DEEP DIVE INTO RDDs

AGENDA

- Introduction to RDDs
- Ways to create RDDs
- Transformations and Actions
- Life Cycle of a Spark Program
- Persistence and Caching
- Data Serialization
- Lazy Evaluation
- Fault Tolerance
- Lineage Graph
- Task Locality
- Working with Partitions
- Types of RDDs
- Pair RDDs

RDD (Resilient Distributed Dataset)

- ➢ **RDDs are the core of Spark**
  - ✓ **Resilient** – If data in memory is lost, it can be recreated (or recomputed)
  - ✓ **Distributed** – Stored in memory across the cluster
  - ✓ **Dataset** – Initial data can come from a file or created programmatically

  - ✓ Immutable collections partitioned across cluster that can be rebuilt (re-computed) if a partition is lost
  - ✓ Created by transforming data in stable storage using data flow operators (map, filter, group-by, …)
  - ✓ Can be cached across parallel operations

- ➢ **RDDs are fundamental unit of data in Spark**

- ➢ **Most of the Spark programming is performing operations on RDDs**

Ways to create RDD # 1

PARALLELIZE

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

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

```
// 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

Ways to create RDD # 2

READ FROM TEXT FILE

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

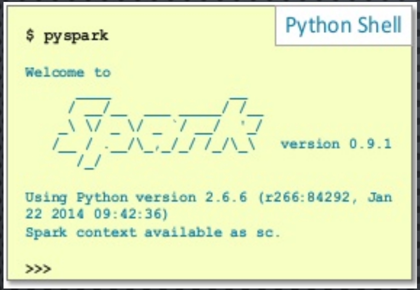
```
// Read a local txt file in Scala
val linesRDD = sc.textFile("/path/to/README.md")
```

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

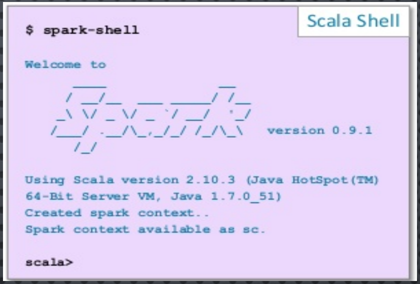
- There are other methods to read data from HDFS, C*, S3, HBase, etc.
- It could be a file or a directory with wildcards as well

Ways to work with Spark

- ➢ **Spark Shell**
  - ✓ Interactive REPL (read–eval–print loop) for data exploraton
  - ✓ Python or Scala



```
$ pyspark

Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 0.9.1
      /_/

Using Python version 2.6.6 (r266:84292, Jan
22 2014 09:42:36)
Spark context available as sc.

>>>
```



```
$ spark-shell

Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 0.9.1
      /_/

Using Scala version 2.10.3 (Java HotSpot(TM)
64-Bit Server VM, Java 1.7.0_51)
Created spark context..
Spark context available as sc.

scala>
```

- ➢ **Spark Applications**
  - ✓ For Large Scale data processing
  - ✓ Python or Scala or Java or SQL or R

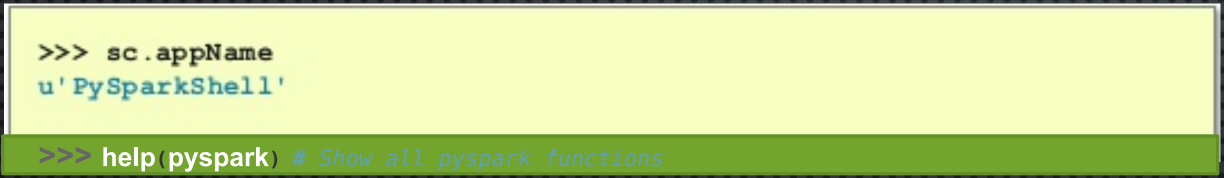
```
$ bin/spark-submit \
    --class com.example.MyApp \
    --master spark://myhost:7077 \
    --name "My Spark App" \
    --executor-memory 1G \
    --total-executor-cores 4 \
    --conf spark.shuffle.spill=false \
    myApp.jar
```

Spark Context

- ➢ **Every Spark application requires a Spark Context**
  - ✓ The main entry point to Spark API

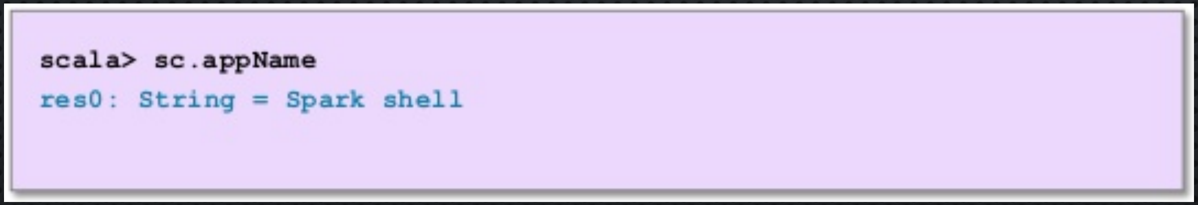
- ➢ **Spark Shell provides a pre configured Spark Context called sc**

- ➢ **pyspark**



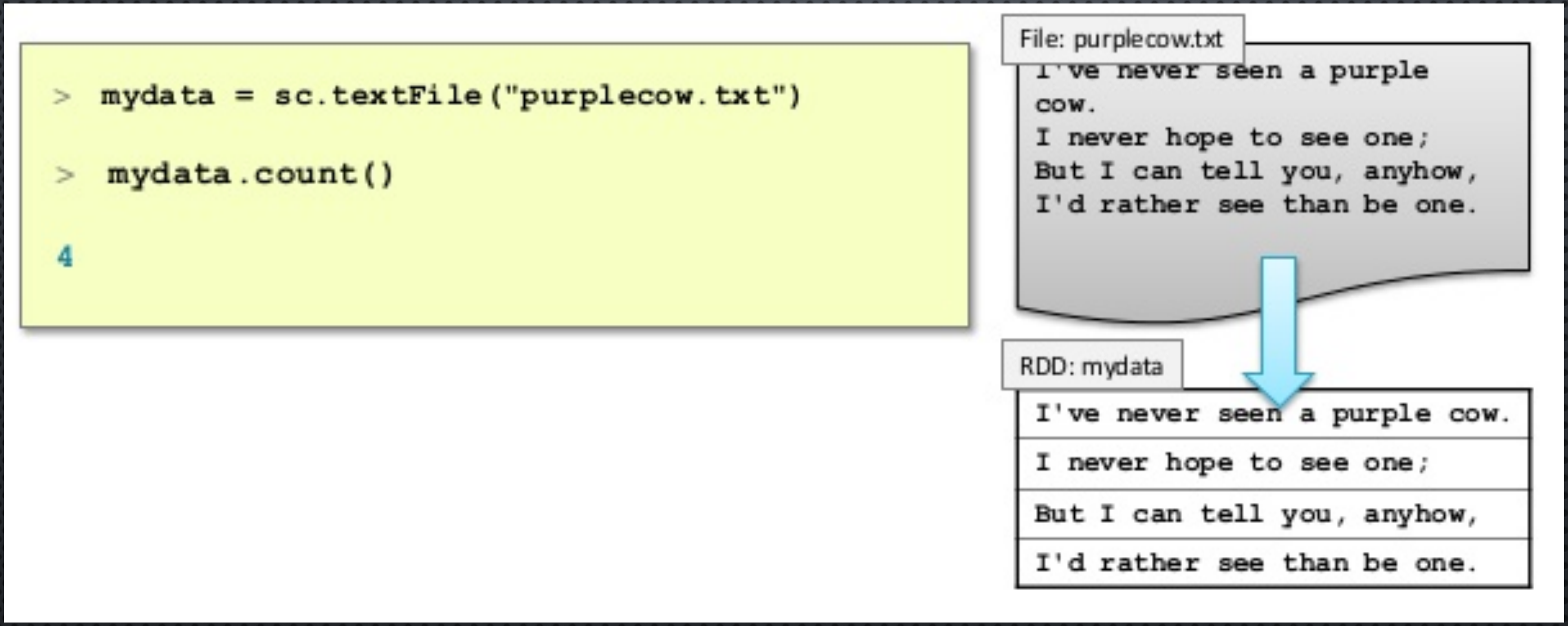
```
>>> sc.appName
u'PySparkShell'
>>> help(pyspark) # Show all pyspark functions
```

- ➢ **spark-shell (Scala)**



```
scala> sc.appName
res0: String = Spark shell
```

Example: A File Based RDD



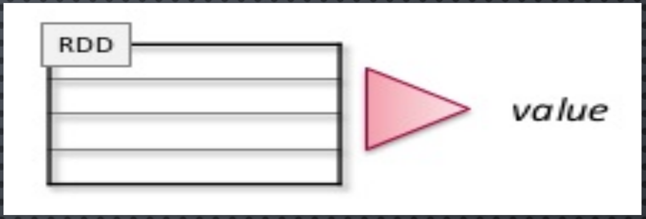
```
> mydata = sc.textFile("purplecow.txt")

> mydata.count()

4
```

File: purplecow.txt
I've never seen a purple
cow.
I never hope to see one;
But I can tell you, anyhow,
I'd rather see than be one.
RDD: mydata
I've never seen a purple cow.
I never hope to see one;
But I can tell you, anyhow,
I'd rather see than be one.

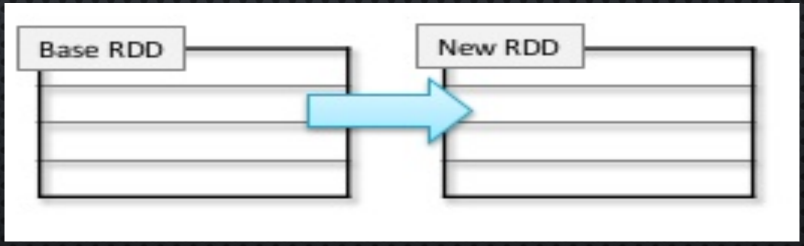
RDD Operations

➢ **Two types of RDD operations**

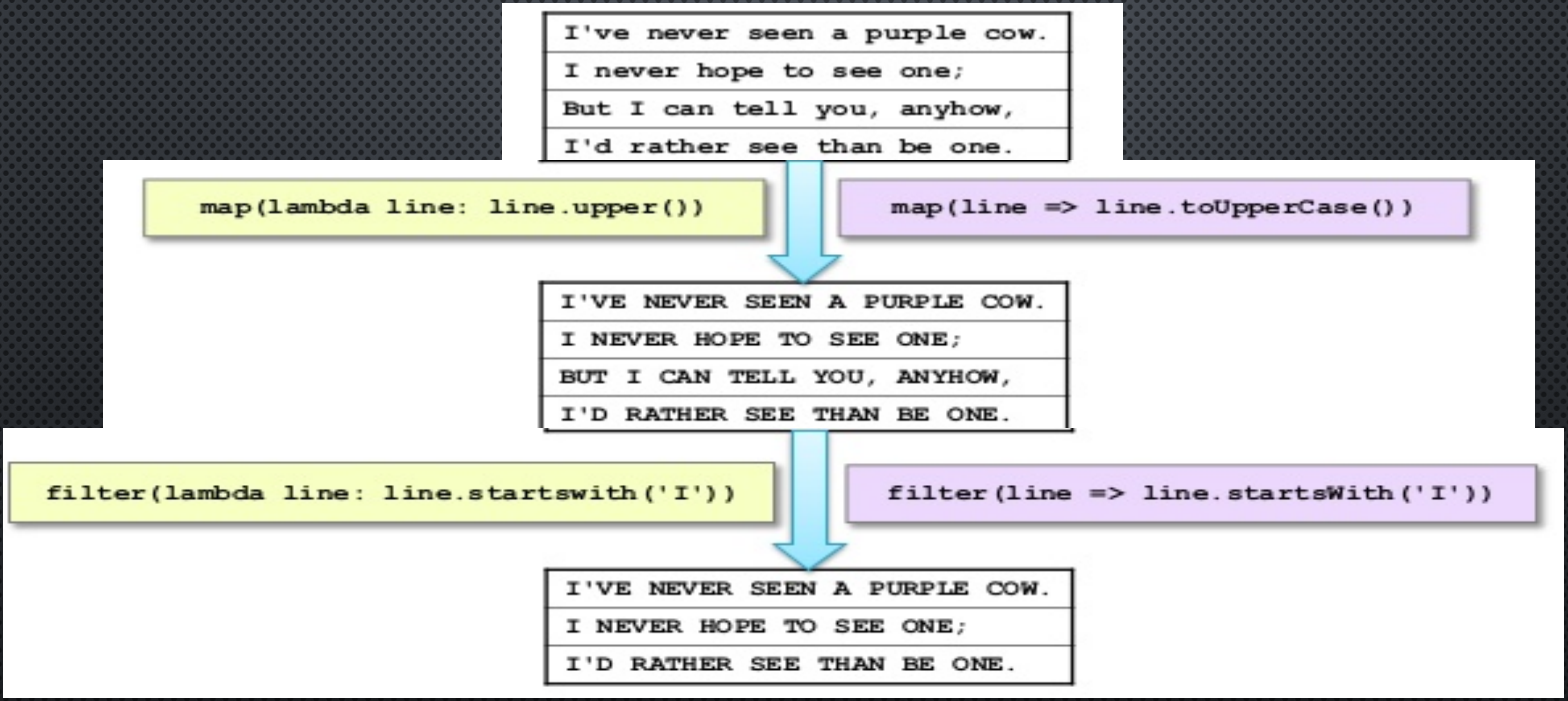
- ✓ Actions – Return values
  - ✓ count
  - ✓ take(n)



- ✓ Transformations – Define new RDDs based on the current one
  - ✓ filter
  - ✓ map
  - ✓ reduce



Example: Map and Filter Transformations



```
I've never seen a purple cow.
I never hope to see one;
But I can tell you, anyhow,
I'd rather see than be one.
```

```
map(lambda line: line.upper())
```

```
map(line => line.toUpperCase())
```

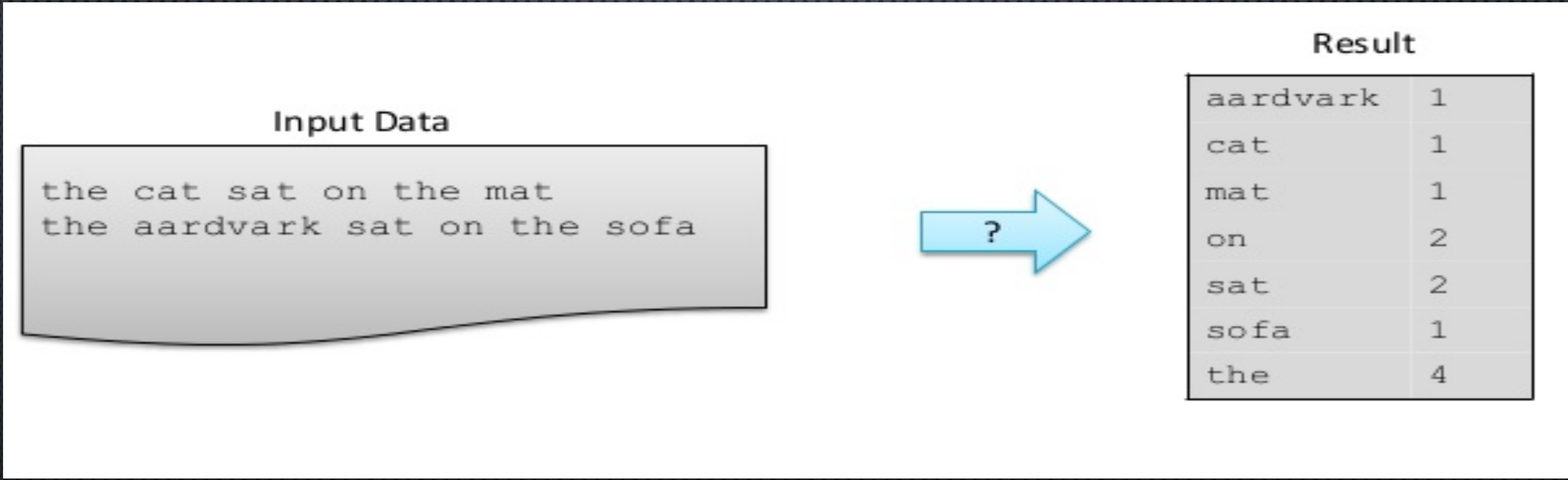
```
I'VE NEVER SEEN A PURPLE COW.
I NEVER HOPE TO SEE ONE;
BUT I CAN TELL YOU, ANYHOW,
I'D RATHER SEE THAN BE ONE.
```

```
filter(lambda line: line.startswith('I'))
```

```
filter(line => line.startsWith('I'))
```

```
I'VE NEVER SEEN A PURPLE COW.
I NEVER HOPE TO SEE ONE;
I'D RATHER SEE THAN BE ONE.
```

Example: MapReduce Wordcount

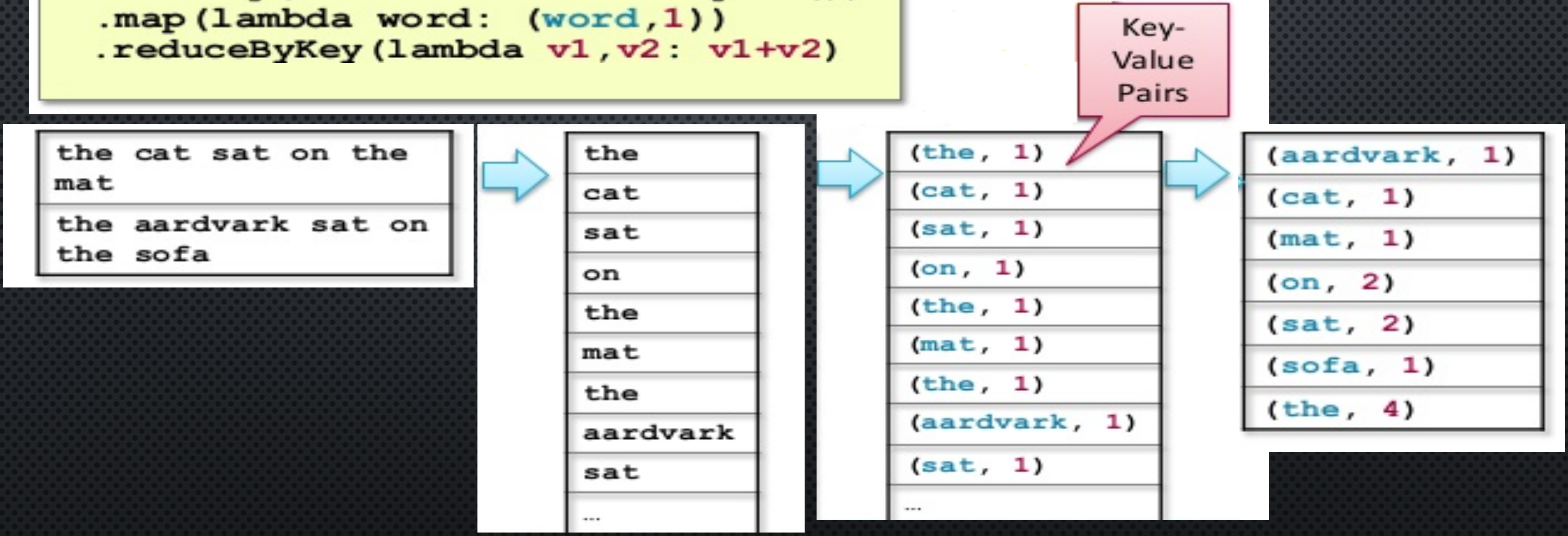


```
the cat sat on the mat
the aardvark sat on the sofa
```

aardvark	1
cat	1
mat	1
on	2
sat	2
sofa	1
the	4

Example: MapReduce Wordcount

```
> counts = sc.textFile(file)
  .flatMap(lambda line: line.split())
  .map(lambda word: (word,1))
  .reduceByKey(lambda v1,v2: v1+v2)
```



Spark Basic Transformations

```
>  nums = sc.parallelize([1, 2, 3])

# Pass each element through a function
>  squares = nums.map(lambda x: x*x)    // {1, 4, 9}

# Keep elements passing a predicate
>  even = squares.filter(lambda x: x % 2 == 0)  // {4}

# flatMap each element
>  lines = sc.parallelize(["hello world", "hi"])
>  words = lines.flatMap(lambda line: line.split(" "))
>  words.first()
       # "hello"
```

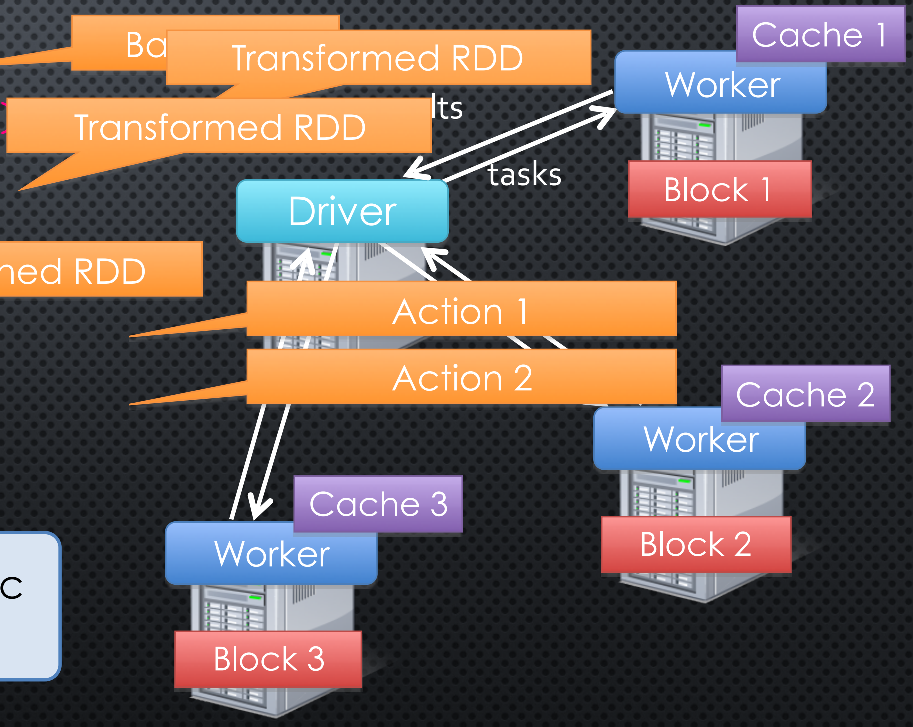
Spark Basic Actions

```
> nums = sc.parallelize([1, 2, 3])
# Retrieve RDD contents as a local collection
> nums.collect() # => [1, 2, 3]
# Return first K elements
> nums.take(2)   # => [1, 2]
# Count number of elements
> nums.count()   # => 3
# Merge elements with an associative function
> nums.reduce(lambda x, y: x + y)  # => 6
# Write elements to a text file
> nums.saveAsTextFile("hdfs://file.txt")
```

Example: Log Mining

➢ Load error messages from a log into memory, then interactively search for various patterns

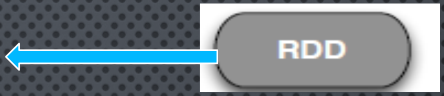
```
val lines = spark.textFile(“hdfs://...”)
val errors = lines.filter(_.startsWith(“ERROR”))
val messages = errors.map(_.split(‘\t’)(2))
val cachedMsgs = messages.cache()

cachedMsgs.filter(_.contains(“foo”)).count
cachedMsgs.filter(_.contains(“bar”)).count
. . .
```

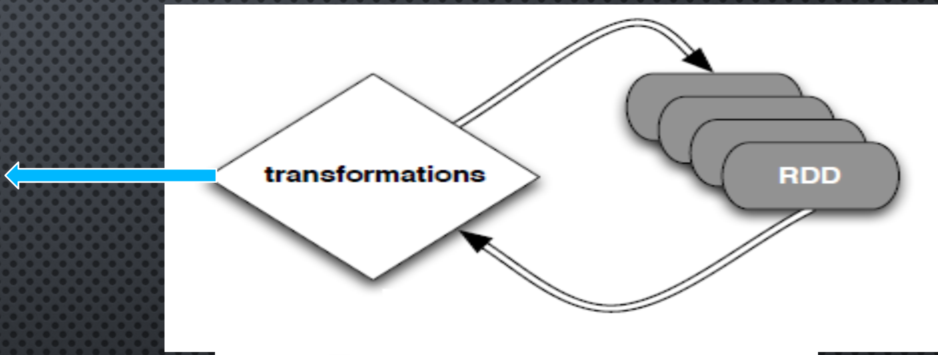


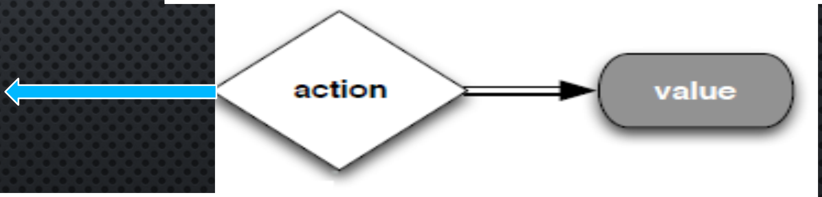
Result: full-text search of Wikipedia in <1 sec (vs 20 sec for on-disk data)

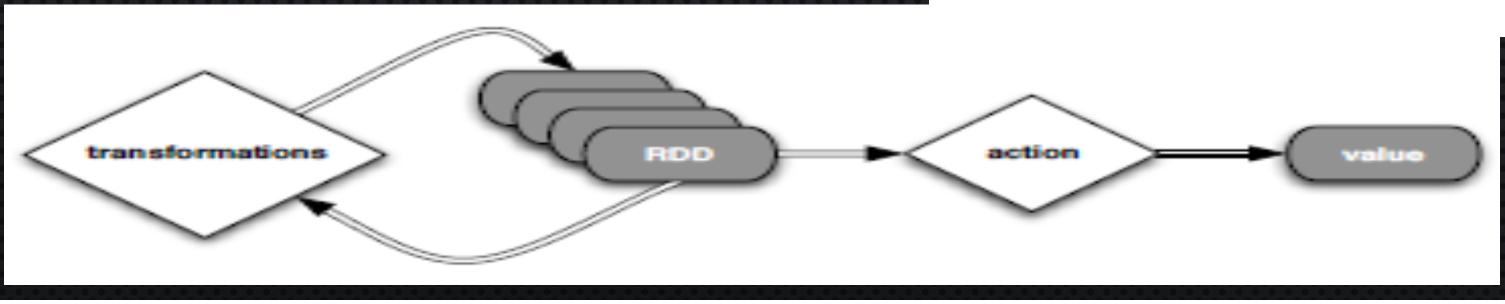
Looking RDD Transformations from other perspective

```
// base RDD
lines = spark.textFile("hdfs://...")

// transformed RDDs
errors = lines.filter(_.startsWith("ERROR"))
messages = errors.map(_.split('\t'))
cachedMsgs = messages.cache()

// action 1
cachedMsgs.filter(_.contains("foo")).count
```

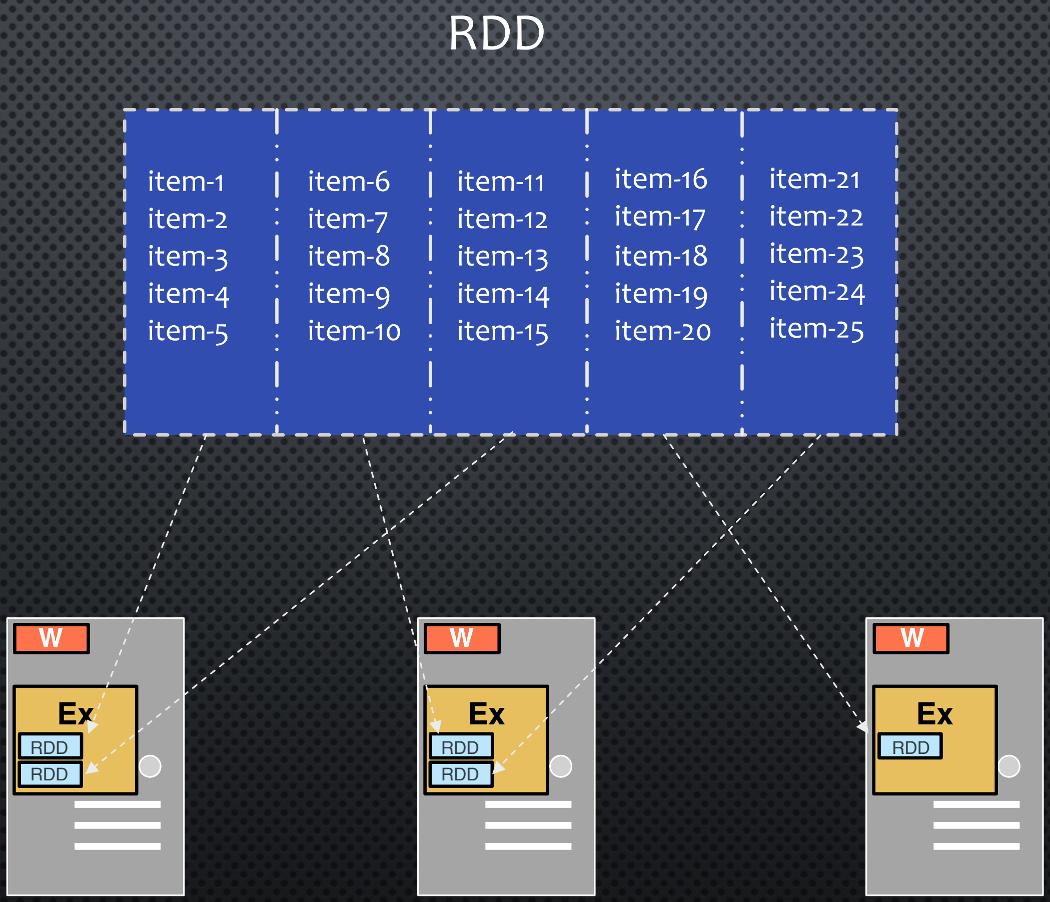


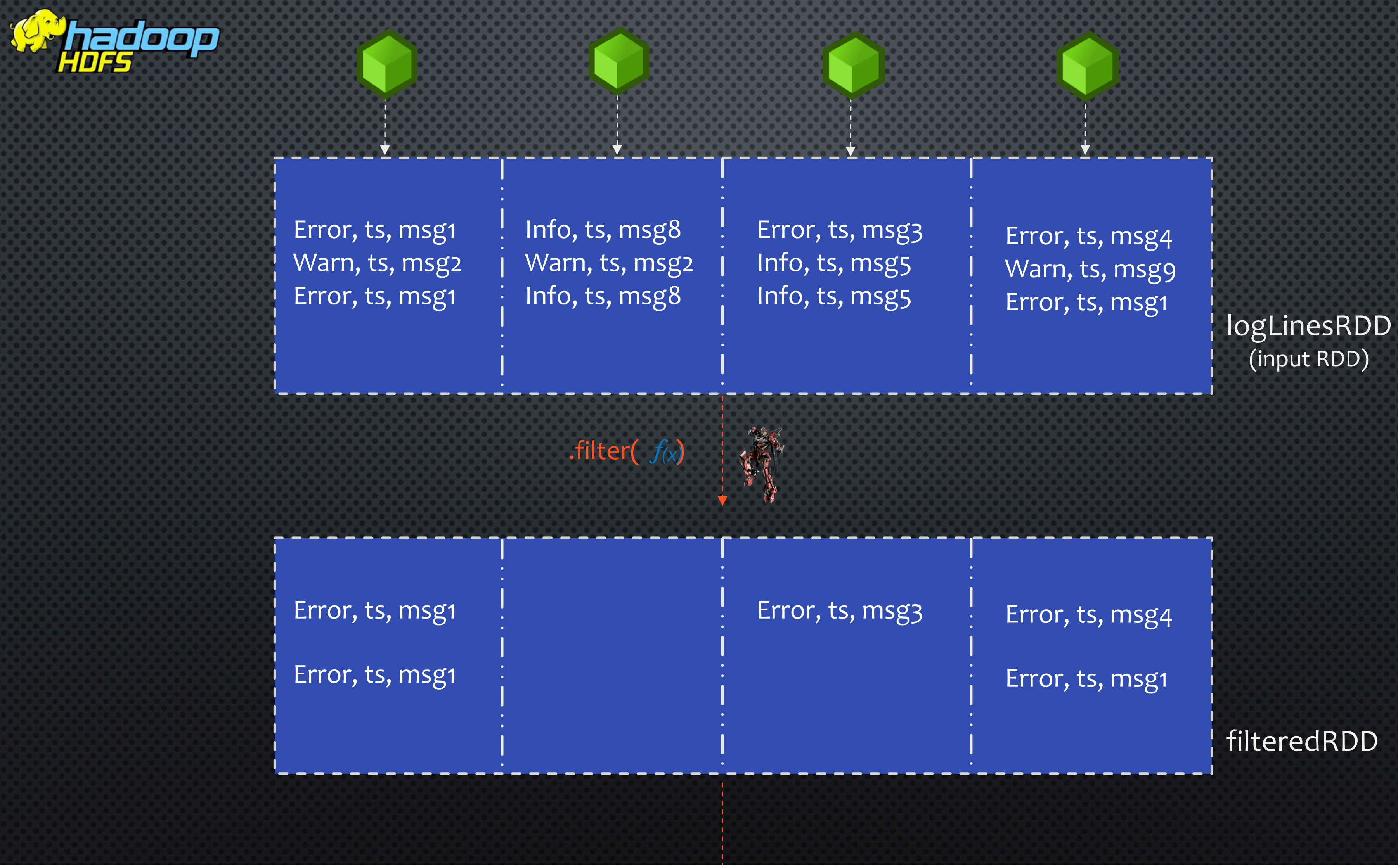


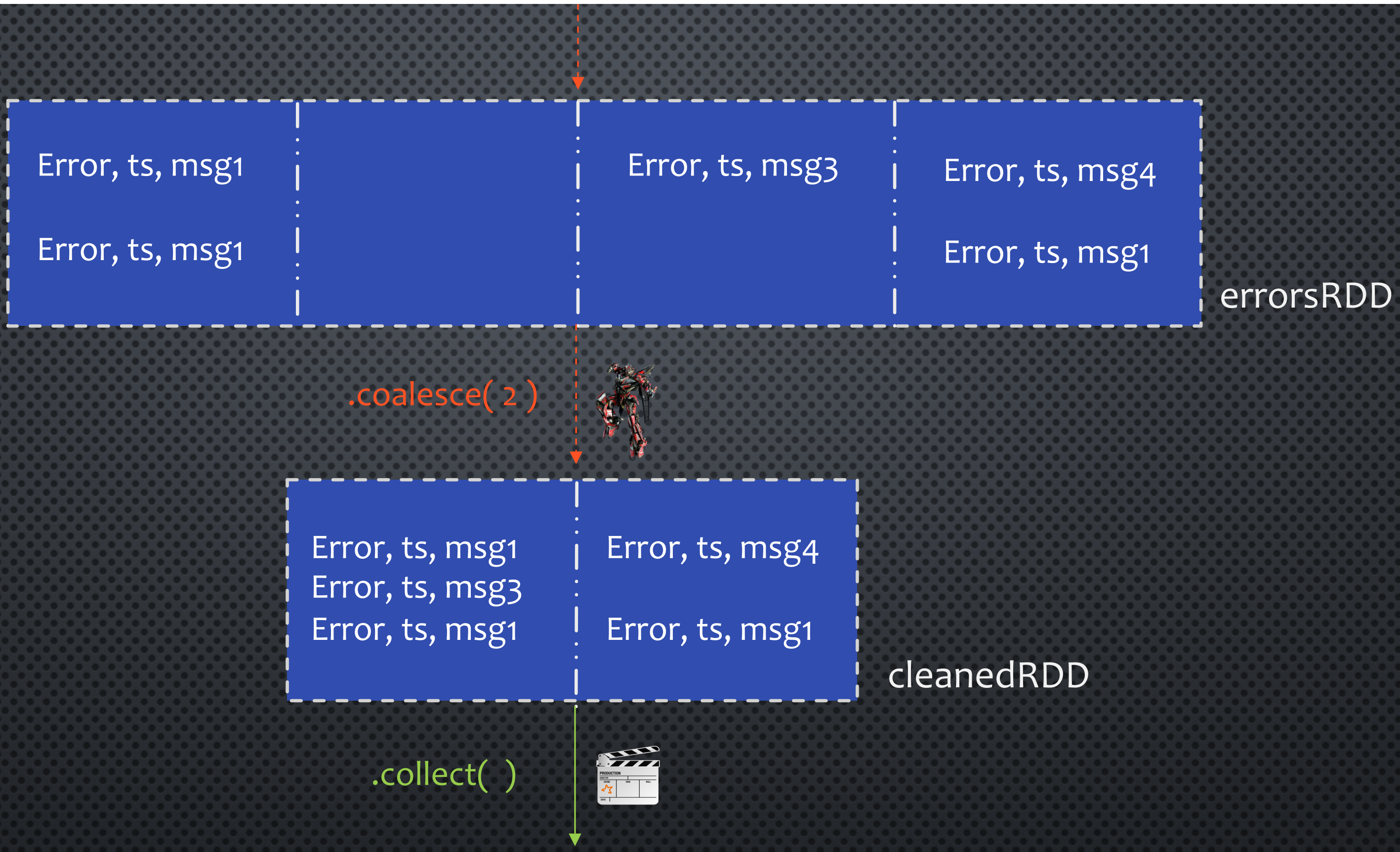


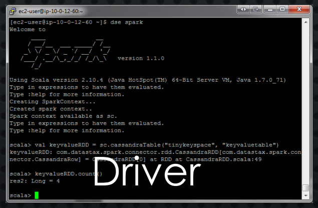


more partitions = more parallelism

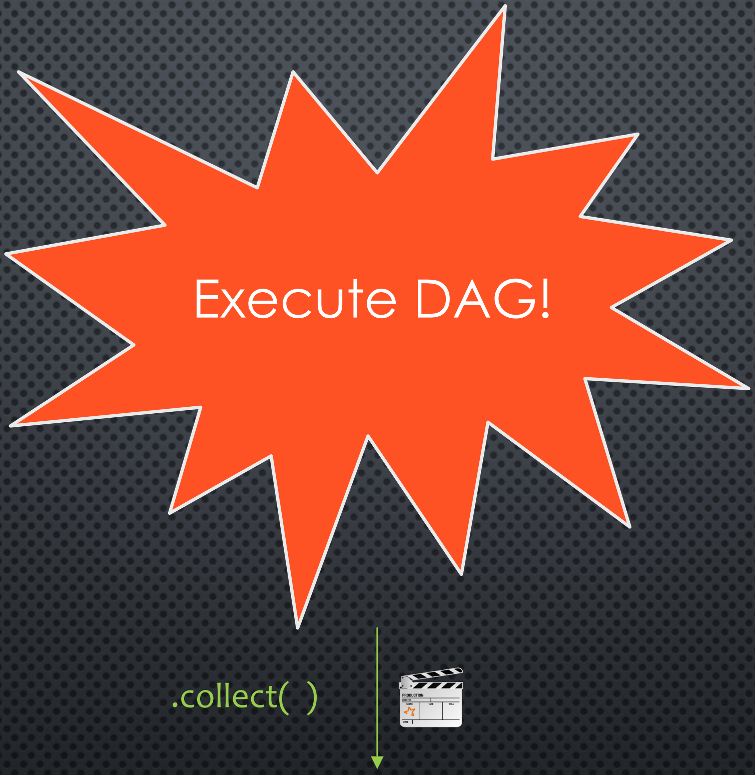


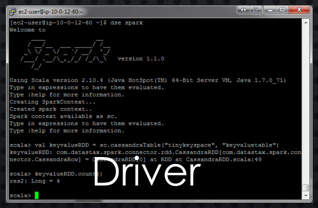

hadoop HDFS
Error, ts, msg1
Warn, ts, msg2
Error, ts, msg1
Info, ts, msg8
Warn, ts, msg2
Info, ts, msg8
Error, ts, msg3
Info, ts, msg5
Info, ts, msg5
Error, ts, msg4
Warn, ts, msg9
Error, ts, msg1
logLinesRDD
(input RDD)
.filter(f(x))
Error, ts, msg1
Error, ts, msg1
Error, ts, msg3
Error, ts, msg4
Error,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



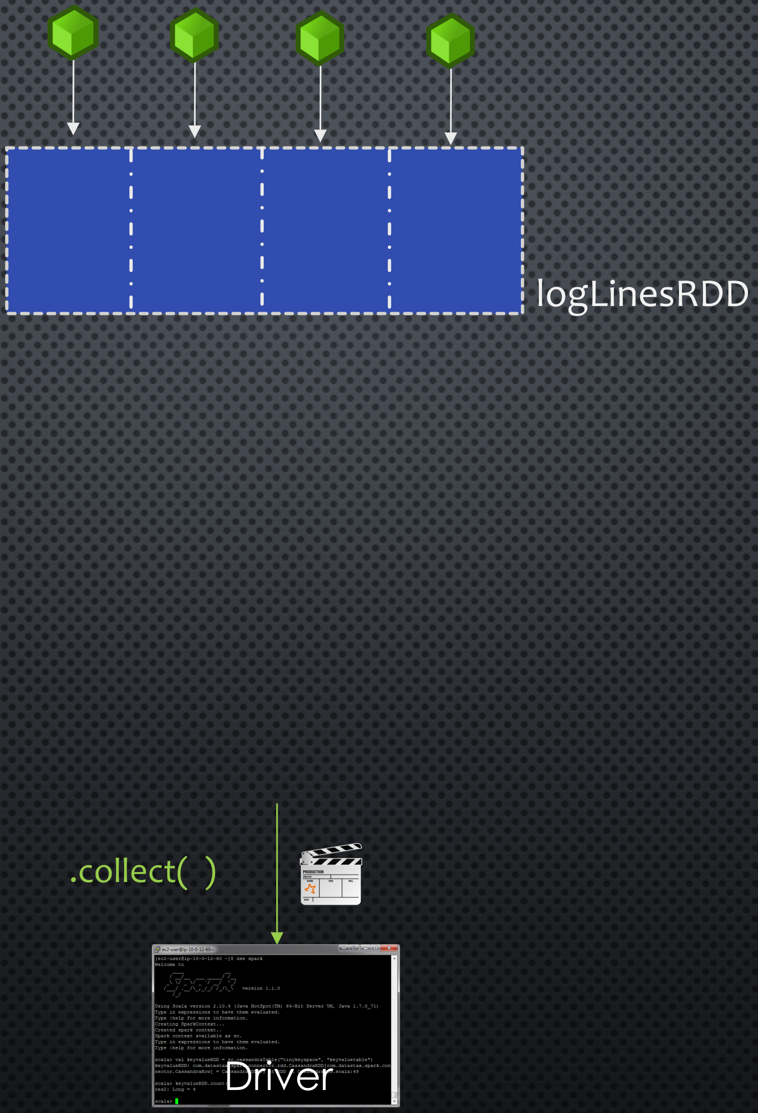


```
[ec2-user@ip-10-0-12-60 ~]$ dse spark
Welcome to
                                  version 1.1.0

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> 
```

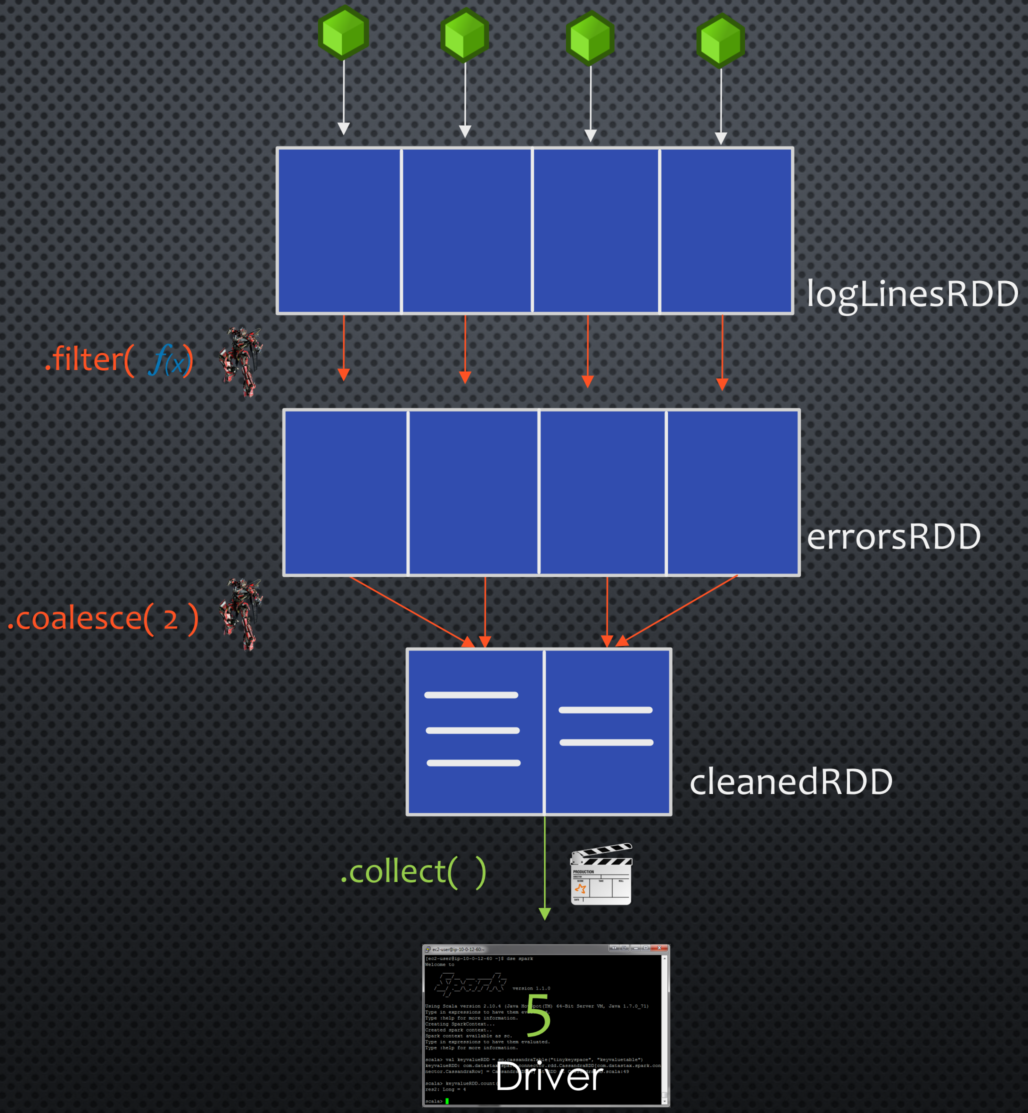




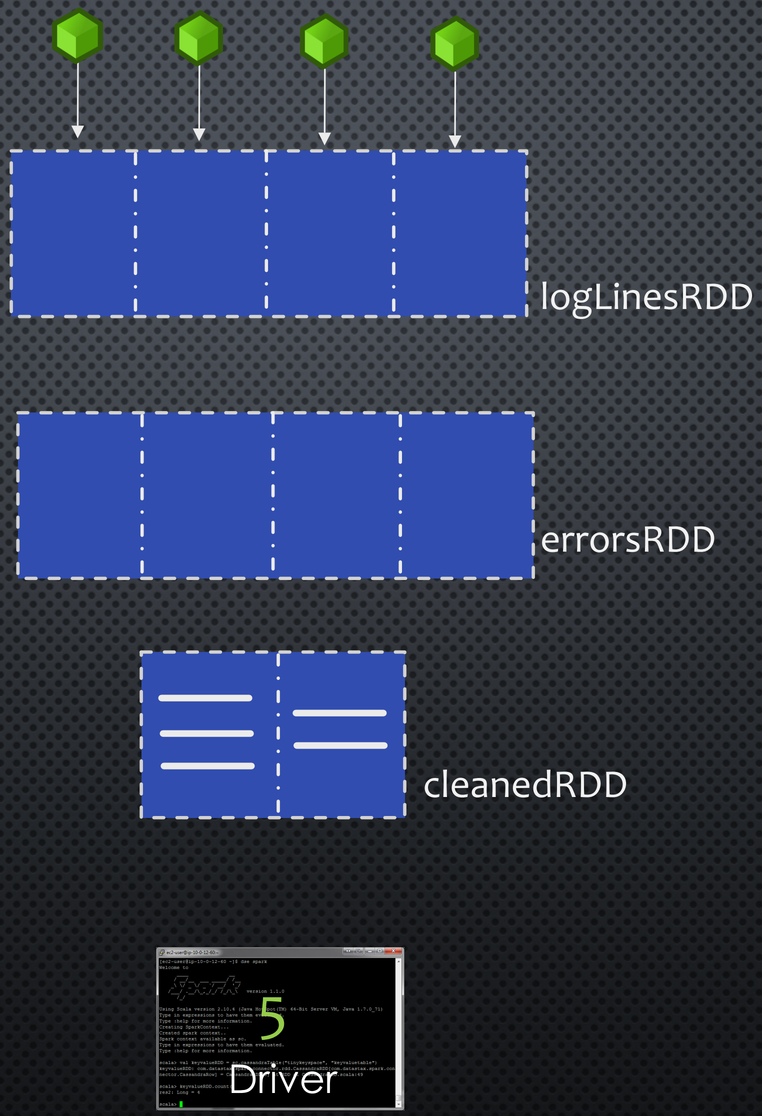
```
[ec2-user@ip-10-0-12-60 ~]$ dse spark
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.1.0
      /_/

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> 
```



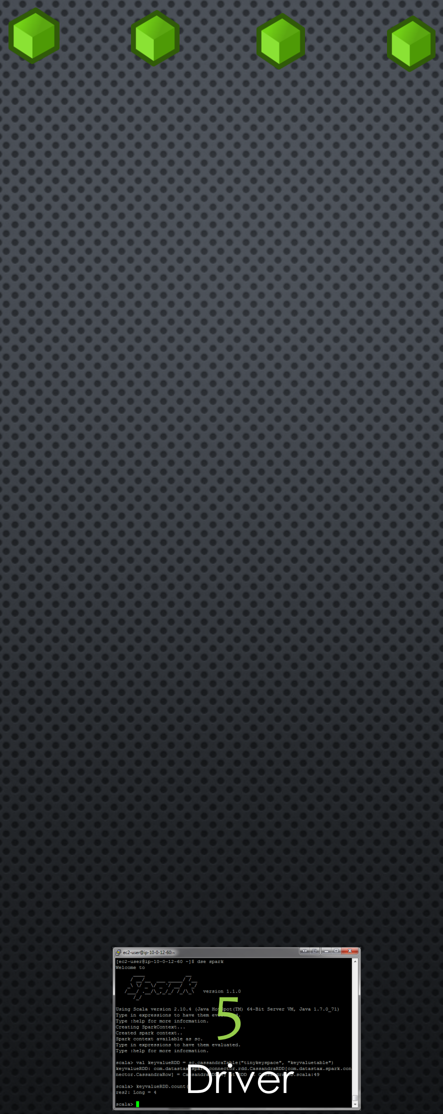
```
ec2-user@ip-10-0-12-60:~
[ec2-user@ip-10-0-12-60 ~]$ dse spark
Welcome to
      Spark  version 1.1.0

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>
```



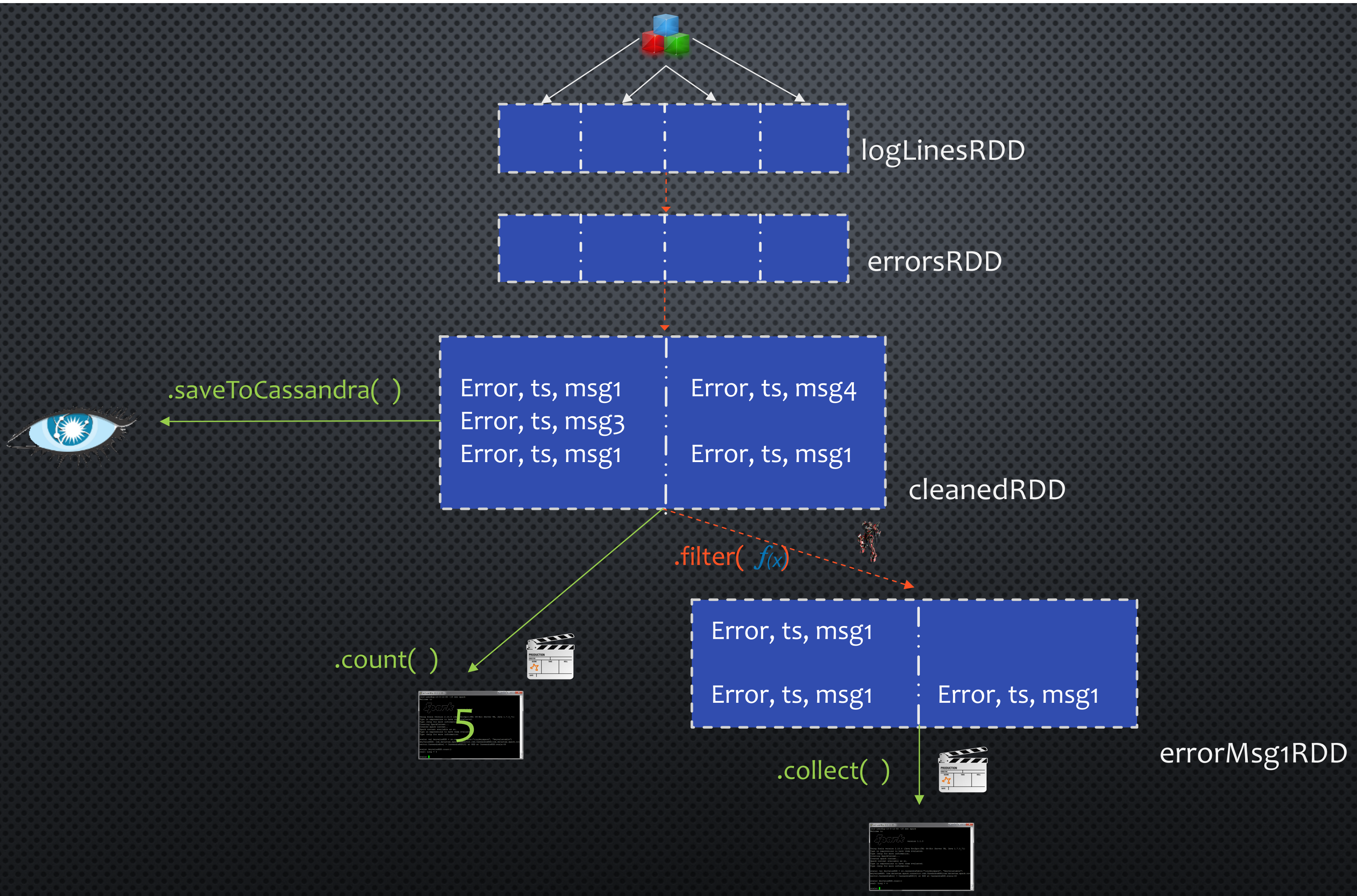
```
[ec2-user@ip-10-0-12-60 ~]$ dse spark
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.1.0
      /_/

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>
```

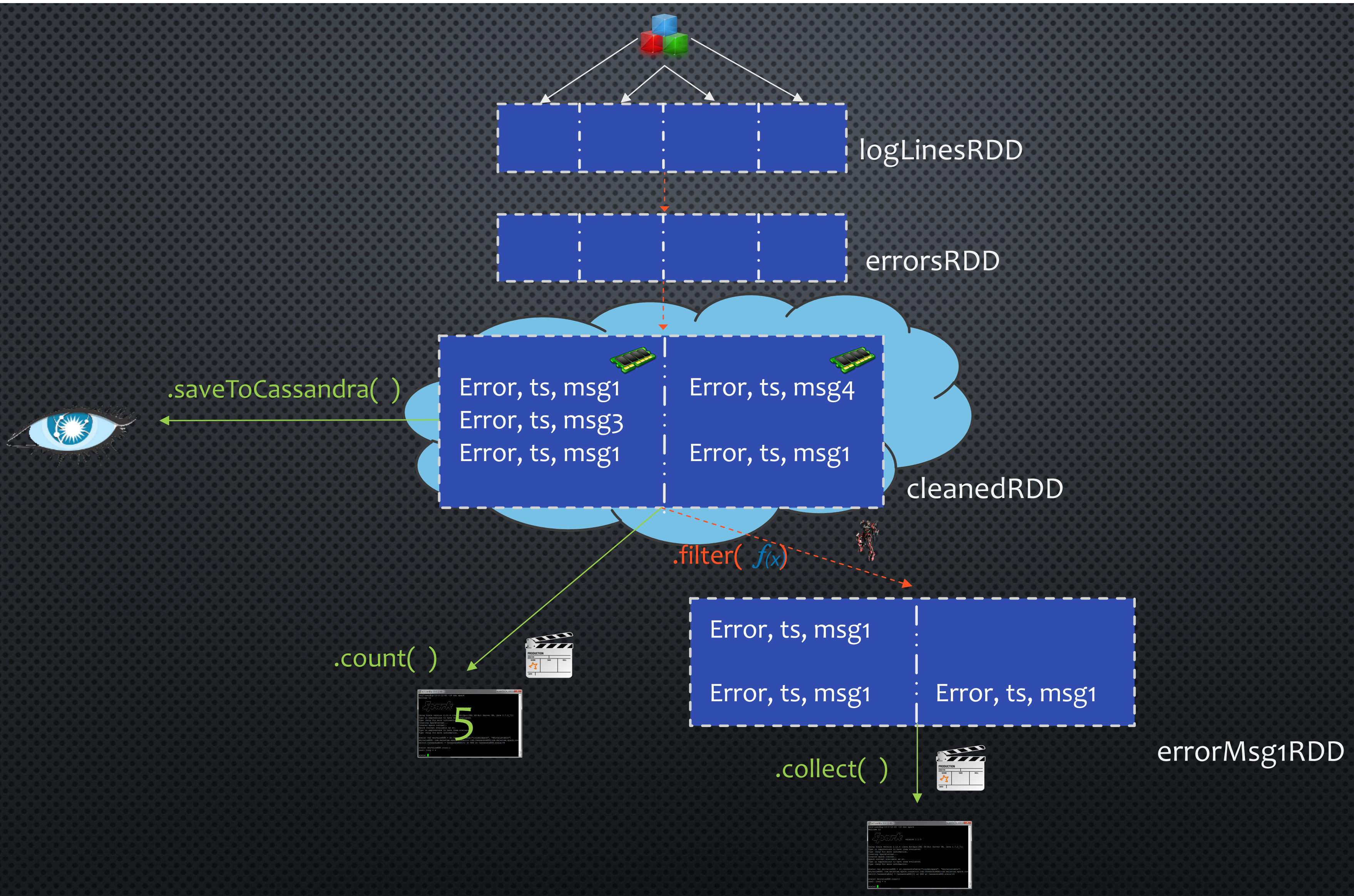


```
ec2-user@ip-10-0-12-60:~
[ec2-user@ip-10-0-12-60 ~]$ dse spark
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.1.0
      /_/

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>
```



```
[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> 
```


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( )

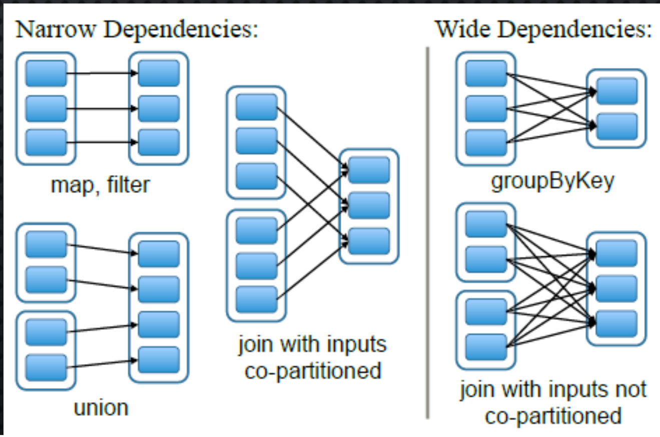

logLinesRDD
errorsRDD
.saveToCassandra( )
Error, ts, msg1
Error, ts, msg3
Error, ts, msg1
Error, ts, msg4
Error, ts, msg1
cleanedRDD
.filter(f(x))
.count( )
5
Error, ts, msg1
Error, ts, msg1
Error, ts, msg1
errorMsg1RDD
.collect( )

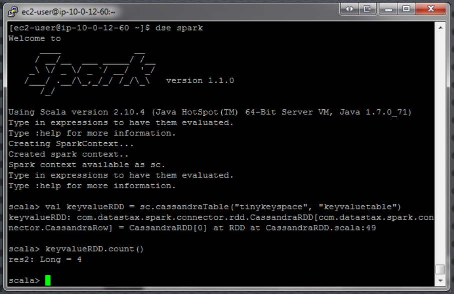
Lazy Evaluation

- Transformations on RDDs are lazily evaluated, meaning that Spark will not begin to execute until it sees an action.
- When a transformation is called, Spark internally records meta-data to indicate this operation has been requested
- Can only be redeemed by an action called upon itself or its downstream RDD.
- Lazy evaluation is used to reduce the # of passes it has to take over the data by grouping operations together
- Note, in MapReduce developers have to spend a lot of time thinking about how to group together operations to minimize the # of MR passes
- In Spark, there's no substantial benefit to writing a single complex map, instead of chaining together many simple operations
- A task with all its dependencies are largely omitted if the partition it generates is already cached.

Lineage Graph

- ➢ A graph-based representation for RDDs
- ➢ Pieces of information for each RDD
  - ✓ a set of partitions
  - ✓ a set of dependencies on parent RDDs
  - ✓ a function for computing it from its parents
  - ✓ metadata about its partitioning scheme and data placement
- ➢ Narrow dependencies
  - ✓ each partition of the parent RDD is used by at most one partition of the child RDD
- ➢ Wide dependencies
  - ✓ multiple child partitions may depend on it
- ➢RDDs will be rebuilt on failures using *lineage graph*


Narrow Dependencies:
Wide Dependencies:
map, filter
union
join with inputs co-partitioned
groupByKey
join with inputs not co-partitioned



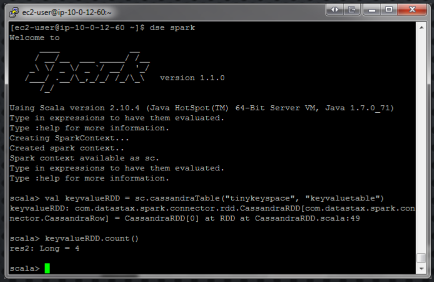
```
[ec2-user@ip-10-0-12-60 ~]$ dse spark
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.1.0
      /_/

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> 
```

```
// Read input file
scala> val input = sc.textFile("input.txt")
```

```
// Split into words and remove empty lines
scala> val tokenized = input.map(line => line.split(" ")).filter(words => words.size > 0)
```

```
// Extract the first word from each line (the log level) and do a count
scala> val counts = tokenized.map(words => (words(0), 1)).reduceByKey{ case a, b) => a + b }
```

To display the lineage of an RDD, Spark provides a `toDebugString` method:



```
[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>
```

```
scala> input.toDebugString

res85: String =
(2) data.text MappedRDD[292] at textFile at <console>:13
 | data.text HadoopRDD[291] at textFile at <console>:13

scala> counts.toDebugString
res84: String =
(2) ShuffledRDD[296] at reduceByKey at <console>:17
 +-(2) MappedRDD[295] at map at <console>:17
    | FilteredRDD[294] at filter at <console>:15
    | MappedRDD[293] at map at <console>:15
    | data.text MappedRDD[292] at textFile at <console>:13
    | data.text HadoopRDD[291] at textFile at <console>:13
```

LINEAGE

- Spark's internal scheduler may truncate the lineage of the RDD graph if an existing RDD has already been persisted in cluster memory or on disk.
- Spark can "short circuit" in this case and just begin computing based on the persisted RDD.
- A second case when this truncation can happen is when an RDD is already materialized as a side-effect of an earlier shuffle, even if it was not explicitly persist() 'ed. This is an under-the-hood optimization that takes advantage of the fact that Spark shuffle outputs are written to disk, and exploits the fact that many times portions of the RDD graph are re-computed.

RDD Interface

Scientific Answer: RDD is an Interface!

1. Set of *partitions* (“aka splits in Hadoop”)
2. List of *dependencies* on parent RDDs
3. Function to *compute* a partition given its parent(s)
4. Optional *preferred locations*
5. Optional *partitioning info* (Partitioner)

“Lineage”

“Optimized Execution”