

Data at the Speed of your Users

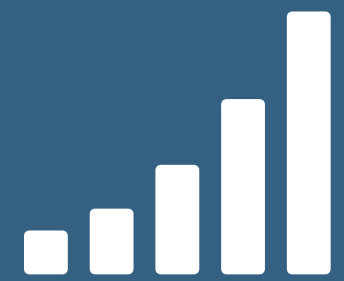
Apache Cassandra and Spark for simple, distributed, near real-time stream processing.

Rustam Aliyev

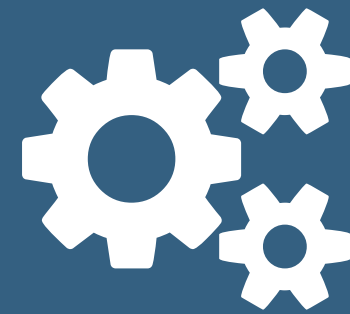
Solution Architect at **DATASTAX** 

 @rstml

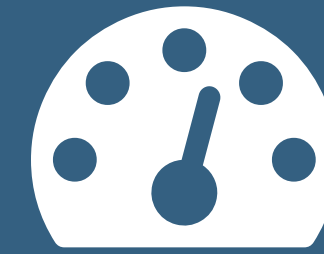
Big Data?



Volume



Variety



Velocity

Velocity = Near Real Time

Near Real Time?

0.5 sec ≤ Near Real Time ≤ 60 sec

Use Cases



Web Analytics

Dynamic Pricing

Recommendation

Fraud Detection

Architecture

Architecture Goals

Low Latency

High Availability

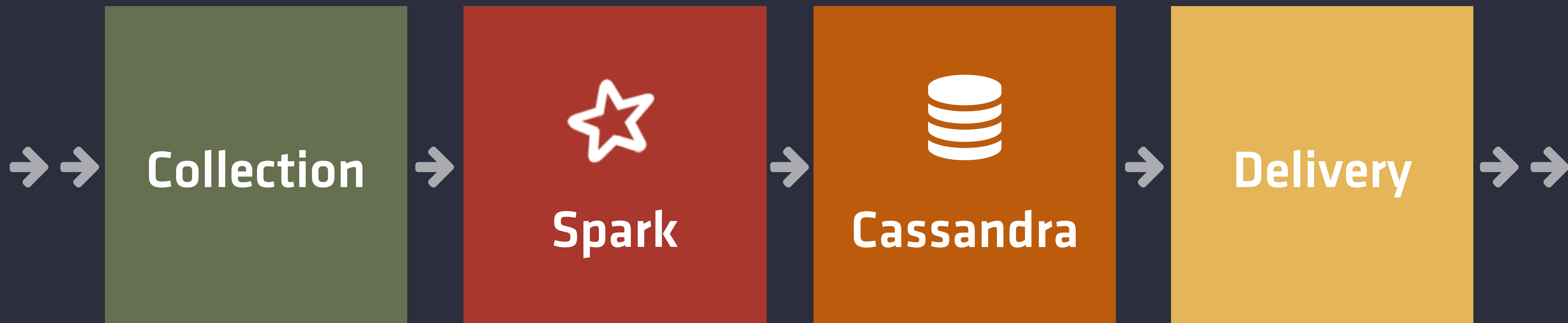
Horizontal Scalability

Simplicity

Stream Processing



Stream Processing



A low-angle, upward-looking photograph of several massive, ancient Egyptian columns. The columns are covered in hieroglyphs and have papyrus-bundle capitals. They are set against a clear blue sky. The image is used as a background for a title card, with a semi-transparent dark blue overlay.

Cassandra

Distributed Database

Data Model

Partition

Partition Key

Cell 1

Cell 2

Cell 3

...

Partition

Nexus 5

os:
Android

storage:
32GB

version:
4.4

weight:
130g

sort order on disk

Table

Nexus 5

OS:
Android

storage:
32GB

version:
4.4

weight:
130g

iPhone 6

OS:
iOS

storage:
64GB

version:
8.0

weight:
129g

OS:

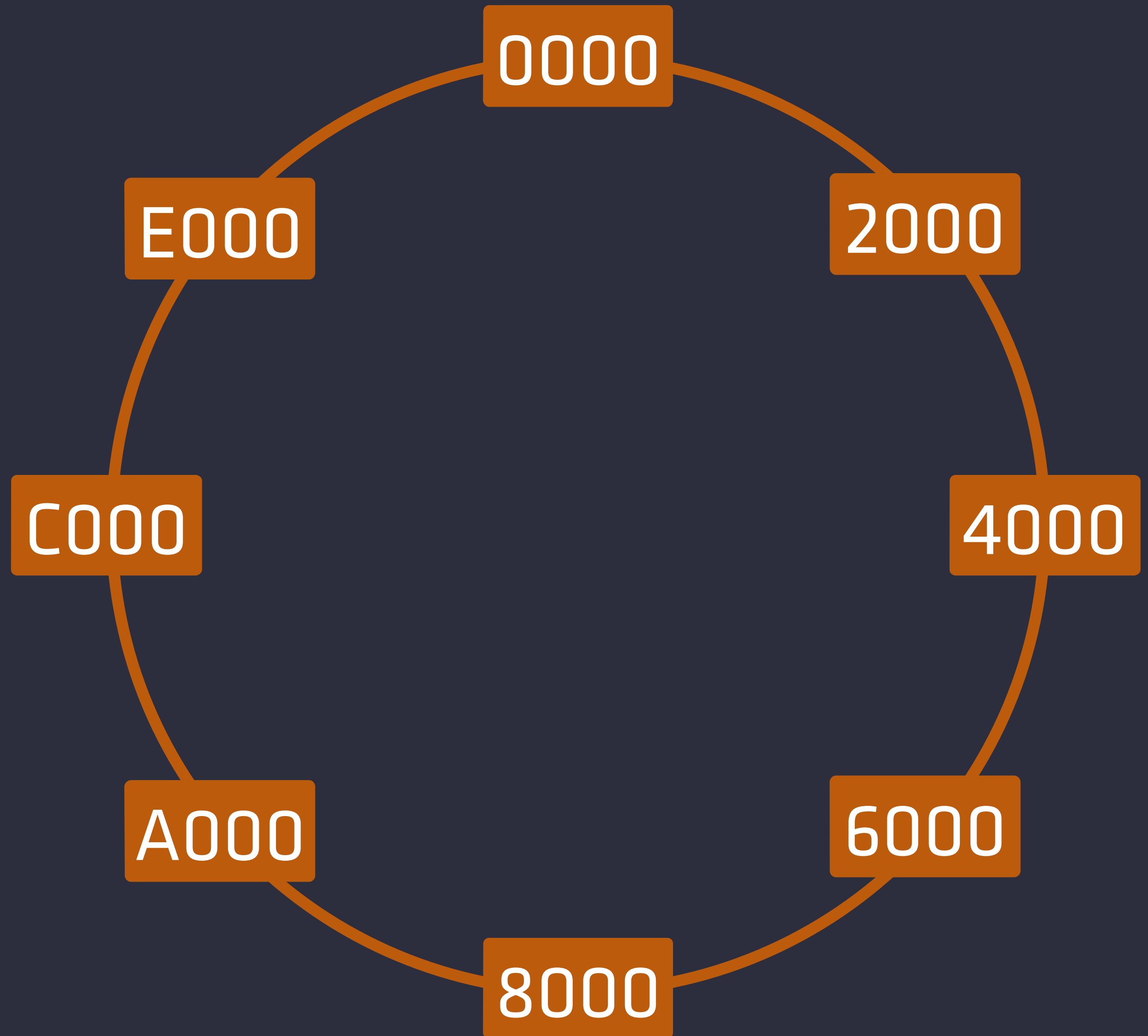
memory:

version:

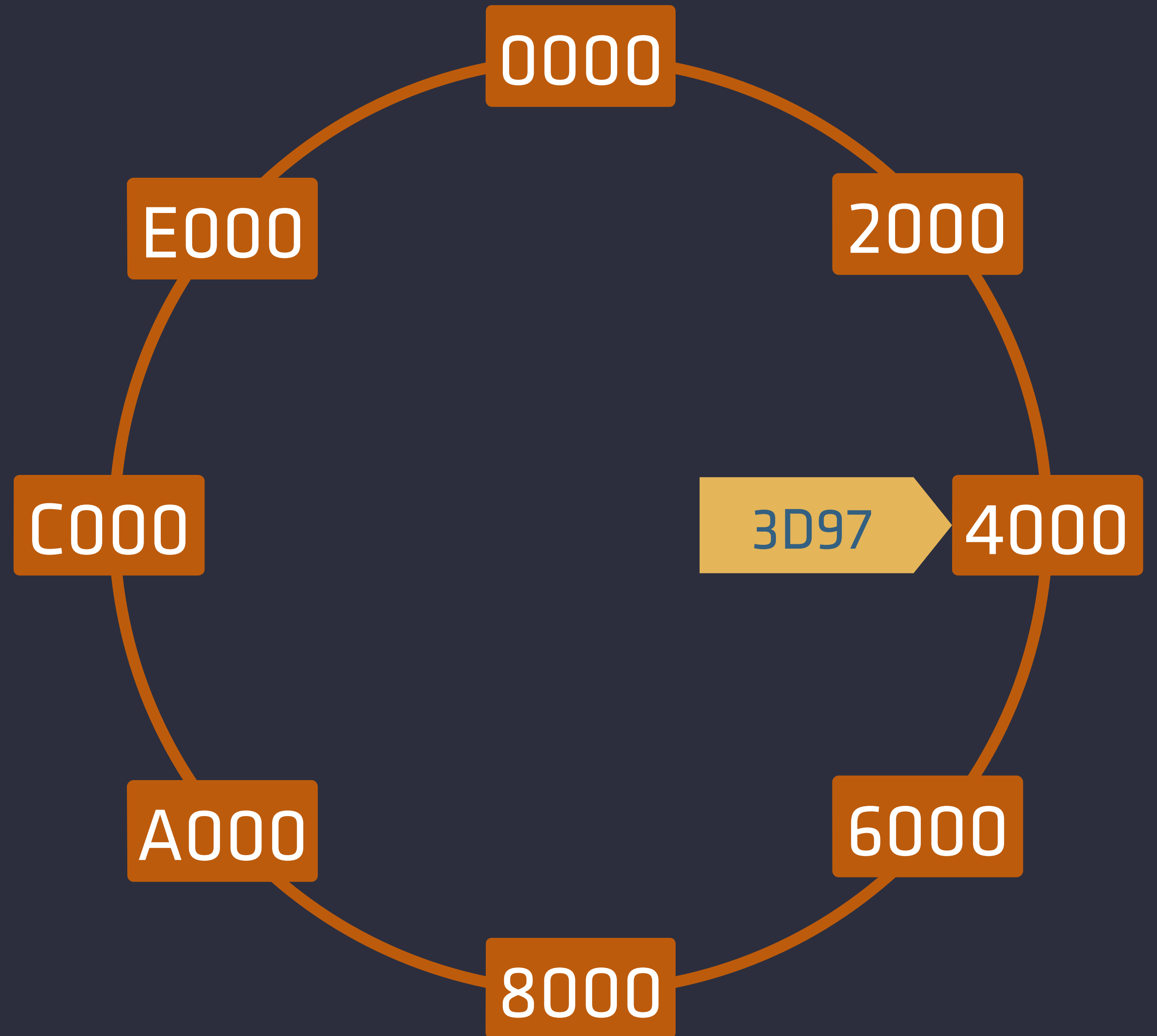
weight:

Distribution

Nexus 5

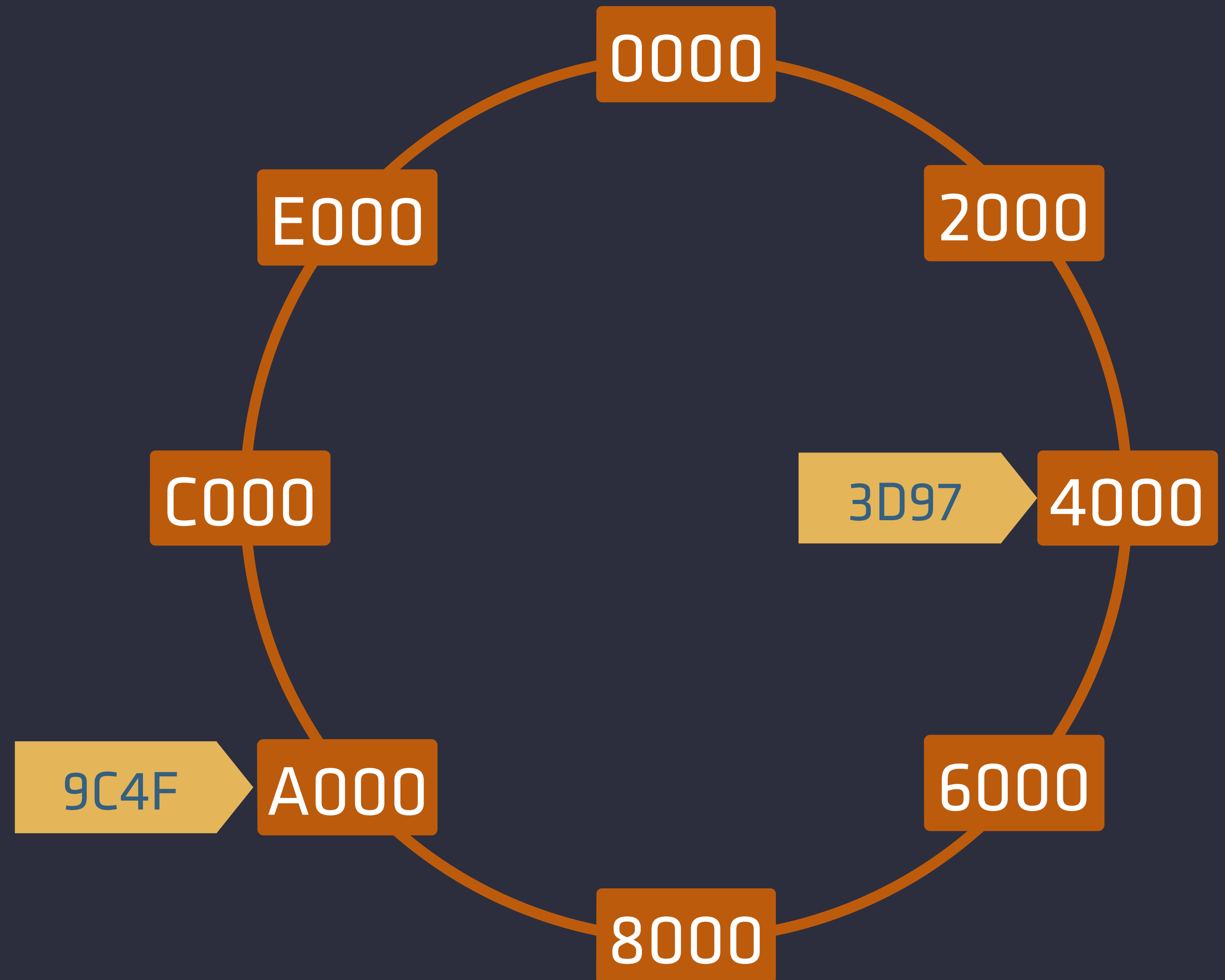


iPhone 6

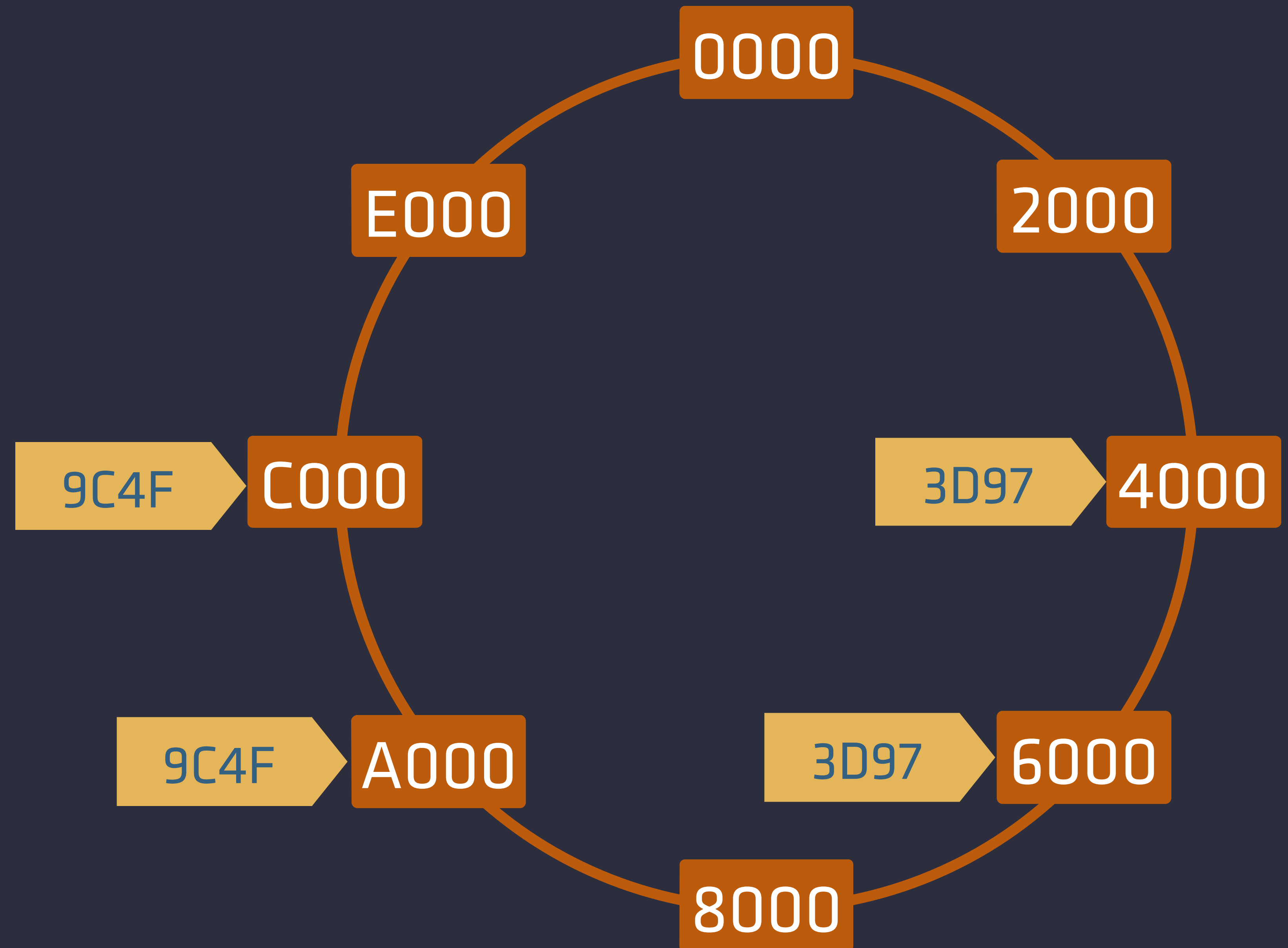


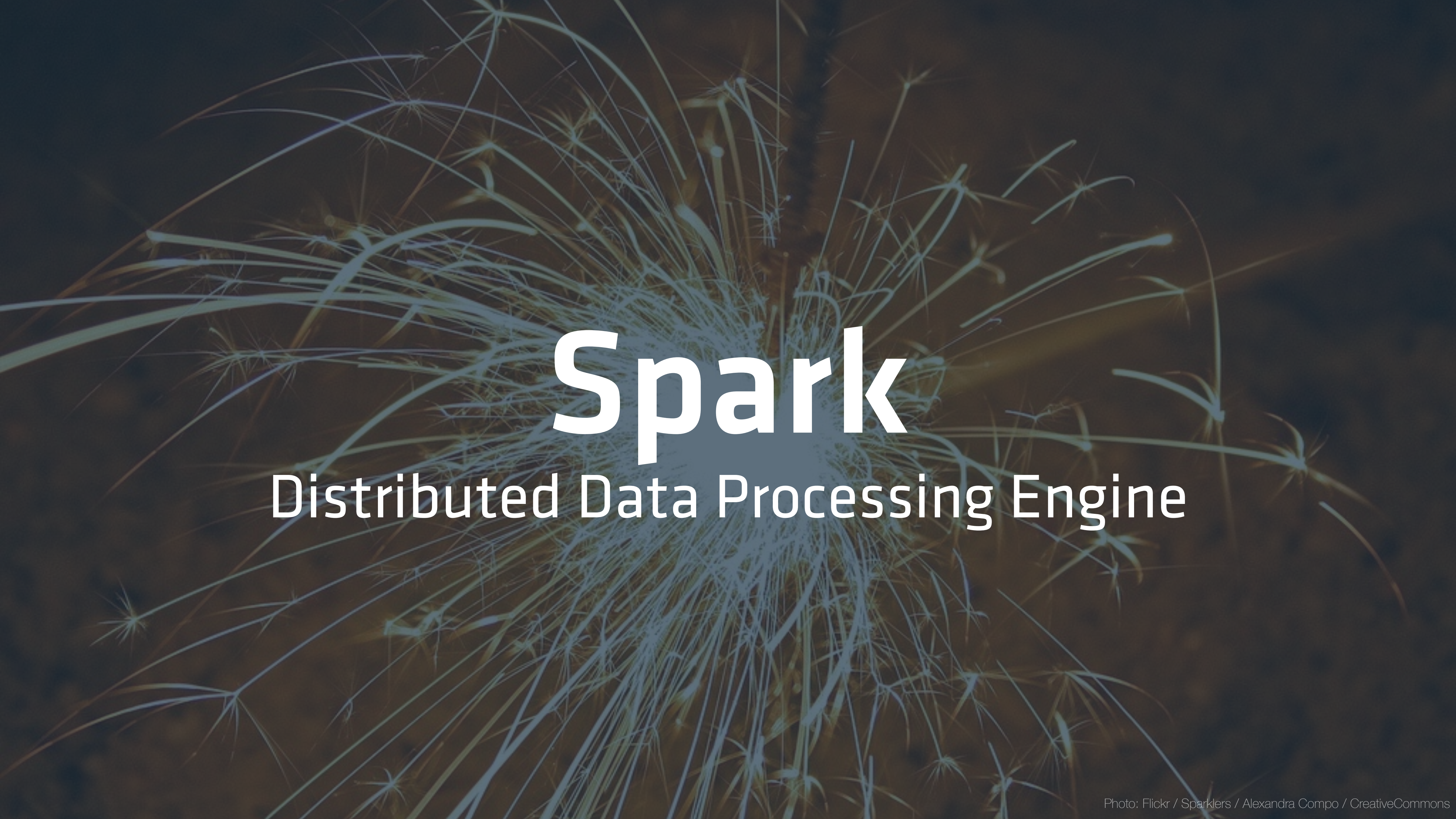
Replication

1 replica



2 replicas



The background of the slide features a large, stylized Spark logo, which is a cluster of thin, light blue lines radiating from a central point, resembling a starburst or a spark. The logo is set against a dark, textured background that looks like a close-up of dry grass or reeds. The text "Spark" is written in a large, white, sans-serif font, and "Distributed Data Processing Engine" is written in a smaller, white, sans-serif font below it.

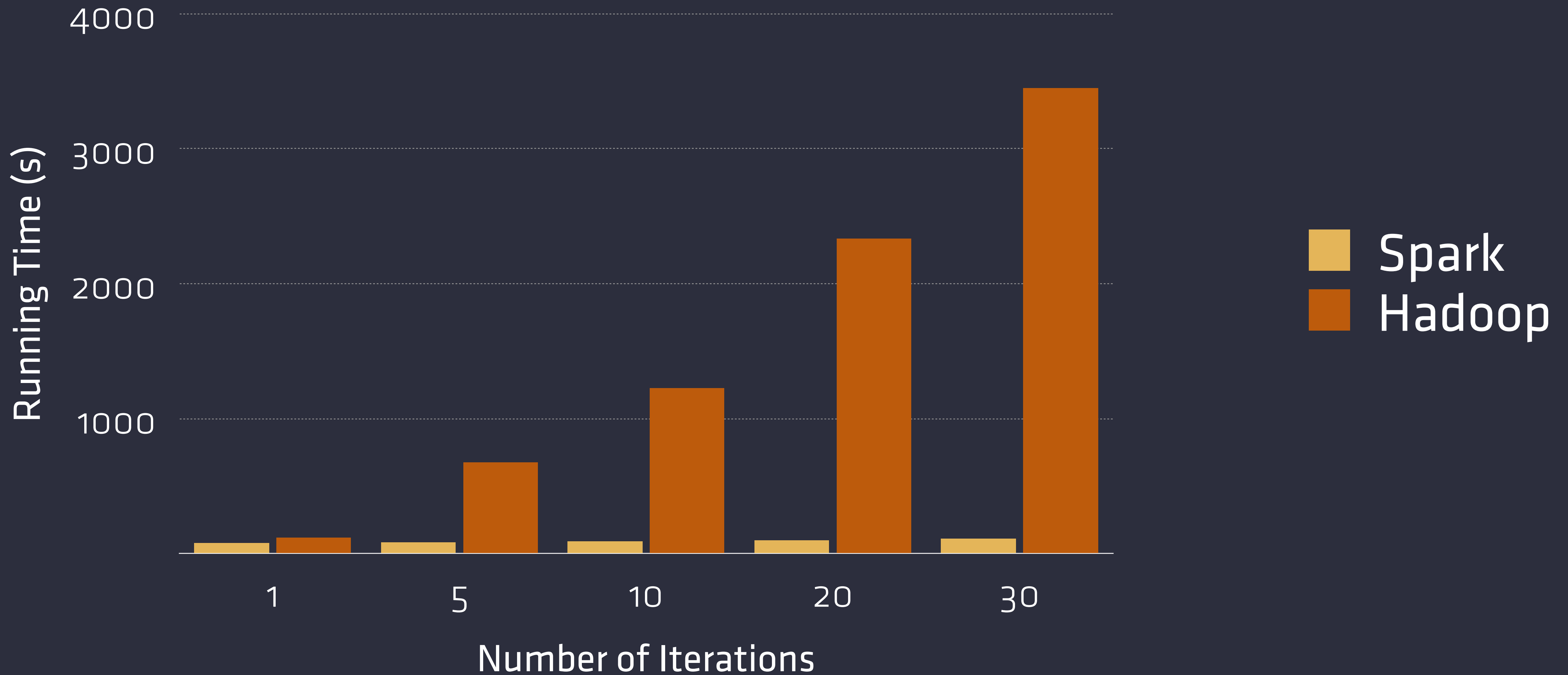
Spark

Distributed Data Processing Engine

Fast

In-memory

Logistic Regression



Easy

map

filter

groupBy

sort

union

join

leftOuterJoin

rightOuterJoin

reduce

count

fold

reduceByKey

groupByKey

cogroup

cross

zip

sample

take

first

partitionBy

mapWith

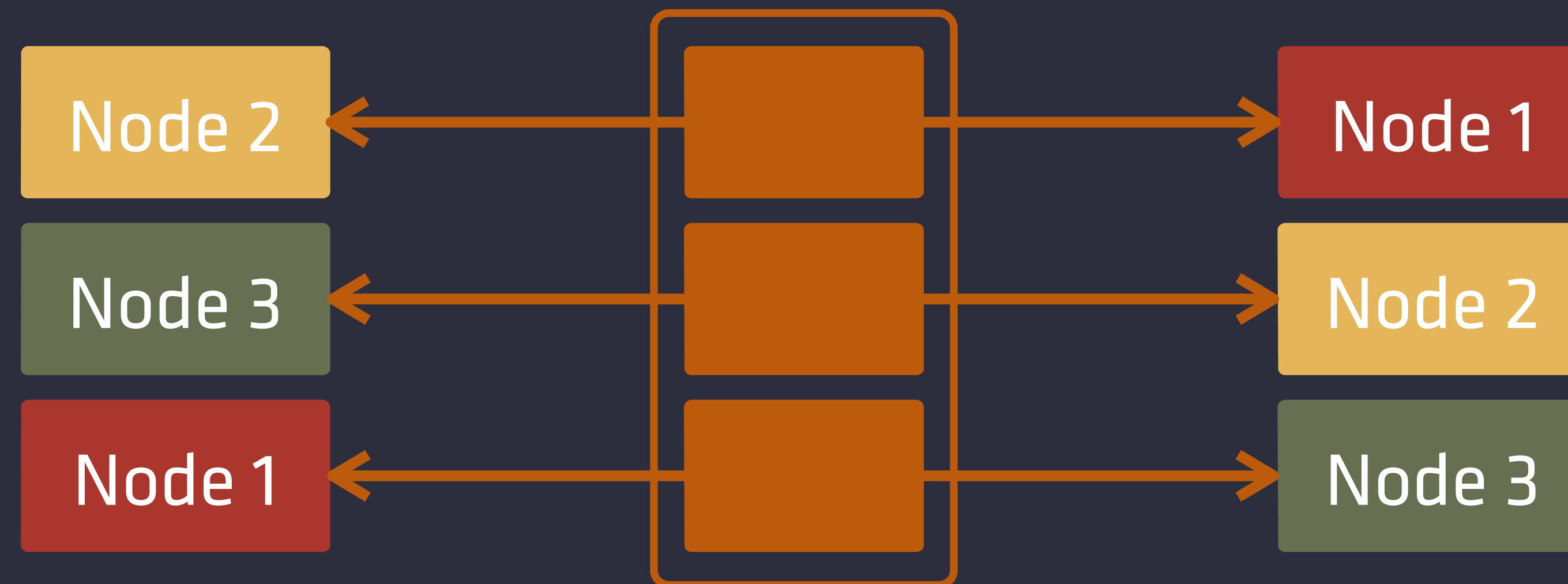
pipe

save

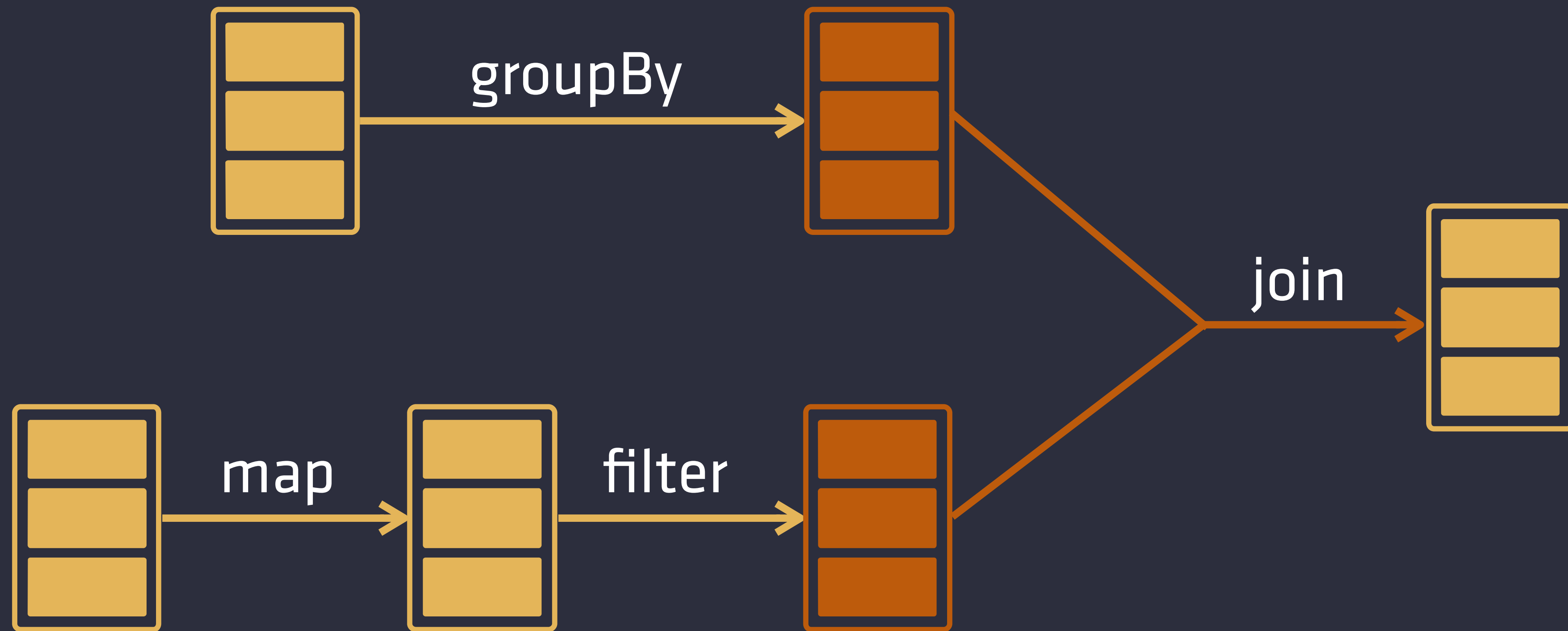
...

RDD

Resilient Distributed Datasets



Operator DAG



Disk RDD



Memory RDD

Spark Streaming

Micro-batching



Spark + Cassandra

DataStax Spark Cassandra Connector

If you write a Spark application that needs access to Cassandra, this library is for you

366 commits
18 branches
9 releases
11 contributors

branch: master
spark-cassandra-connector / +

Merge branch 'b1.1' of github.com:datastax/spark-cassandra-connector

pkolacz authored 13 hours ago latest commit aa1e1eb2eb

doc	Merge branch 'b1.0' into wip-201b1.0-streaming-status	4 days ago
project	Set version to 1.2.0-SNAPSHOT	13 hours ago
sbt	Fix issue #180 (maybe) by adding a shell script to download sbt for u...	15 days ago
scripts	Made the assembly integration more precise, and disabled by default f...	19 days ago
spark-cassandra-connector-d...	Replace internal spark Logging with own class	a day ago
spark-cassandra-connector-ja...	Merge branch 'b1.0' into wip-133-explicit-functions	a month ago
spark-cassandra-connector/src	Replace internal spark Logging with own class	a day ago
.gitignore	#99 Created new java module, separated out the java code to that alon...	2 months ago

<> Code

[Issues](#) 61

[Pull Requests](#) 5

[Wiki](#)

[Pulse](#)

[Graphs](#)

SSH clone URL

git@github.com:datastax/s

You can clone with [HTTPS](#), [SSH](#), or [Subversion](#). ?

[Clone in Desktop](#)

[Download ZIP](#)

<https://github.com/datastax/spark-cassandra-connector>



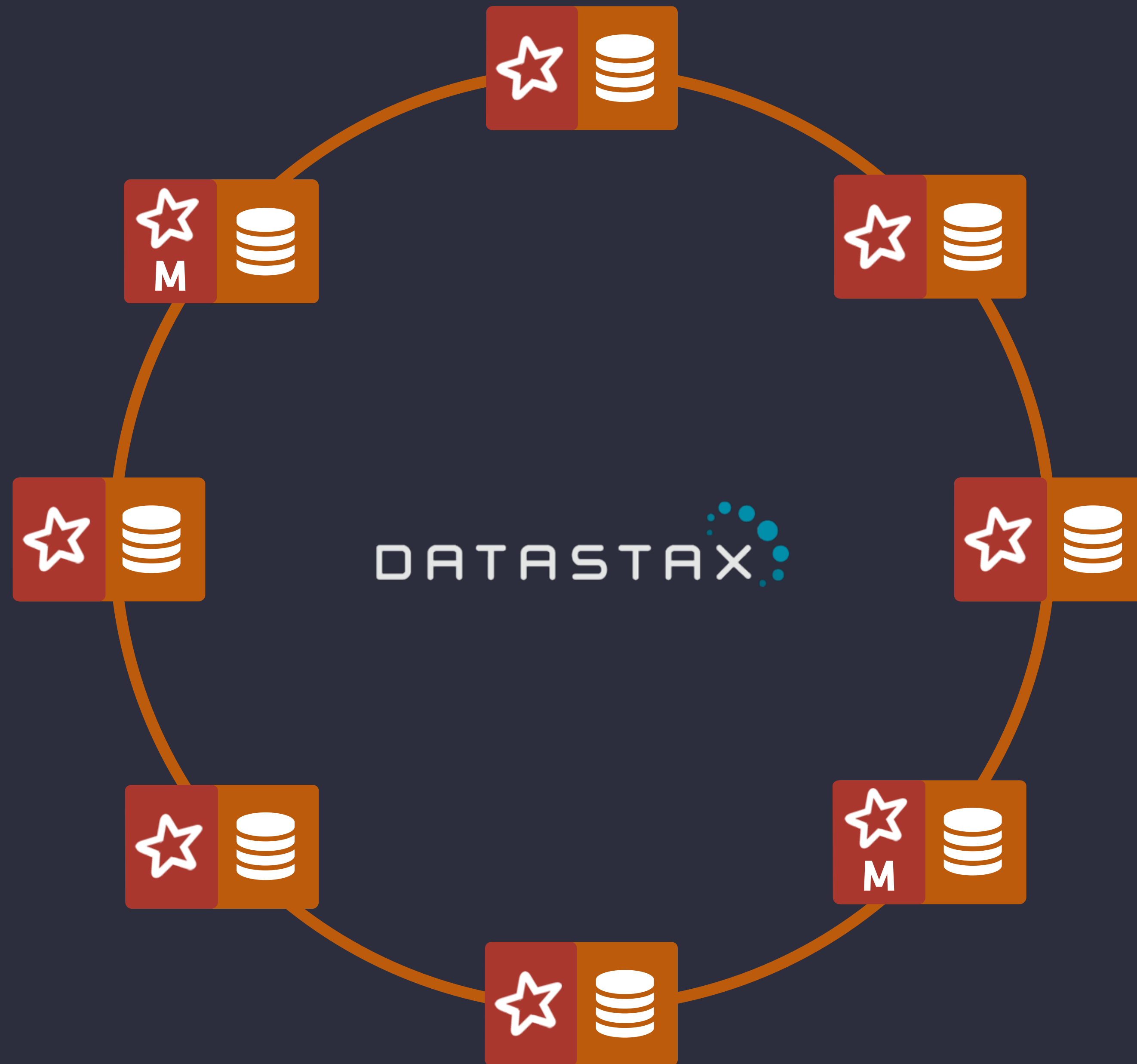
Cassandra



Spark Worker



Spark Master & Worker



Demo

 Twitter Analytics

Cassandra Data Model

#hashtag

ALL:
7139

2014-09-21:
220

2014-09-20:
309

2014-09-19:
129

sort order

```
CREATE TABLE hashtags (  
    hashtag text,  
    interval text,  
    mentions counter,  
    PRIMARY KEY((hashtag), interval)  
) WITH CLUSTERING ORDER BY (interval DESC);
```

Processing Data Stream


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

val sc = new SparkConf()
    .setMaster("spark://127.0.0.1:7077")
    .setAppName("Twitter-Demo")
    .setJars("demo-assembly-1.0.jar")
    .set("spark.cassandra.connection.host", "127.0.0.1")

val ssc = new StreamingContext(sc, Seconds(2))

val stream = TwitterUtils.
    createStream(ssc, None, Nil, storageLevel = StorageLevel.MEMORY_ONLY_SER_2)

val hashTags = stream.flatMap(tweet =>
    tweet.getText.toLowerCase.split(" ").
    filter(tags.contains(Seq("#iphone", "#android"))))

val tagCounts = hashTags.map((_, 1)).reduceByKey(_ + _)

val tagCountsAll = tagCounts.map{
    case (tag, mentions) => (tag, mentions, "ALL")
}

tagCountsAll.saveToCassandra(
```

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

val sc = new SparkConf()
    .setMaster("spark://127.0.0.1:7077")
    .setAppName("Twitter-Demo")
    .setJars("demo-assembly-1.0.jar")
    .set("spark.cassandra.connection.host", "127.0.0.1")

val ssc = new StreamingContext(sc, Seconds(2))

val stream = TwitterUtils.
    createStream(ssc, None, Nil, storageLevel = StorageLevel.MEMORY_ONLY_SER_2)

val hashTags = stream.flatMap(tweet =>
    tweet.getText.toLowerCase.split(" ").
    filter(tags.contains(Seq("#iphone", "#android"))))

val tagCounts = hashTags.map((_, 1)).reduceByKey(_ + _)

val tagCountsAll = tagCounts.map{
    case (tag, mentions) => (tag, mentions, "ALL")
}

tagCountsAll.saveToCassandra(
```

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

val sc = new SparkConf()
    .setMaster("spark://127.0.0.1:7077")
    .setAppName("Twitter-Demo")
    .setJars("demo-assembly-1.0.jar")
    .set("spark.cassandra.connection.host", "127.0.0.1")

val ssc = new StreamingContext(sc, Seconds(2))

val stream = TwitterUtils.
    createStream(ssc, None, Nil, storageLevel = StorageLevel.MEMORY_ONLY_SER_2)

val hashTags = stream.flatMap(tweet =>
    tweet.getText.toLowerCase.split(" ").
    filter(tags.contains(Seq("#iphone", "#android"))))

val tagCounts = hashTags.map((_, 1)).reduceByKey(_ + _)

val tagCountsAll = tagCounts.map{
    case (tag, mentions) => (tag, mentions, "ALL")
}

tagCountsAll.saveToCassandra(
```



```
val ssc = new StreamingContext(sc, Seconds(2))

val stream = TwitterUtils.
  createStream(ssc, None, Nil, storageLevel = StorageLevel.MEMORY_ONLY_SER_2)

val hashTags = stream.flatMap(tweet =>
  tweet.getText.toLowerCase.split(" ").
  filter(tags.contains(Seq("#iphone", "#android"))))

val tagCounts = hashTags.map((_, 1)).reduceByKey(_ + _)

val tagCountsAll = tagCounts.map{
  case (tag, mentions) => (tag, mentions, "ALL")
}

tagCountsAll.saveToCassandra(
  "demo_ks", "hashtags", Seq("hashtag", "mentions", "interval"))

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

```
val ssc = new StreamingContext(sc, Seconds(2))

val stream = TwitterUtils.
  createStream(ssc, None, Nil, storageLevel = StorageLevel.MEMORY_ONLY_SER_2)

val hashTags = stream.flatMap(tweet =>
  tweet.getText.toLowerCase.split(" ").
  filter(tags.contains(Seq("#iphone", "#android"))))

val tagCounts = hashTags.map((_, 1)).reduceByKey(_ + _)

val tagCountsByDay = tagCounts.map{
  case (tag, mentions) => (tag, mentions, DateTime.now.toString("yyyyMMdd"))
}

tagCountsByDay.saveToCassandra(
  "demo_ks", "hashtags", Seq("hashtag", "mentions", "interval"))

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

```
val ssc = new StreamingContext(sc, Seconds(2))

val stream = TwitterUtils.
  createStream(ssc, None, Nil, storageLevel = StorageLevel.MEMORY_ONLY_SER_2)

val hashTags = stream.flatMap(tweet =>
  tweet.getText.toLowerCase.split(" ").
  filter(tags.contains(Seq("#iphone", "#android"))))

val tagCounts = hashTags.map((_, 1)).reduceByKey(_ + _)

val tagCountsAll = tagCounts.map{
  case (tag, mentions) => (tag, mentions, "ALL")
}

tagCountsAll.saveToCassandra(
  "demo_ks", "hashtags", Seq("hashtag", "mentions", "interval"))

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


Questions ?