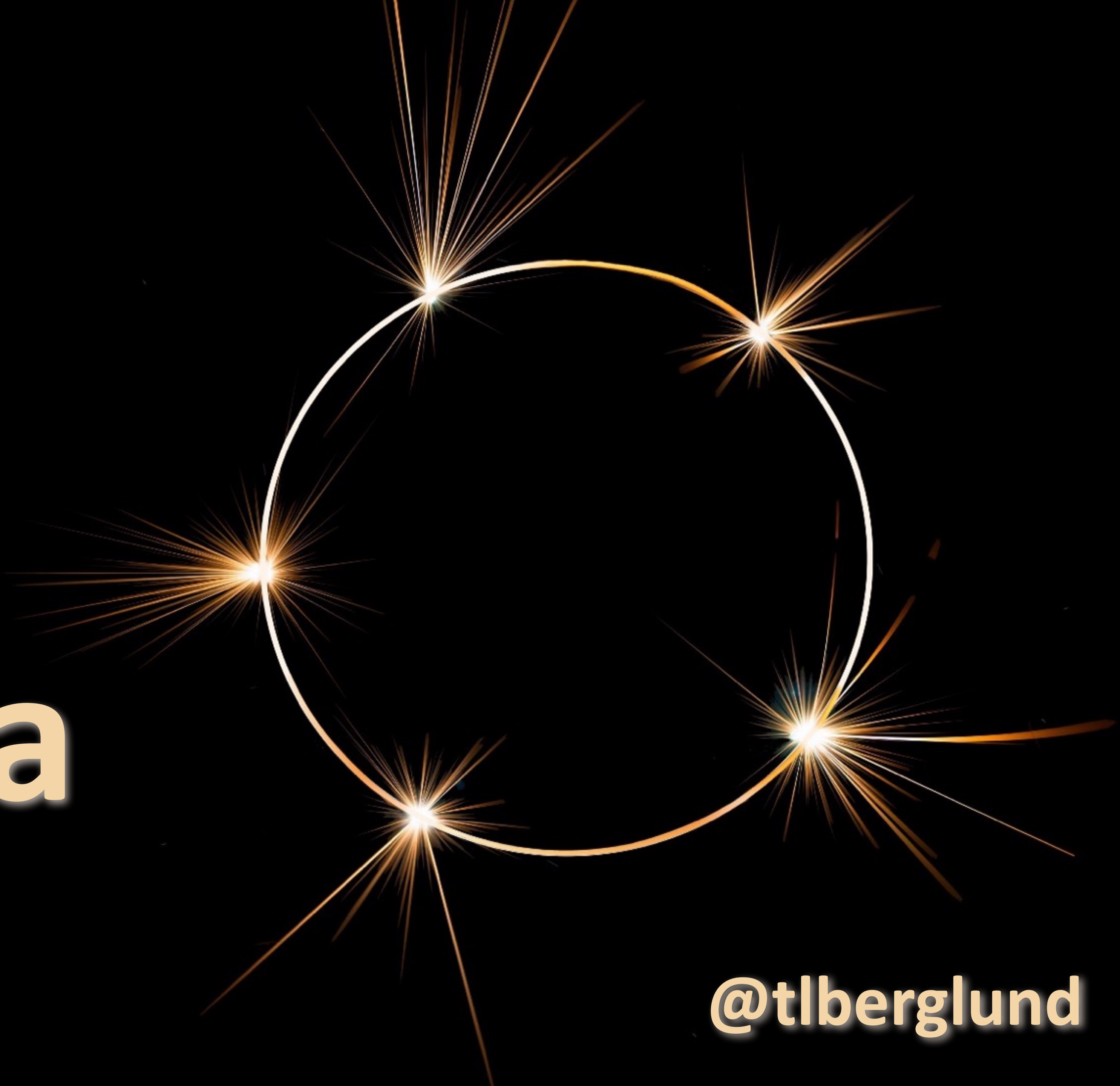




# Cassandra and Spark

@tlberglund

# Cassandra and Spark



@tlberglund

CASSANDRA

# WHEN?

- YOU HAVE TOO MUCH DATA
- IT CHANGES A LOT
- THE SYSTEM IS ALWAYS ON

# WHAT:

- DISTRIBUTED DATABASE
- TRANSACTIONAL/ONLINE
- LOW LATENCY, HIGH VELOCITY

# 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));
```



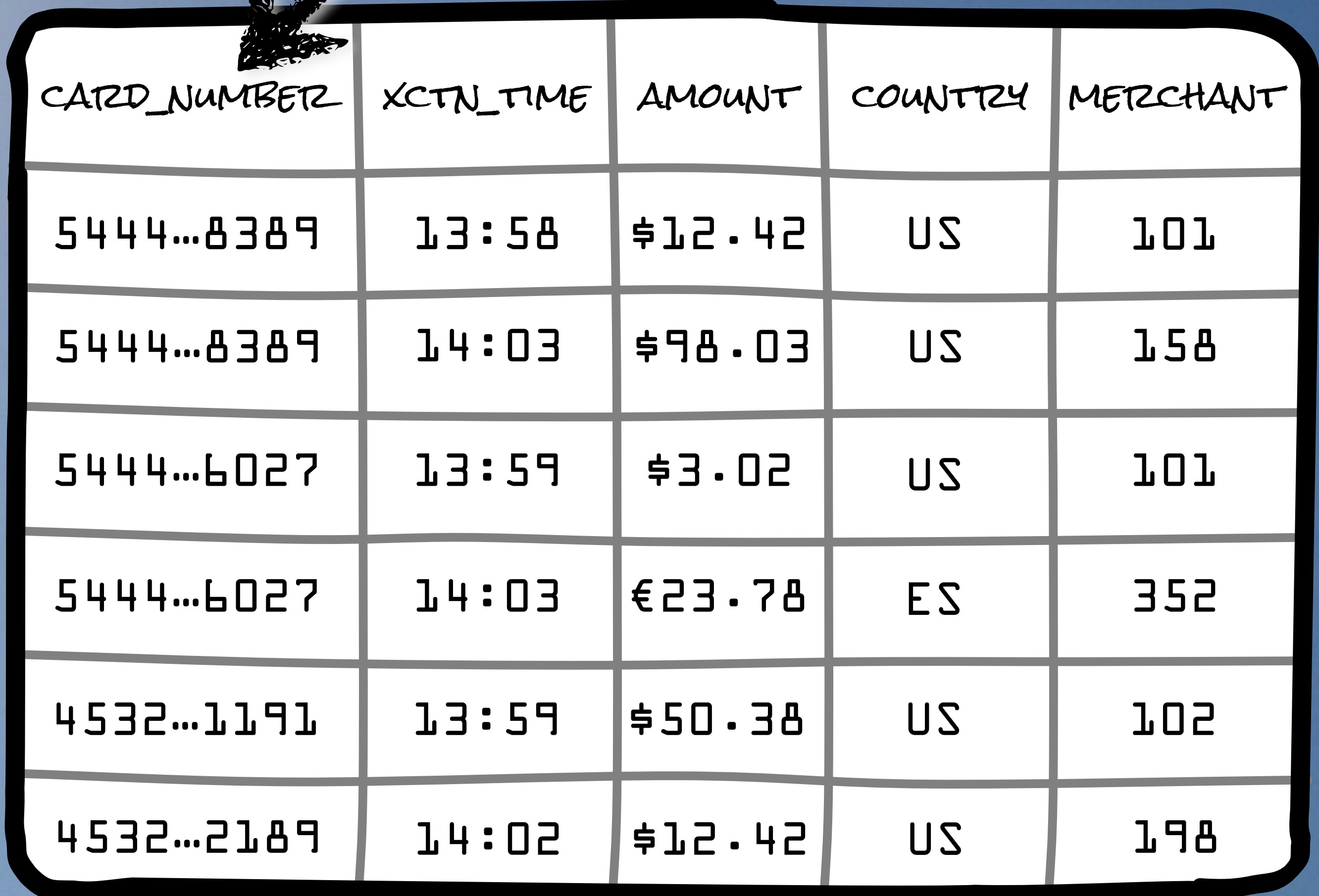
# PRIMARY KEY

```
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

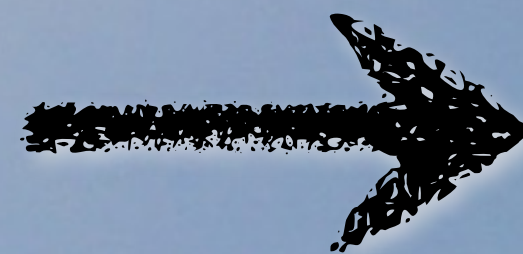
PARTITION



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

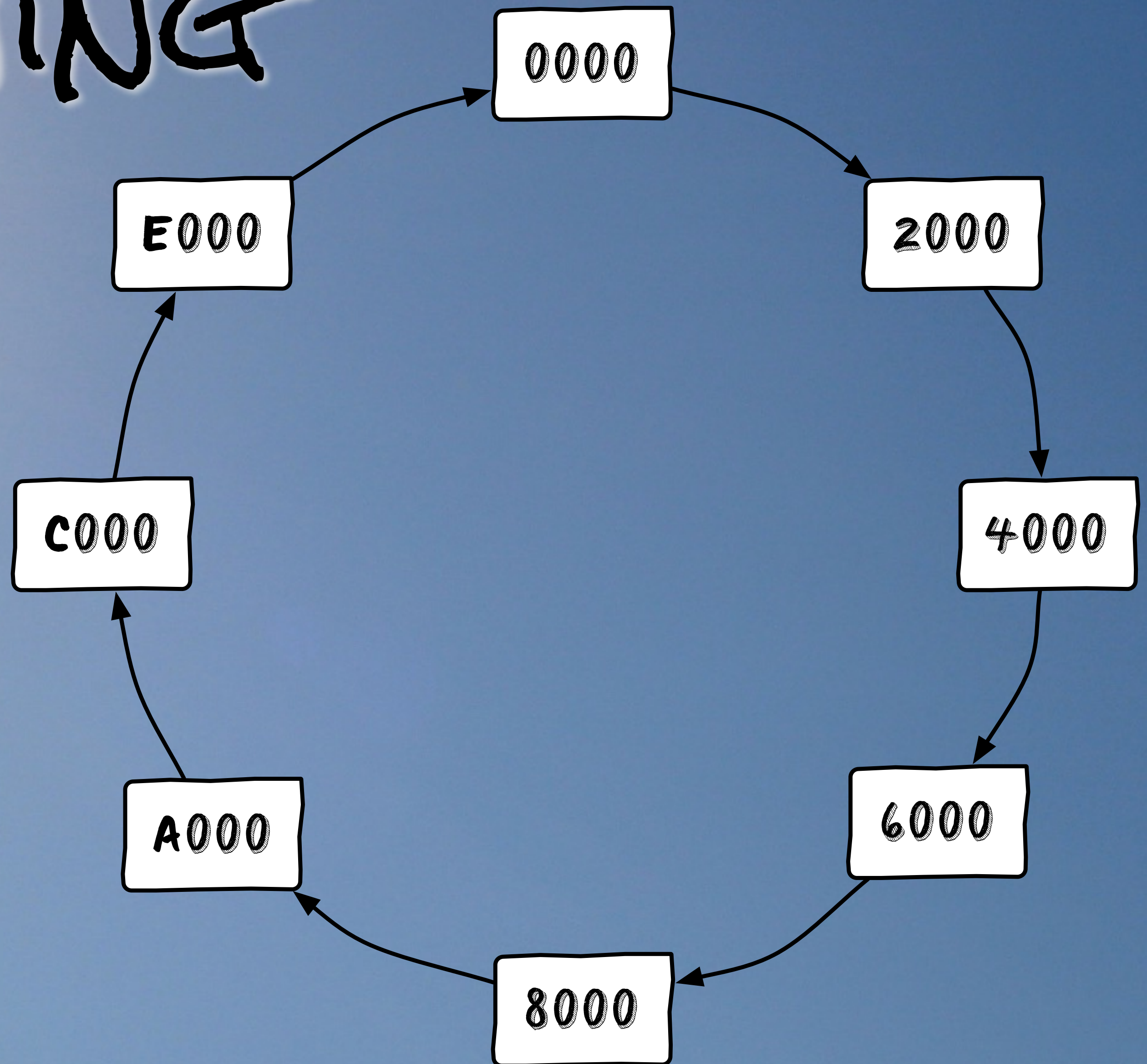
PARTITION



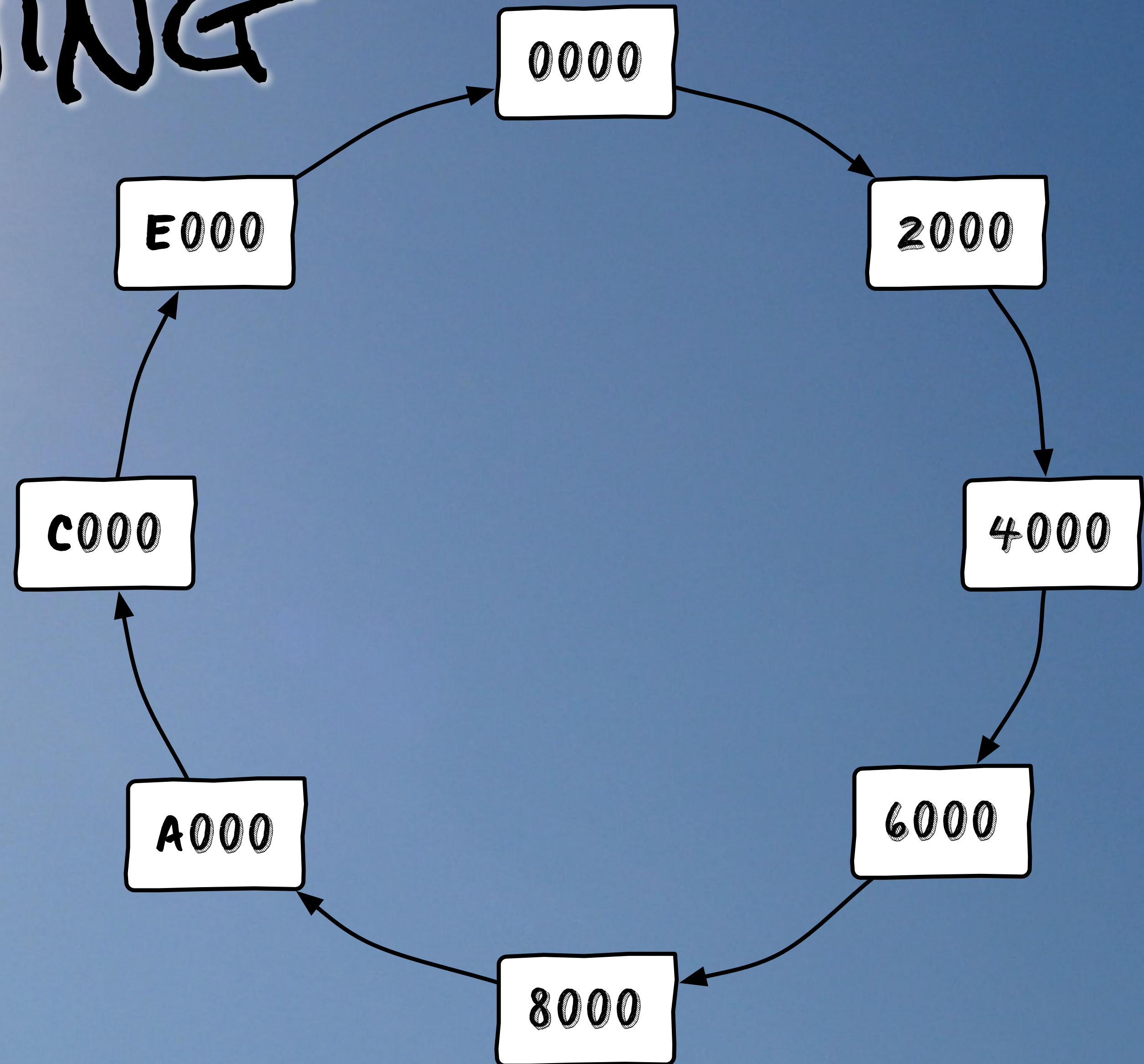
CARD_NUMBER	XCTN_TIME	AMOUNT	COUNTRY	MERCHANT
5444...8389	13:58	\$12.42	US	101
5444...8389	14:03	\$98.03	US	158
5444...6027	13:59	\$3.02	US	101
5444...6027	14:03	€23.78	ES	352
4532...1191	13:59	\$50.38	US	102
4532...2189	14:02	\$12.42	US	198

# PARTITIONING

$F(x) \rightarrow 5D93$   
 $\uparrow$   
5444...8389  
14:03 \$98.03 US 158



# PARTITIONING



5D93



5444...8389

14:03

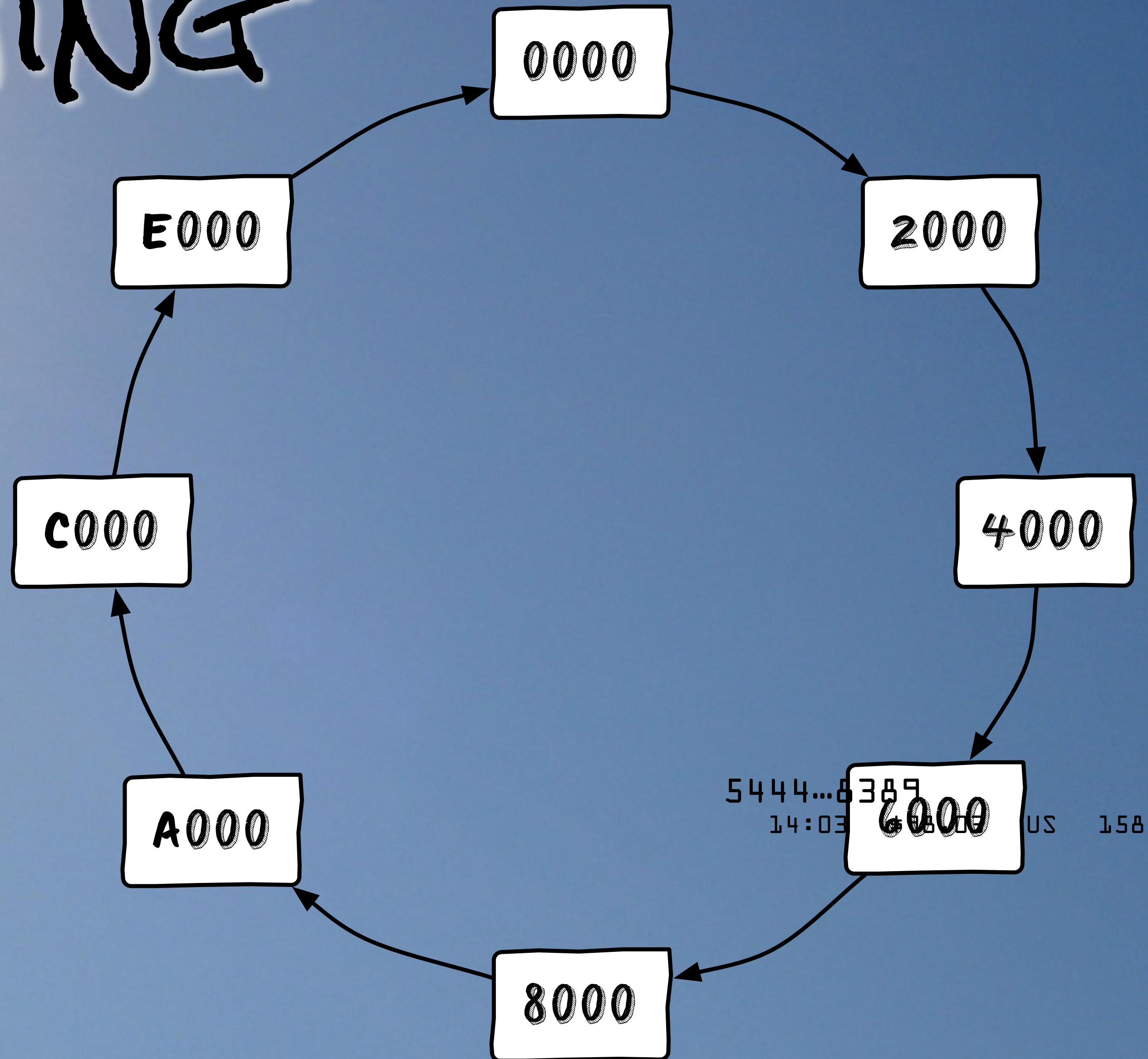
\$98.03

US

158

# PARTITIONING

$F(x) \rightarrow 5D93$   
 $\uparrow$   
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

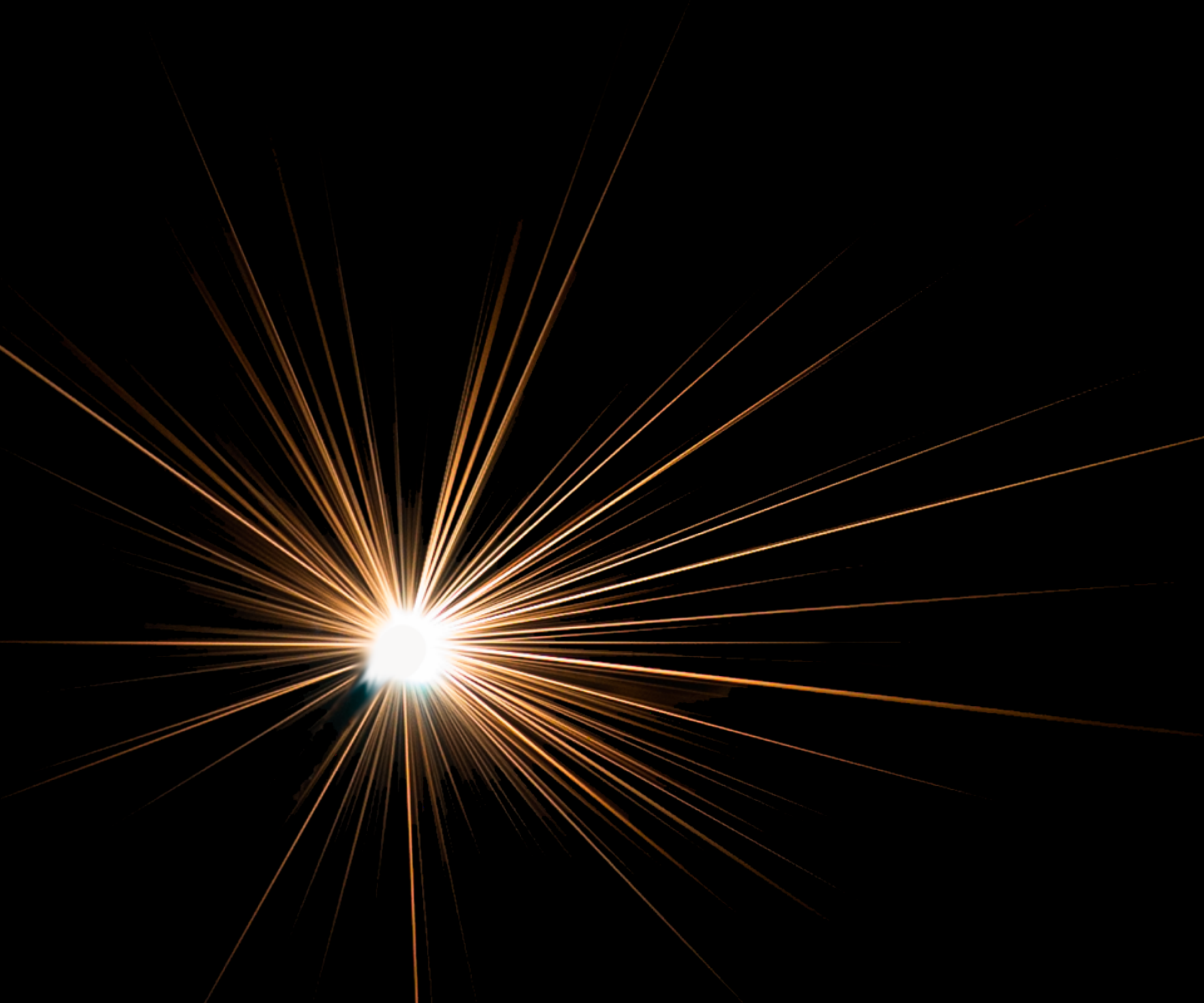
# PARTITIONS

- 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 MAP REDUCE~~

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

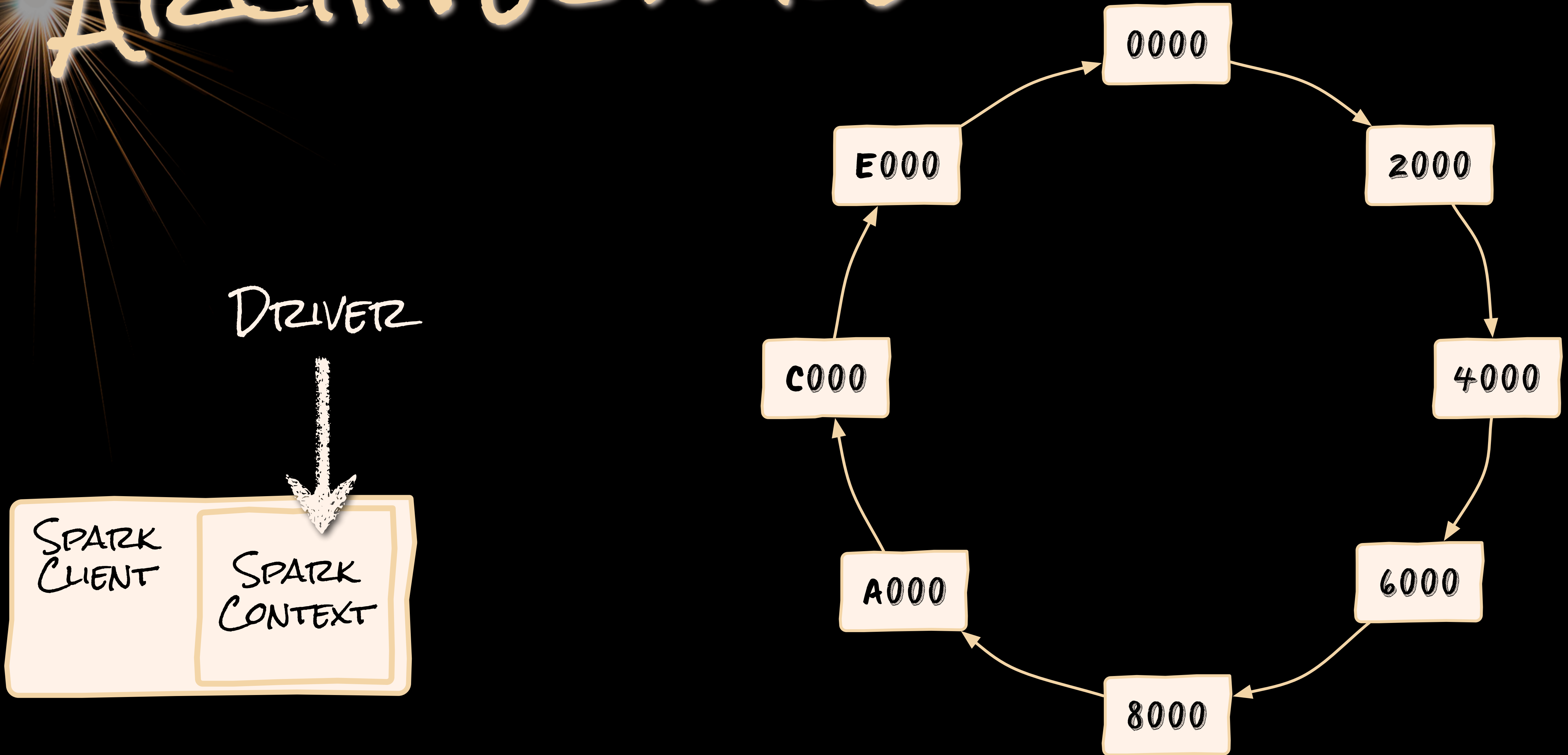




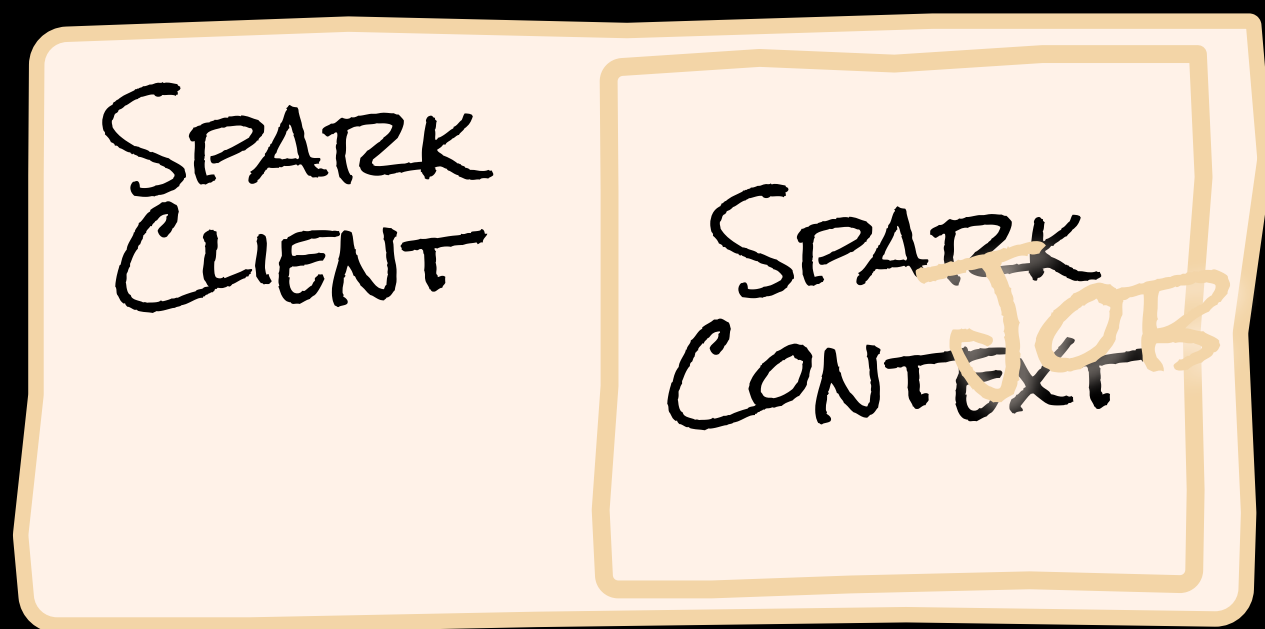
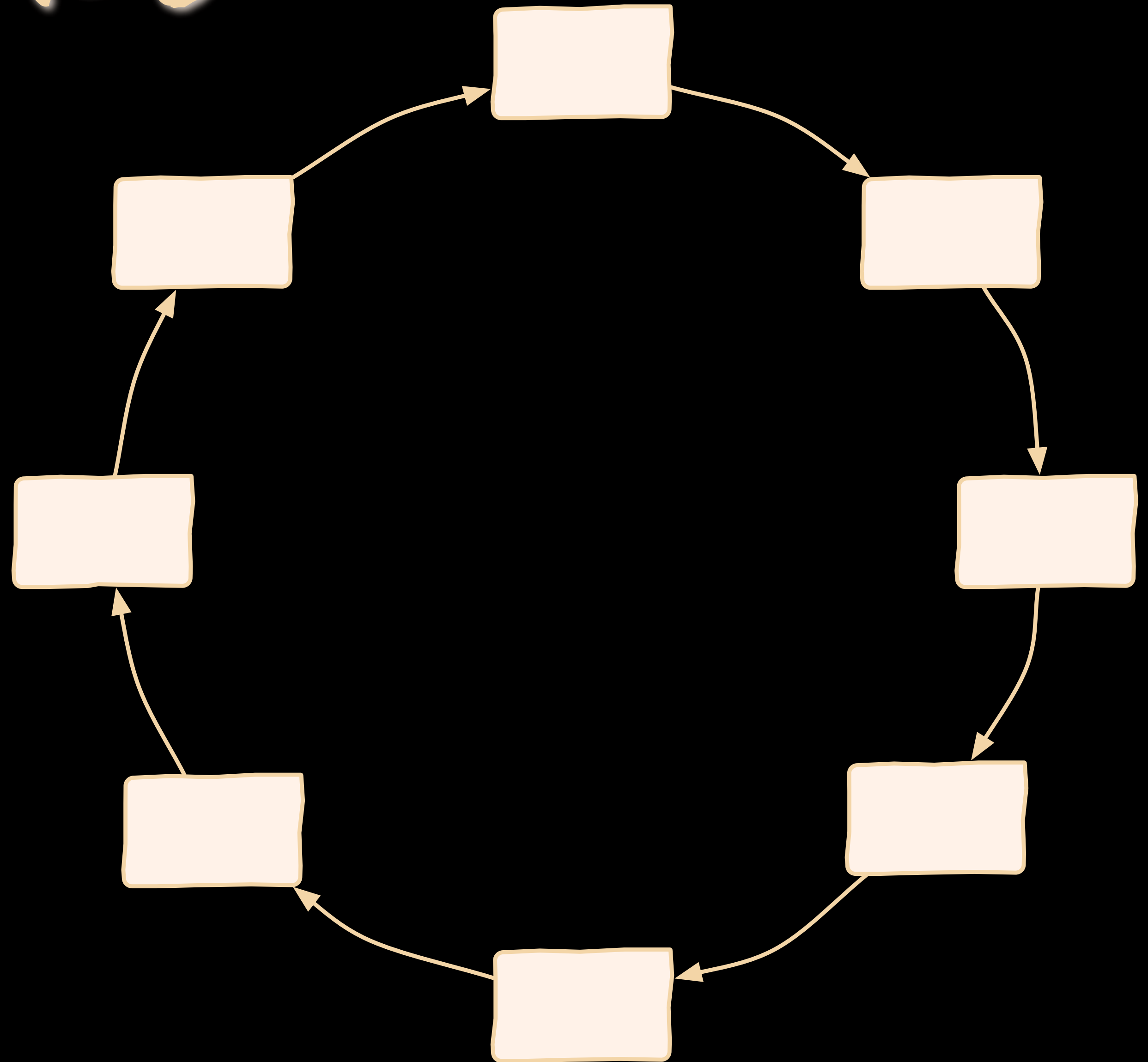
PARADIGM :

DISTRIBUTE ALL OF THIS  
STORAGE AND COMPUTATION

# ARCHITECTURE



# ARCHITECTURE



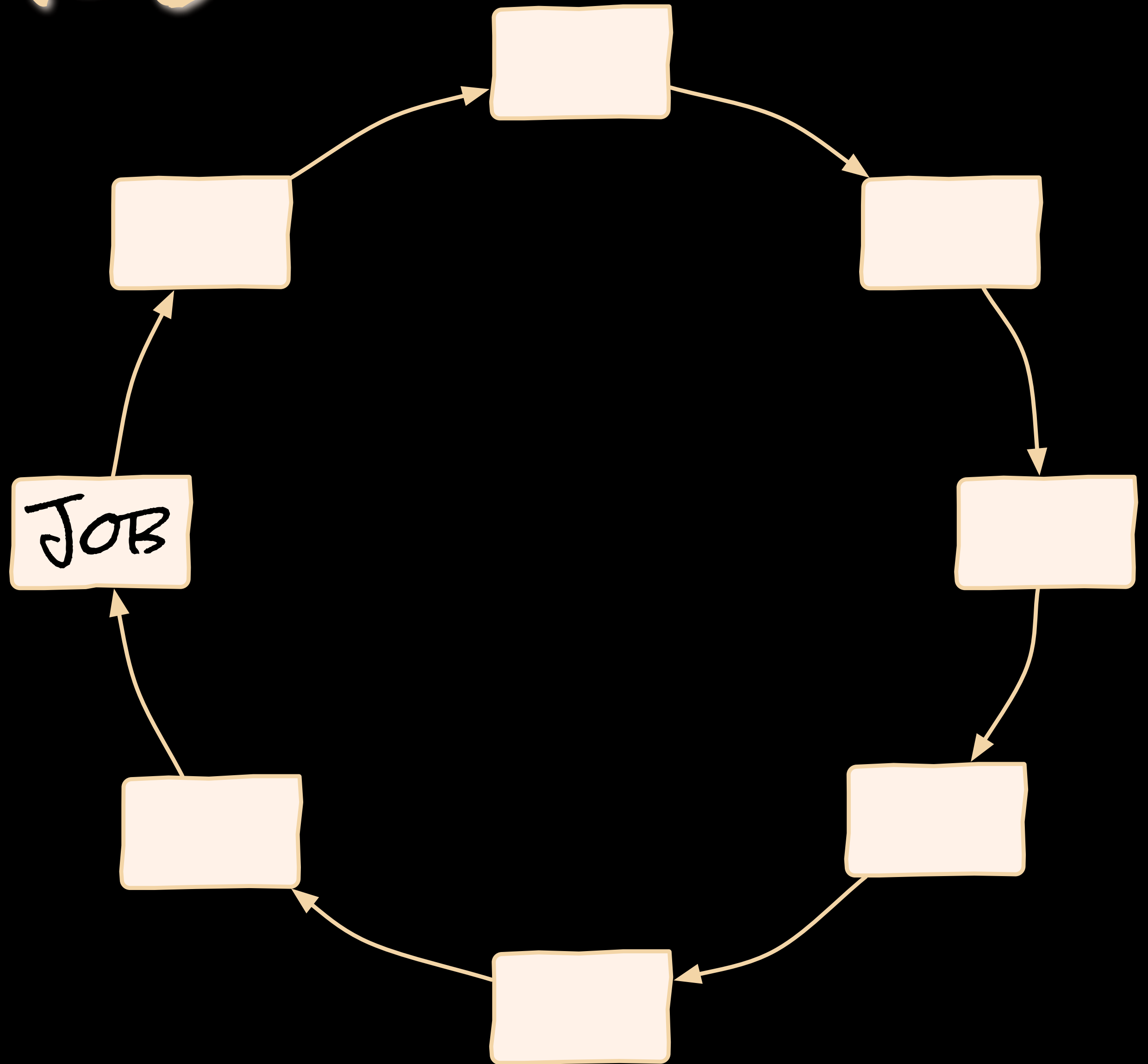
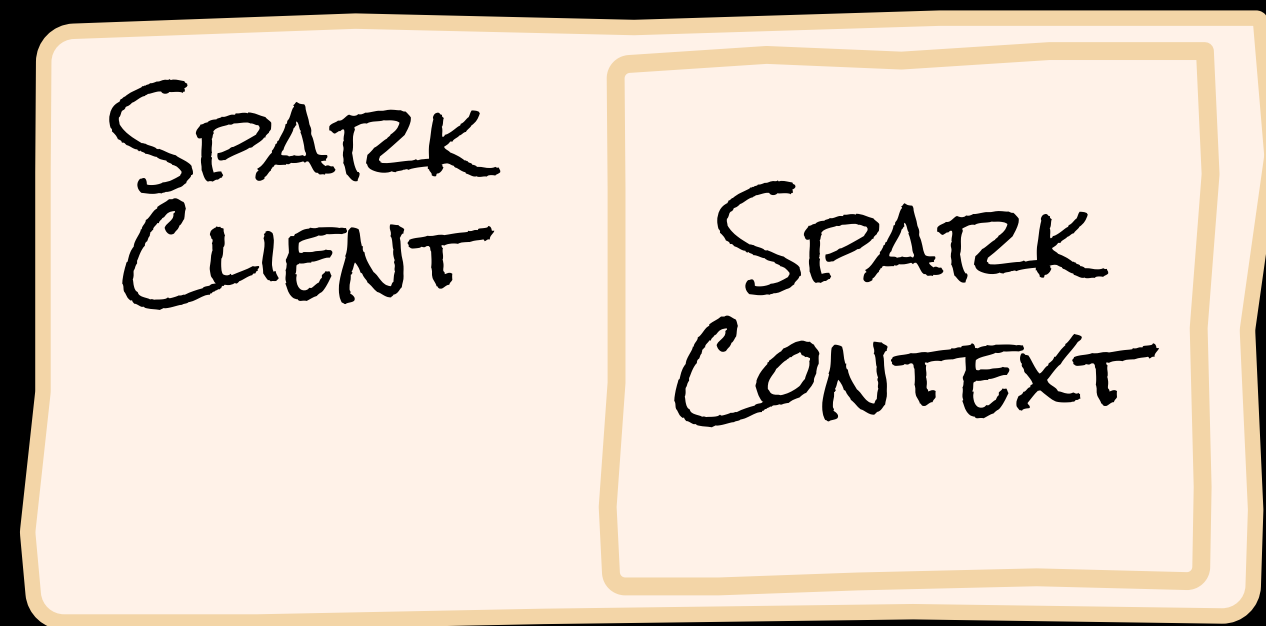
# ARCHITECTURE

CLUSTER  
MANAGER

JOB

SPARK  
CLIENT

SPARK  
CONTEXT



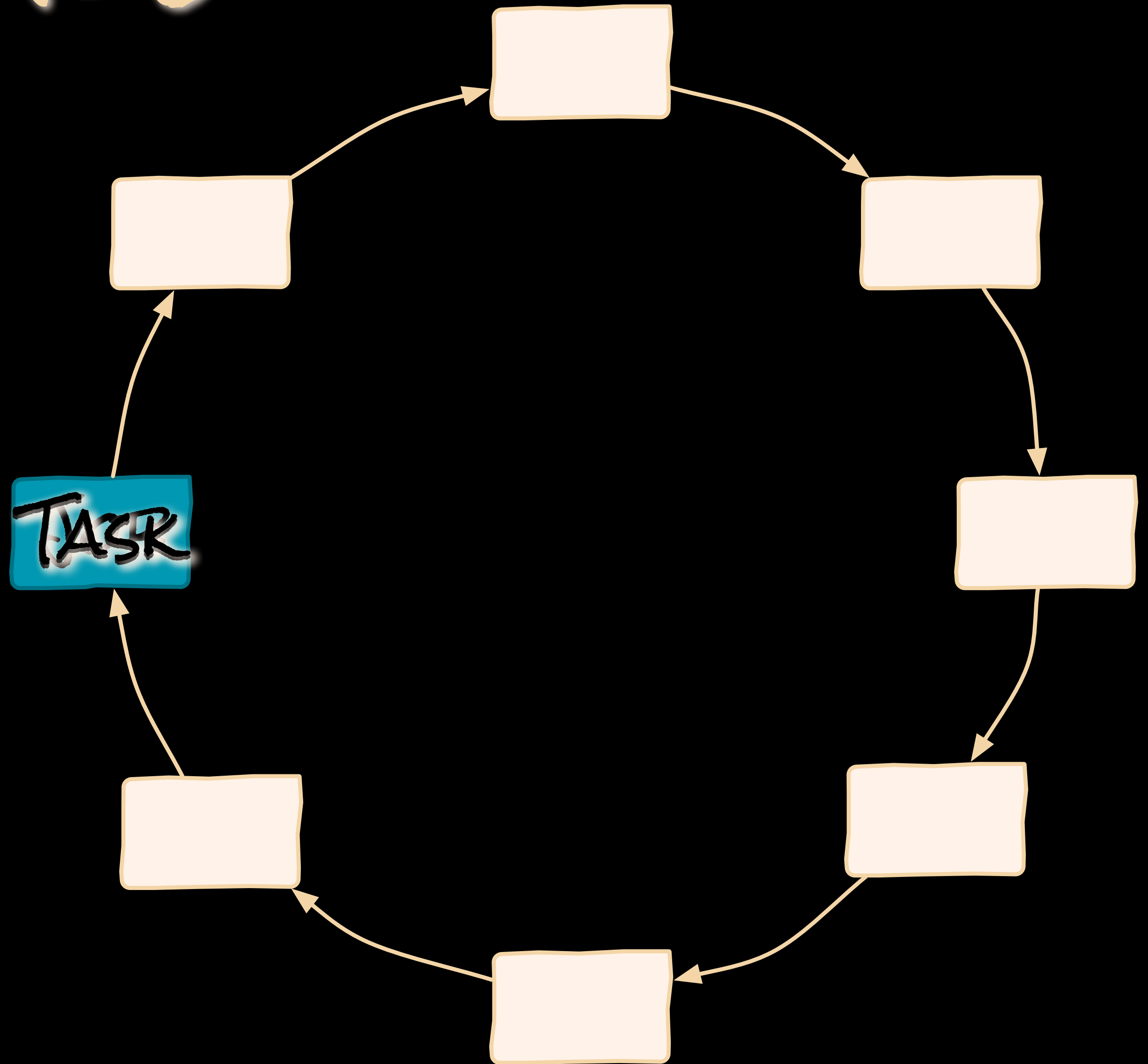
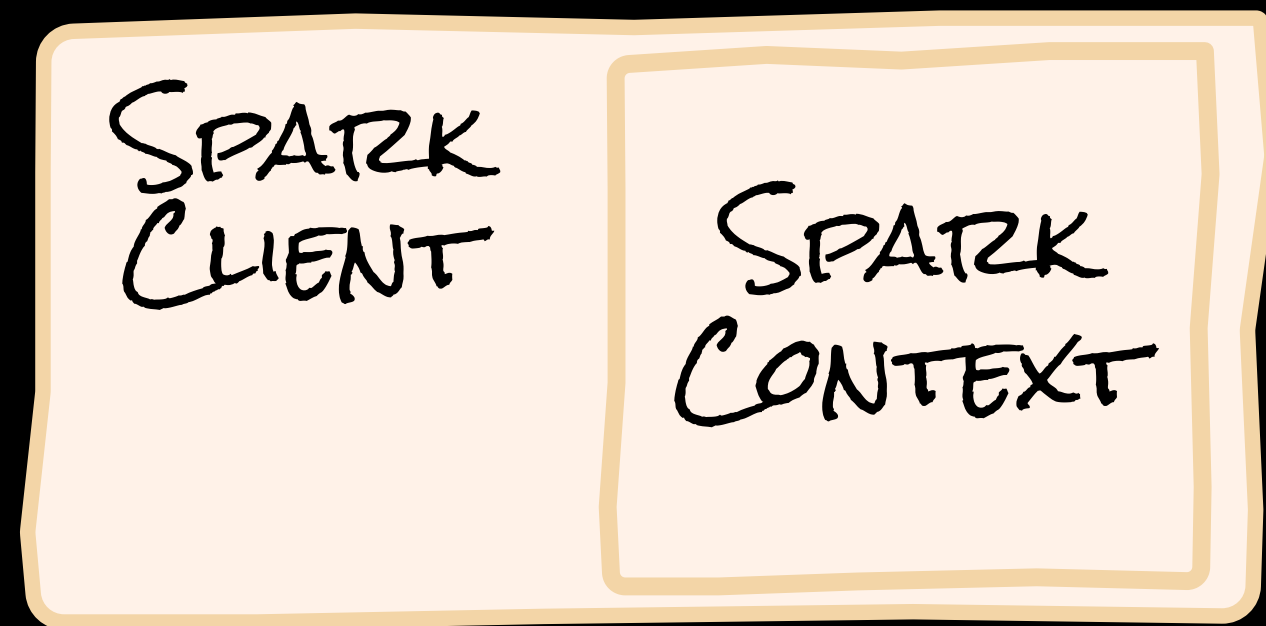
# ARCHITECTURE

CLUSTER  
MANAGER

TASK

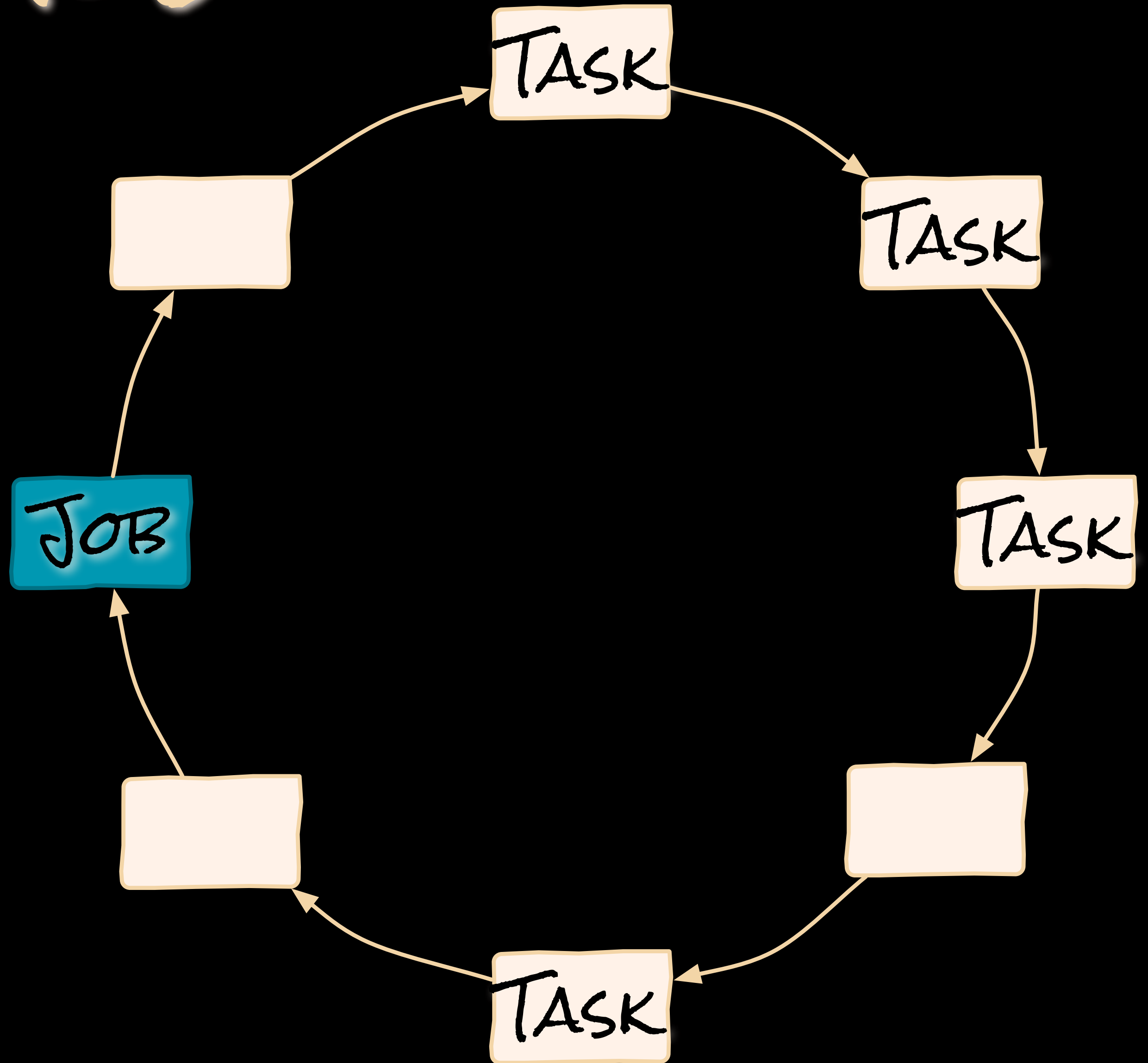
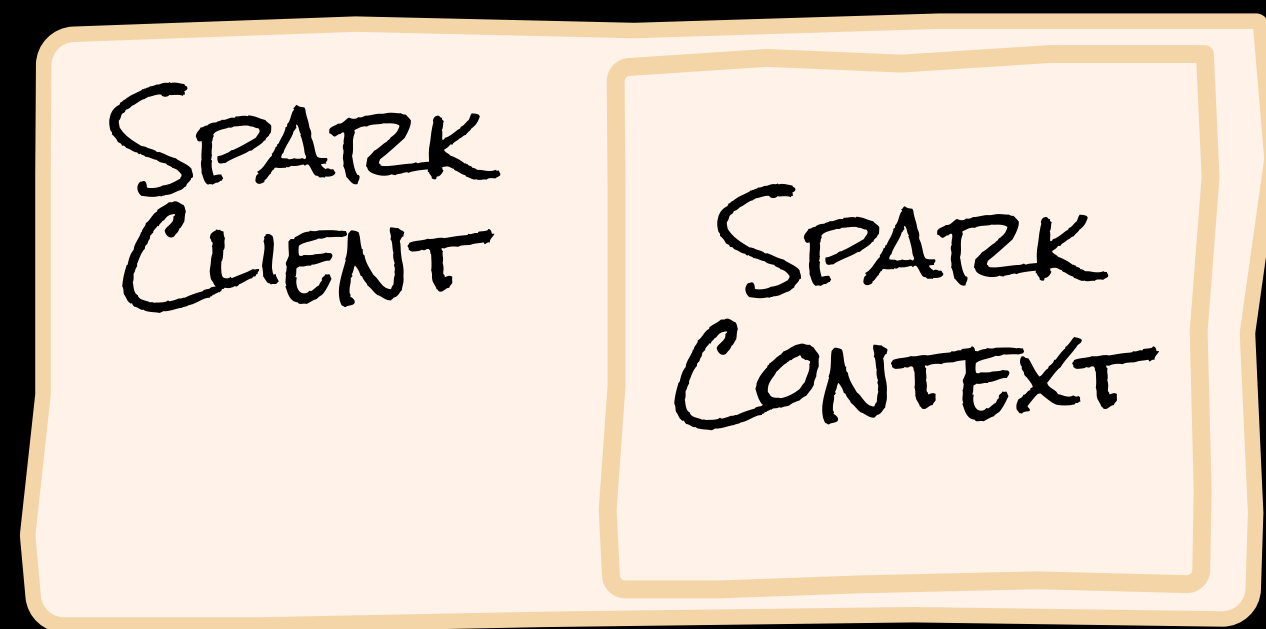
SPARK  
CLIENT

SPARK  
CONTEXT

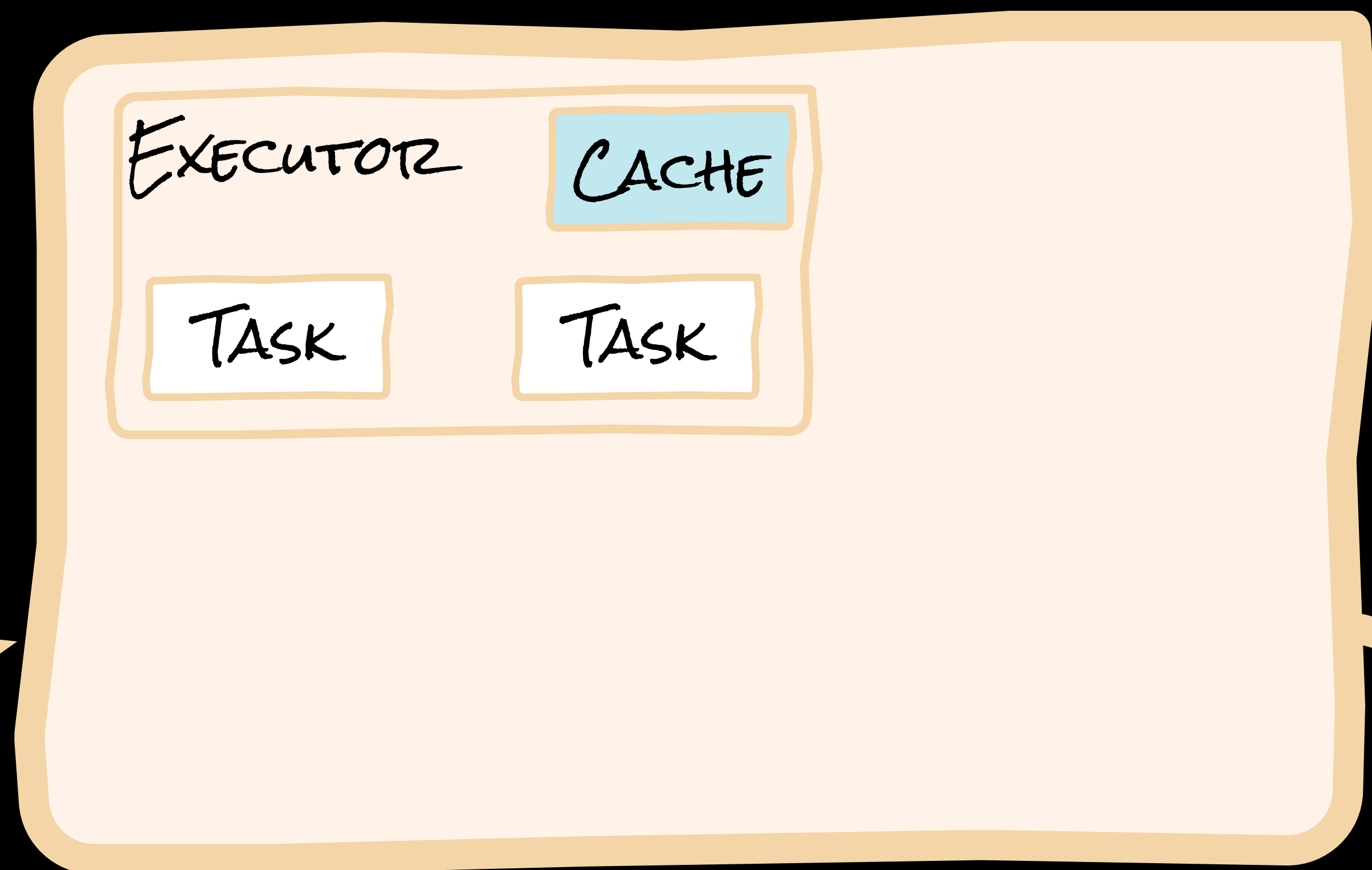


# ARCHITECTURE

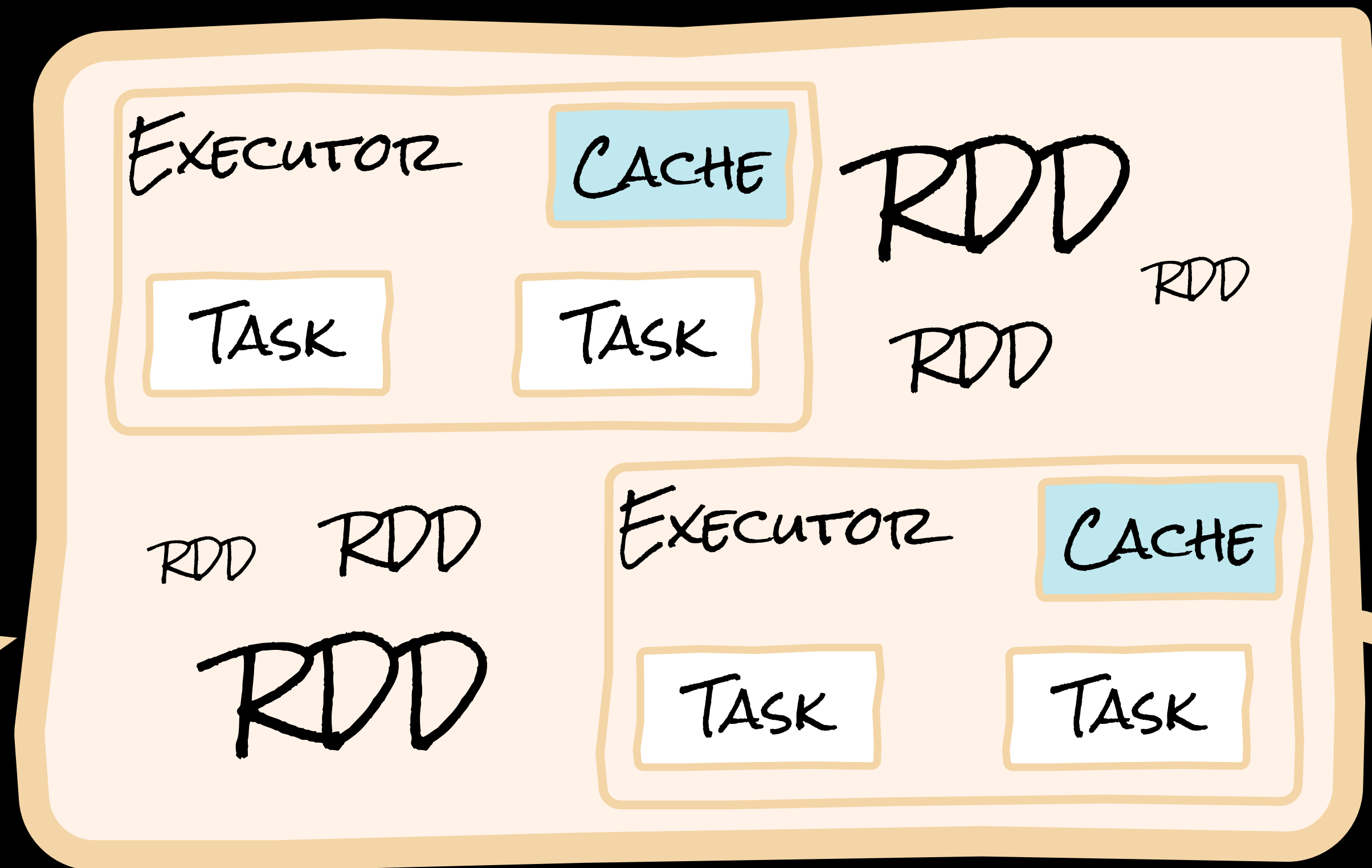
CLUSTER  
MANAGER



# WORKER NODE

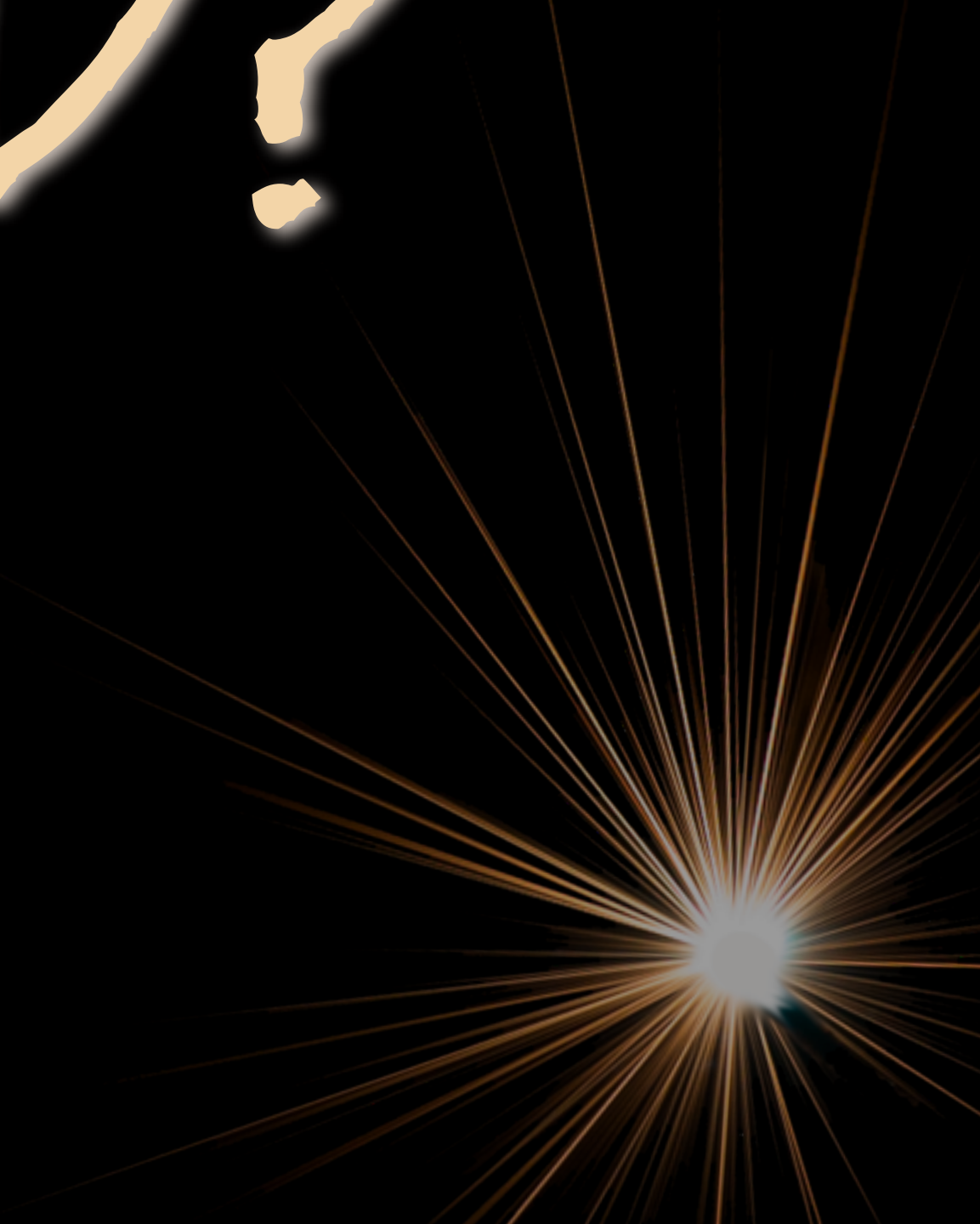


# WORKER NODE





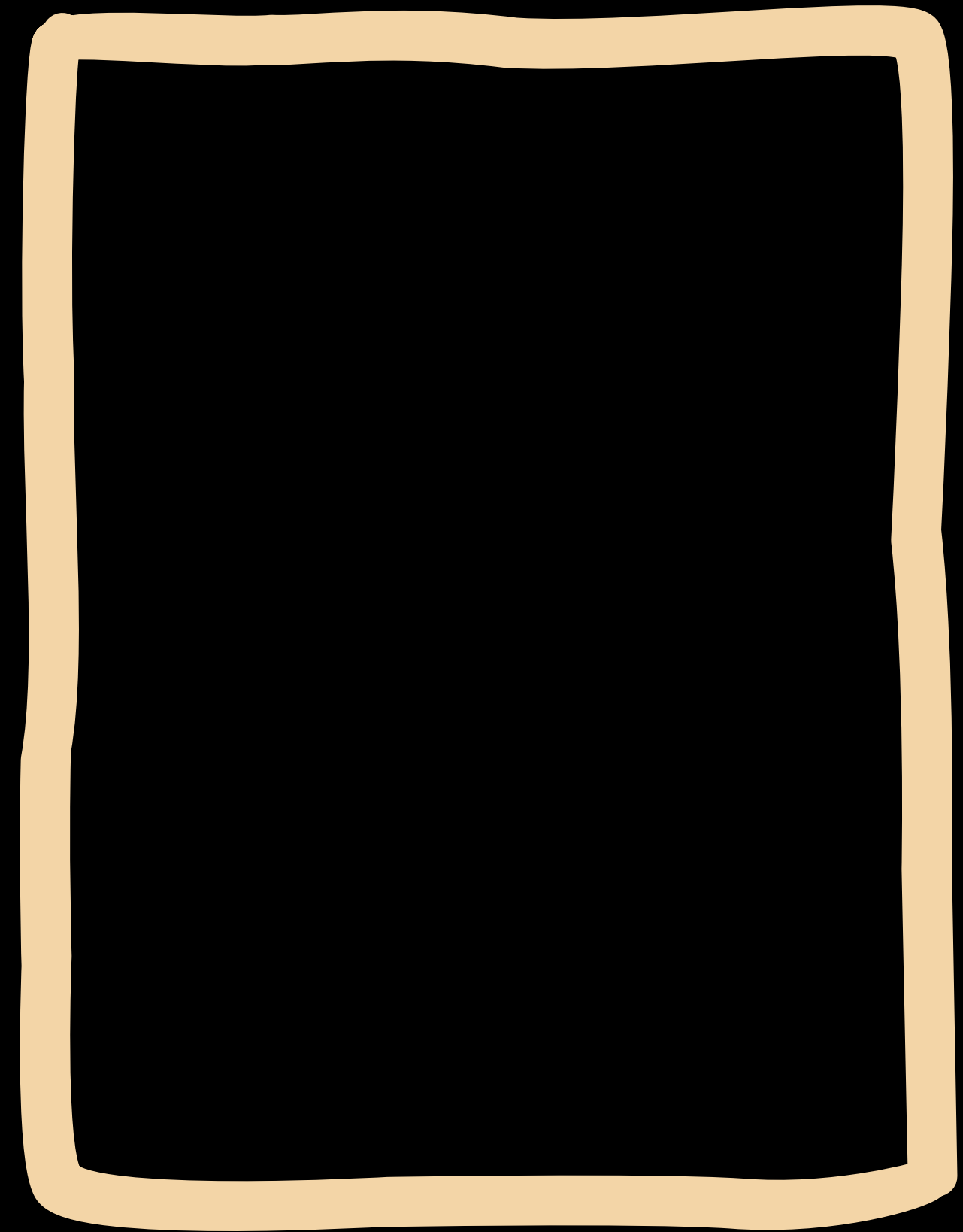
WHAT'S AN RDD?





# WHAT'S AN RDD?

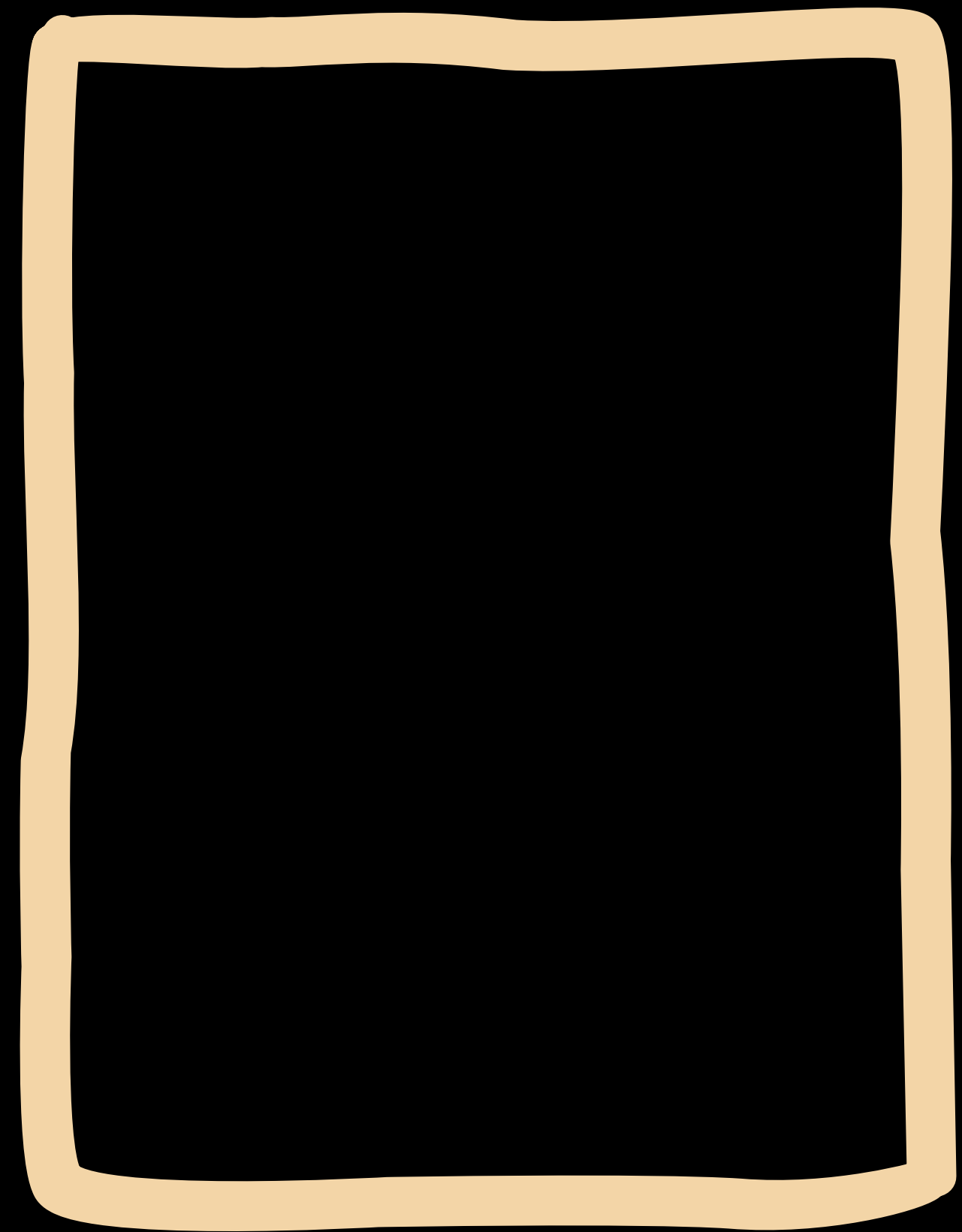
THIS IS





# WHAT'S AN RDD?

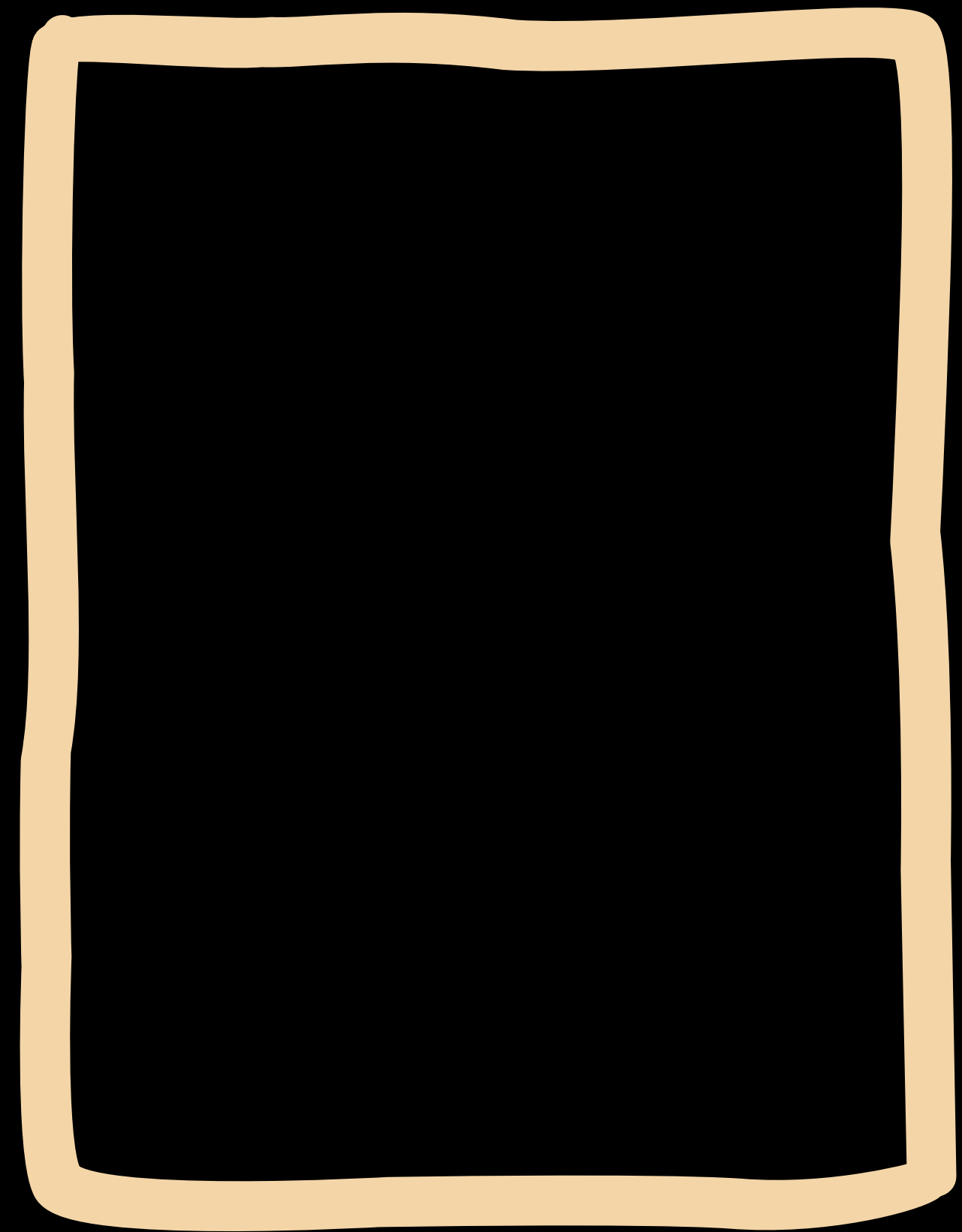
BIGGER THAN A  
COMPUTER





# WHAT'S AN RDD?

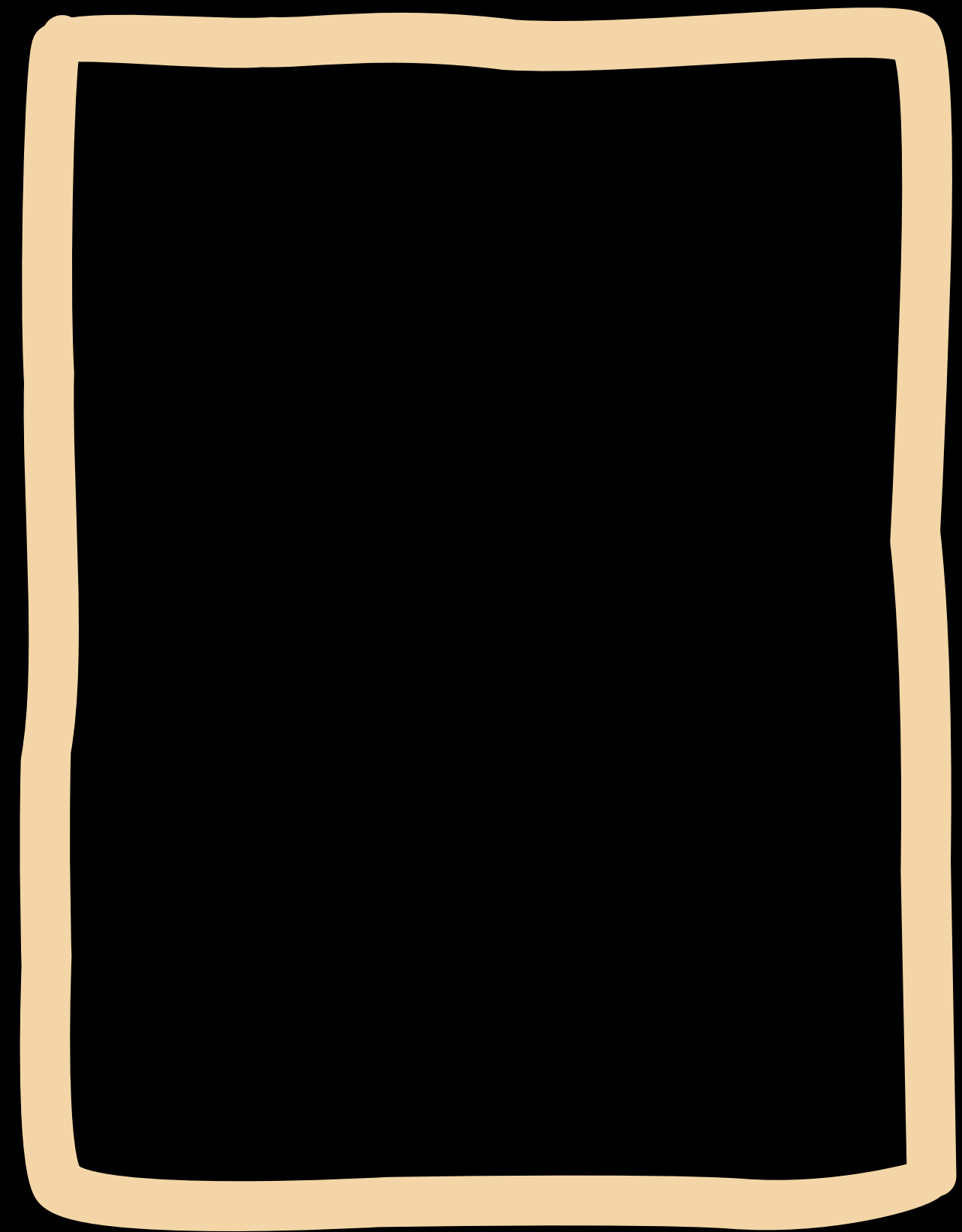
READ FROM AN  
INPUT SOURCE?





# WHAT'S AN RDD?

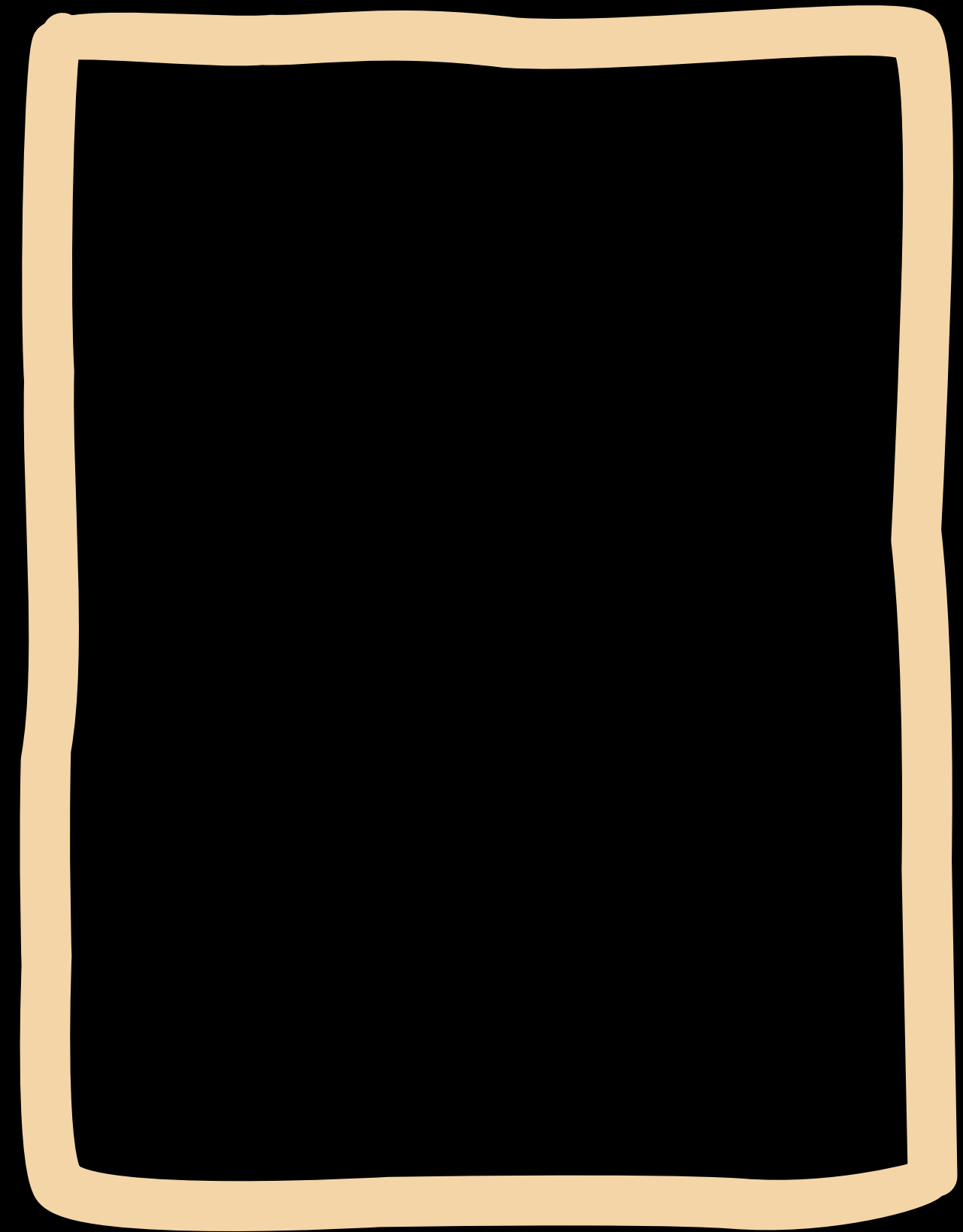
THE OUTPUT OF  
A PURE  
FUNCTION?





# WHAT'S AN RDD?

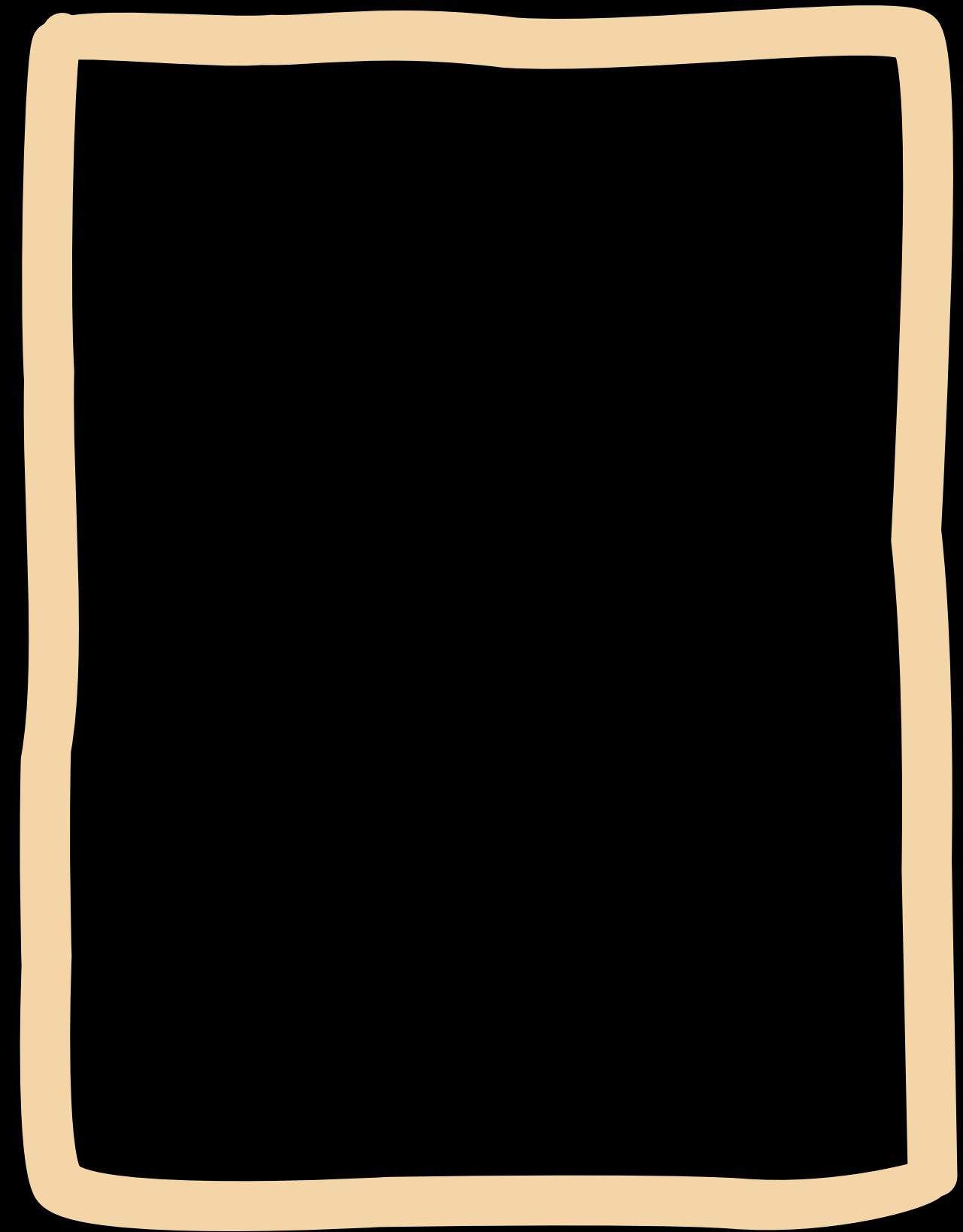
IMMUTABLE





# WHAT'S AN RDD?

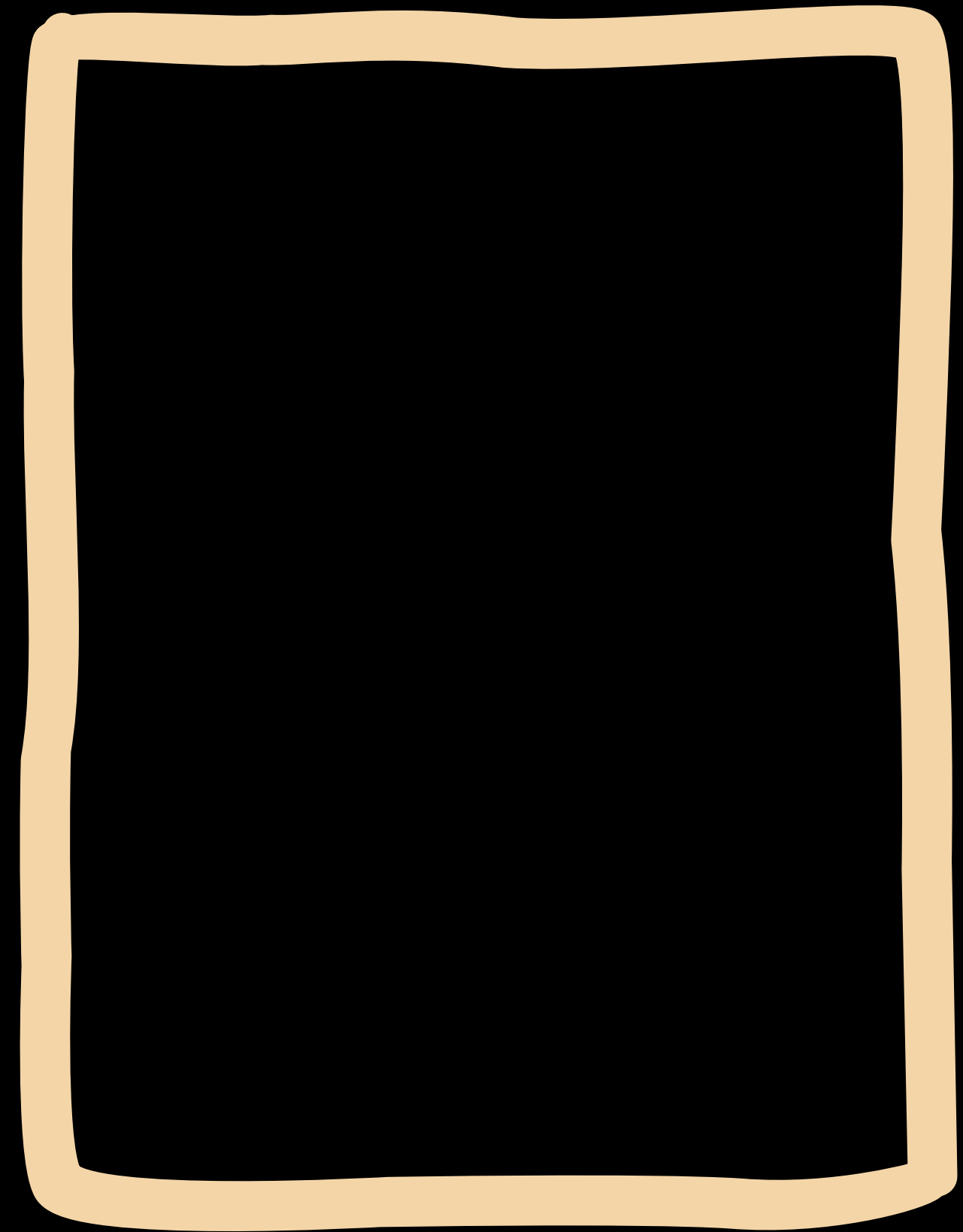
TYPED





# WHAT'S AN RDD?

ORDERED

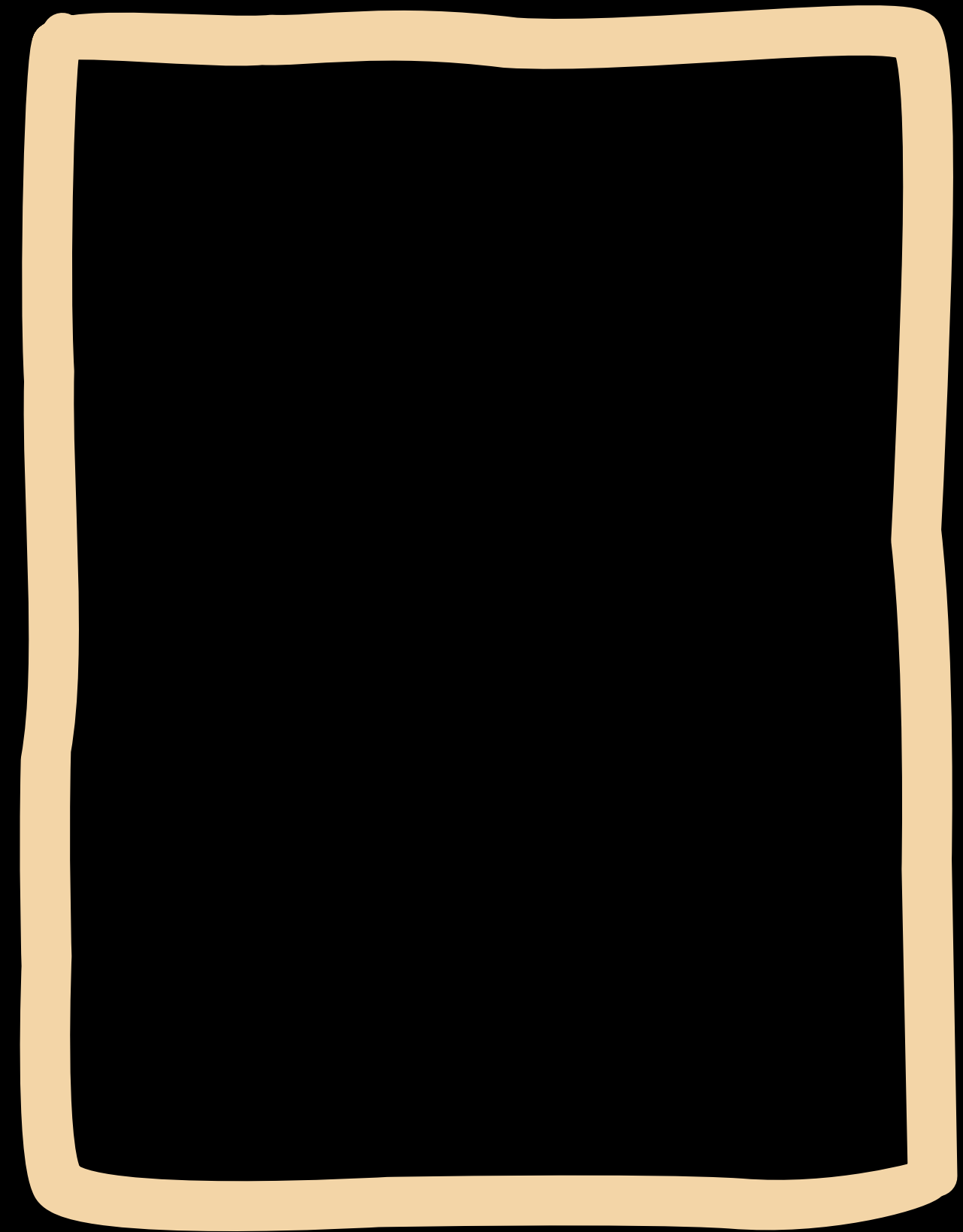






# WHAT'S AN RDD?

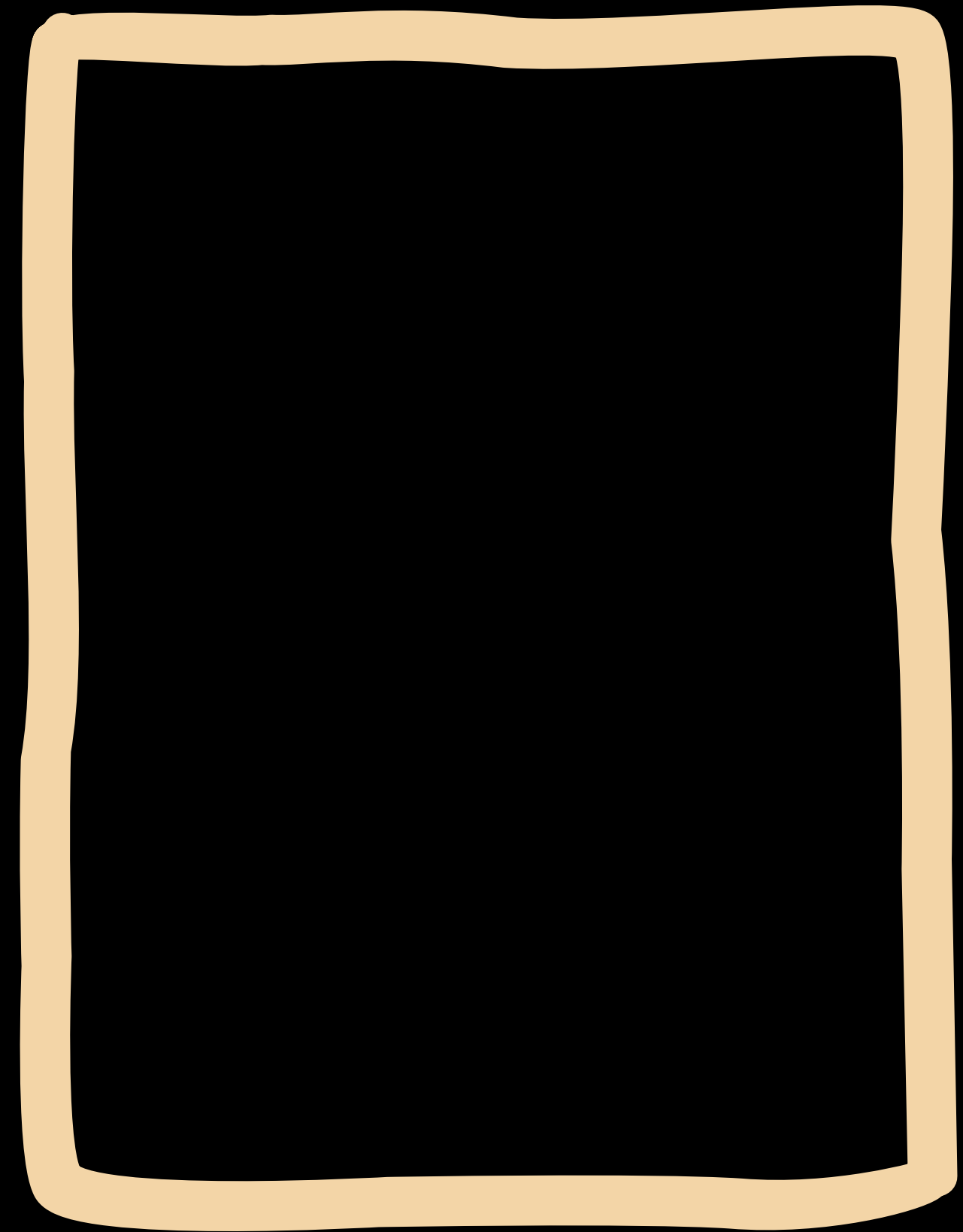
FUNCTIONS  
COMPRIZE A  
GRAPH





# WHAT'S AN RDD?

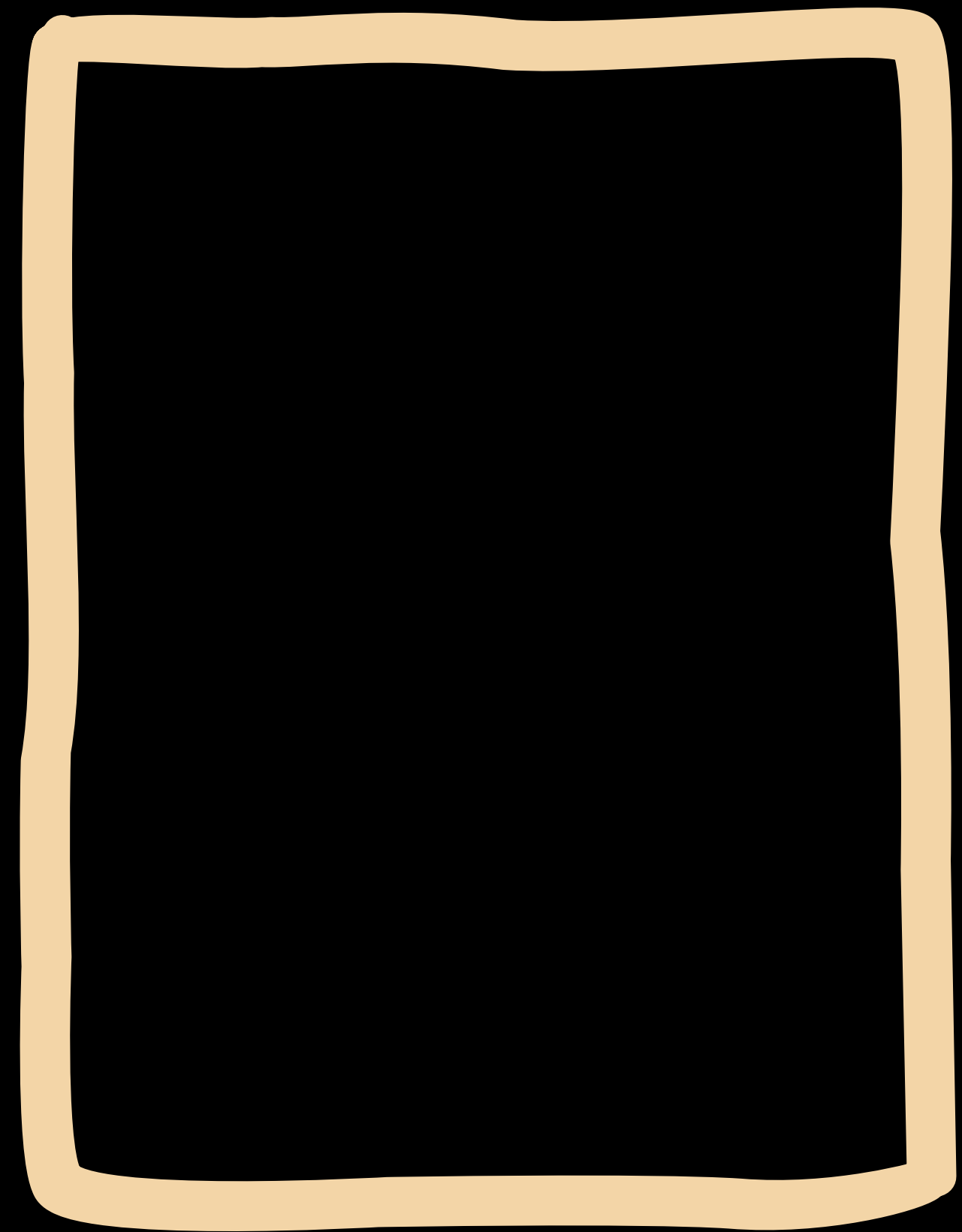
LAZILY  
EVALUATED





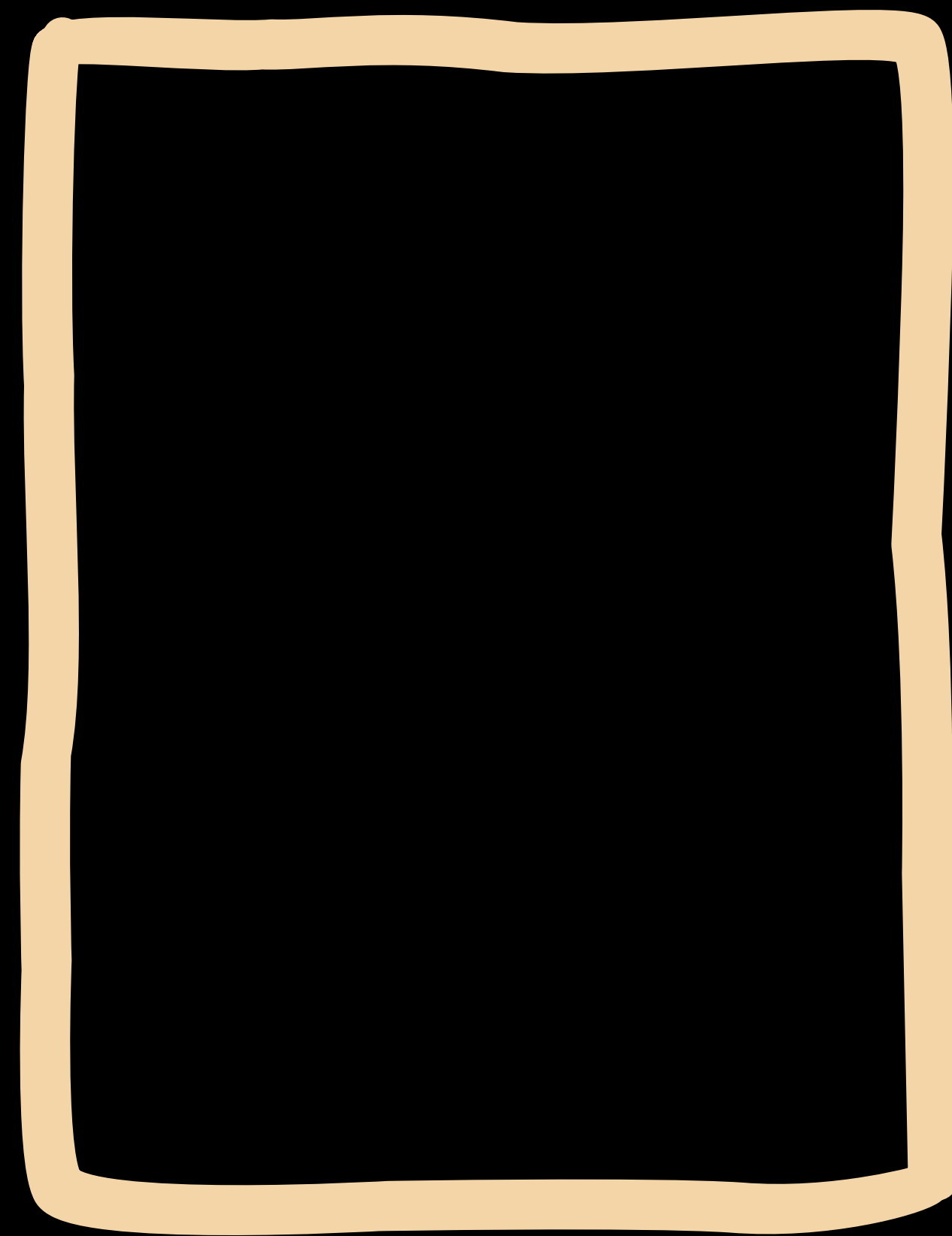
# WHAT'S AN RDD?

COLLECTION OF  
THINGS





DISTRIBUTED, HOW?



# DISTRIBUTED, HOW?

HASH  
THESE

5444 5676 0686 8389:

List(xn-1, xn-2, xn-3)

4532 4569 7030 1191:

List(xn-4, xn-5, xn-6)

5444 5607 6517 6027:

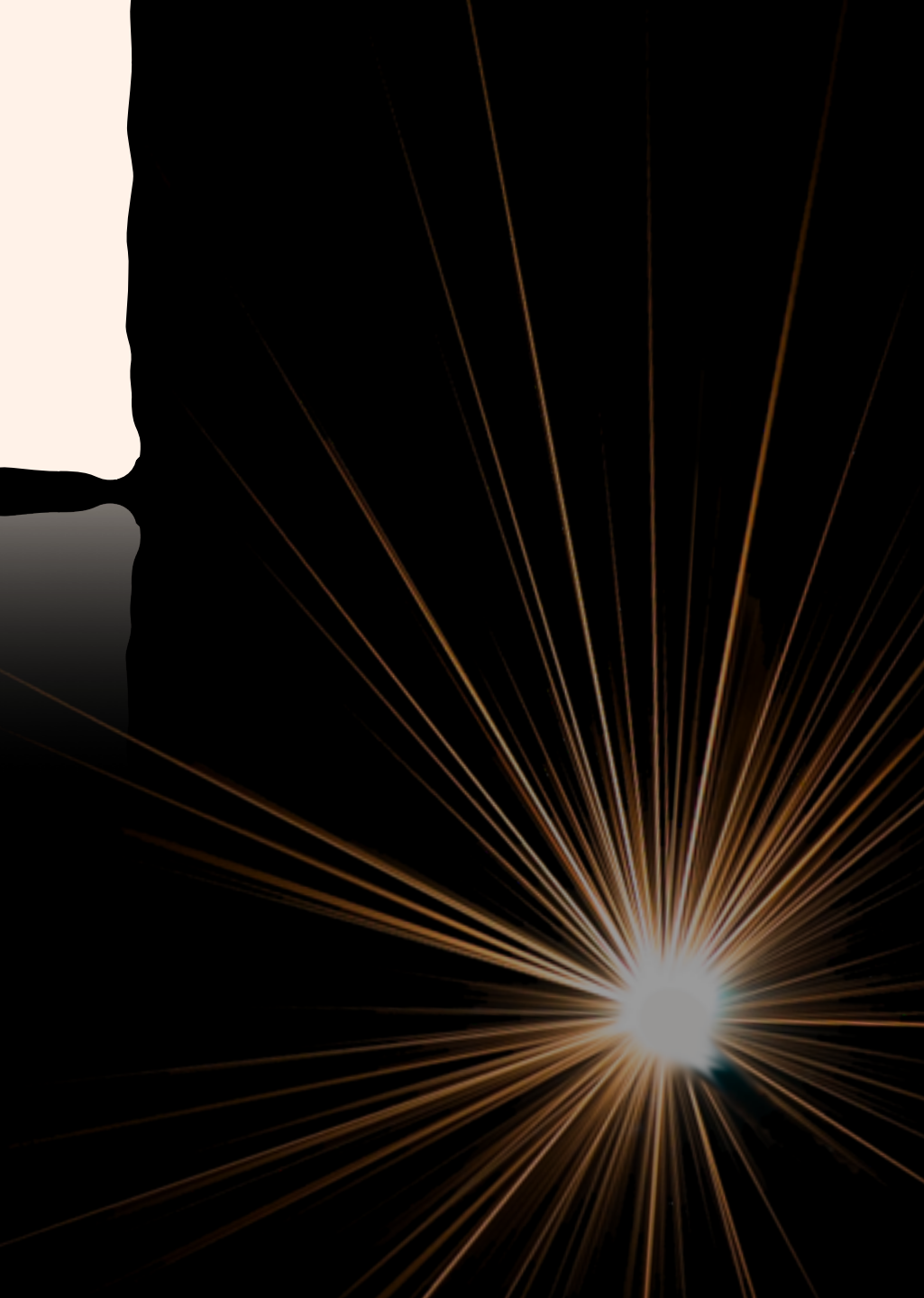
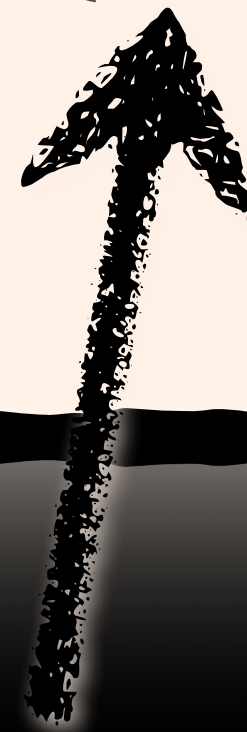
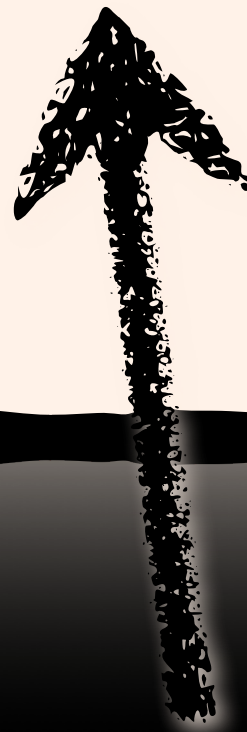
List(xn-7, xn-8, xn-9)

4532 4577 0122 2189:

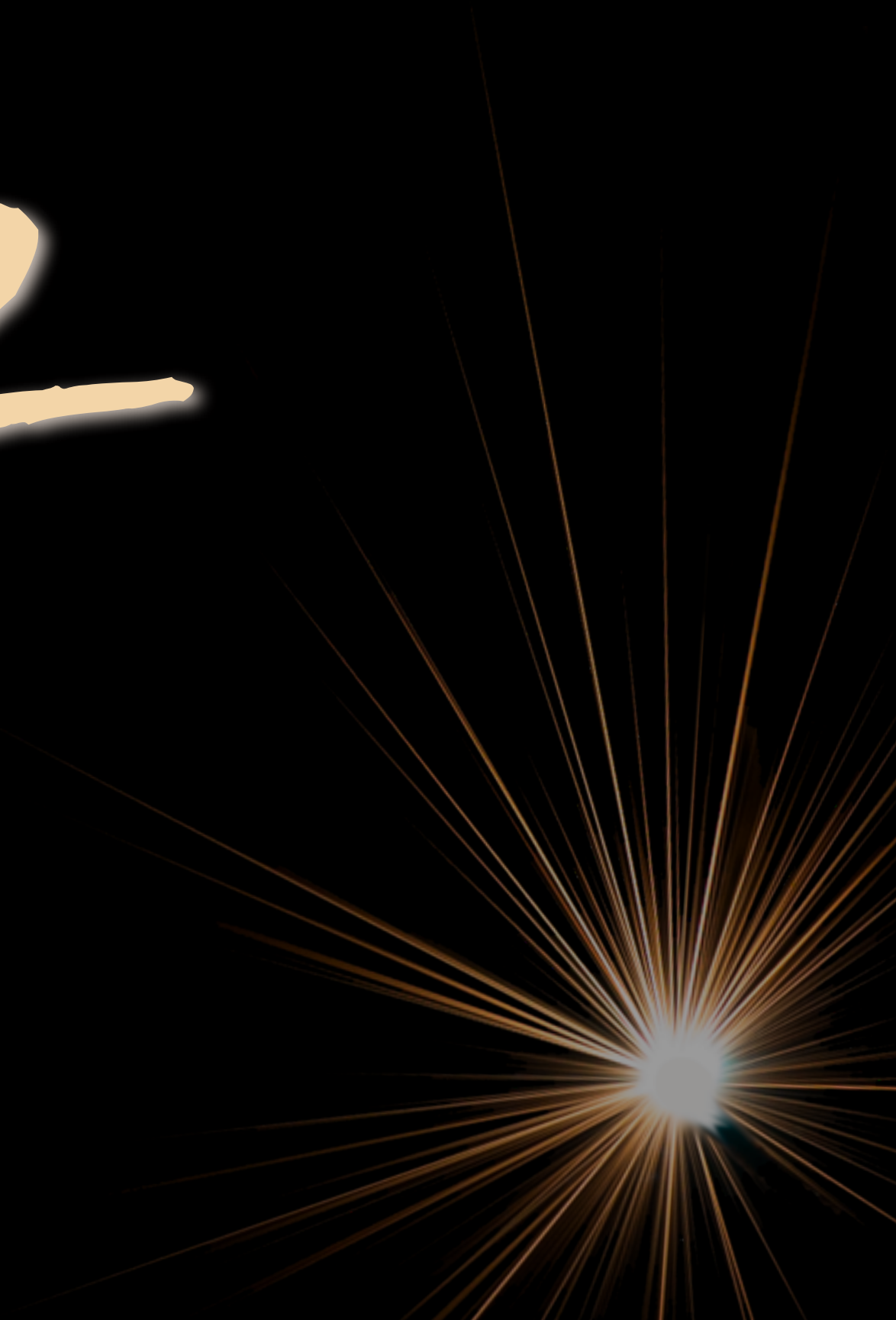
List(xn-10, xn-11, xn-12)

```
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")
```



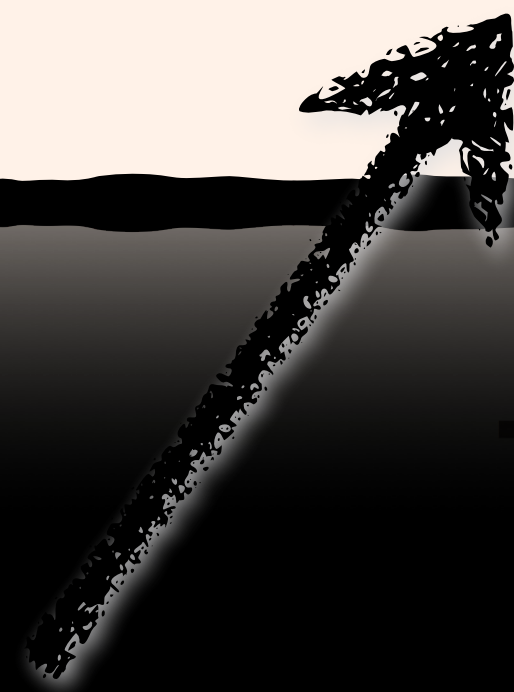


# PROJECTION

```
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)

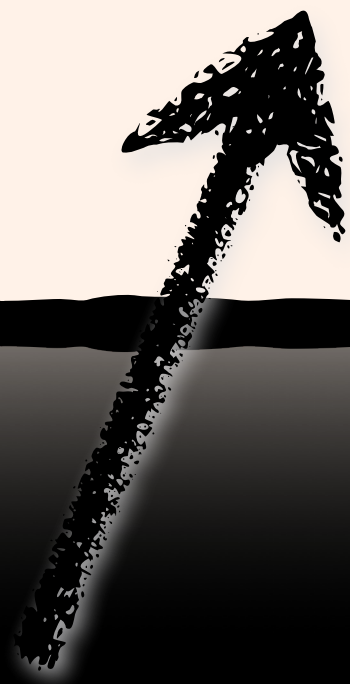
val countries = xctns.map(x => (x._1, x._2))
```

# 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