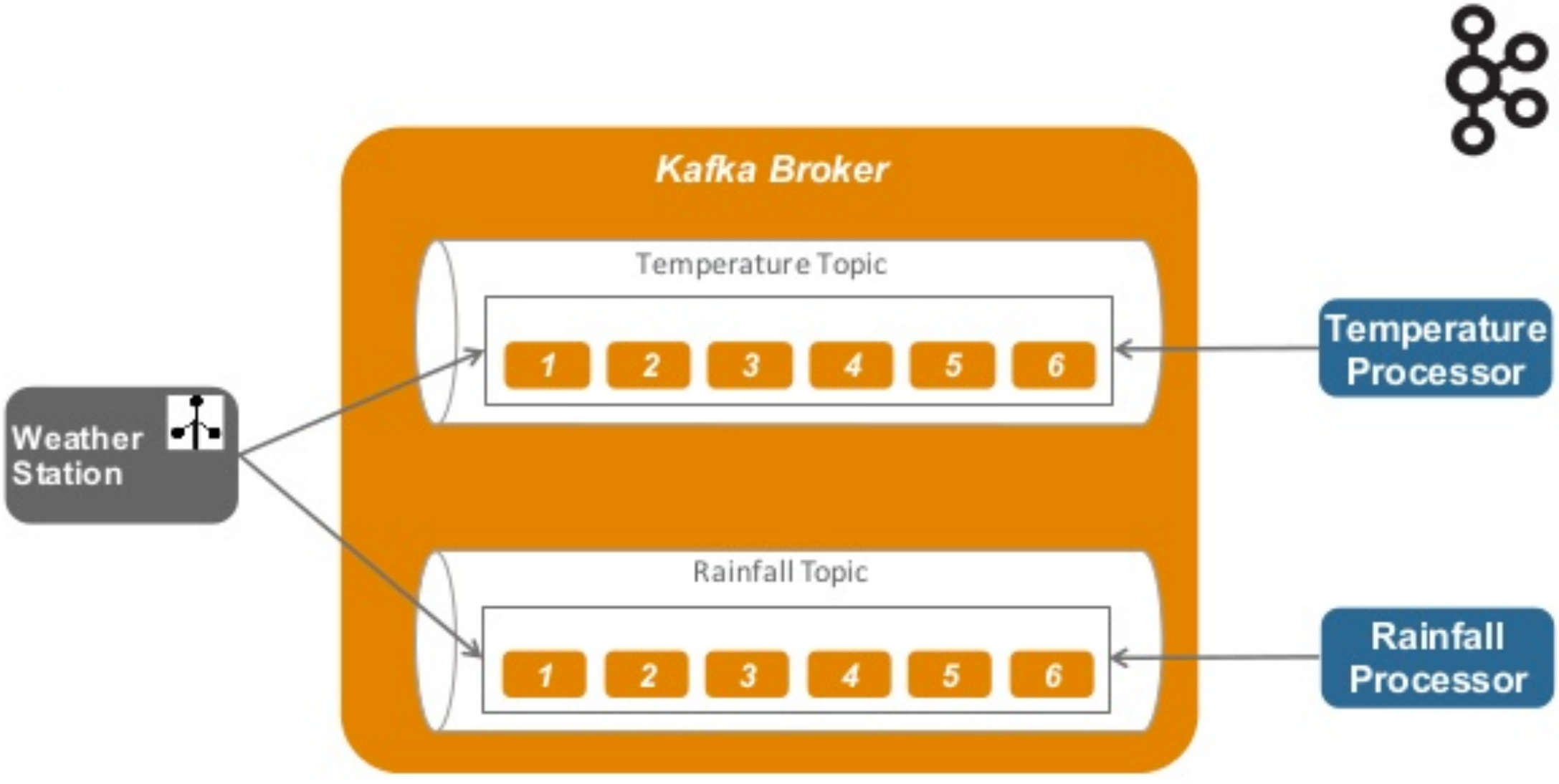

FLUME

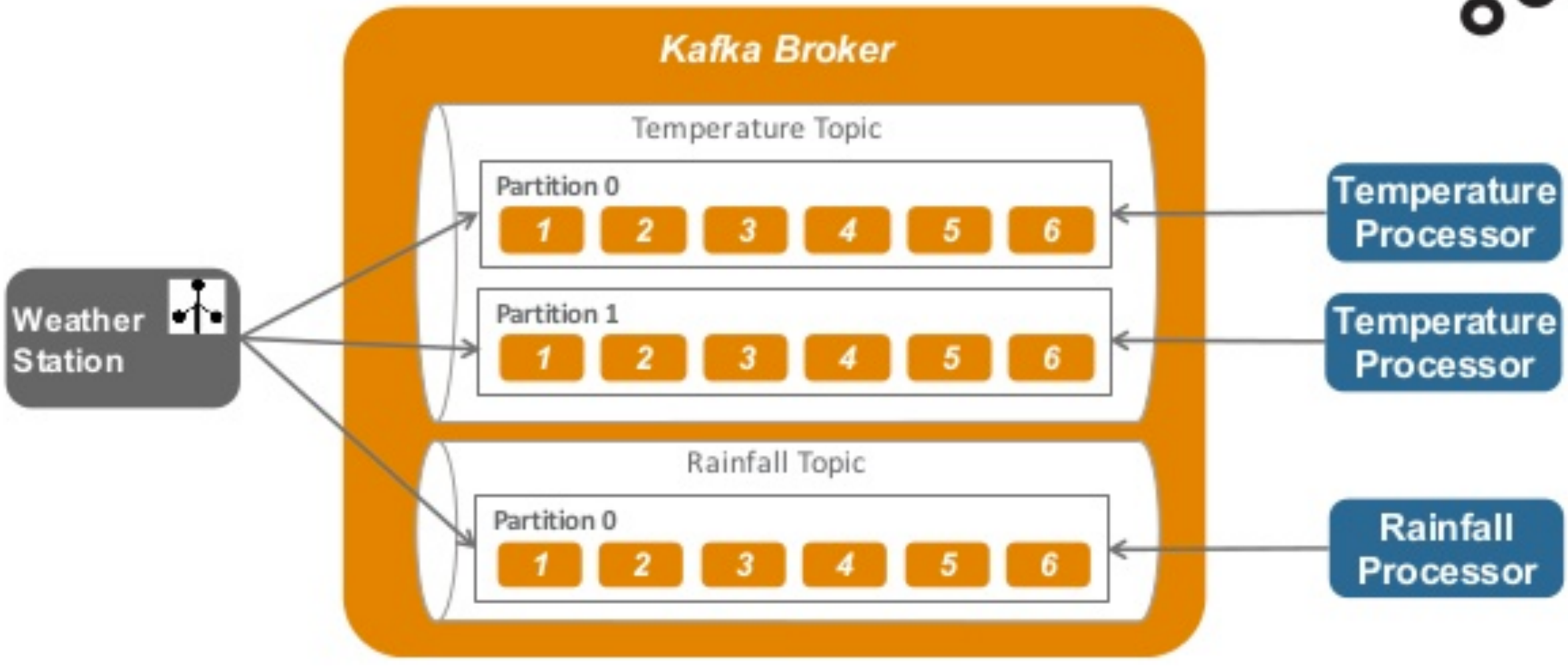

MQTT.ORG

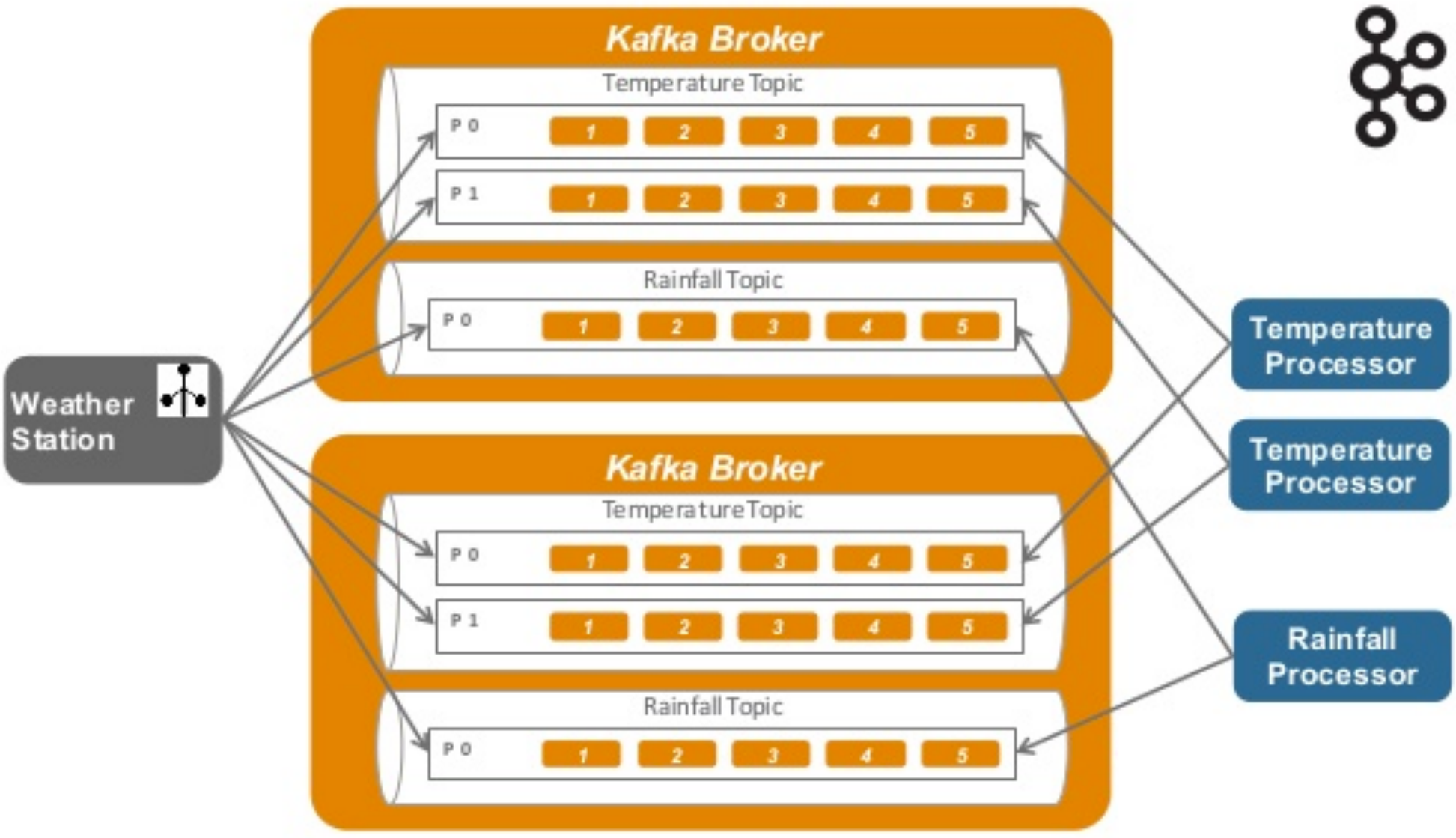
Apache kafka

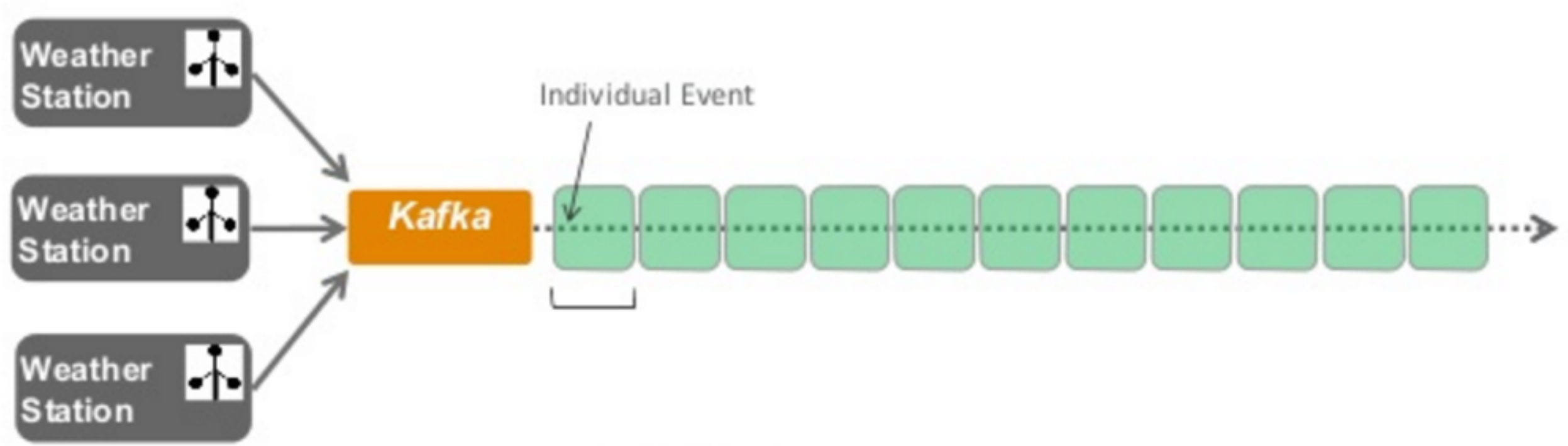
- Distributed publish-subscribe messaging system
- Processing real time activity strean
  metrics, collections, social media s
- Initially by LinkedIn, now Apache p
- Does not use JMS API and standards
- Kafka maintains feeds of messages in topics




Kafka Broker
Temperature Topic
1
2
3
4
5
6
Weather Station
Temperature Processor
Rainfall Topic
1
2
3
4
5
6
Rainfall Processor


Kafka Broker
Temperature Topic
Partition 0
1
2
3
4
5
6
Partition 1
1
2
3
4
5
6
Rainfall Topic
Partition 0
1
2
3
4
5
6
Weather Station
Temperature Processor
Temperature Processor
Rainfall Processor


Kafka Broker
Temperature Topic
P 0
1
2
3
4
5
P 1
1
2
3
4
5
Rainfall Topic
P 0
1
2
3
4
5
Weather Station
Kafka Broker
Temperature Topic
P 0
1
2
3
4
5
P 1
1
2
3
4
5
Rainfall Topic
P 0
1
2
3
4
5
Temperature Processor
Temperature Processor
Rainfall Processor


Weather Station
Weather Station
Weather Station
Kafka
Individual Event


amazon
web services
KINESIS


akka


TCP/IP

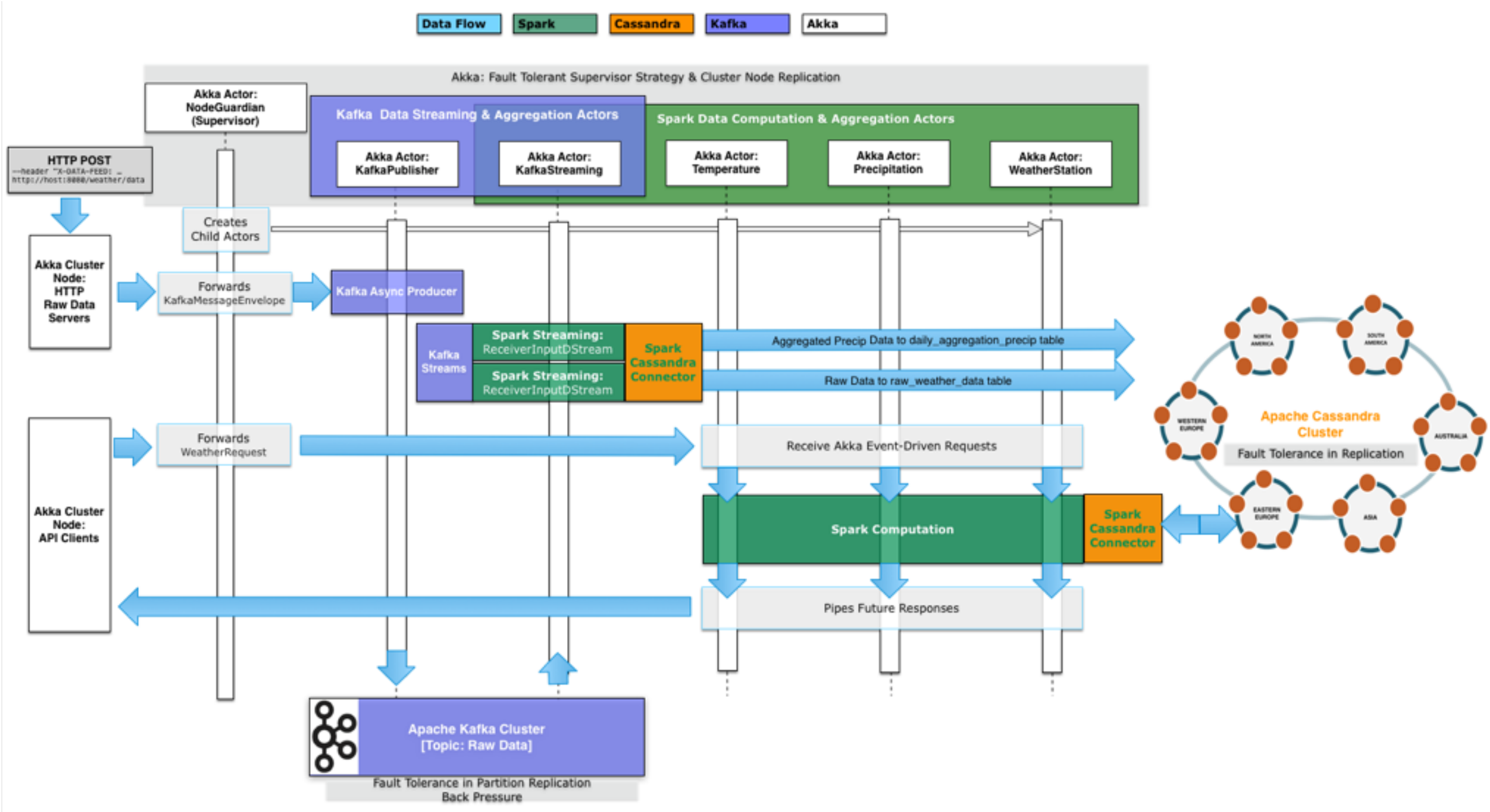

kafka


ØMQ


Spark
Streaming


FLUME


MQTT.ORG



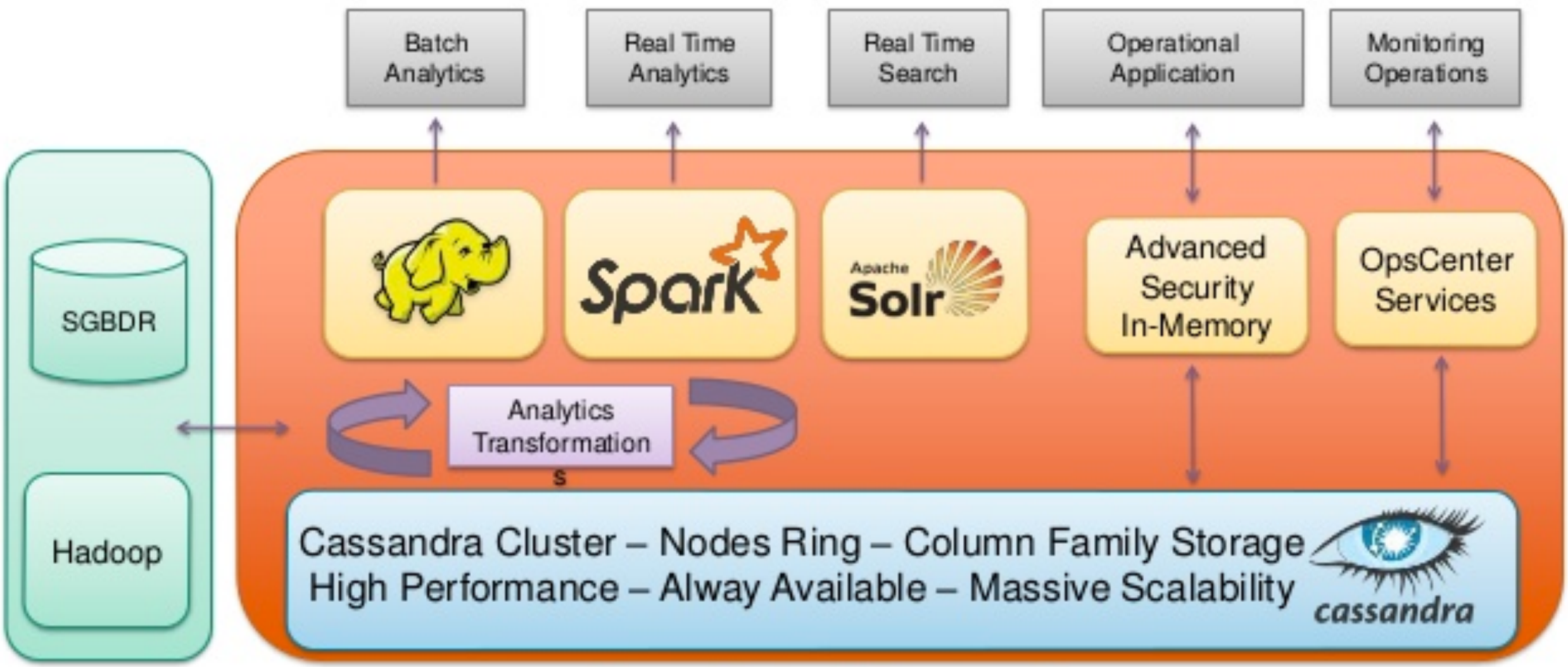
https://raw.githubusercontent.com/killrweather/killrweather/master/diagrams/Asynchronous_Fault_Tolerant_Data_Pipeline.png

Combining spark and cassandra

- Spark
  - Great for analyzing large amount of data
- Cassandra
  - Great for storing large amount of data


DATASTAX

DATASTAX

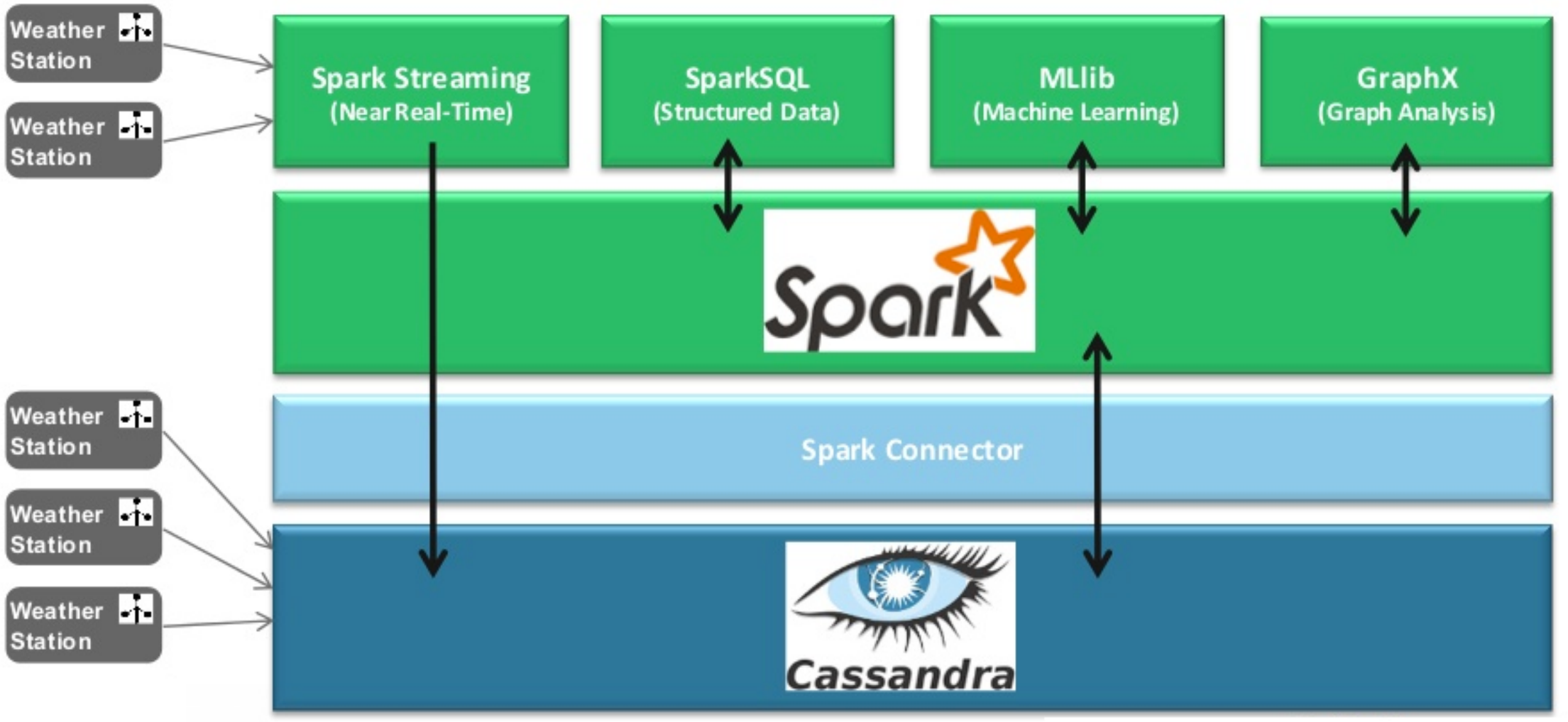

Batch Analytics
Real Time Analytics
Real Time Search
Operational Application
Monitoring Operations
SGBDR
Hadoop
Spark
Apache Solr
Advanced Security In-Memory
OpsCenter Services
Analytics Transformation
s
Cassandra Cluster – Nodes Ring – Column Family Storage
High Performance – Alway Available – Massive Scalability
cassandra

Spark Streaming
(Near Real-Time)

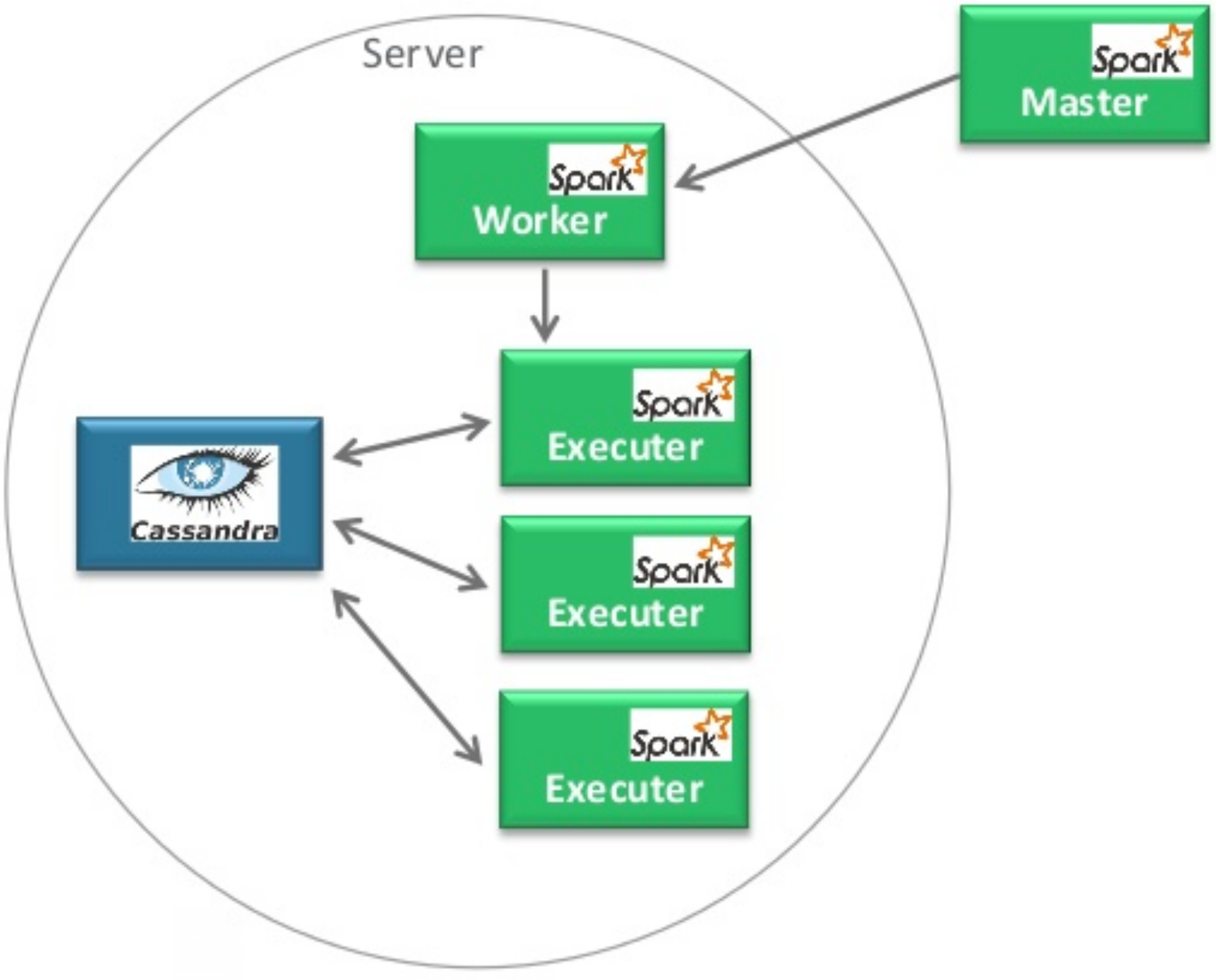
SparkSQL
(Structured Data)

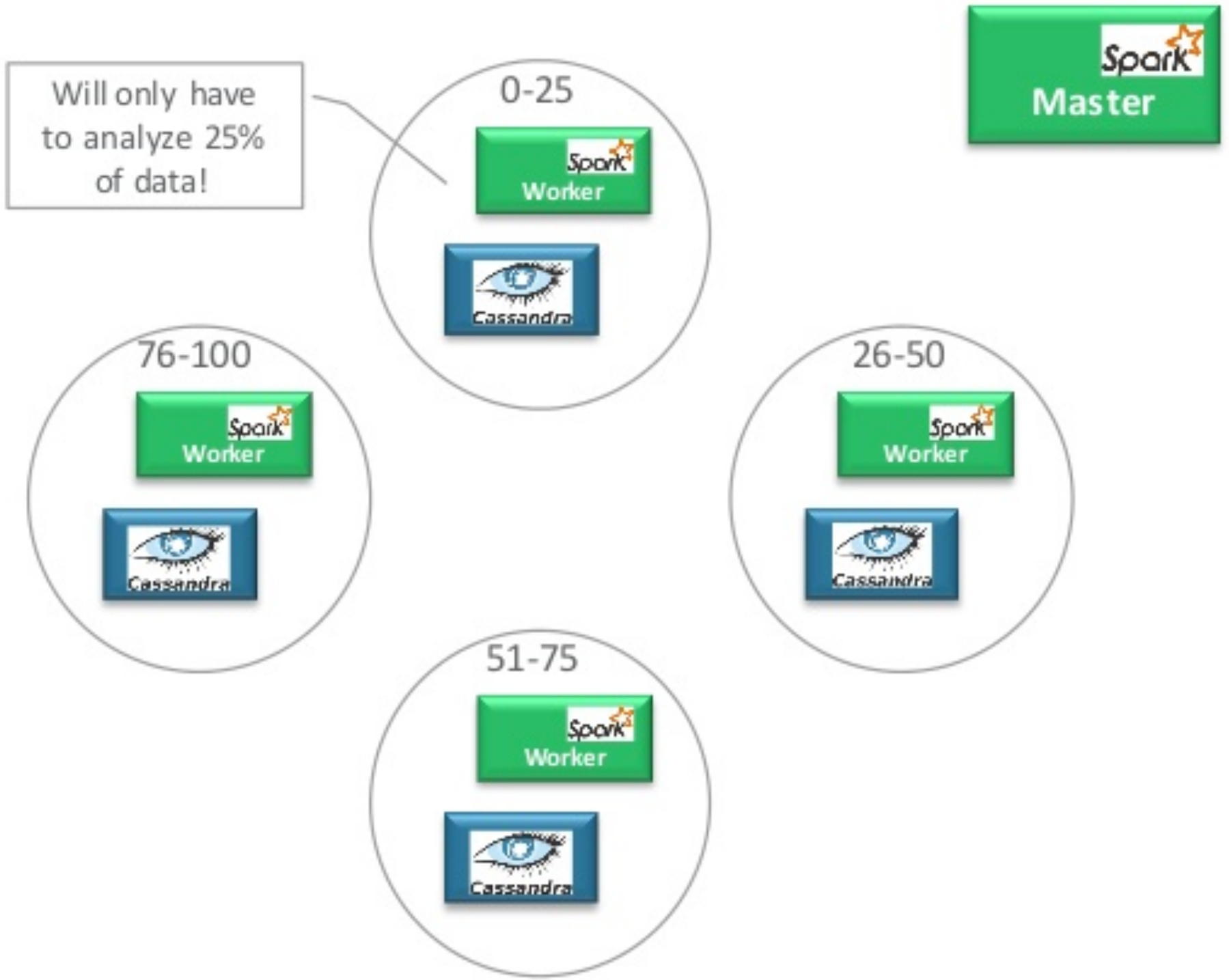
MLlib
(Machine Learning)

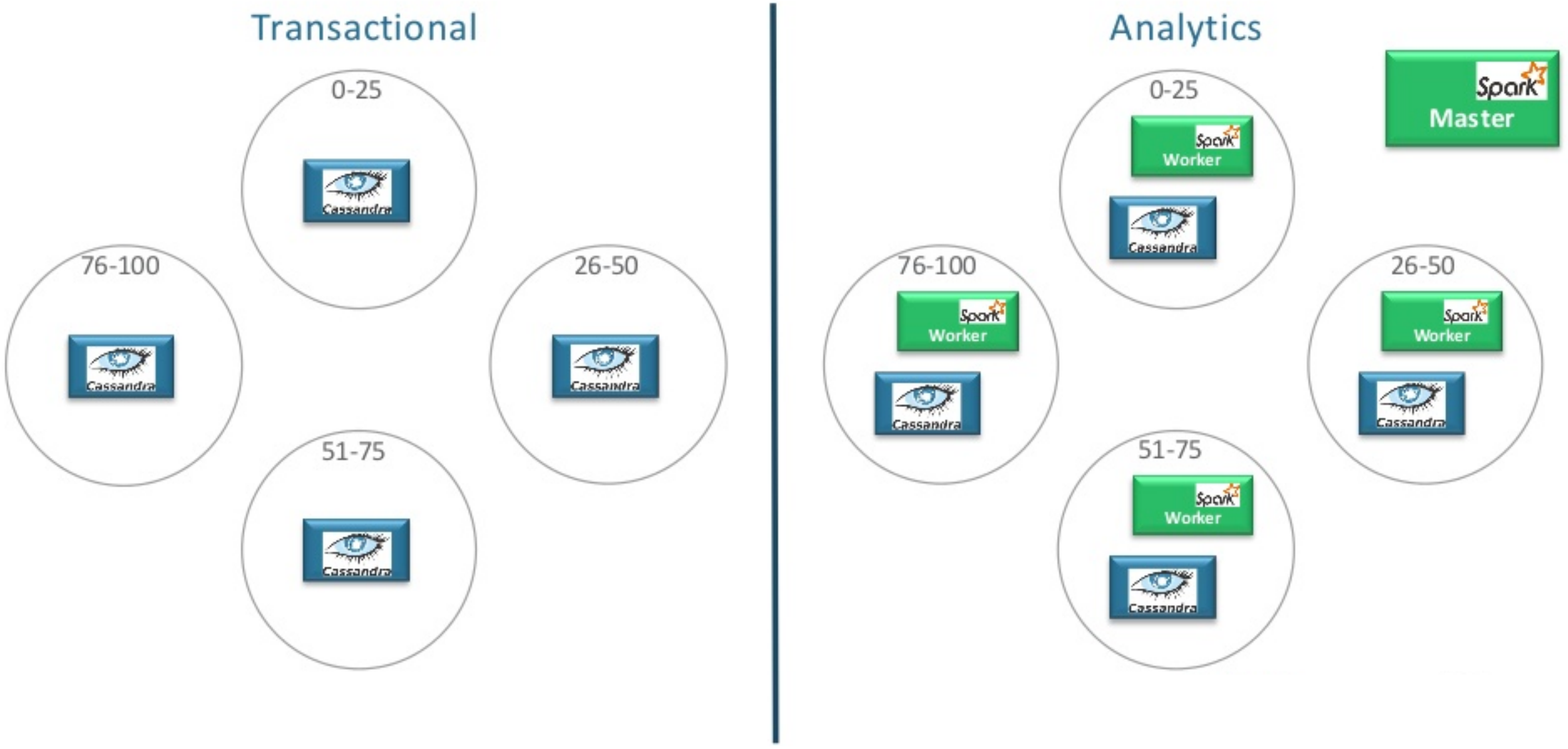
GraphX
(Graph Analysis)


Weather Station
Weather Station
Spark Streaming
(Near Real-Time)
SparkSQL
(Structured Data)
MLlib
(Machine Learning)
GraphX
(Graph Analysis)
Spark
Weather Station
Weather Station
Weather Station
Spark Connector
Cassandra

Spark and Cassandra architecture

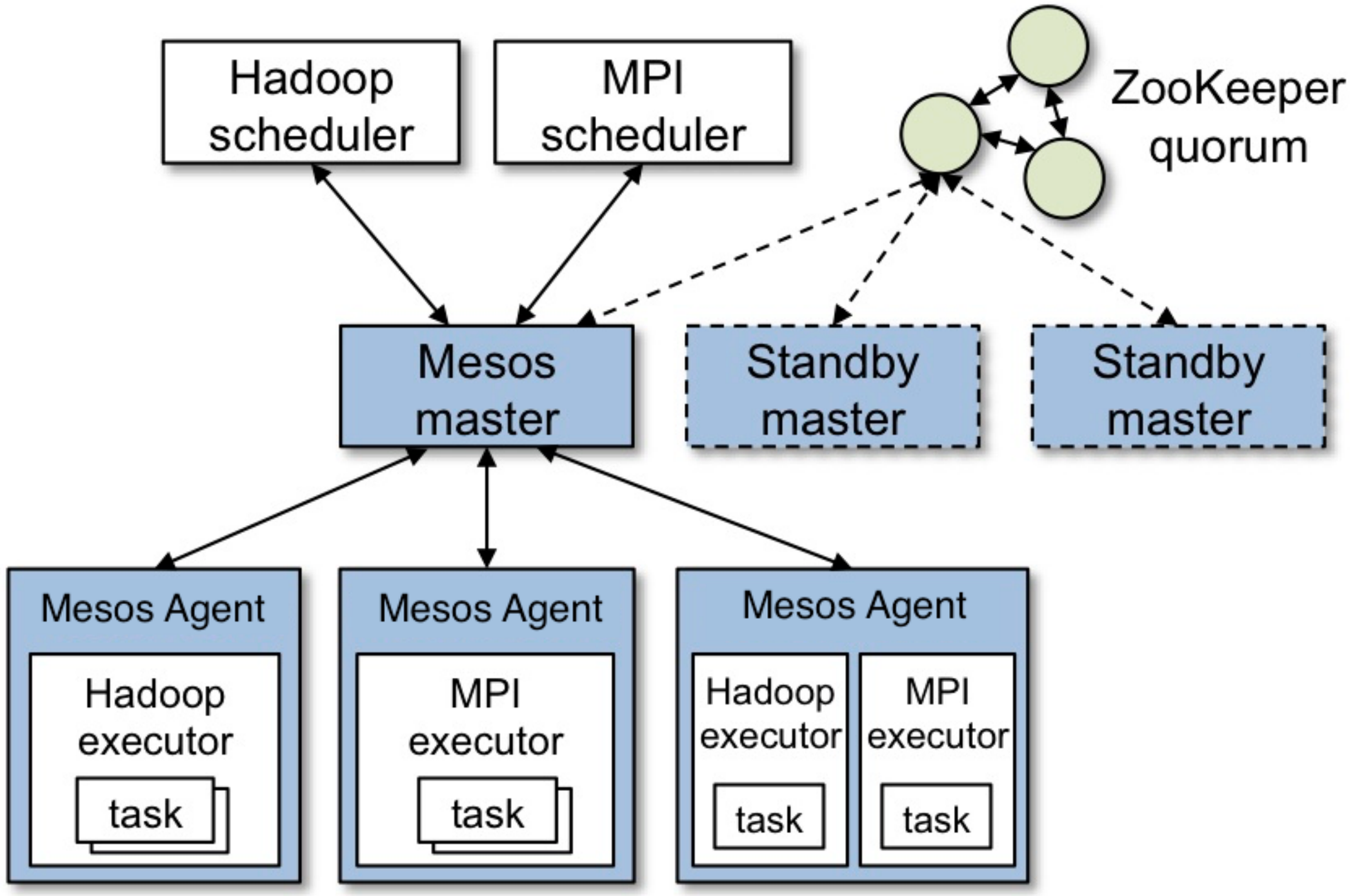



Will only have to analyze 25% of data!
0-25
Spark
Worker
Cassandra
Spark
Master
76-100
Spark
Worker
Cassandra
26-50
Spark
Worker
Cassandra
51-75
Spark
Worker
Cassandra


Transactional
0-25
Cassandra
76-100
Cassandra
26-50
Cassandra
51-75
Cassandra
Analytics
Spark
Master
0-25
Spark
Worker
Cassandra
76-100
Spark
Worker
Cassandra
26-50
Spark
Worker
Cassandra
51-75
Spark
Worker
Cassandra

cassandra and spark

	Cassandra	Cassandra & Spark
Joins and Unions	No	Yes
Transformations	Limited	Yes
Outside Data Integration	No	Yes
Aggregations	Limited	Yes


Hadoop scheduler
MPI scheduler
ZooKeeper quorum
Mesos master
Standby master
Standby master
Mesos Agent
Hadoop executor
task
Mesos Agent
MPI executor
task
Mesos Agent
Hadoop executor
task
MPI executor
task

AMER-PROD
Jane Doe

- Dashboard
- Services
- Jobs
- Universe

RESOURCES

- Nodes
- Networking
- Security

SYSTEM

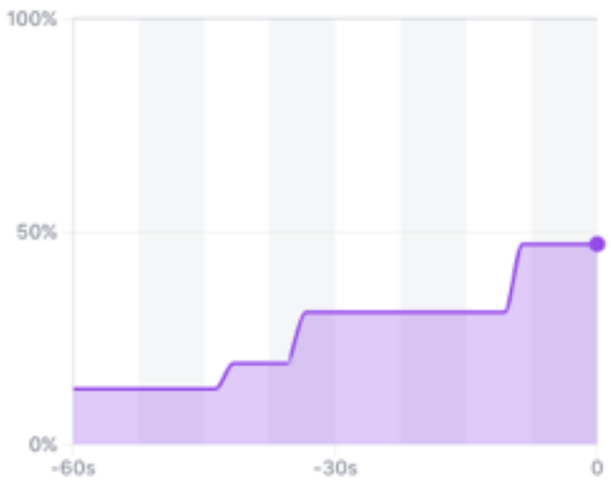
- Cluster
- Components
- Settings
- Organization

Dashboard

MESOSPHERE DC/OS

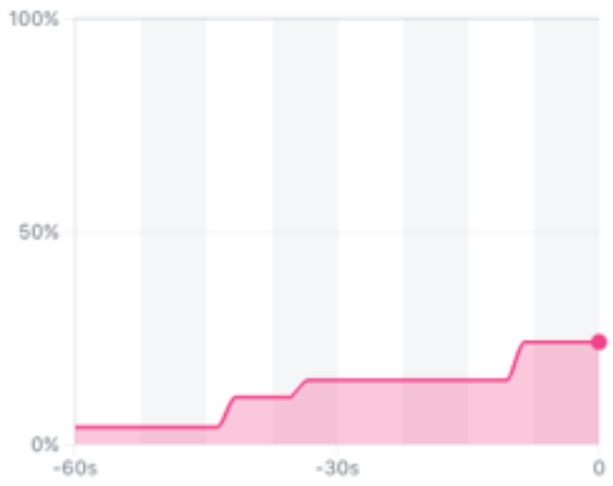
CPU Allocation

50%
40 of 80 Shares



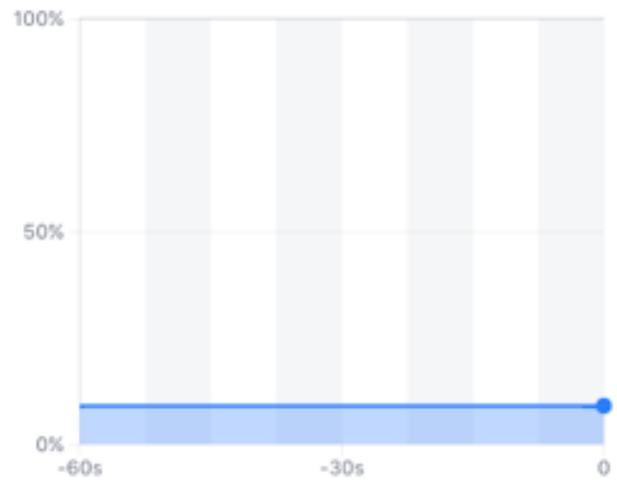
Memory Allocation

24%
70 GiB of 270 GiB



Disk Allocation

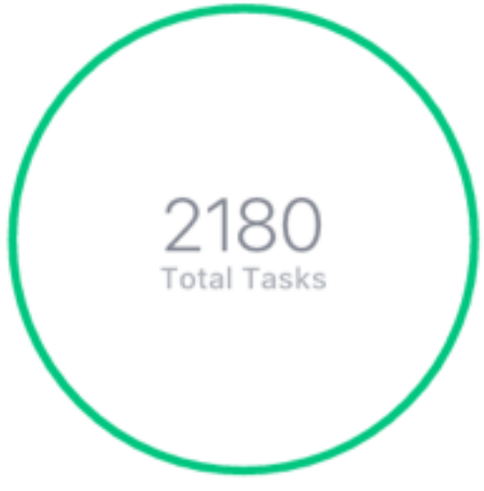
9%
200 GiB of 2080 GiB



Services Health

arangodb3	Healthy
cassandra	Healthy
kafka	Healthy
nginx	Healthy
confluent-kafka	Healthy
jenkins	Healthy
linkerd	Healthy

Tasks



Component Health

Admin Router Agent	Healthy
Admin Router Master	Healthy
Admin Router Reloader	Healthy
Admin Router Reloader	Healthy
Admin Router Reloader Timer	Healthy

EMEA-PROD
Jane Doe

- Dashboard
- Services
- Jobs
- Universe
  - Packages
  - Installed

RESOURCES

- Nodes
- Networking
- Security

SYSTEM

- Cluster
- Components
- Settings
- Organization

Packages

MESOSPHERE DC/OS

Apache Spark
1.0.7-2.1.0
INSTALL PACKAGE

Datastax Ent Max
1.0.16-5.0.2
INSTALL PACKAGE

Confluent Kafka
0.9.6-3.1.2
INSTALL PACKAGE

Elastic Stack
1.0.5-5.2.1
INSTALL PACKAGE

Apache HDFS
1.0.0-2.6.0
INSTALL PACKAGE

Couchbase
4.6.0
INSTALL PACKAGE

Apache Flink
1.2.0-1.0
INSTALL PACKAGE

Alluxio
1.4.0
INSTALL PACKAGE

Lightbend Reactive
2.1.0

Redis
3.0.7-0.0.1

Jenkins
3.0.2-2.32.2

Basho Riak
2.0.0

Services › redis-service Waiting (0 of 1)

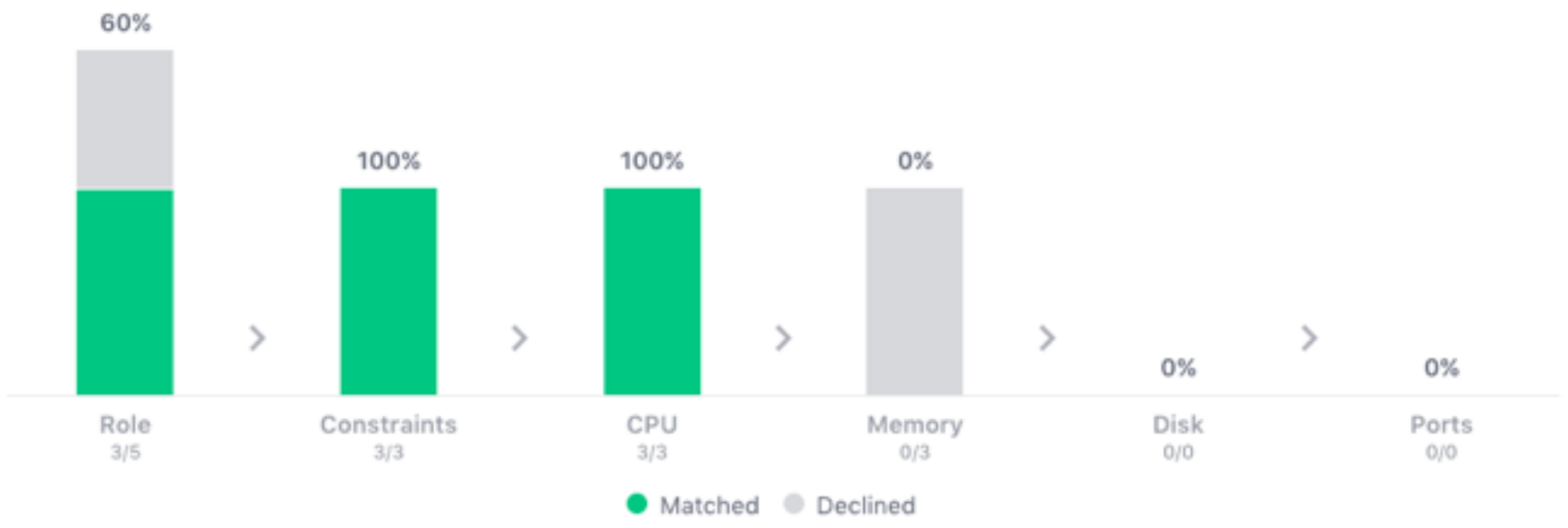
MESOSPHERE DC/OS

Instances Configuration Debug

Recent Resource Offers (5)

When you attempt to deploy a service, DC/OS waits for offers to match the resources your service requires. If the offer does not satisfy the requirement, it is declined and DC/OS retries. Learn more.

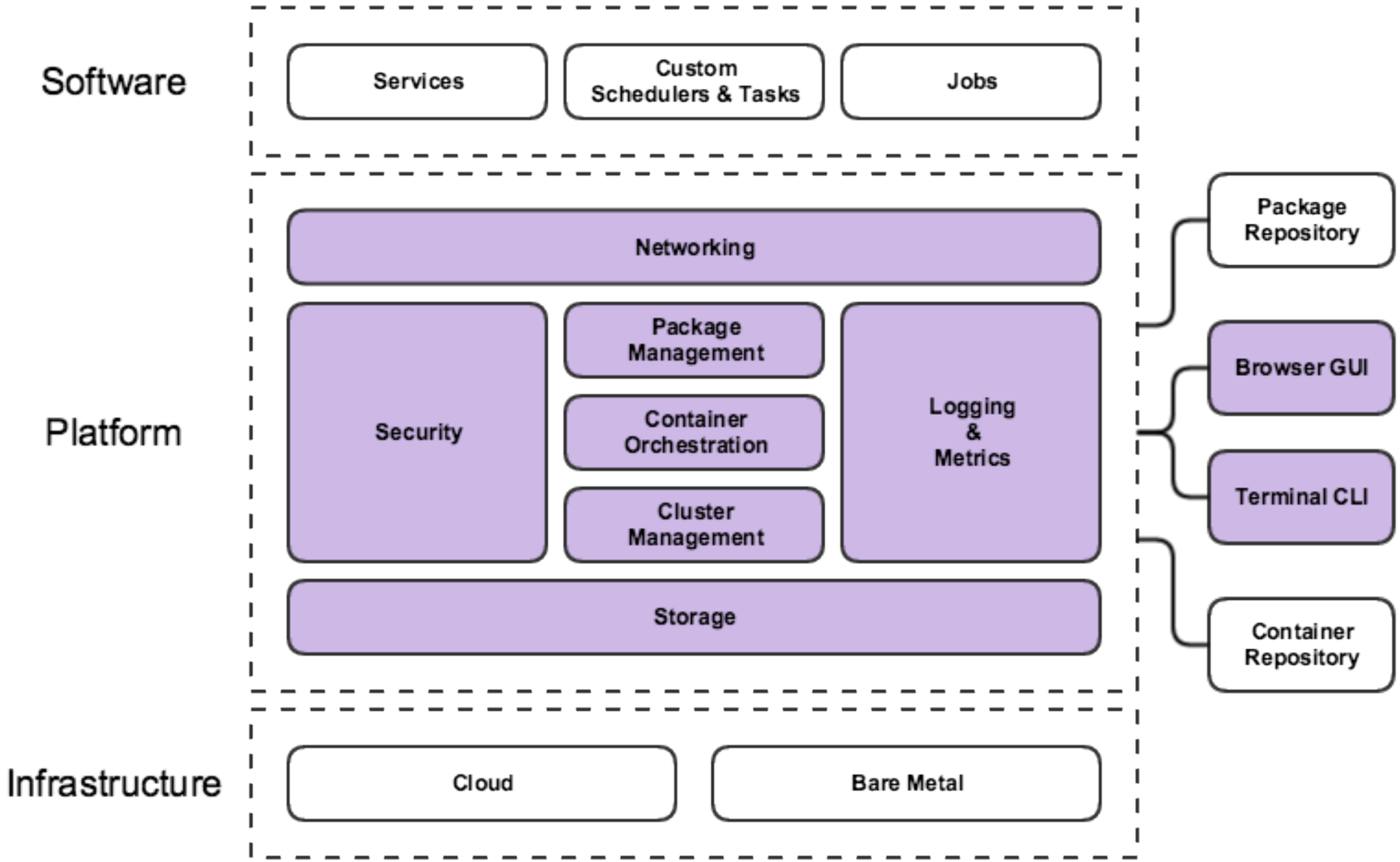
Summary



Details

HOST ▲	ROLE	CONSTRAINT	CPU	MEM	DISK	PORT	RECEIVED
10.0.1.155	✓	✓	✓	✗	✓	✓	2 minutes ago
10.0.3.205	✓	✓	✓	✗	✓	✓	2 minutes ago
10.0.3.241	✓	✓	✓	✗	✓	✓	2 minutes ago
10.0.4.213	✗	✓	✗	✗	✓	✓	2 minutes ago

DC/OS Architecture Layers



SMACK AGENDA

- SMACK stack overview
- Bigdata characteristics
- Cassandra NoSQL database
- Spark Streaming
- Kafka and Akka event sourcing
- Mesos Cluster management
- Mesos Scheduling and execution

October 5-6, 2017
Indian Institute of Science, Bangalore
www.modsummit.com

Register early and get the best discounts

April 23-28, 2018
Indian Institute of Science, Bangalore
www.developersummit.com