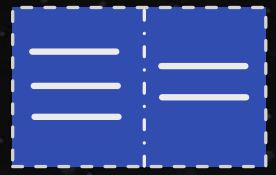


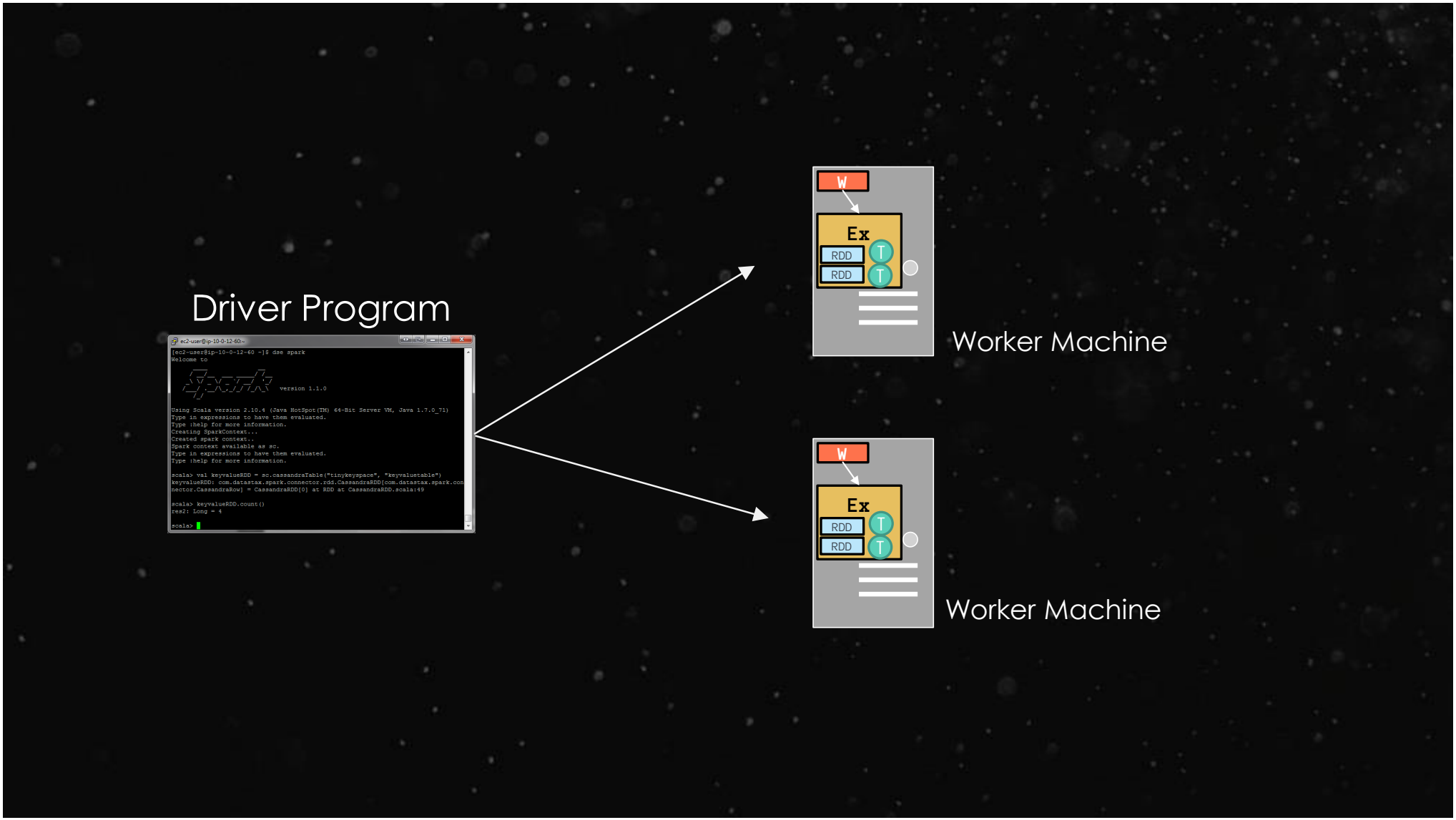
RDD FUNDAMENTALS

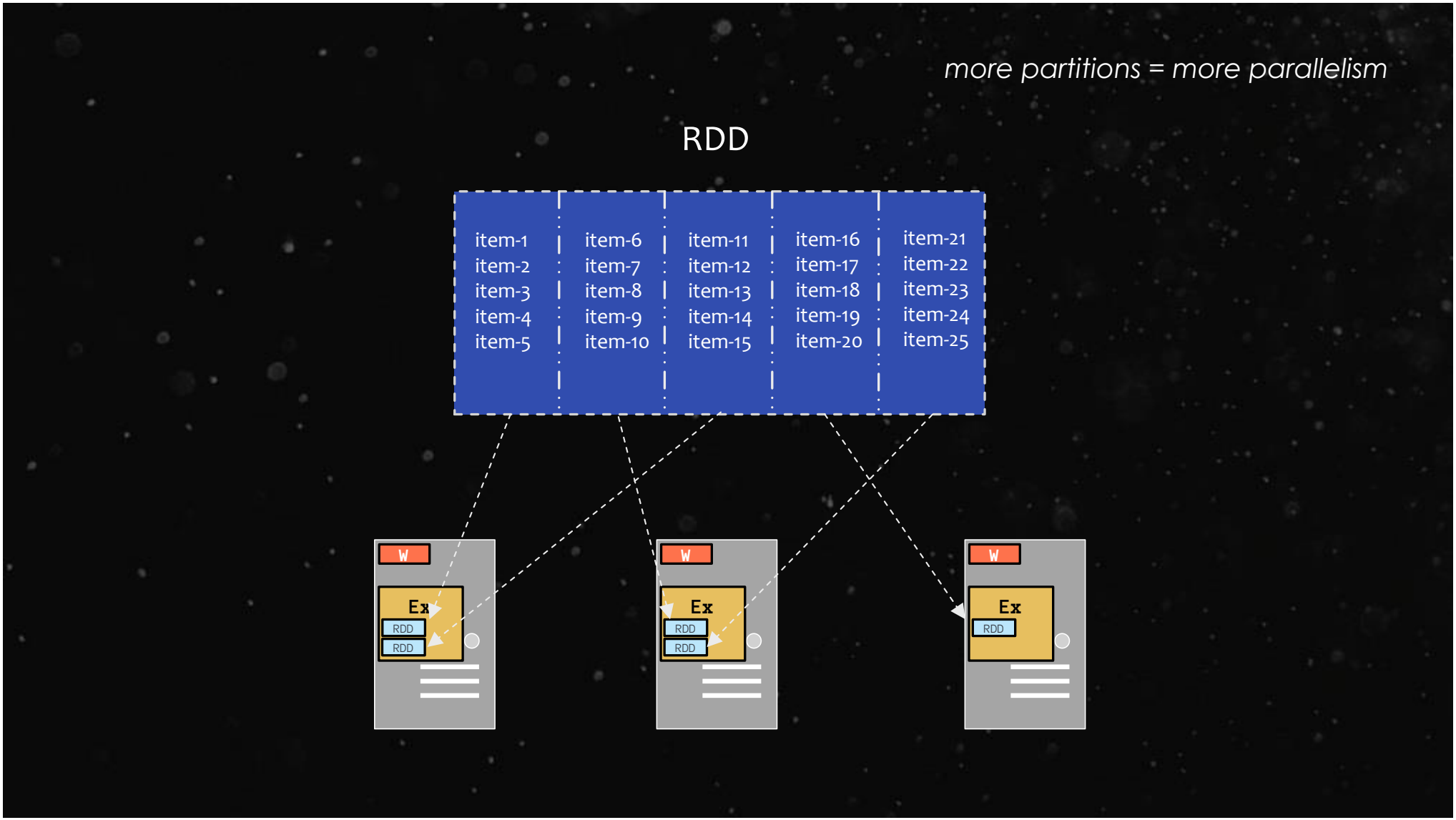
databricks

INTERACTIVE SHELL

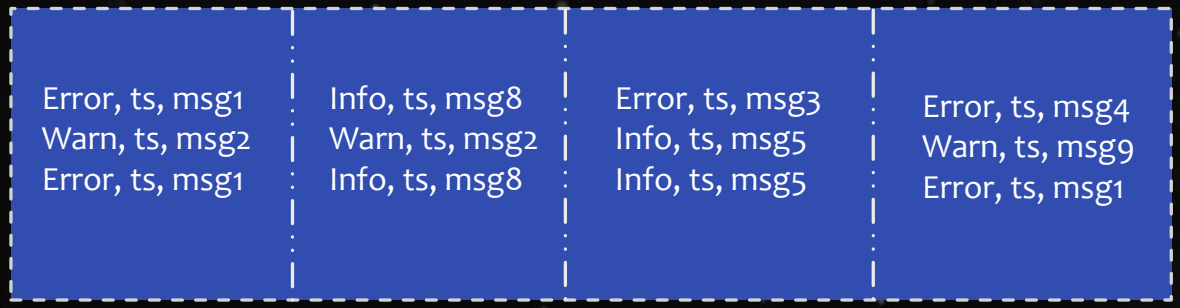
```
ubuntu@ip-10-0-53-24: ~
ubuntu@ip-10-0-53-24:~$ dse spark
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 0.9.1
      /_/

Using Scala version 2.10.3 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_51)
Type in expressions to have them evaluated.
Type :help for more information.
Creating SparkContext...
Created spark context..
Spark context available as sc.
Type in expressions to have them evaluated.
Type :help for more information.

scala> val myRDD = sc.cassandraTable("tinykeyspace", "keyvaluetable")
myRDD: com.datastax.bdp.spark.CassandraRDD[com.datastax.bdp.spark.CassandraRow] = Cassan
draRDD[0] at RDD at CassandraRDD.scala:32

scala> myRDD.count()
res2: Long = 5

scala>
```

(Scala & Python only)


Driver Program
ec2-user@ip-10-0-12-60:~
[ec2-user@ip-10-0-12-60 ~]$ dse spark
Welcome to
version 1.1.0
Using Scala version 2.10.4 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_71)
Type in expressions to have them evaluated.
Type :help for more information.
Creating SparkContext...
Created spark context..
Spark context available as sc.
Type in expressions to have them evaluated.
Type :help for more information.
scala> val keyvalueRDD = sc.cassandraTable("tinykeyspace", "keyvaluetable")
keyvalueRDD: com.datastax.spark.connector.rdd.CassandraRDD[com.datastax.spark.con
nector.CassandraRow] = CassandraRDD[0] at RDD at CassandraRDD.scala:49
scala> keyvalueRDD.count()
res2: Long = 4
scala>
W
Ex
RDD
RDD
T
T
Worker Machine
W
Ex
RDD
RDD
T
T
Worker Machine


more partitions = more parallelism
RDD
item-1
item-2
item-3
item-4
item-5
item-6
item-7
item-8
item-9
item-10
item-11
item-12
item-13
item-14
item-15
item-16
item-17
item-18
item-19
item-20
item-21
item-22
item-23
item-24
item-25
W
Ex
RDD
RDD
W
Ex
RDD
RDD
W
Ex
RDD

RDD w/ 4 partitions



logLinesRDD

An RDD can be created 2 ways:

- Parallelize a collection
- Read data from an external source (S3, C*, HDFS, etc)

PARALLELIZE

```python
# Parallelize in Python
wordsRDD = sc.parallelize([“fish", “cats“, “dogs”])
```

```scala
// Parallelize in Scala
val wordsRDD= sc.parallelize(List("fish", "cats", "dogs"))
```

```java
// Parallelize in Java
JavaRDD<String> wordsRDD = sc.parallelize(Arrays.asList(“fish", “cats“, “dogs”));
```

- Take an existing in-memory collection and pass it to SparkContext's parallelize method
- Not generally used outside of prototyping and testing since it requires entire dataset in memory on one machine

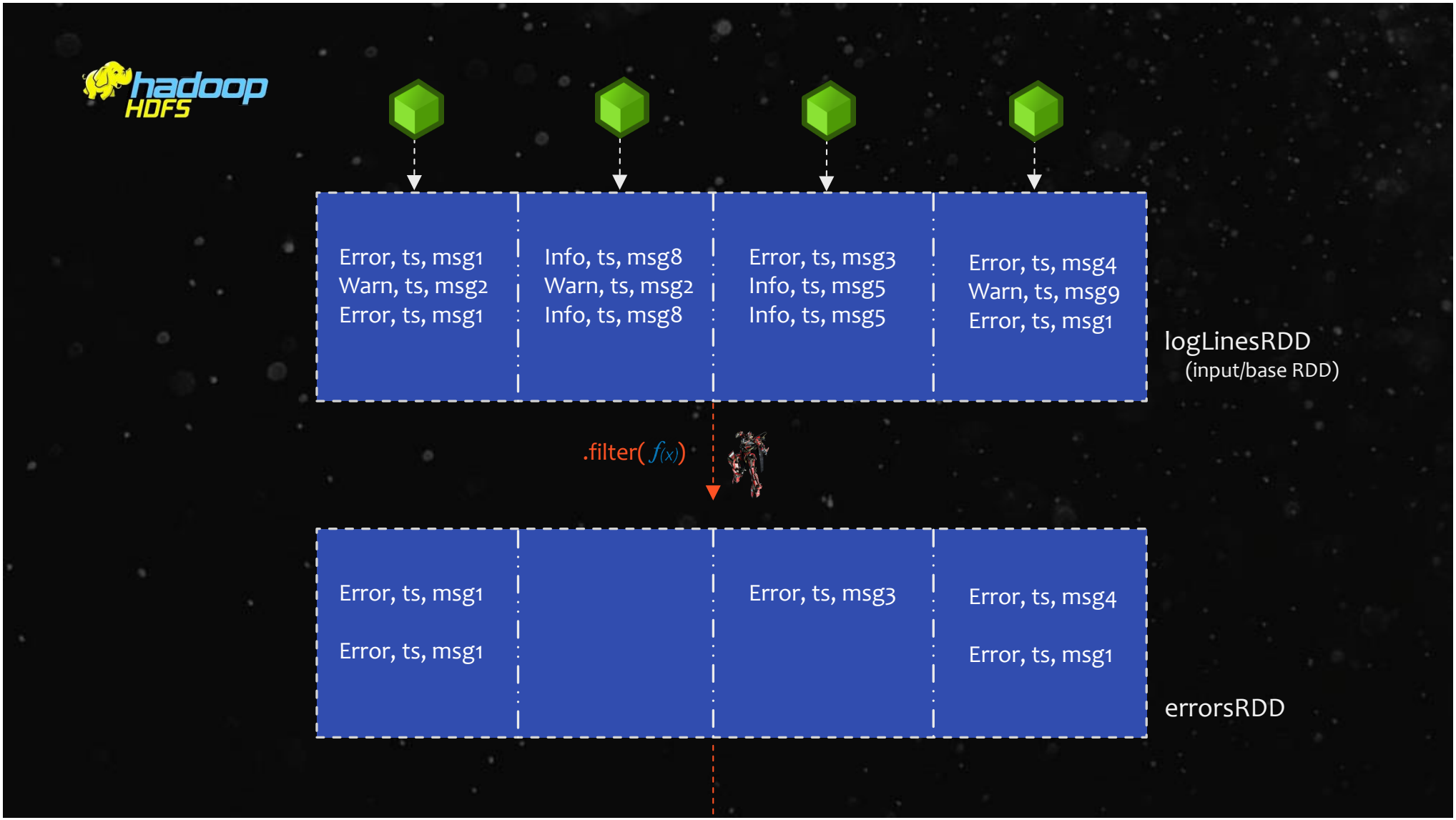
READ FROM TEXT FILE

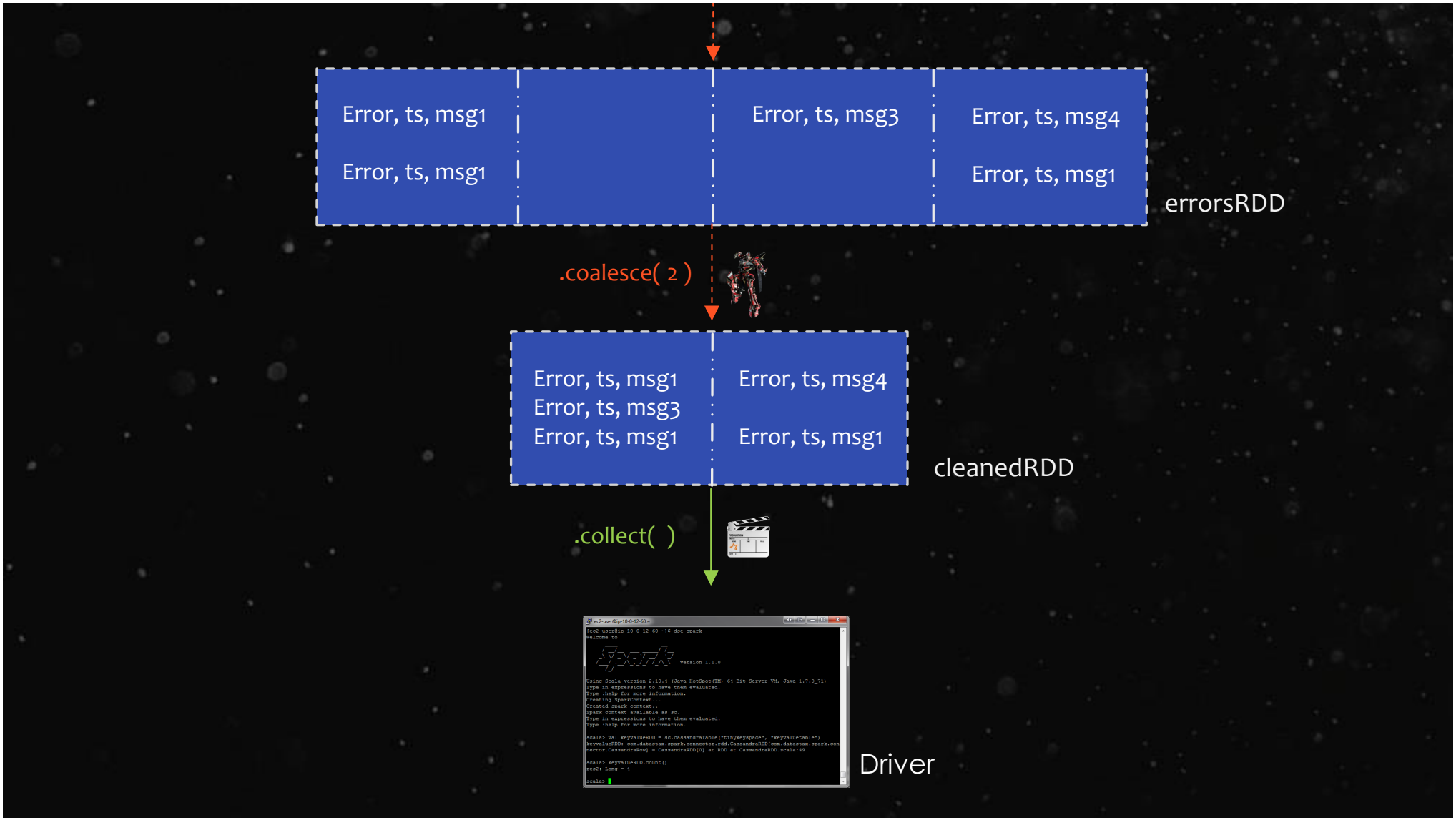
```python
# Read a local txt file in Python
linesRDD = sc.textFile("/path/to/README.md")
```

```scala
// Read a local txt file in Scala
val linesRDD = sc.textFile("/path/to/README.md")
```

```java
// Read a local txt file in Java
JavaRDD<String> lines = sc.textFile("/path/to/README.md");
```

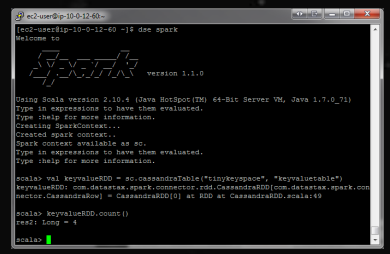
- There are other methods to read data from HDFS, C*, S3, HBase, etc.


hadoop
HDFS
Error, ts, msg1
Warn, ts, msg2
Error, ts, msg1
Info, ts, msg8
Warn, ts, msg2
Info, ts, msg8
Error, ts, msg3
Info, ts, msg5
Info, ts, msg5
Error, ts, msg4
Warn, ts, msg9
Error, ts, msg1
logLinesRDD
(input/base RDD)
.filter(f(x))
Error, ts, msg1
Error, ts, msg1
Error, ts, msg3
Error, ts, msg4
Error, ts, msg1
errorsRDD

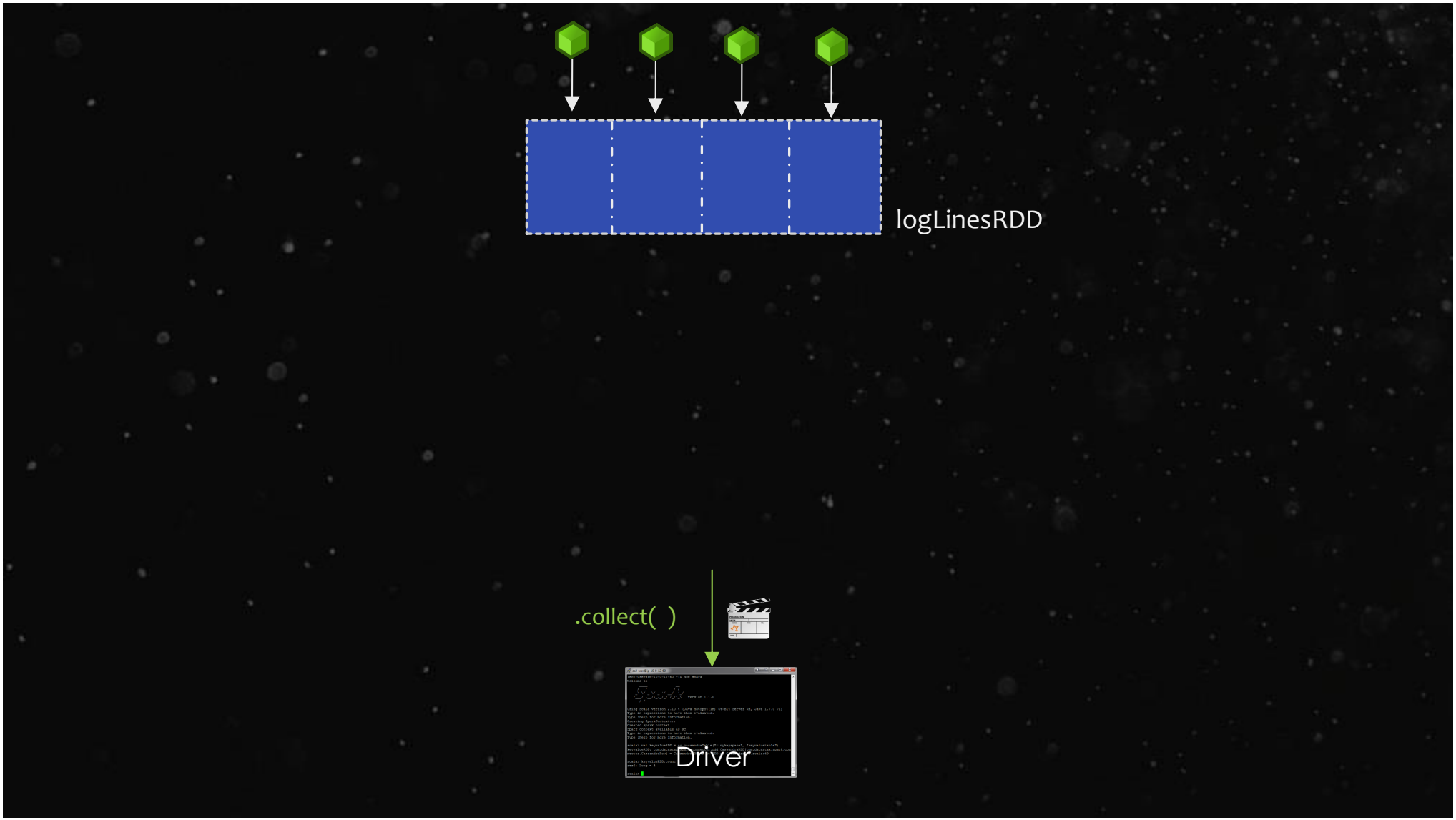

Error, ts, msg1
Error, ts, msg1
Error, ts, msg3
Error, ts, msg4
Error, ts, msg1
errorsRDD
.coalesce(2)
Error, ts, msg1
Error, ts, msg3
Error, ts, msg1
Error, ts, msg4
Error, ts, msg1
cleanedRDD
.collect( )
Driver

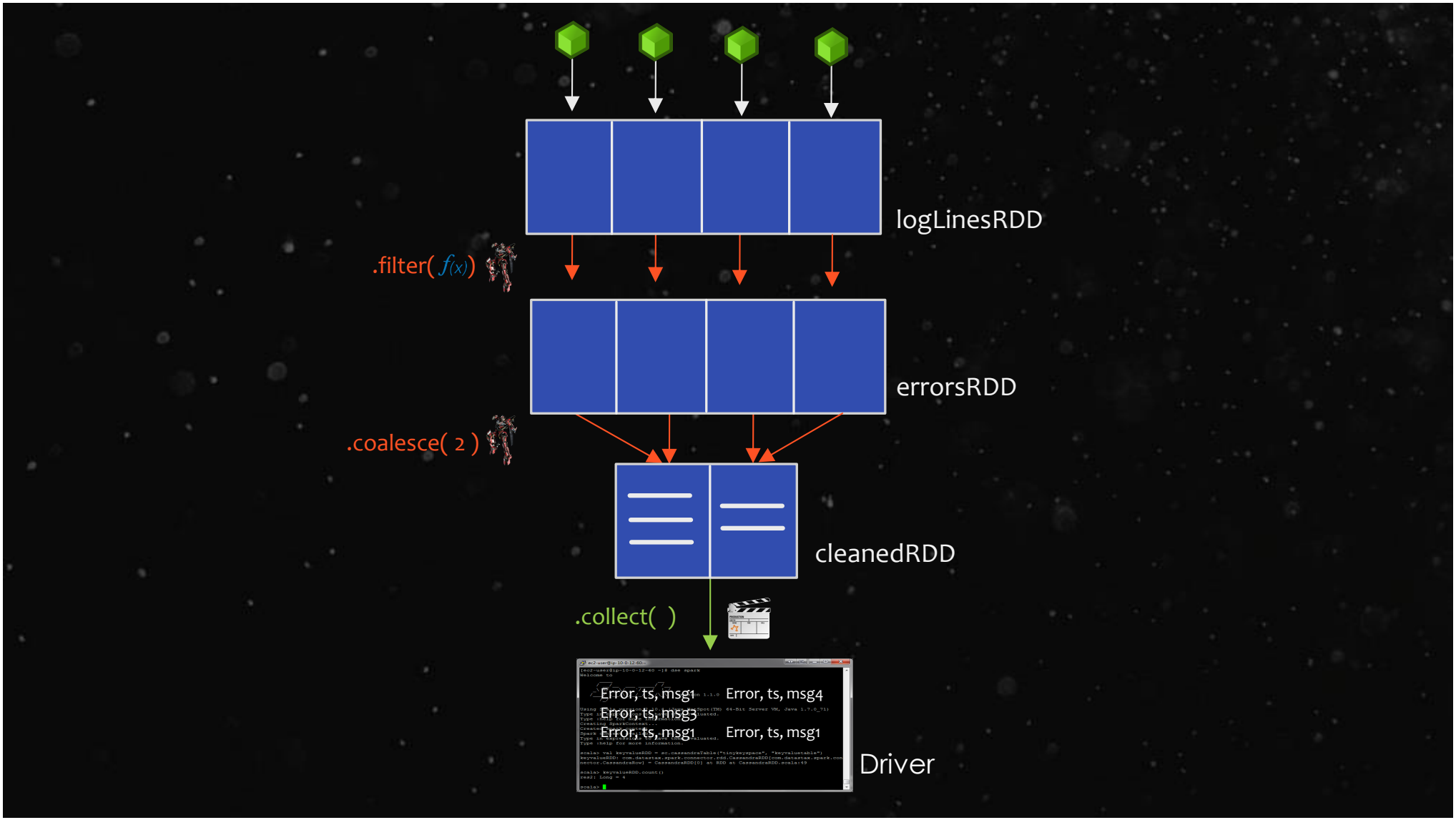
Execute DAG!

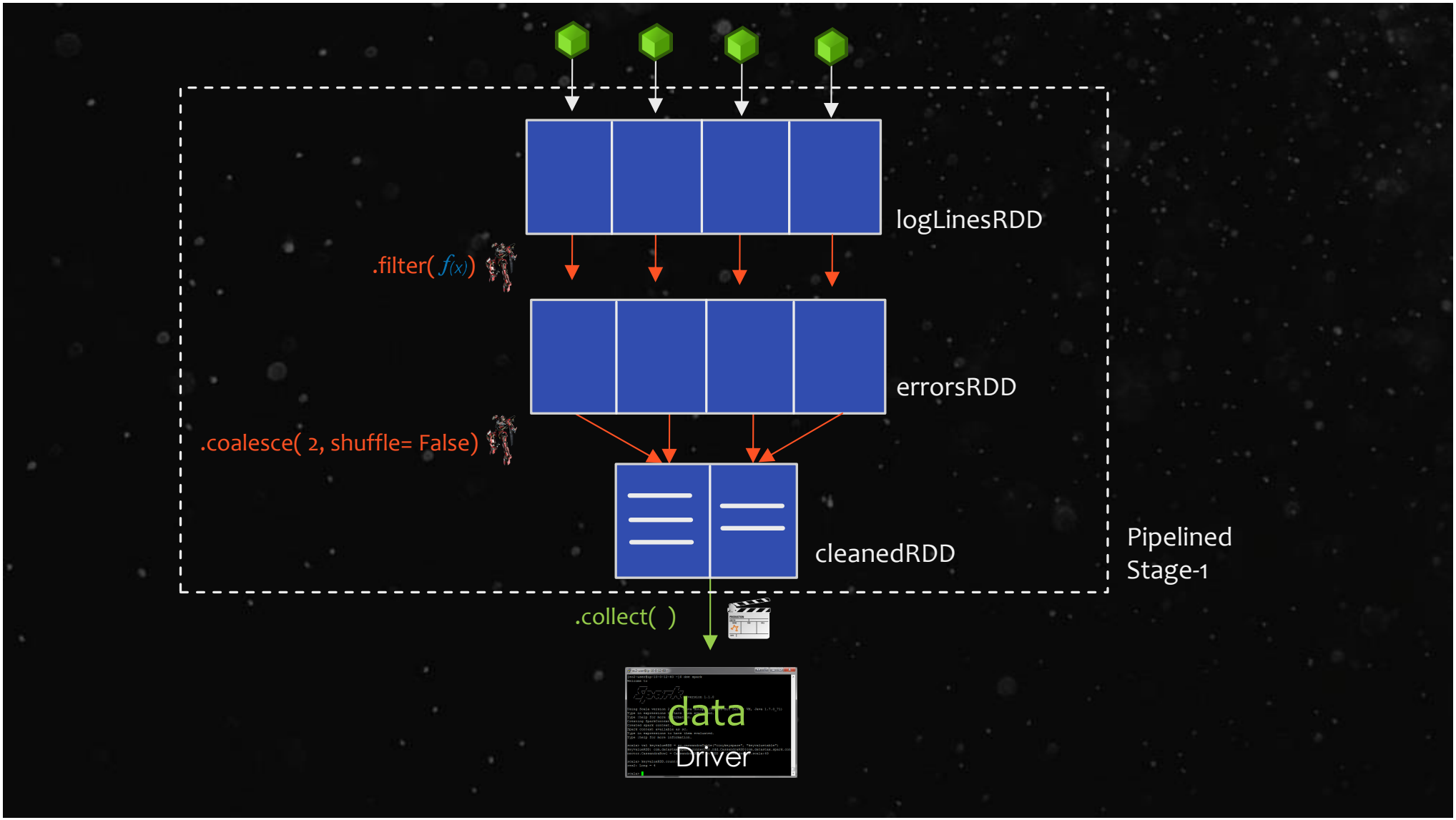
.collect( )

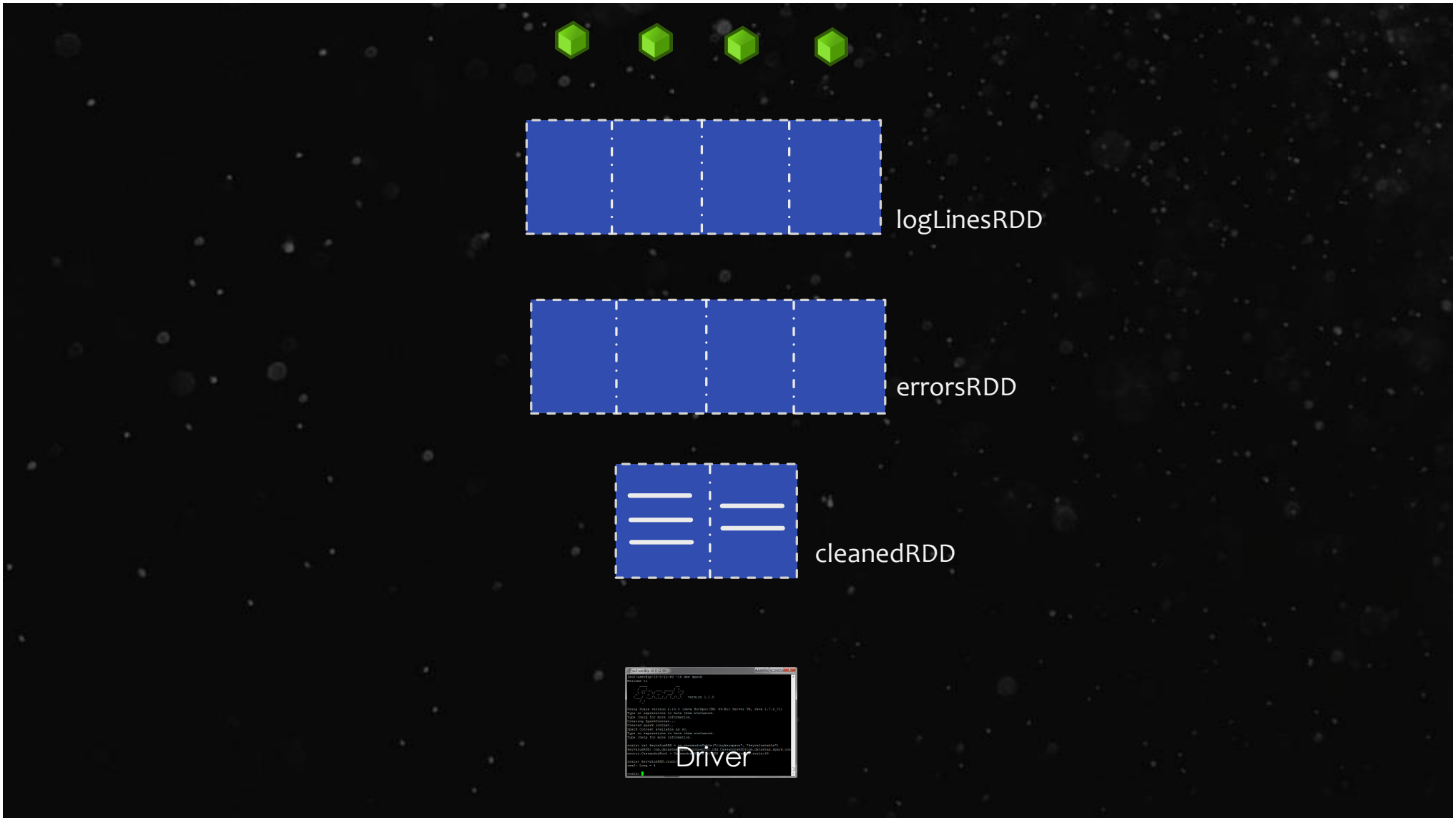


Driver

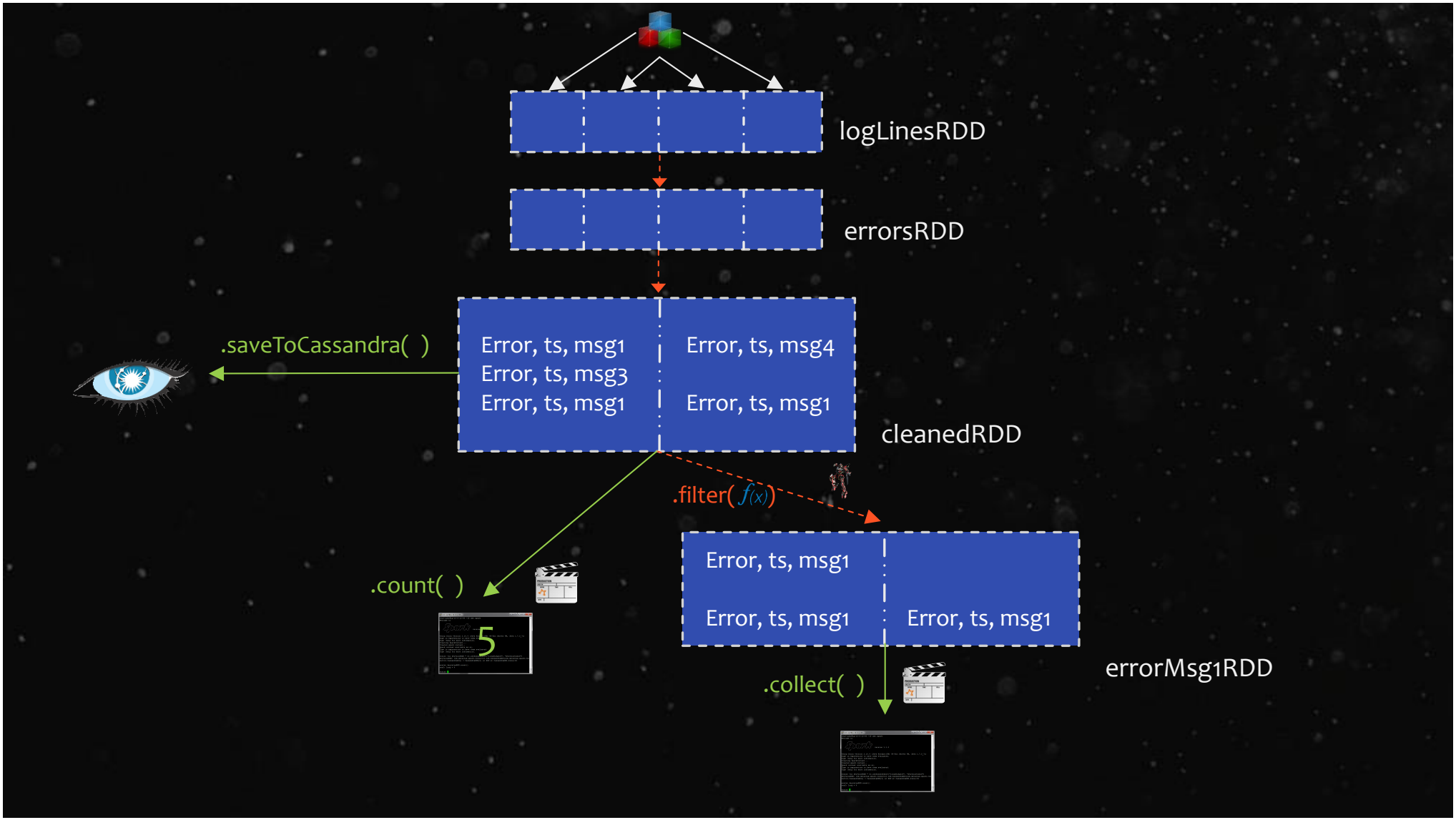

logLinesRDD
.collect( )
Driver

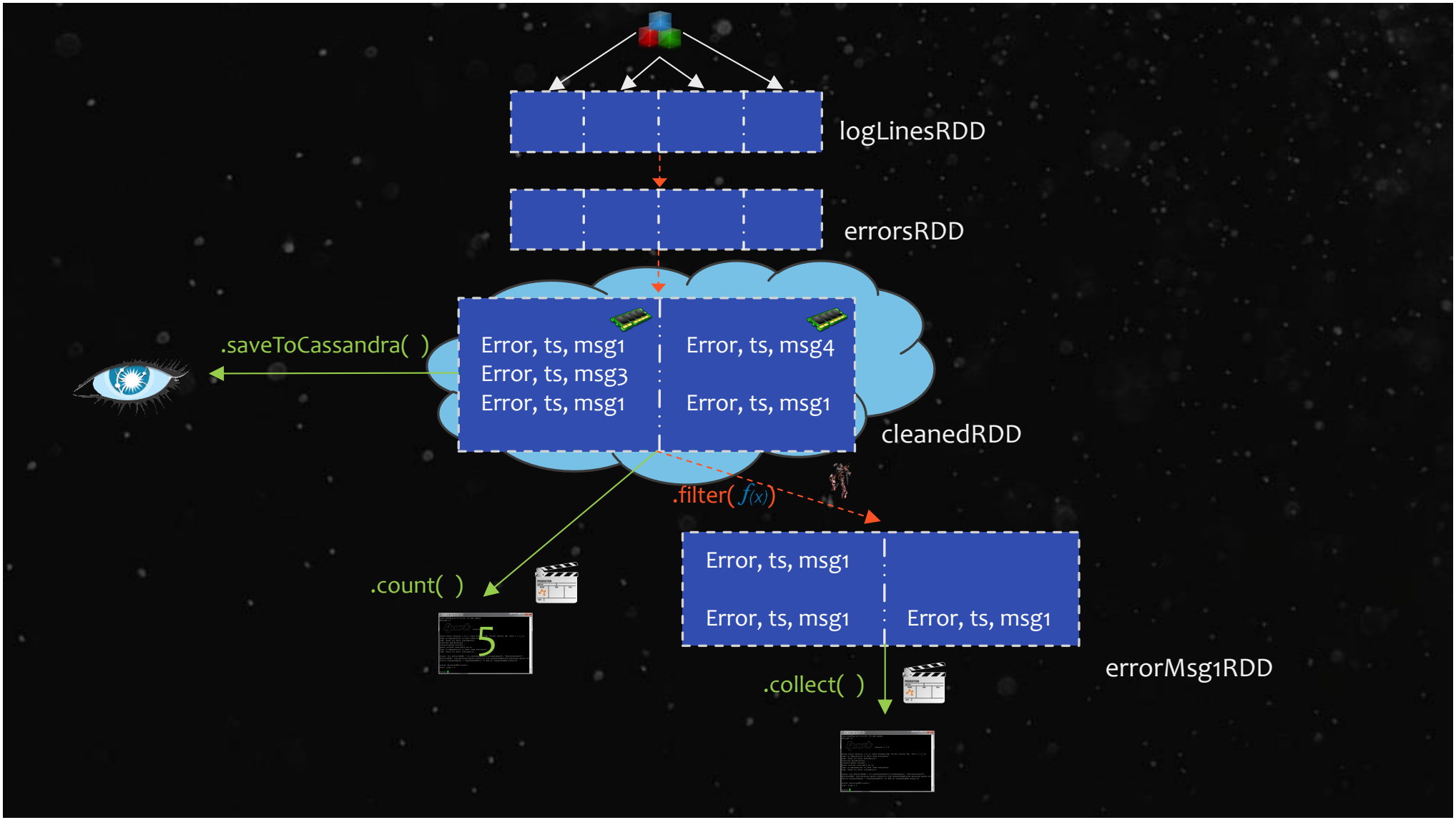

logLinesRDD
.filter(f(x))
errorsRDD
.coalesce(2)
cleanedRDD
.collect( )
Error, ts, msg1
Error, ts, msg4
Error, ts, msg3
Error, ts, msg1
Error, ts, msg1
Driver


logLinesRDD
.filter(f(x))
errorsRDD
.coalesce(2, shuffle= False)
cleanedRDD
Pipelined
Stage-1
.collect( )
data
Driver


logLinesRDD
errorsRDD
cleanedRDD
Driver

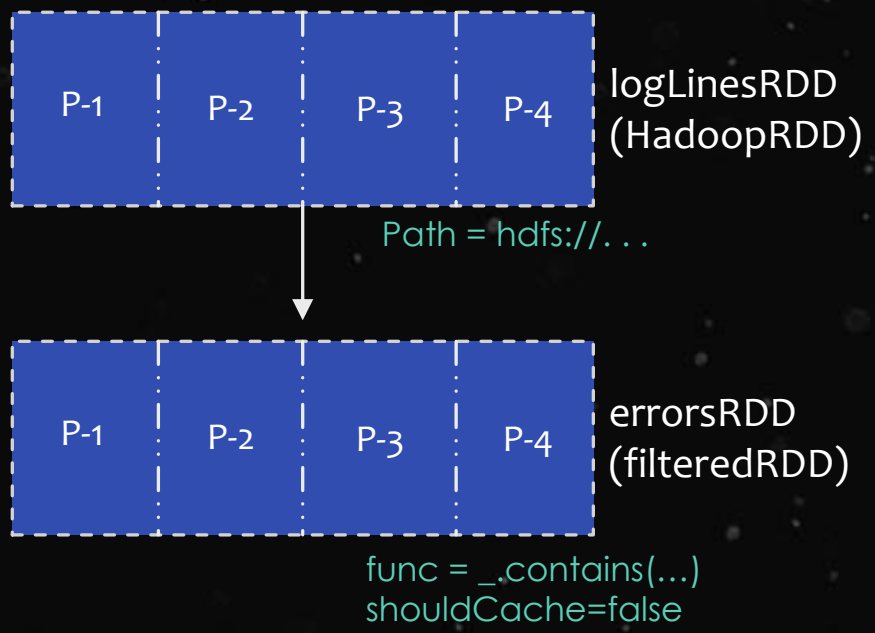

data
Driver


logLinesRDD
errorsRDD
.saveToCassandra( )
Error, ts, msg1
Error, ts, msg3
Error, ts, msg1
Error, ts, msg4
Error, ts, msg1
cleanedRDD
.filter(f(x))
.count( )
5
Error, ts, msg1
Error, ts, msg1
Error, ts, msg1
errorMsg1RDD
.collect( )

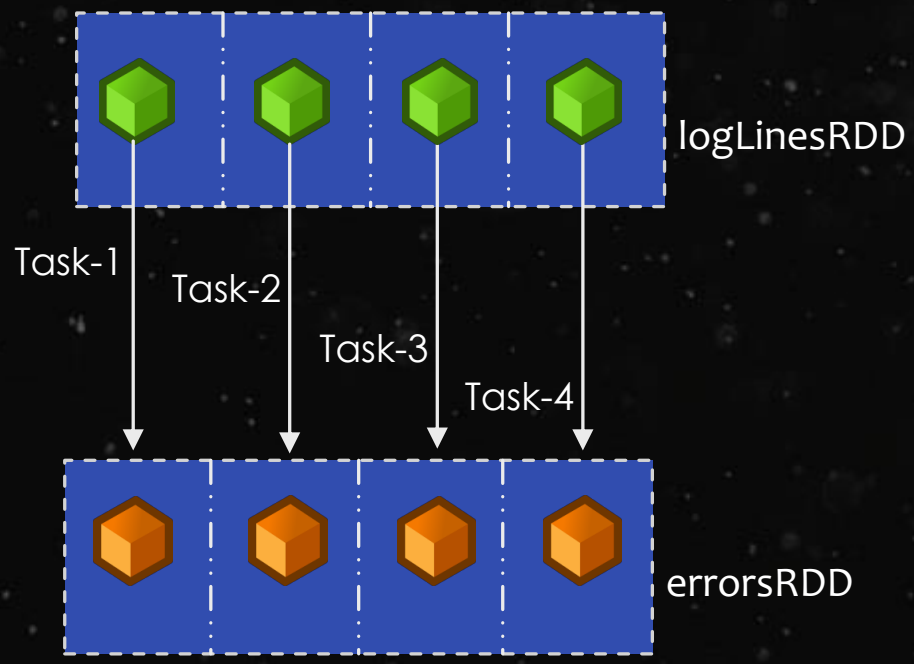

logLinesRDD
errorsRDD
.saveToCassandra( )
Error, ts, msg1
Error, ts, msg3
Error, ts, msg1
Error, ts, msg4
Error, ts, msg1
cleanedRDD
.filter(f(x))
.count( )
5
Error, ts, msg1
Error, ts, msg1
Error, ts, msg1
errorMsg1RDD
.collect( )

RDD GRAPH

Dataset-level view:



Partition-level view:



LIFECYCLE OF A SPARK PROGRAM

1) Create some input RDDs from external data or parallelize a collection in your driver program.

2) Lazily transform them to define new RDDs using transformations like `filter()` or `map()`

3) Ask Spark to `cache()` any intermediate RDDs that will need to be reused.

4) Launch actions such as `count()` and `collect()` to kick off a parallel computation, which is then optimized and executed by Spark.

TRANSFORMATIONS (lazy)

map()	intersection()	cartesion()
flatMap()	distinct()	pipe()
filter()	groupByKey()	coalesce()
mapPartitions()	reduceByKey()	repartition()
mapPartitionsWithIndex()	sortByKey()	partitionBy()
sample()	join()	...
union()	cogroup()	...

- Most transformations are element-wise (they work on one element at a time), but this is not true for all transformations

ACTIONS

```
reduce()
collect()
count()
first()
take()
takeSample()
saveToCassandra()
```

```
takeOrdered()
saveAsTextFile()
saveAsSequenceFile()
saveAsObjectFile()
countByKey()
foreach()
...
```

TYPES OF RDDS

- HadoopRDD
- FilteredRDD
- MappedRDD
- PairRDD
- ShuffledRDD
- UnionRDD
- PythonRDD

- DoubleRDD
- JdbcRDD
- JsonRDD
- SchemaRDD
- VertexRDD
- EdgeRDD

- CassandraRDD *(DataStax)*
- GeoRDD *(ESRI)*
- EsSpark *(ElasticSearch)*