Persisting with Cassandra

Ahmad Alkilani
DATA ARCHITECT

@akizl

Apache Cassandra

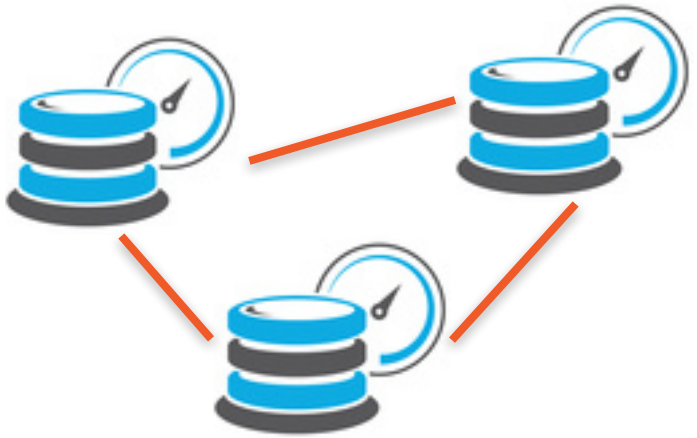
Use Cases

Data Modeling with Cassandra

Spark Cassandra Connector

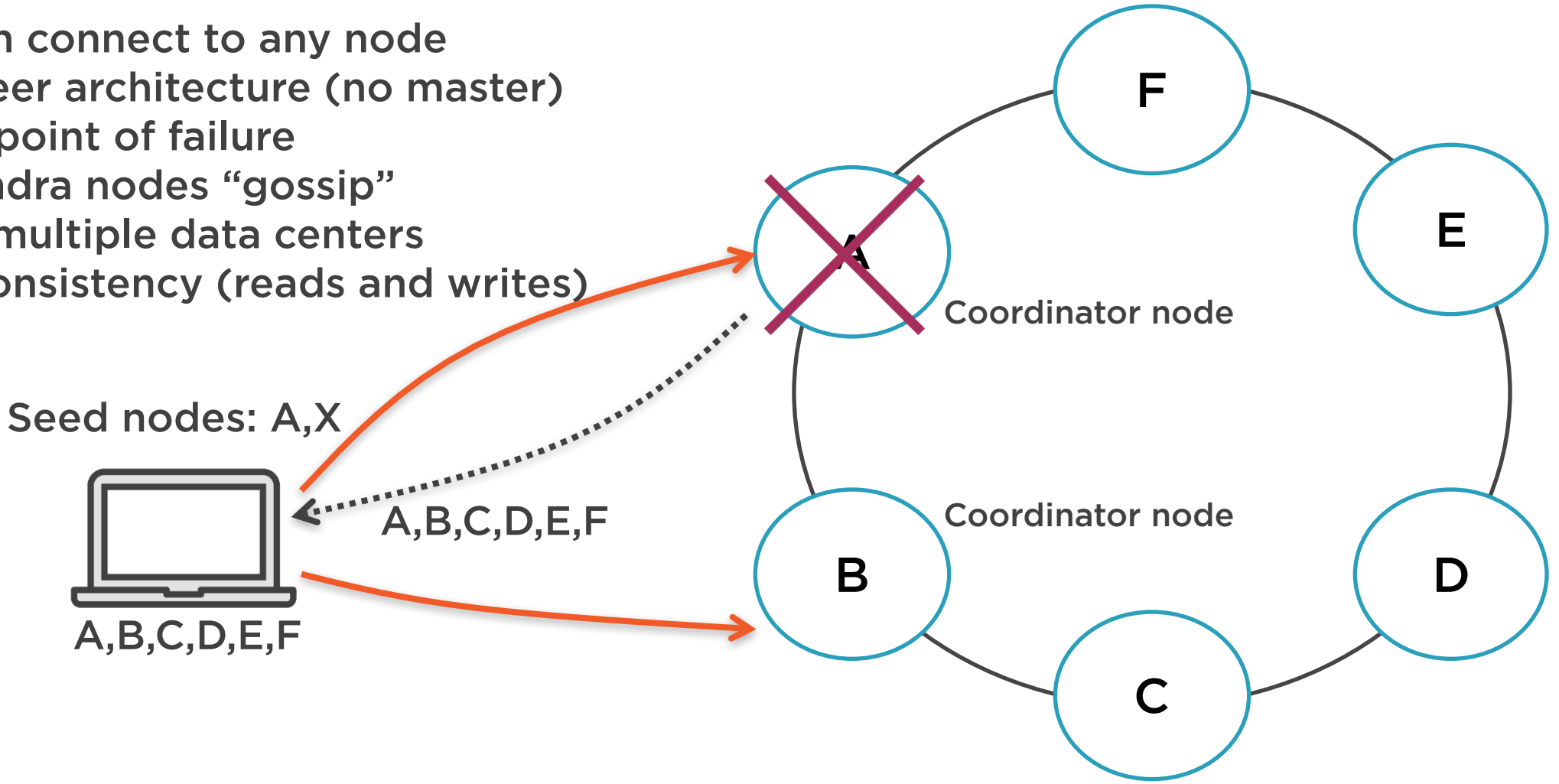
Serving Layer with Batch and Realtime Views

Distributed database management system designed for large amounts of data, providing high availability with no single point of failure.



Apache Cassandra

- Clients can connect to any node
- Peer-to-peer architecture (no master)
- No single point of failure
  - Cassandra nodes “gossip”
- Can span multiple data centers
- Tunable consistency (reads and writes)



Relational Database vs. Cassandra

Relational	Cassandra
Database	Keyspace
Table	Column Family
Primary Key	Row Key
Column Name	Column Name/Key
Column Value	Column Value

Spark Cassandra Connector

Seamless integration with Spark

Exposes Cassandra tables as RDDs

Customizable Object Mapping

Map Cassandra rows to objects of user defined classes

CQL, Spark SQL and DataFrames

Allows for advanced optimizations like "joinWithCassandraTable"

Spark Cassandra Connector

Using Spark Data Sources API

```
val df = sqlContext
  .read
  .format("org.apache.spark.sql.cassandra")
  .options(Map( "table" -> “commentsByUser", "keyspace" -> "test" ))
  .load()
```

Spark Cassandra Connector

Using Spark SQL

```
sqlContext.sql(
  """CREATE TEMPORARY TABLE commentsByUser
    |USING org.apache.spark.sql.cassandra
    |OPTIONS (
    |  table " commentsByUser ",
    |  keyspace "test",
    |  pushdown "true"
    |)""".stripMargin)

val df = sqlContext.sql(s"SELECT * FROM commentsByUser WHERE username = '$username'")
```

Cassandra's Data Model

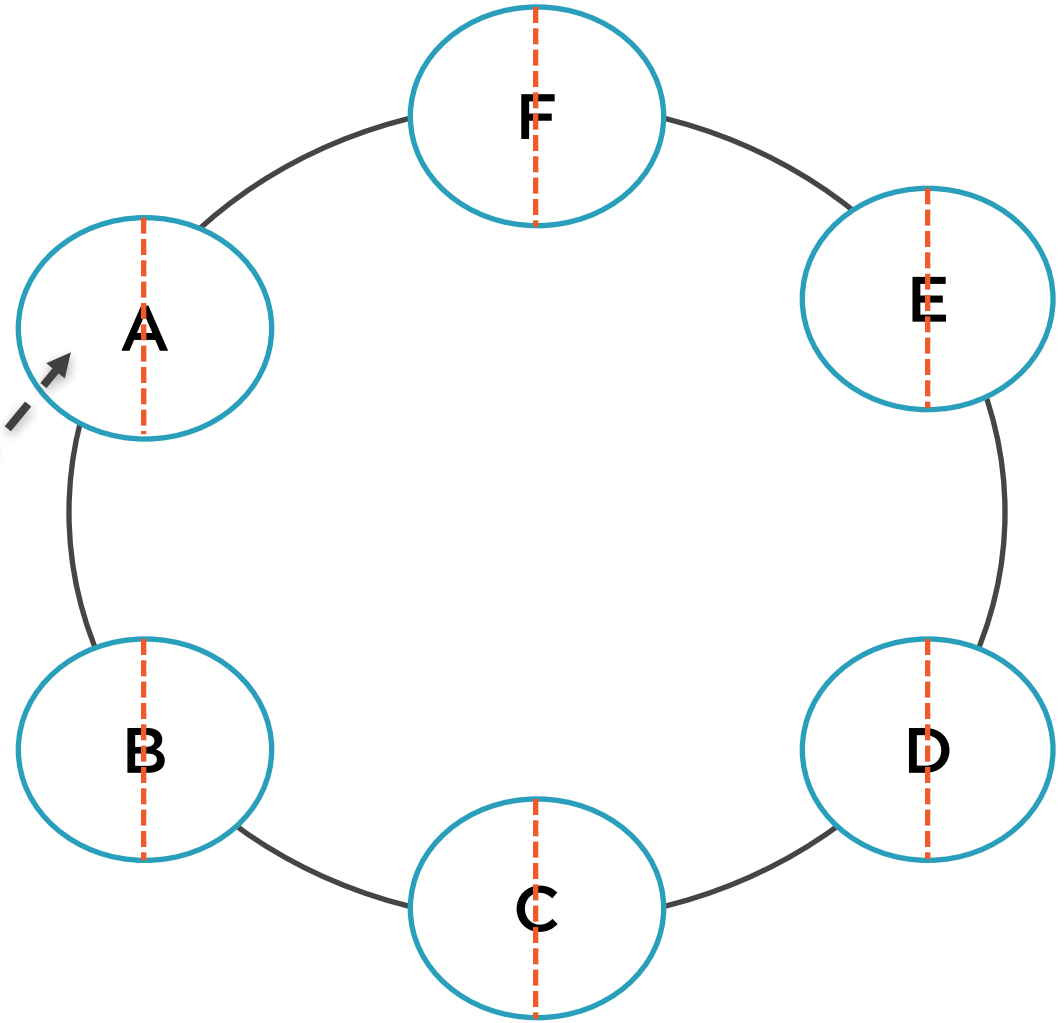
- Cassandra is a row-store
- Cassandra Query Language (CQL)
- Distributed Map
  - Access by Key
  - Partitioner (Murmur3 – consistent hash)

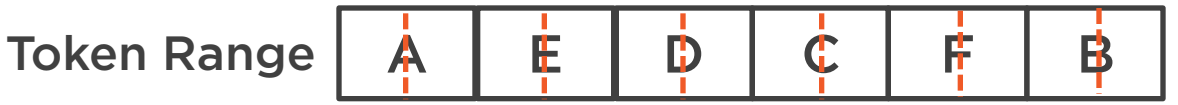
```
CREATE TABLE commentsByUser (
    email text ,
    comment text ,
    username text PRIMARY KEY
);

INSERT INTO commentsByUser
(email, comment, username)
VALUES ('john@doe.com', 'first comment', 'John')

SELECT username
FROM commentsByUser
WHERE username = 'John'
```

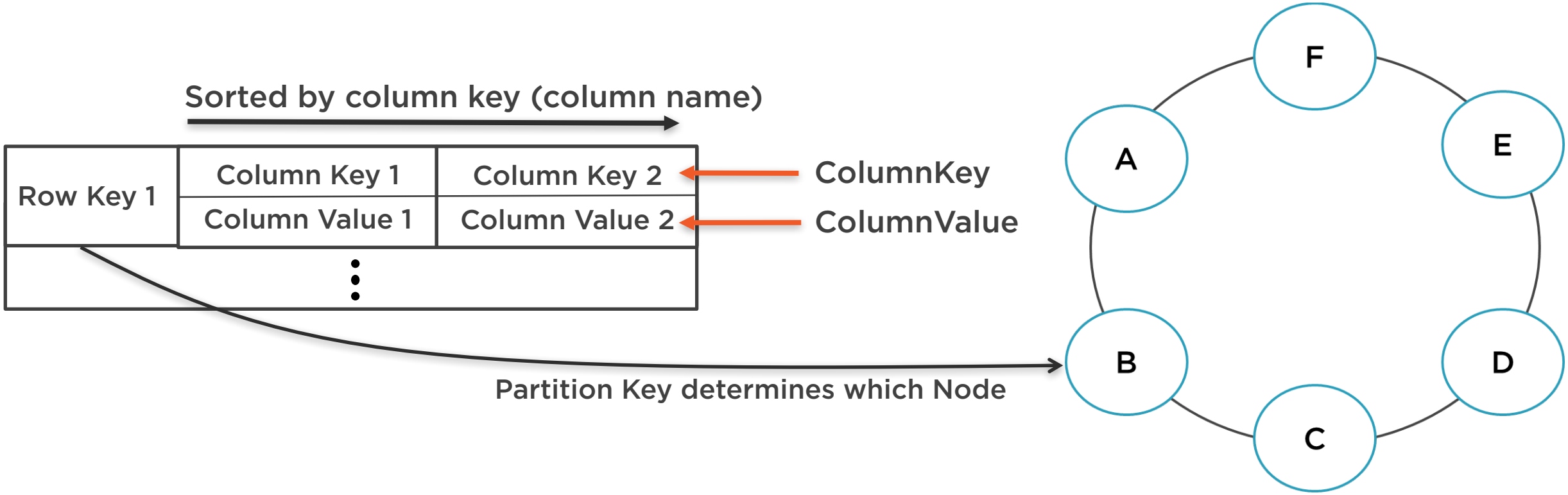
Murmur3Hash('John')





Cassandra's Data Model

Map[RowKey, SortedMap[ColumnKey, ColumnValue]]



Cassandra's Data Model

Map[RowKey, SortedMap[ColumnKey, ColumnValue]]

```
CREATE TABLE commentsByUser (
    email text ,
    comment text ,
    username text PRIMARY KEY
);
```

Also defines Partition Key

```
PRIMARY KEY (email, username)
```

Partition Key

commentsByUser

	comment	email
John	first comment	john@doe.com

ColumnKey

ColumnValue

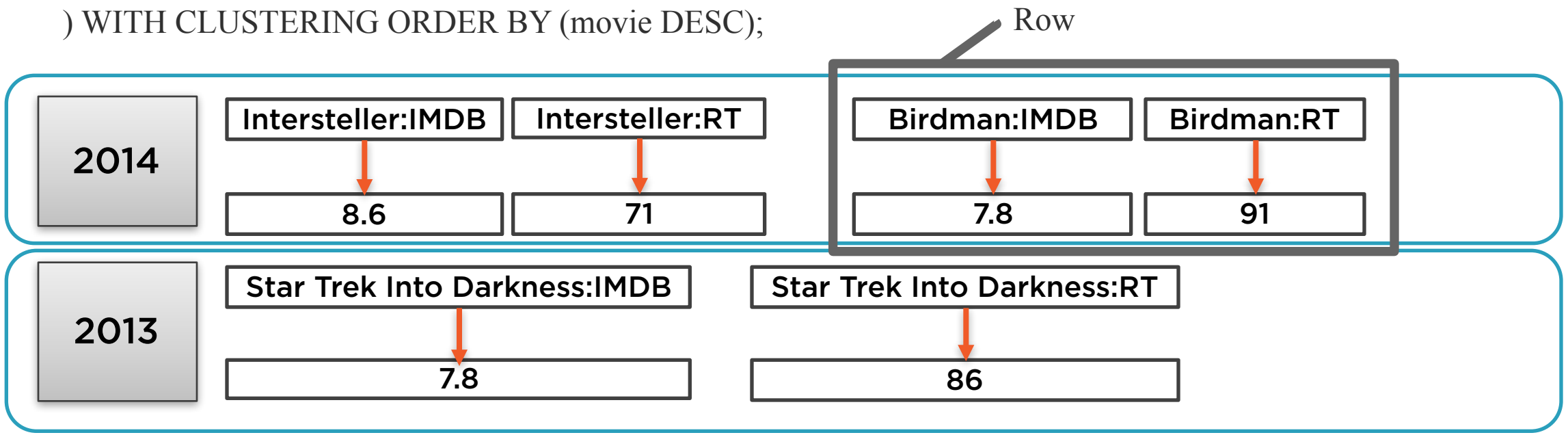
Partition Key (Row Key)

Composite Primary Key

Map[RowKey, SortedMap[ColumnKey, ColumnValue]]

```
CREATE TABLE movieRatings_by_year_movie (
    year int ,
    movie text ,
    IMDB float,
    RT int,
    PRIMARY KEY (year, movie)
) WITH CLUSTERING ORDER BY (movie DESC);
```

year	movie	IMDB	RT
2013	Star Trek Into Darkness	7.8	86
2014	Interstellar	8.6	71
2014	Birdman	7.8	91

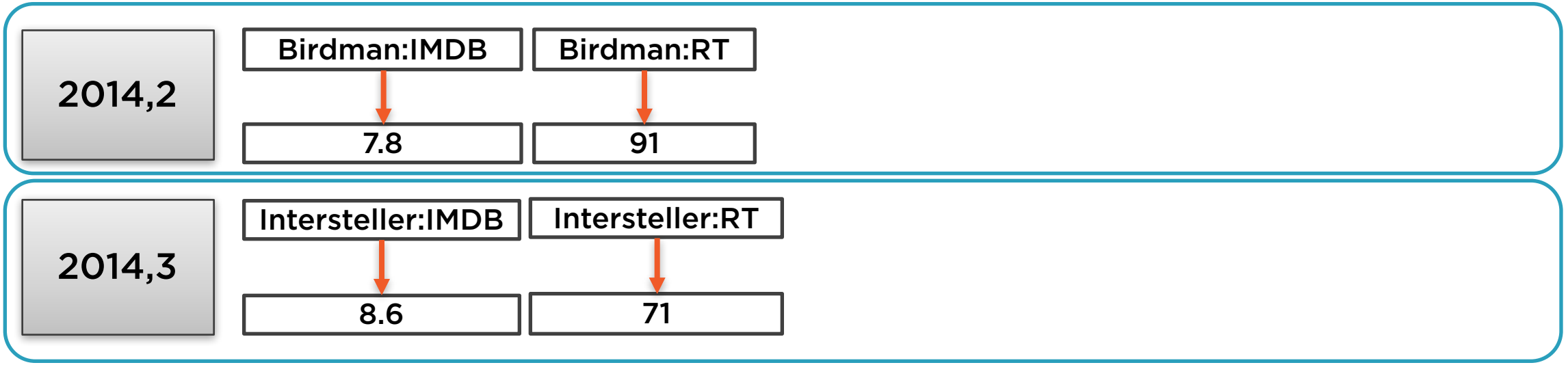


Composite Partition Key

Map[RowKey, SortedMap[ColumnKey, ColumnValue]]

```
CREATE TABLE movieRatings_by_year_movie (
    year int ,
    movie text ,
    IMDB float,
    RT int,
    hash_prefix int,
    PRIMARY KEY ((year, hash_prefix), movie)
) WITH CLUSTERING ORDER BY (movie DESC);
```

year	movie	IMDB	RT
2014	Interstellar	8.6	71
2014	Birdman	7.8	91



Time Series

Map[RowKey, SortedMap[ColumnKey, ColumnValue]]

Time series data stored in Cassandra as a wide-row

[timestamp]

Sorted by column key (column name)

eventsByCompany

Company X	timestamp1	timestamp2	timestamp3	timestamp4
	event1	event2	event3	event4

• • •

Using composite columns

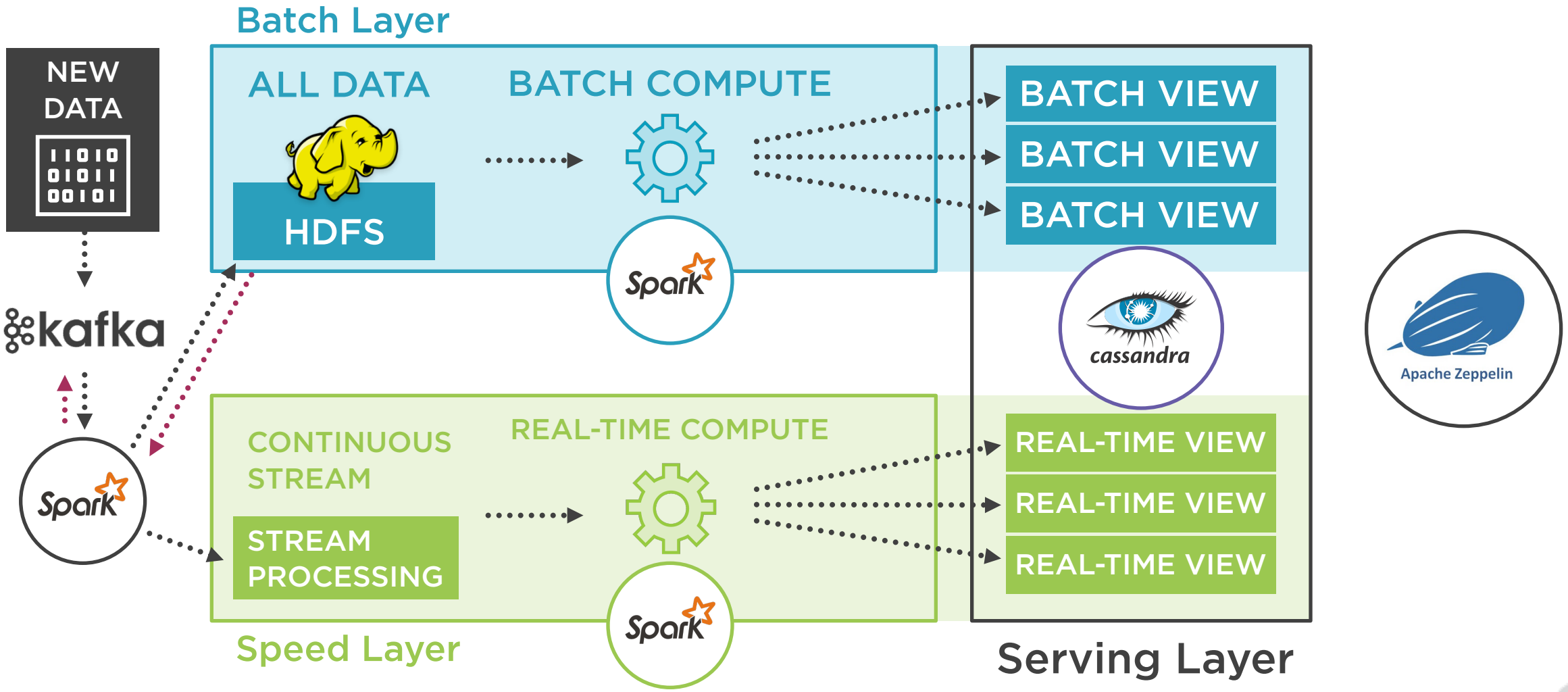
[eventType, timestamp]

eventsByCompanyEventType

Company X	Call | timestamp1	Call | timestamp2	Offer | timestamp1
	{data}	{data}	{data}

• • •

Summary


Batch Layer
NEW DATA
kafka
ALL DATA
BATCH COMPUTE
HDFS
Spark
BATCH VIEW
BATCH VIEW
BATCH VIEW
cassandra
Apache Zeppelin
Spark
CONTINUOUS STREAM
REAL-TIME COMPUTE
STREAM PROCESSING
Spark
REAL-TIME VIEW
REAL-TIME VIEW
REAL-TIME VIEW
Speed Layer
Serving Layer