
Cassandra
and
Spark
@tlberglund


Cassandra
and
Spark
@tlberglund

Cassandra

What:

- Distributed database
- Transactional/online
- Low latency, high velocity

When?

- You have too much data
- It changes a lot
- The system is always on

Data Model

```
CREATE TABLE credit_card_transactions(
  card_number VARCHAR,
  transaction_time TIMESTAMP,
  amount DECIMAL(10,2),
  merchant_id VARCHAR,
  country VARCHAR,
  PRIMARY KEY(card_number,
              transaction_time DESC));
```

Partition Key

```
CREATE TABLE credit_card_transactions(
  card_number VARCHAR,
  transaction_time TIMESTAMP,
  amount DECIMAL(10,2),
  merchant_id VARCHAR,
  country VARCHAR,
  PRIMARY KEY(card_number,
              transaction_time DESC));
```

Partition Key

```
CREATE TABLE credit_card_transactions(
  card_number VARCHAR,
  transaction_time TIMESTAMP,
  amount DECIMAL(10,2),
  merchant_id VARCHAR,
  country VARCHAR,
  PRIMARY KEY(card_number,
              transaction_time DESC));
```

Clustering Column

```
CREATE TABLE credit_card_transactions(
  card_number VARCHAR,
  transaction_time TIMESTAMP,
  amount DECIMAL(10,2),
  merchant_id VARCHAR,
  country VARCHAR,
  PRIMARY KEY(card_number,
              transaction_time DESC));
```

Clustering Column

```
CREATE TABLE credit_card_transactions(
  card_number VARCHAR,
  transaction_time TIMESTAMP,
  amount DECIMAL(10,2),
  merchant_id VARCHAR,
  country VARCHAR,
  PRIMARY KEY(card_number,
              transaction_time DESC));
```

Primary Key

```
CREATE TABLE credit_card_transactions(
  card_number VARCHAR,
  transaction_time TIMESTAMP,
  amount DECIMAL(10,2),
  merchant_id VARCHAR,
  country VARCHAR,
  PRIMARY KEY(card_number,
              transaction_time DESC));
```

Data Model

CARD_NUMBER	XCTN_TIME	AMOUNT	COUNTRY	MERCHANT
5444…8389	13:58	$12.42	US	101
5444…8389	14:03	$98.03	US	158
5444…6027	13:59	$3.02	US	101
5444…6027	14:03	€23.78	ES	352
4532…1191	13:59	$50.38	US	102
4532…2189	14:02	$12.42	US	198

Partition Key

CARD_NUMBER	XCTN_TIME	AMOUNT	COUNTRY	MERCHANT
5444…8389	13:58	$12.42	US	101
5444…8389	14:03	$98.03	US	158
5444…6027	13:59	$3.02	US	101
5444…6027	14:03	€23.78	ES	352
4532…1191	13:59	$50.38	US	102
4532…2189	14:02	$12.42	US	198

Clustering Column

CARD_NUMBER	XCTN_TIME	AMOUNT	COUNTRY	MERCHANT
5444…8389	13:58	$12.42	US	101
5444…8389	14:03	$98.03	US	158
5444…6027	13:59	$3.02	US	101
5444…6027	14:03	€23.78	ES	352
4532…1191	13:59	$50.38	US	102
4532…2189	14:02	$12.42	US	198

Primary Key

CARD_NUMBER	XCTN_TIME	AMOUNT	COUNTRY	MERCHANT
5444…8389	13:58	$12.42	US	101
5444…8389	14:03	$98.03	US	158
5444…6027	13:59	$3.02	US	101
5444…6027	14:03	€23.78	ES	352
4532…1191	13:59	$50.38	US	102
4532…2189	14:02	$12.42	US	198

Partition

CARD_NUMBER	XCTN_TIME	AMOUNT	COUNTRY	MERCHANT
5444…8389	13:58	$12.42	US	101
5444…8389	14:03	$98.03	US	158
5444…6027	13:59	$3.02	US	101
5444…6027	14:03	€23.78	ES	352
4532…1191	13:59	$50.38	US	102
4532…2189	14:02	$12.42	US	198

Partition

CARD_NUMBER	XCTN_TIME	AMOUNT	COUNTRY	MERCHANT
5444…8389	13:58	$12.42	US	101
5444…8389	14:03	$98.03	US	158
5444…6027	13:59	$3.02	US	101
5444…6027	14:03	€23.78	ES	352
4532…1191	13:59	$50.38	US	102
4532…2189	14:02	$12.42	US	198

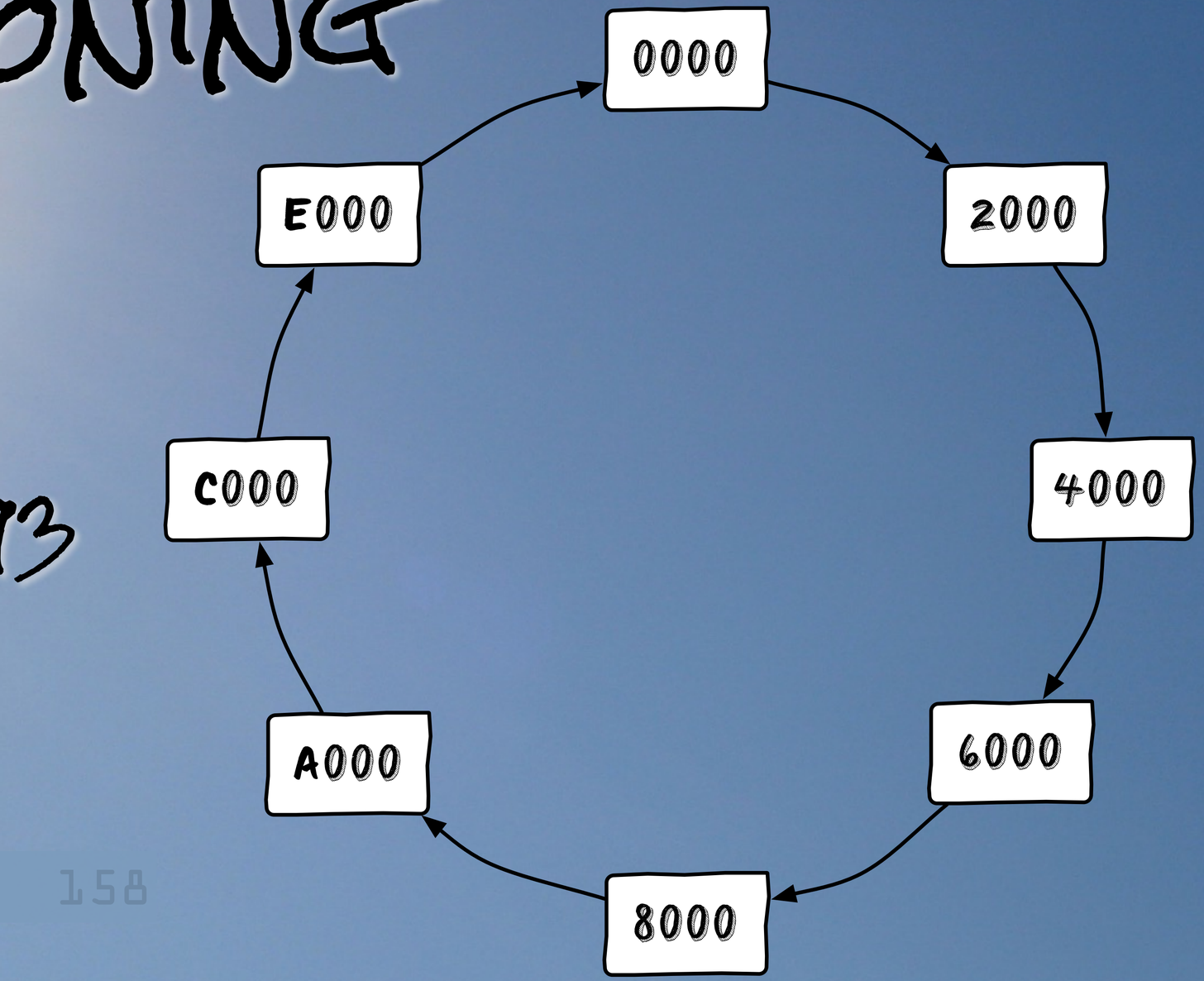
Partition

CARD_NUMBER	XCTN_TIME	AMOUNT	COUNTRY	MERCHANT
5444…8389	13:58	$12.42	US	101
5444…8389	14:03	$98.03	US	158
5444…6027	13:59	$3.02	US	101
5444…6027	14:03	€23.78	ES	352
4532…1191	13:59	$50.38	US	102
4532…2189	14:02	$12.42	US	198

Partition

CARD_NUMBER	XCTN_TIME	AMOUNT	COUNTRY	MERCHANT
5444…8389	13:58	$12.42	US	101
5444…8389	14:03	$98.03	US	158
5444…6027	13:59	$3.02	US	101
5444…6027	14:03	€23.78	ES	352
4532…1191	13:59	$50.38	US	102
4532…2189	14:02	$12.42	US	198

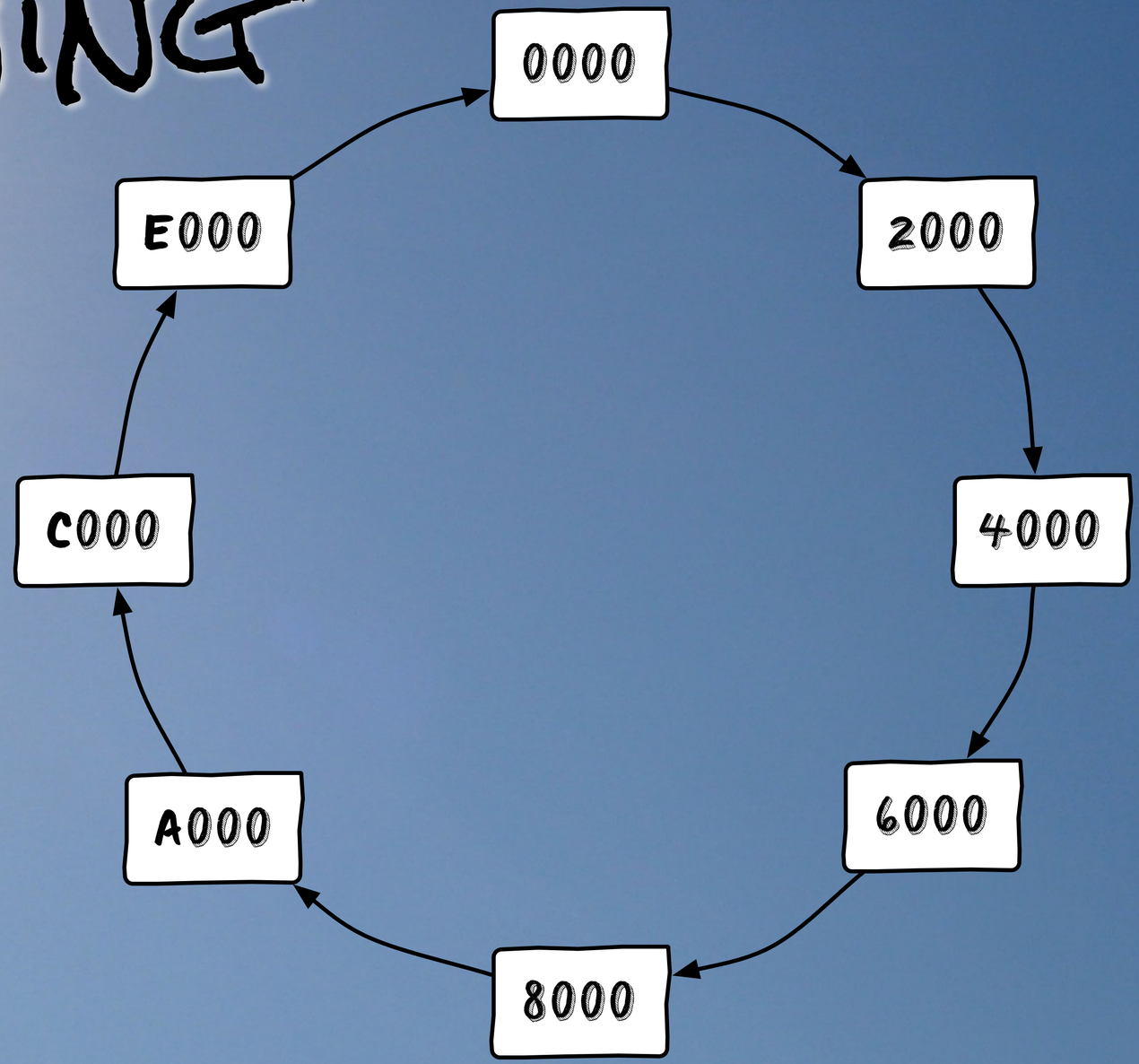
Partitioning



F(X) → 5D93

5444…8389

Partitioning



5D93

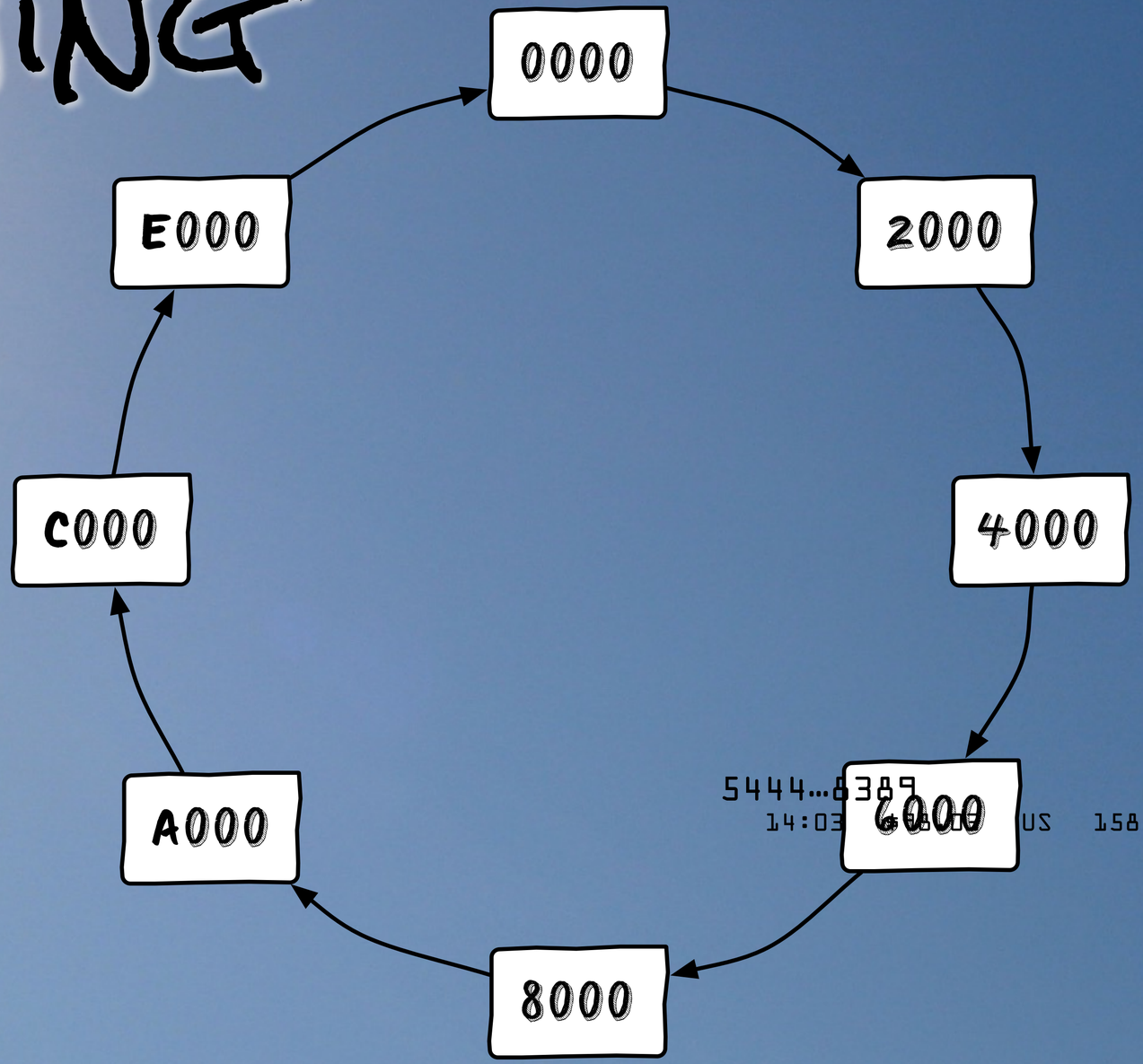
5444…8389

14:03 $98.03 US 158

Partitioning

F(X) → 5D93

5444…8389



Partitioning

Partition

CARD_NUMBER	XCTN_TIME	AMOUNT	COUNTRY	MERCHANT
5444…8389	13:58	$12.42	US	101
5444…8389	14:03	$98.03	US	158
5444…6027	13:59	$3.02	US	101
5444…6027	14:03	€23.78	ES	352
4532…1191	13:59	$50.38	US	102
4532…2189	14:02	$12.42	US	198

Partitions

- Clumps of related data
- Always on one server
- Accessed by consistent hashing

Consequences

- Design for low-latency, high-throughput reads and writes
- Inflexible ad-hoc queries
- Very little aggregation
- Very little secondary indexing


Spark

When do I use it?

- You have too much data
- You don't know the questions
- ~~You don't like MapReduce~~

You're tired of Hadoop lol

What are they saying?

- Distributed, functional
- "In-memory"
- "Real-time"
- Next-generation MapReduce

Paradigm :

Store collections in Resilient Distributed Datasets, or "RDDs"

Paradigm :

Transform RDDs into other RDDs

Paradigm :

Aggregate RDDs with "Actions"

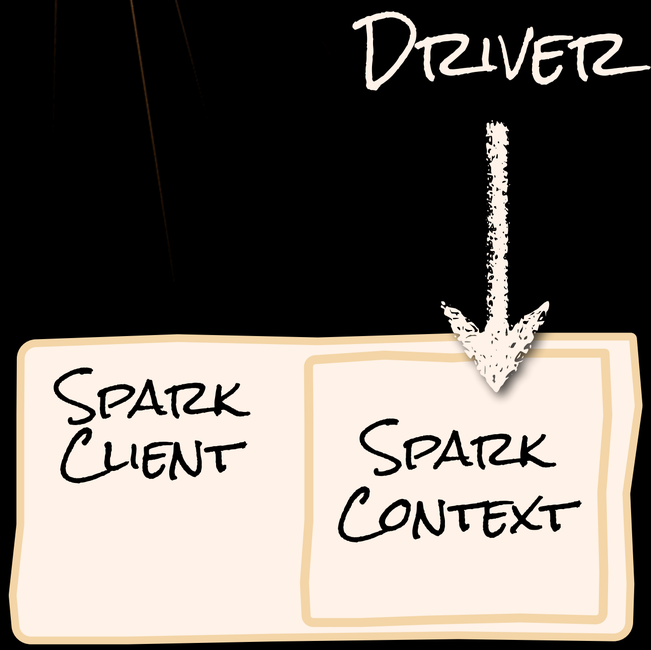
Paradigm :

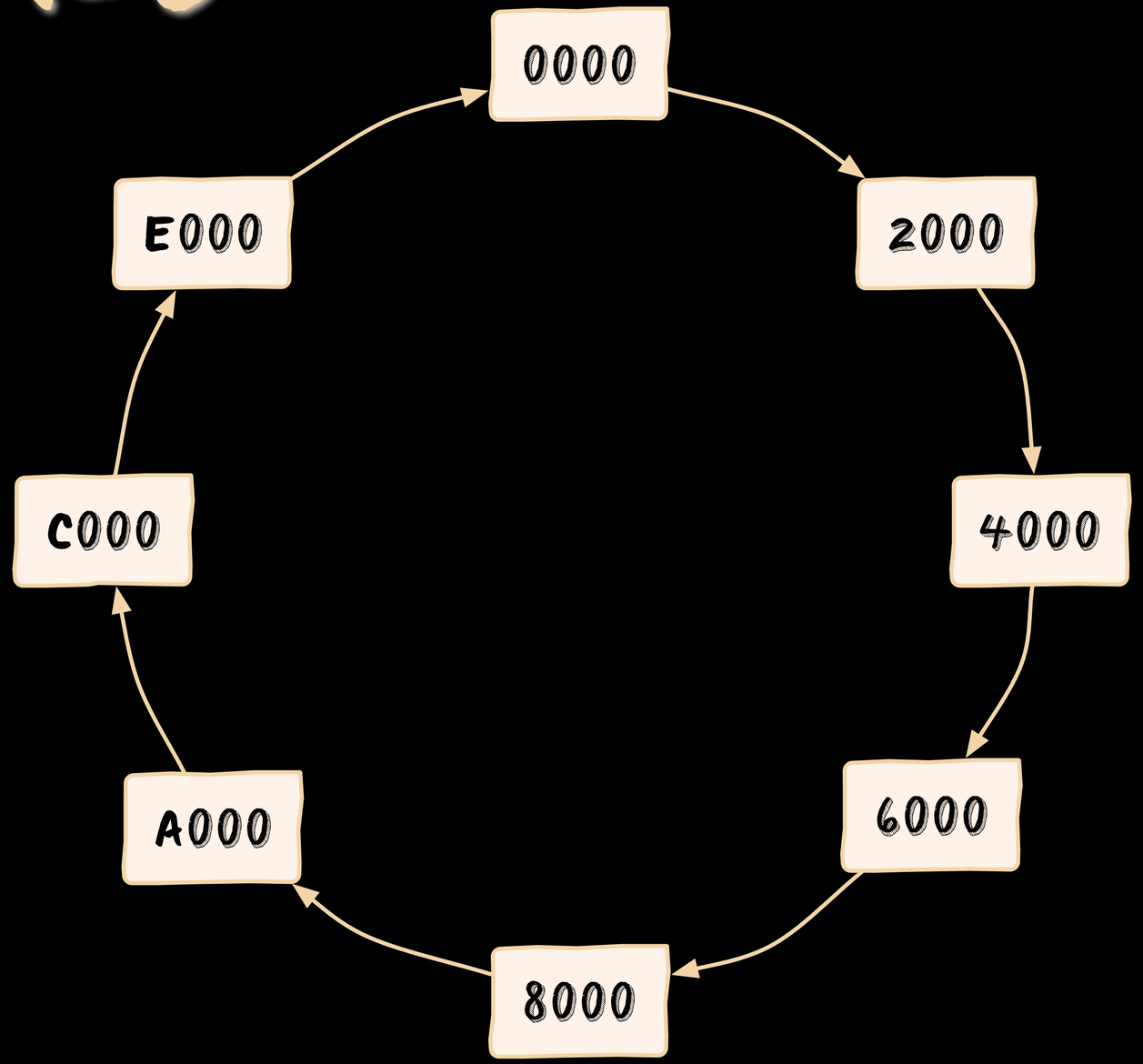
Keep track of a graph of transformations and Actions

Paradigm :

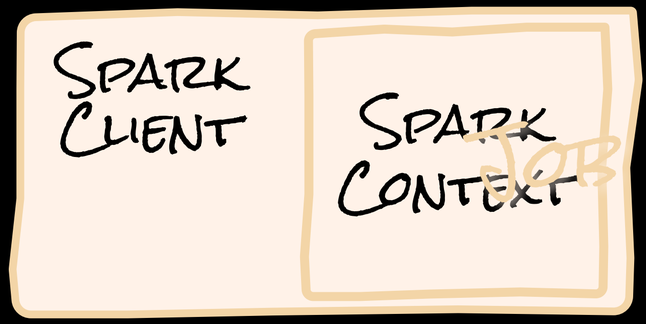
Distribute all of this
storage and computation

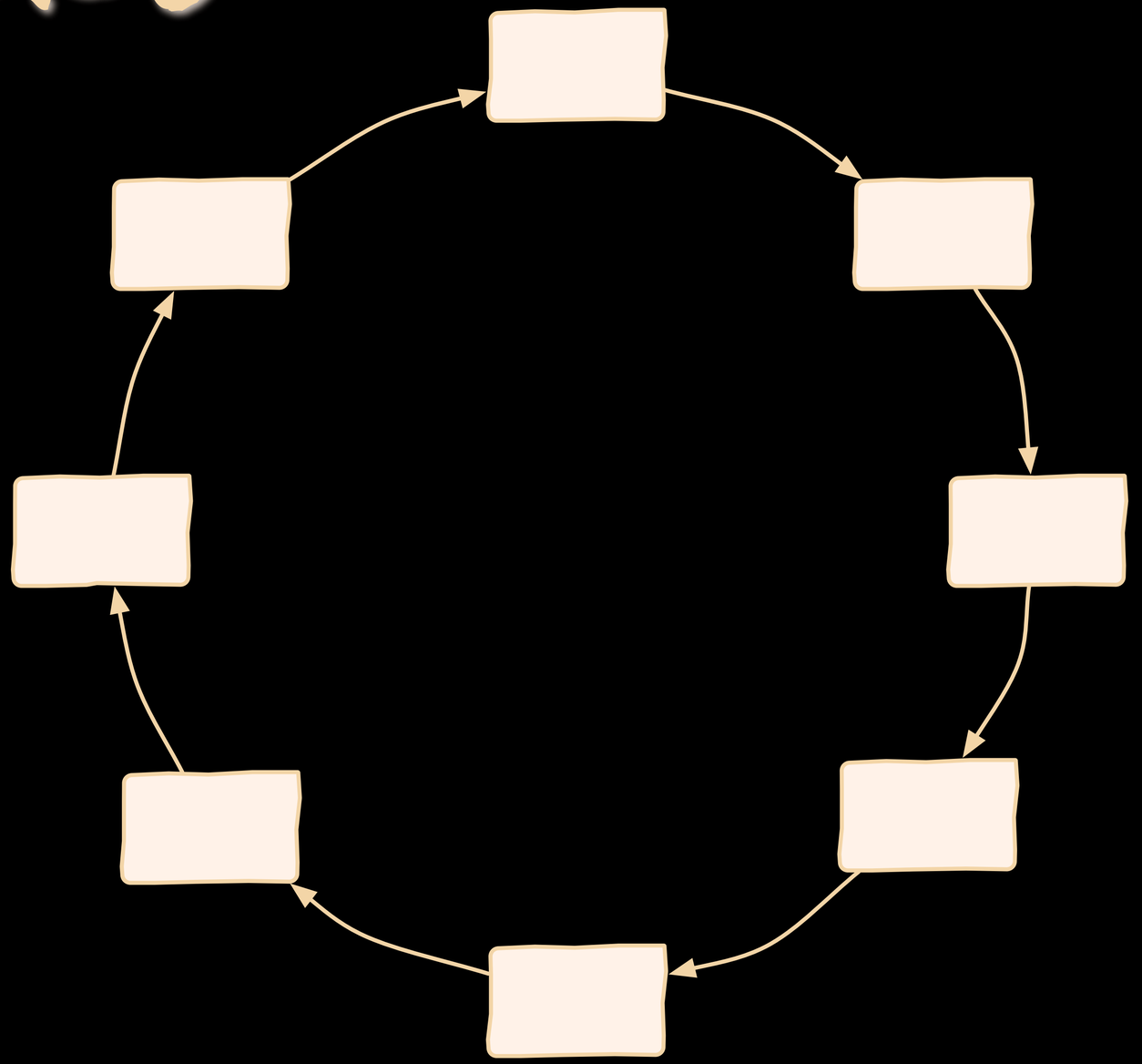
Architecture





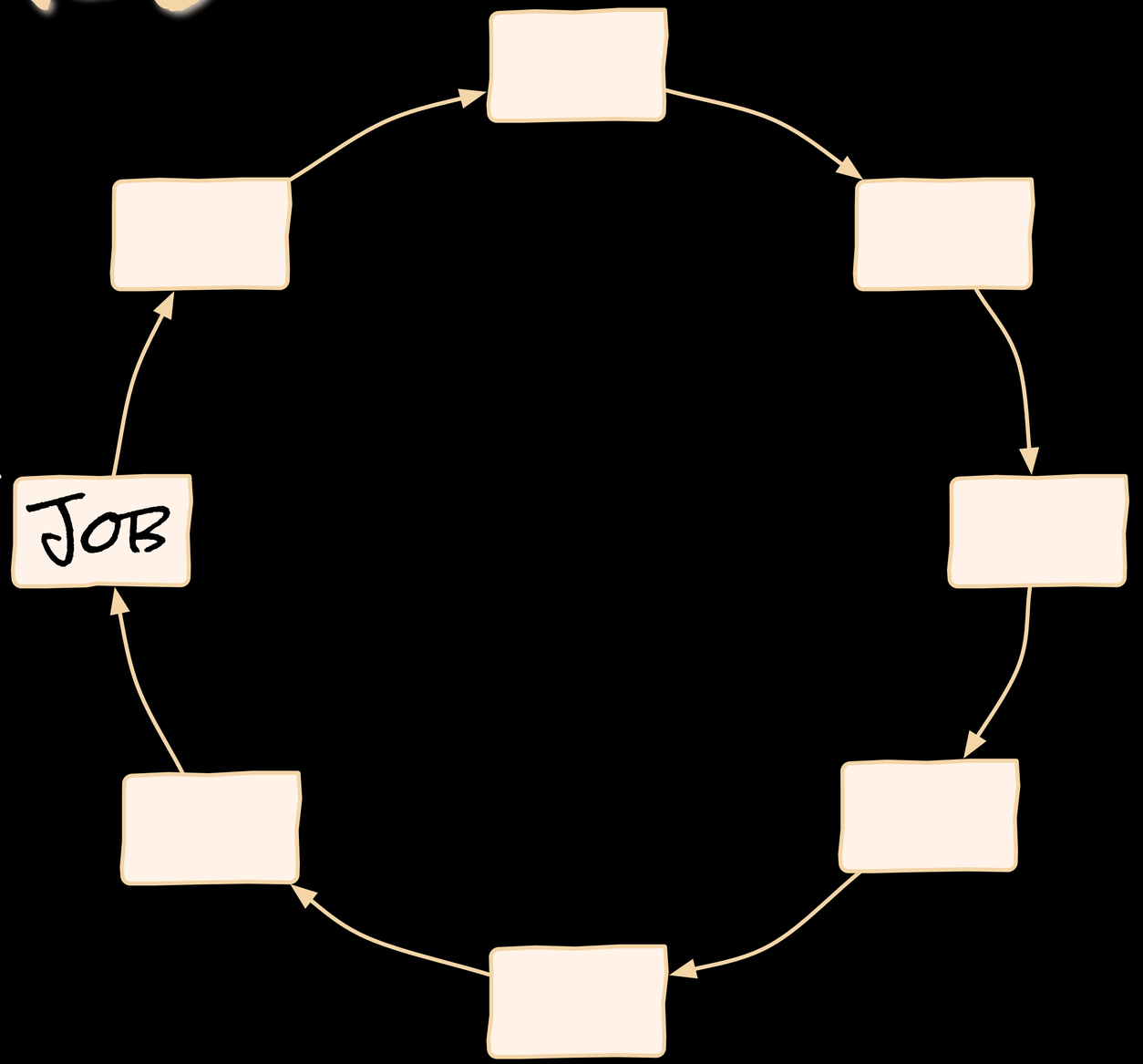
Architecture





Architecture

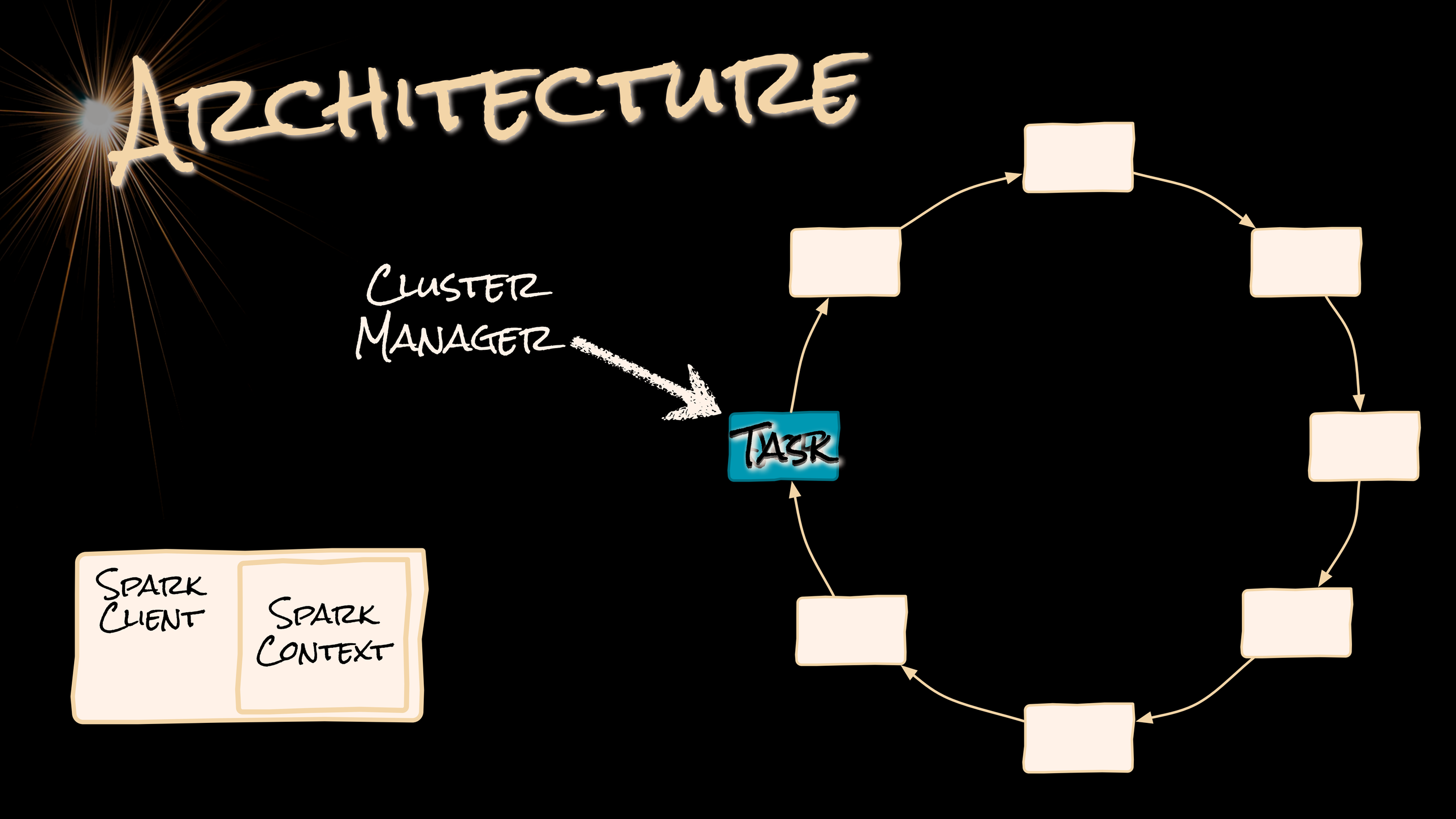
Cluster Manager



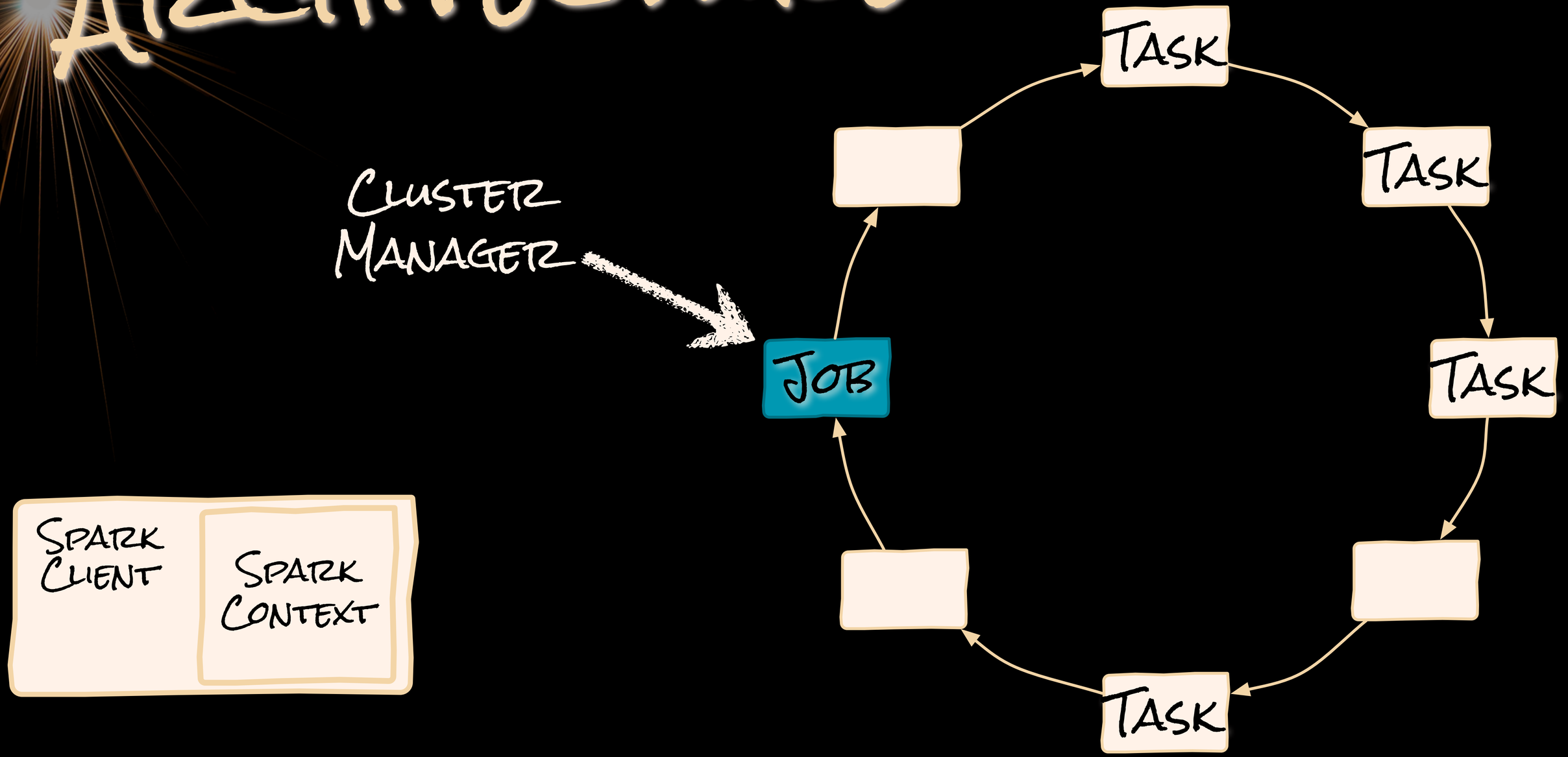
Spark Client

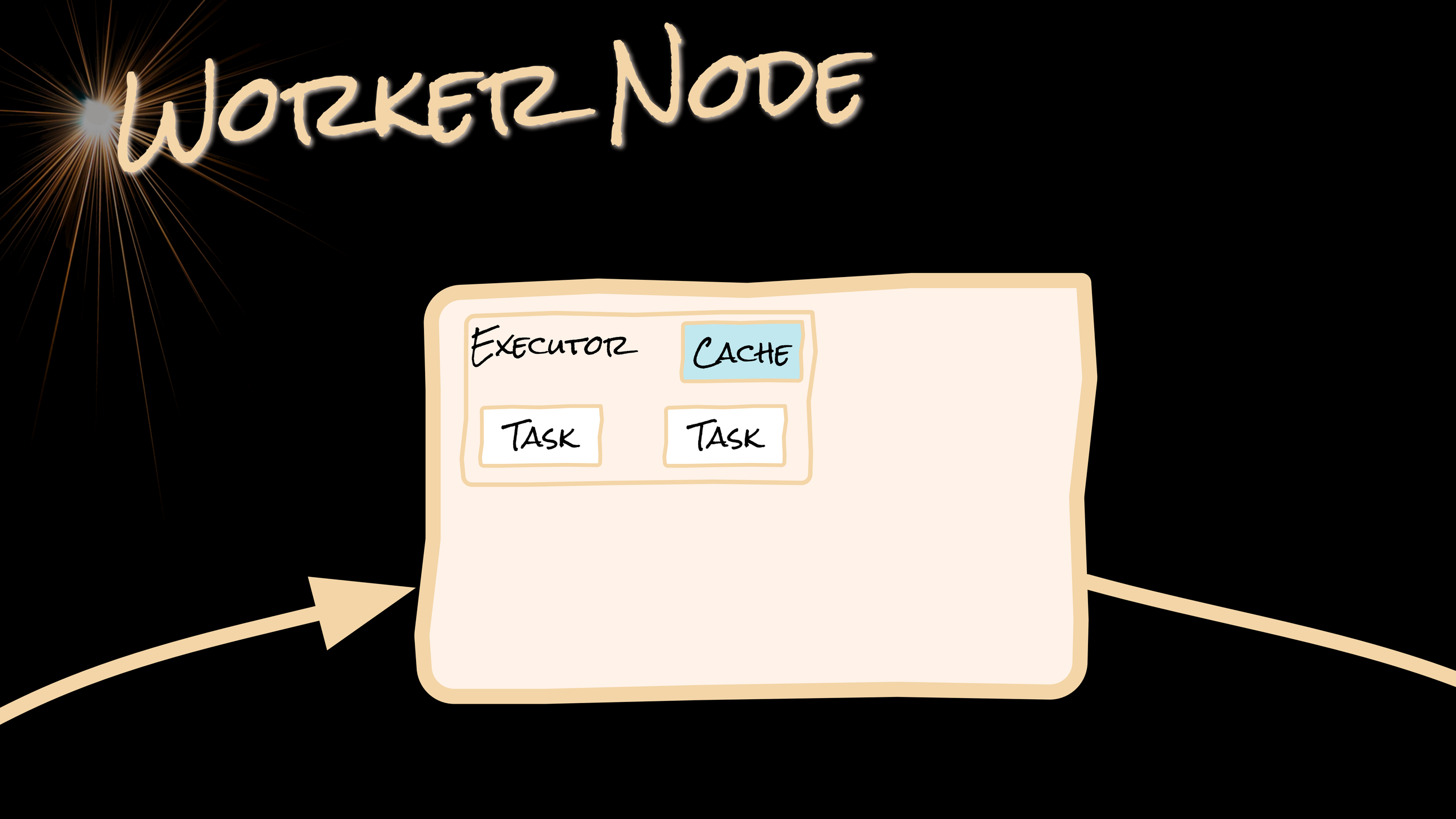
Spark Context

Architecture

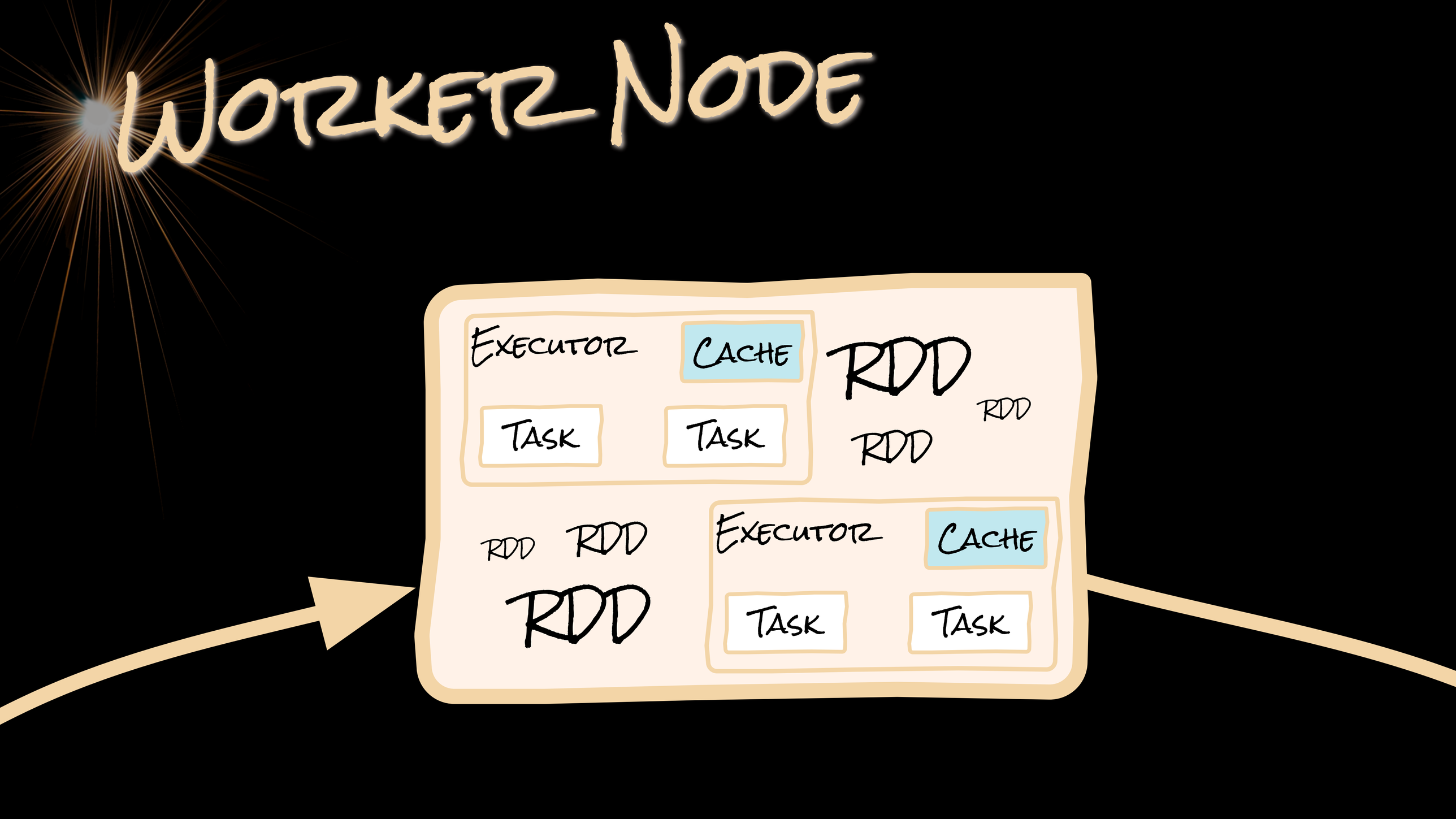


Architecture




Worker Node
Executor
Cache
Task
Task

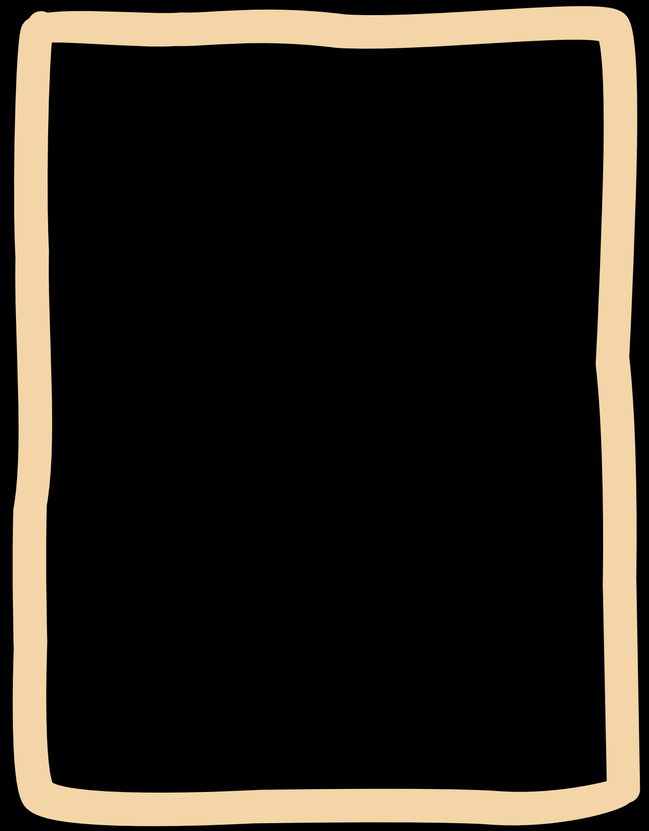
Worker Node


Executor
Cache
Task
Task
RDD
RDD
RDD
RDD
RDD
RDD
Executor
Cache
Task
Task

What's an RDD?

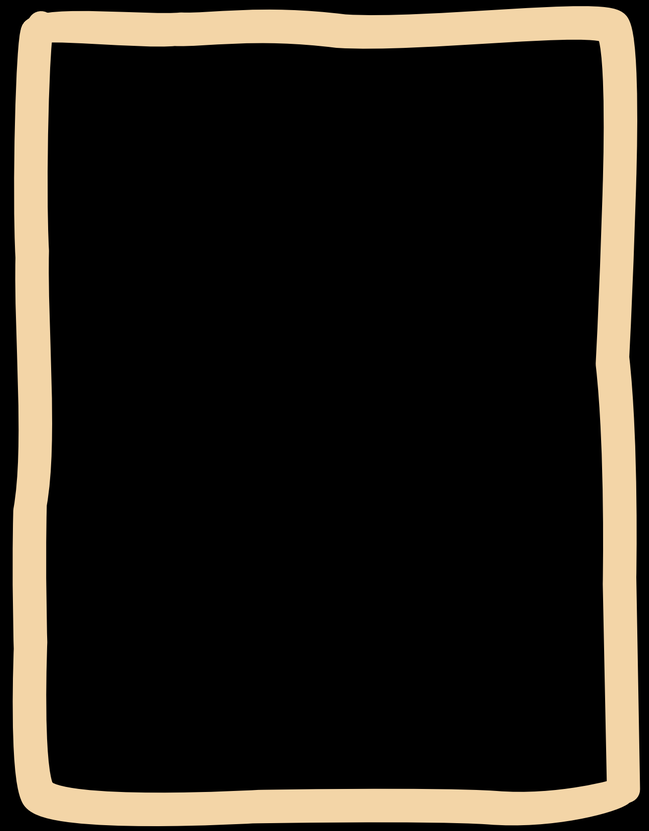
WHAT'S AN RDD?

THIS IS →



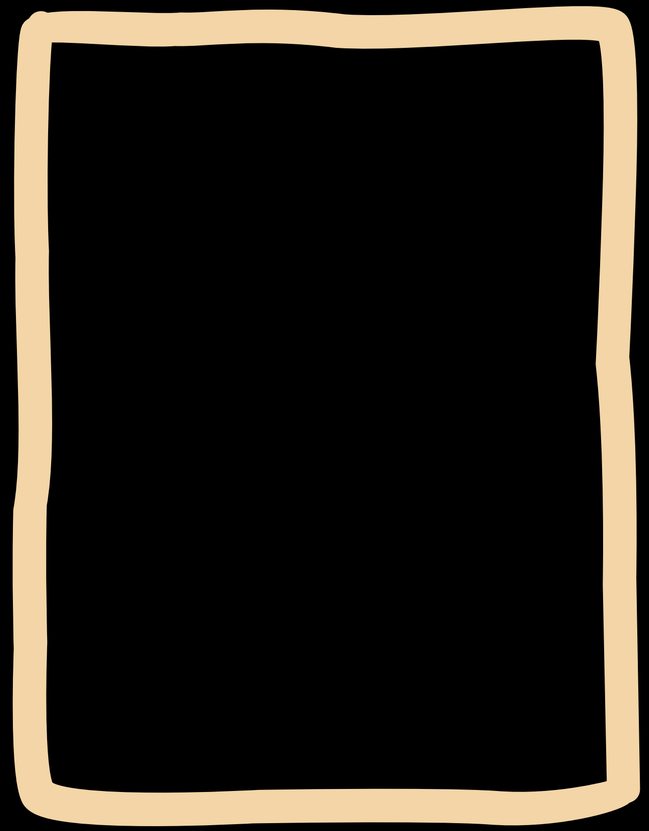
What's an RDD?

Bigger than a computer



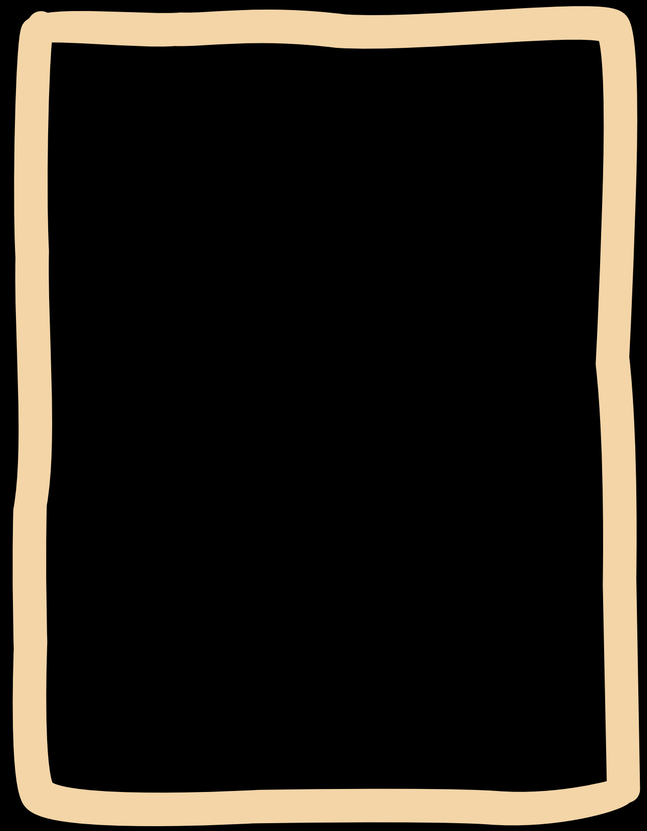
What's an RDD?

Read from an input source?



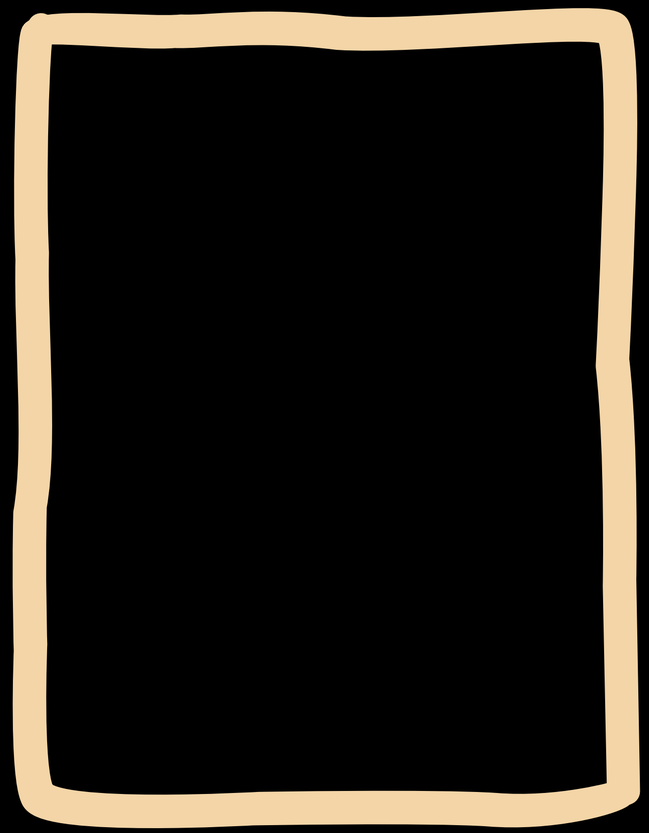
What's an RDD?

The output of a pure function?



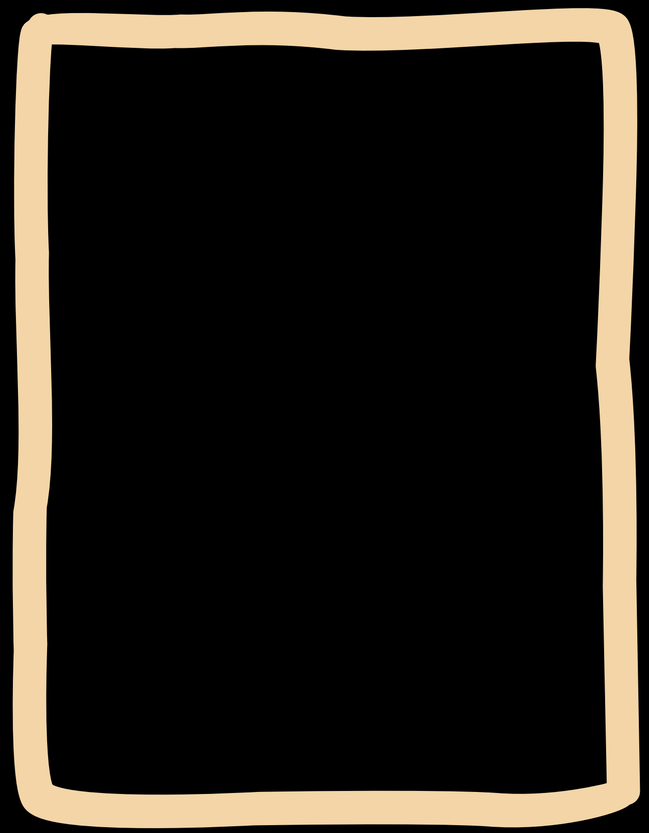
What's an RDD?

Immutable



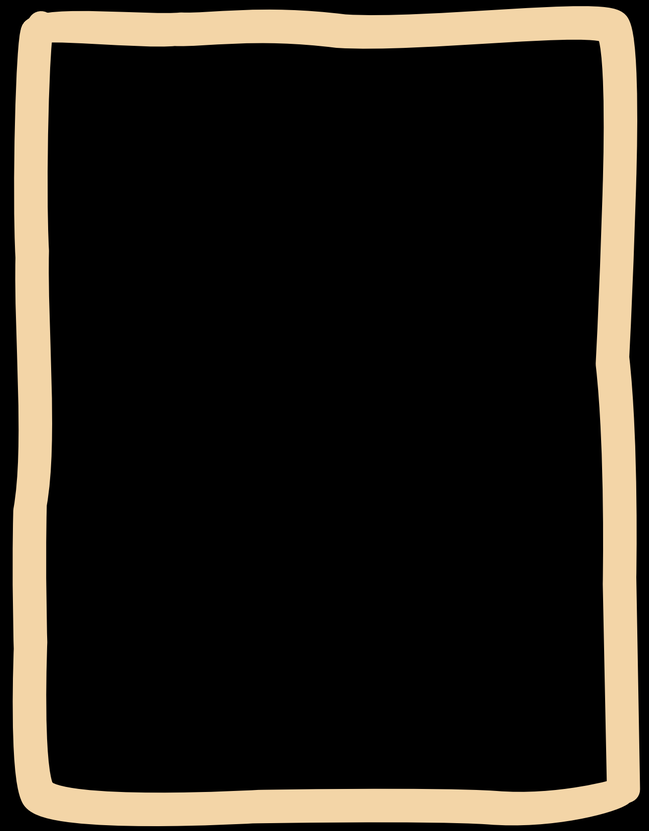
What's an RDD?

Typed



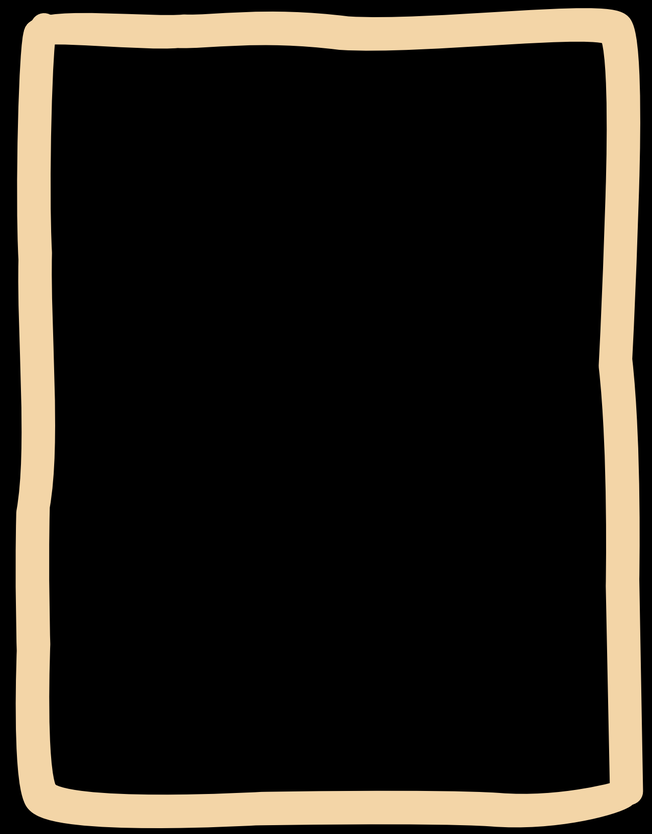
What's an RDD?

- Ordered



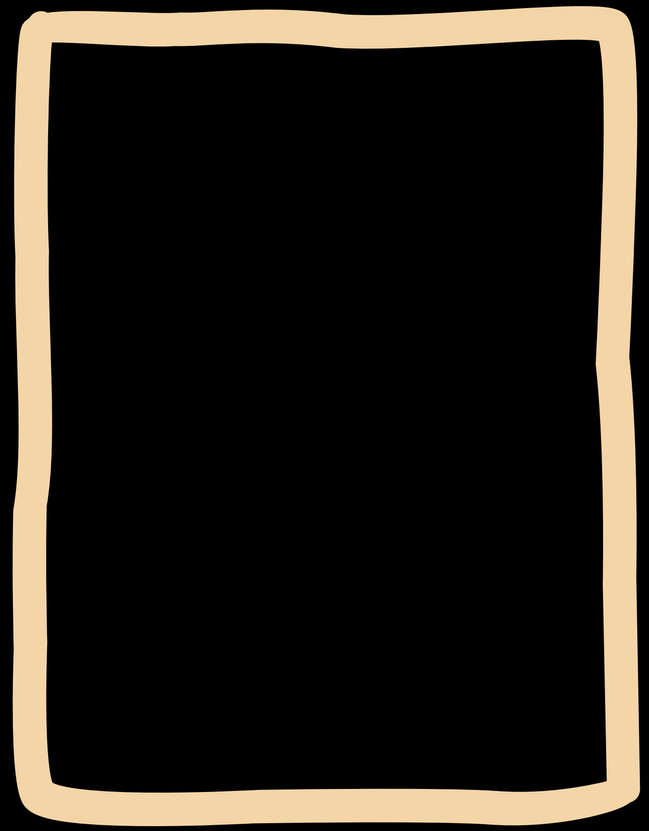
What's an RDD?

Functions comprise a Graph



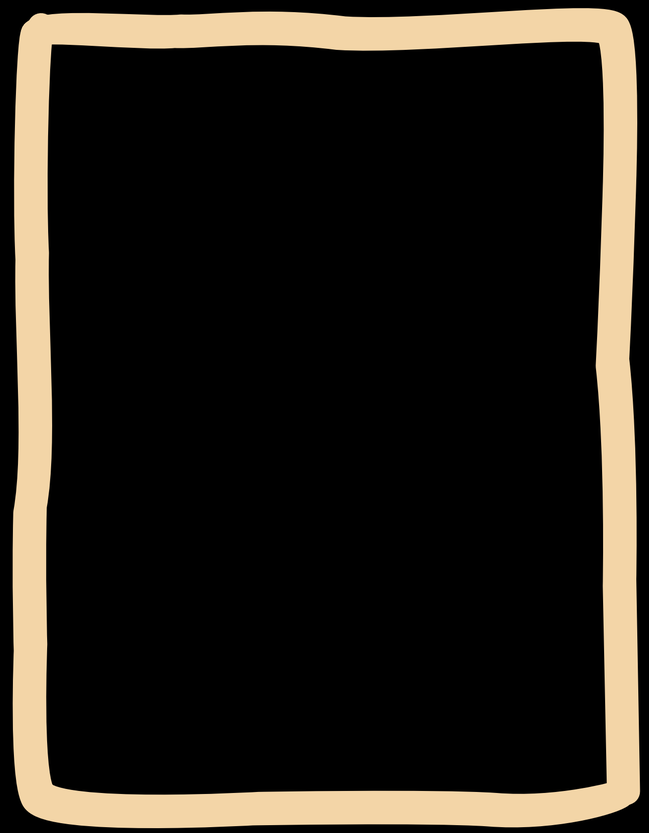
What's an RDD?

Lazily
Evaluated

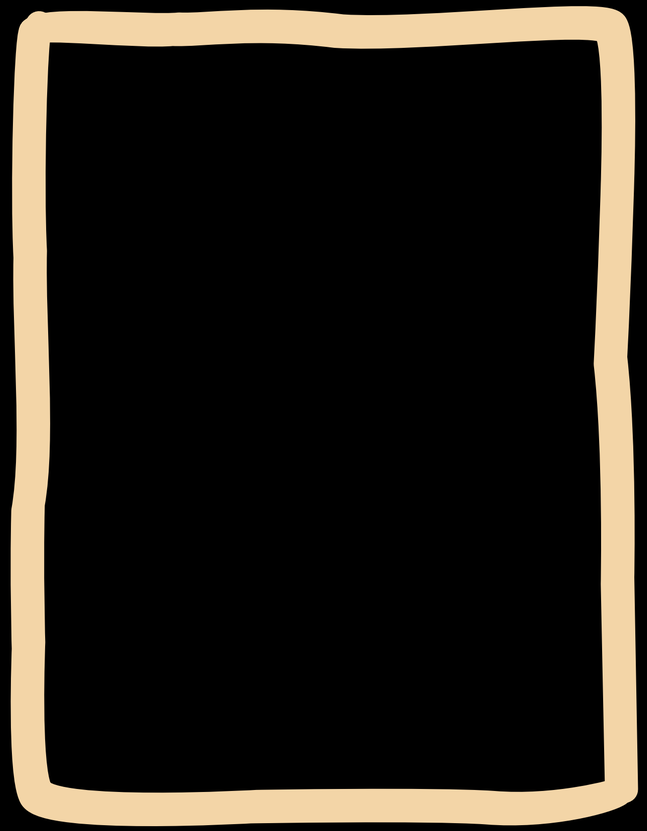


What's an RDD?

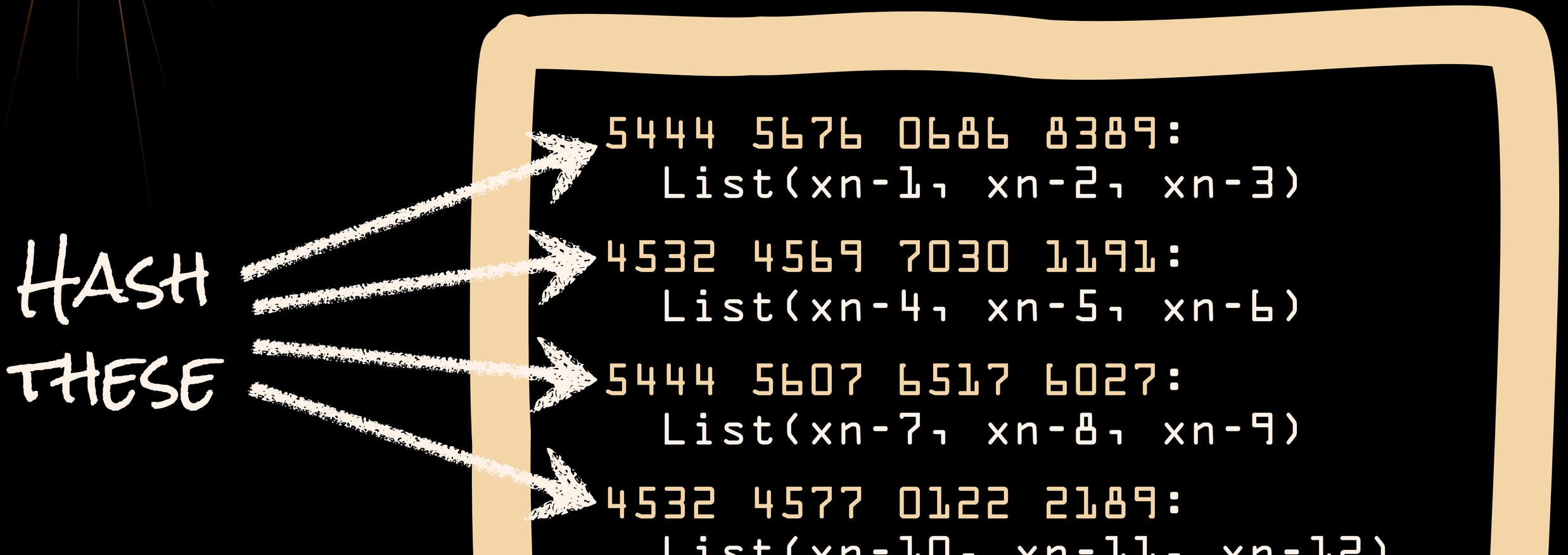
Collection of Things



Distributed, how?

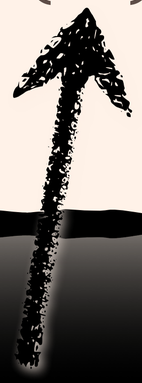


Distributed, how?



```
val conf = new SparkConf(true)
   .set("spark.cassandra.connection.host", “10.0.3.74”)

 val sc = new SparkContext(“spark://127.0.0.1:7077”,
                           “music”, conf)
 val transactions = sc. // cassandra stuff
   .partitionBy(new HashPartitioner(25)).persist()
  }
}
```



Cassandra Connector

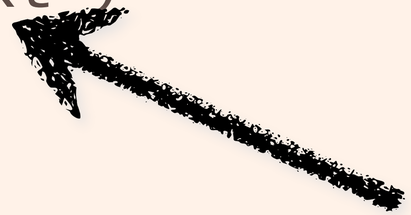
From a Text File

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

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

   val conf = new SparkConf(true)
     .set("spark.cassandra.connection.host", "127.0.0.1")

   val sc = new SparkContext(“spark://127.0.0.1:7077”,
                             “music”, conf)
    val rdd = sc.textFile(“/actual/path/raven.txt”)
 }
}
```



Data Model

```
CREATE TABLE credit_card_transactions(
  card_number VARCHAR,
  transaction_time TIMESTAMP,
  amount DECIMAL(10,2),
  merchant_id VARCHAR,
  country VARCHAR,
  PRIMARY KEY(card_number,
              transaction_time DESC));
```

From a Table

```
val conf = new SparkConf(true)
  .set("spark.cassandra.connection.host", "127.0.0.1")

val sc = new SparkContext(“spark://127.0.0.1:7077”,
                          “music”, conf)
val xctns = sc.cassandraTable(“payments”,
                              “credit_card_transactions")
```

Projection

```
val conf = new SparkConf(true)
  .set("spark.cassandra.connection.host", "127.0.0.1")

val sc = new SparkContext(“spark://127.0.0.1:7077”,
                          “music”, conf)
val xctns = sc.cassandraTable(“payments”,
                              “credit_card_transactions”)
              .select(“card_number”, “country”)
```

Making K / V pairs

```
val conf = new SparkConf(true)
  .set("spark.cassandra.connection.host", "127.0.0.1")

val sc = new SparkContext(“spark://127.0.0.1:7077”,
                          “music”, conf)
val countries = xctns.map(x => (x.getString(“card_number”),
                                x.getString(“country”))
```

Count up Countries

```
val conf = new SparkConf(true)
  .set("spark.cassandra.connection.host", "127.0.0.1")

val sc = new SparkContext(“spark://127.0.0.1:7077”,
                          “music”, conf)
val cCount = xctns.reduce(card, country =>
             // cool analysis redacted!
             )
```

Make k / v pairs

```
val conf = new SparkConf(true)
  .set("spark.cassandra.connection.host", "127.0.0.1")

val sc = new SparkContext(“spark://127.0.0.1:7077”,
                          “music”, conf)
val countries = xctns.map(x => (x._1, x._2))
```

Persisting to a Table

```
val conf = new SparkConf(true)
  .set("spark.cassandra.connection.host", "127.0.0.1")

val sc = new SparkContext(“spark://127.0.0.1:7077”,
                          “music”, conf)
val fraud = cCount.saveToCassandra(“payments”,
                                   “frequent_countries”)
```

Thank you!

@tlberglund