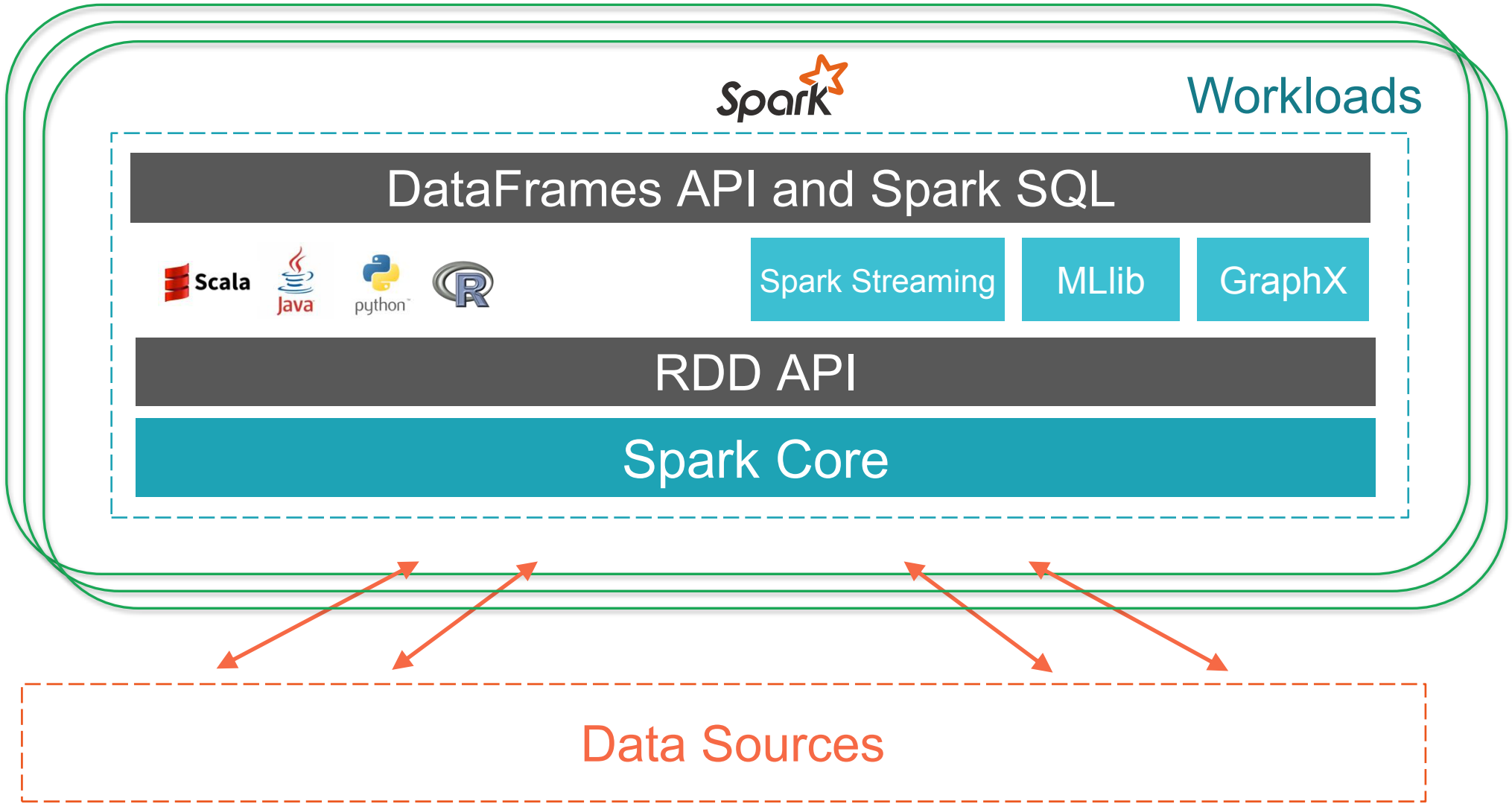
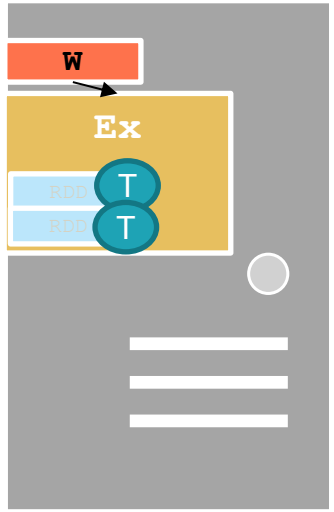
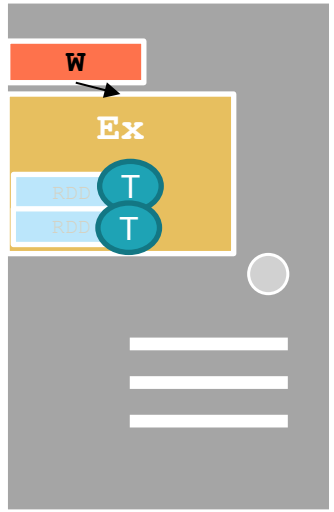
RDD Fundamentals


Spark
Workloads
DataFrames API and Spark SQL
Scala
Java
python
Spark Streaming
MLlib
GraphX
RDD API
Spark Core
Data Sources

Driver Program



Worker Machine

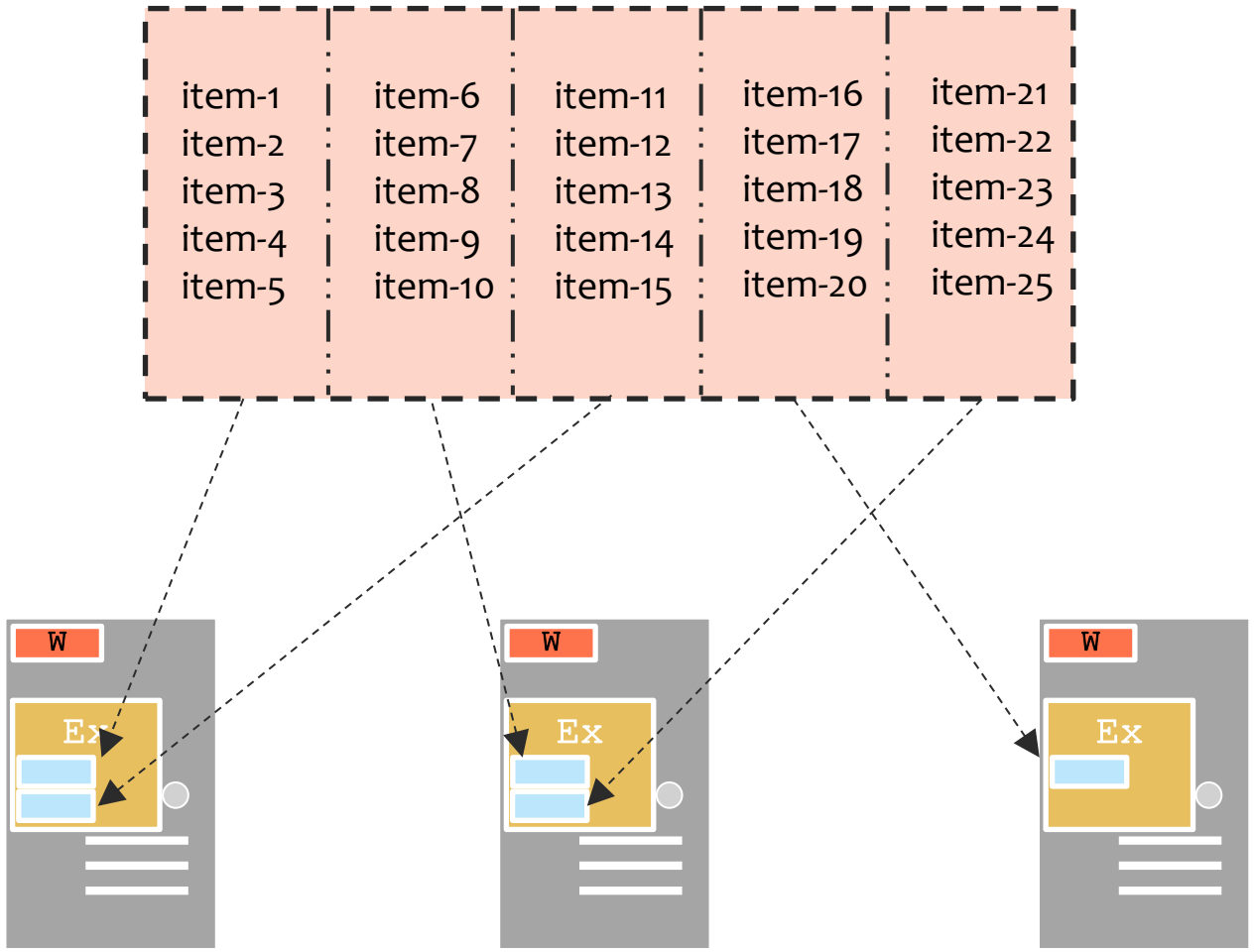


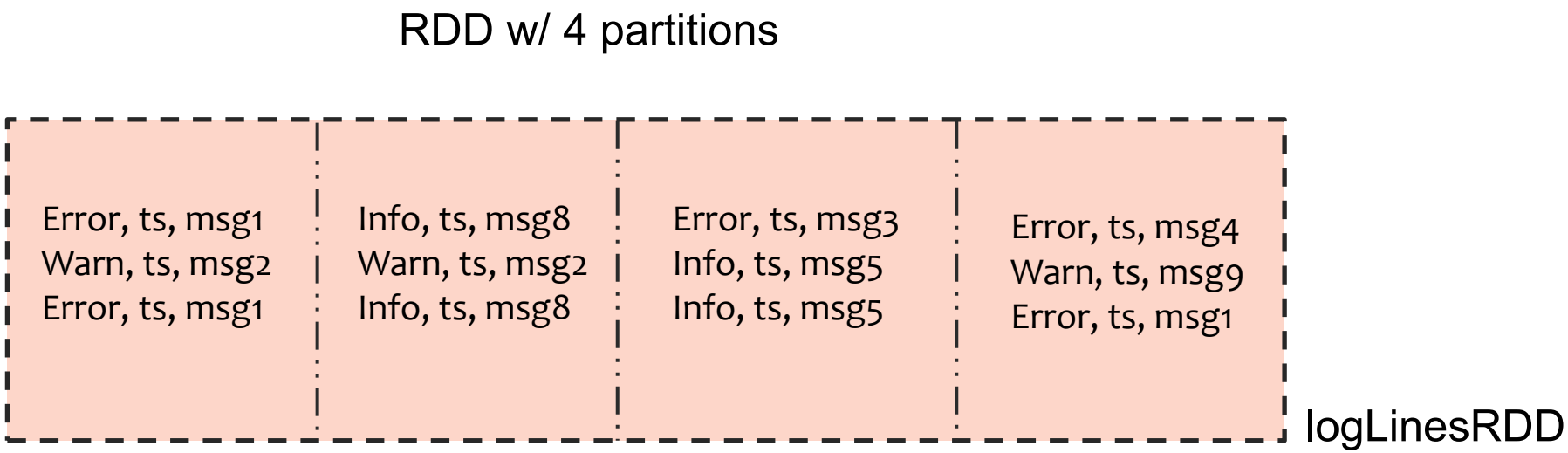
Worker Machine

Resilient Distributed Datasets (RDDs)

- Write programs in terms of operations on distributed data
- Partitioned collections of objects spread across a cluster
- Diverse set of parallel transformations and actions
- Fault tolerant

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
W
Ex
W
Ex



A base RDD can be created 2 ways:

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

Create a Base RDD

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

Parallelize

Take an existing in-memory collection and pass it to SparkContext's parallelize method

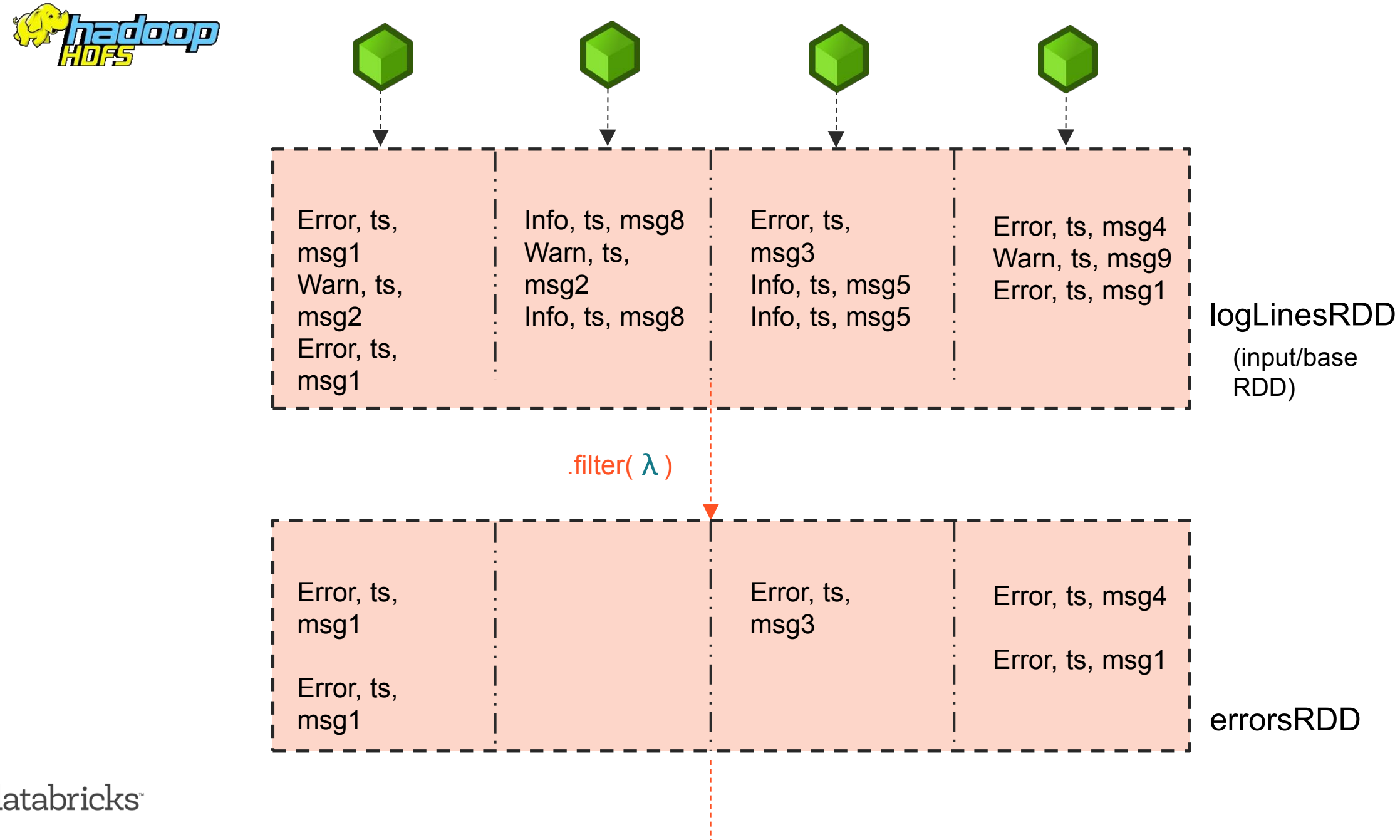
```
# Read a local txt file in Python
linesRDD = sc.textFile("/path/to/README.md")
```

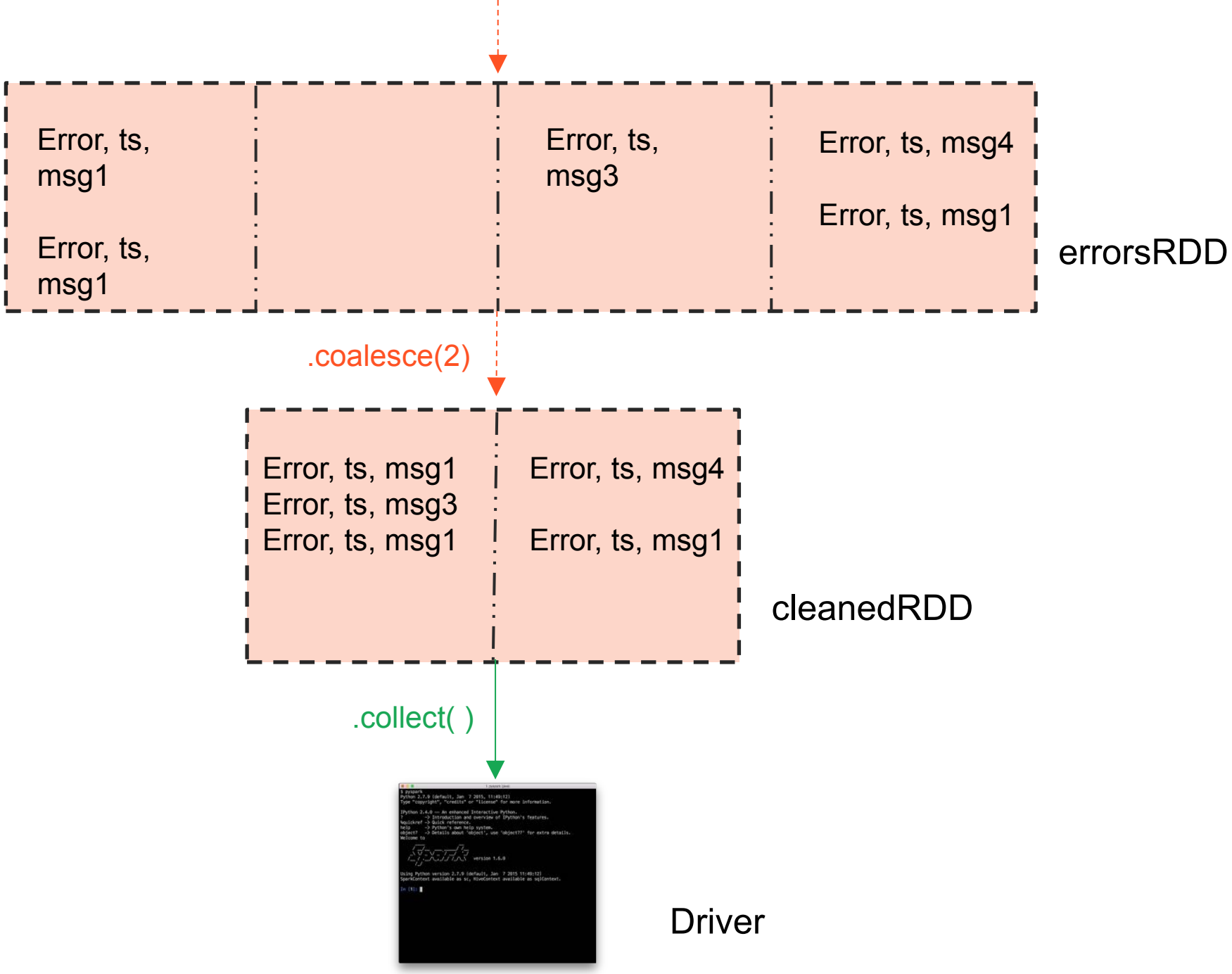
Read from Text File

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

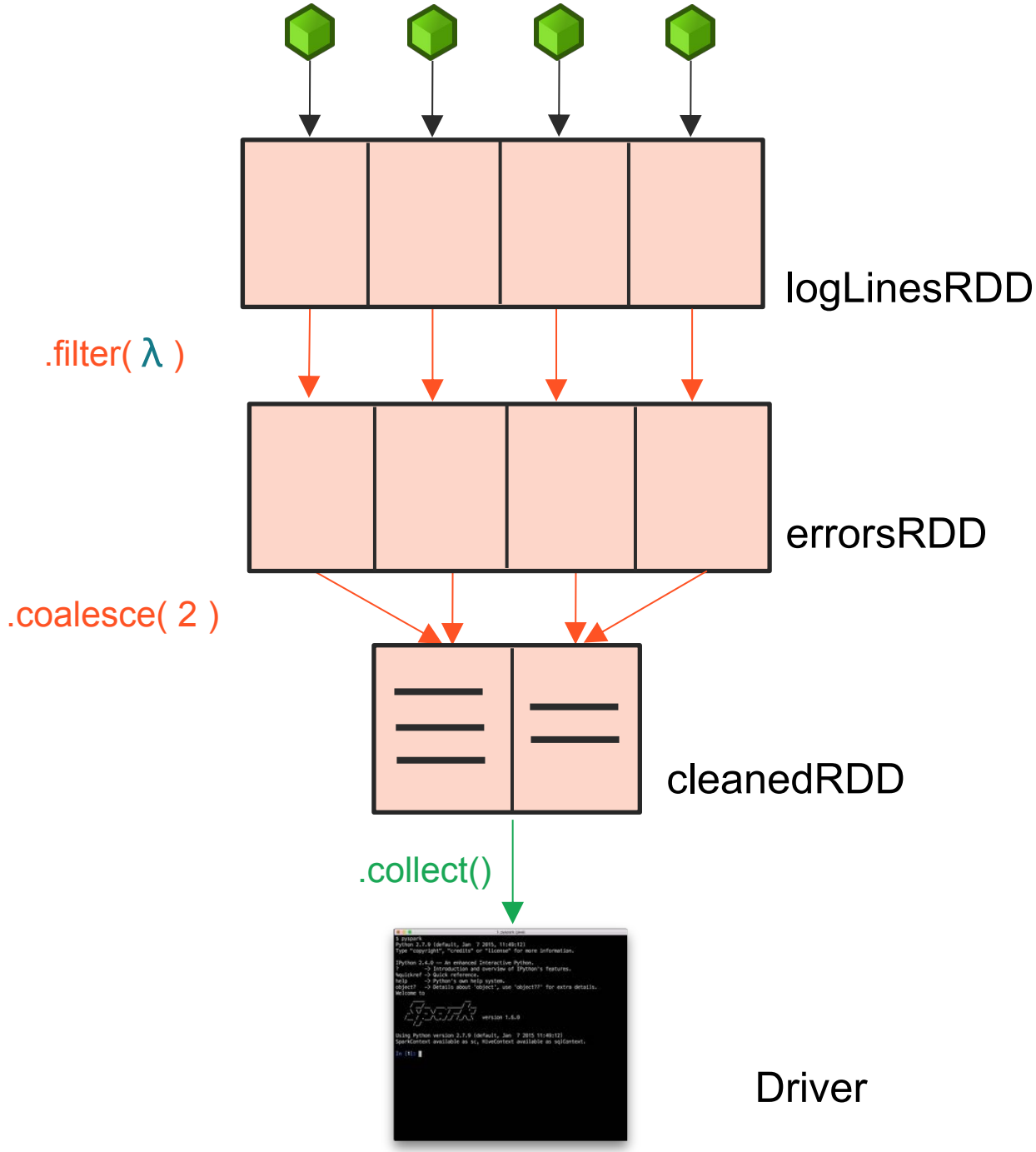
Operations on Distributed Data

- Two types of operations: *transformations* and *actions*
- Transformations are lazy (*not computed immediately*)
- Transformations are executed when an action is run
- Persist (cache) distributed data in memory or disk

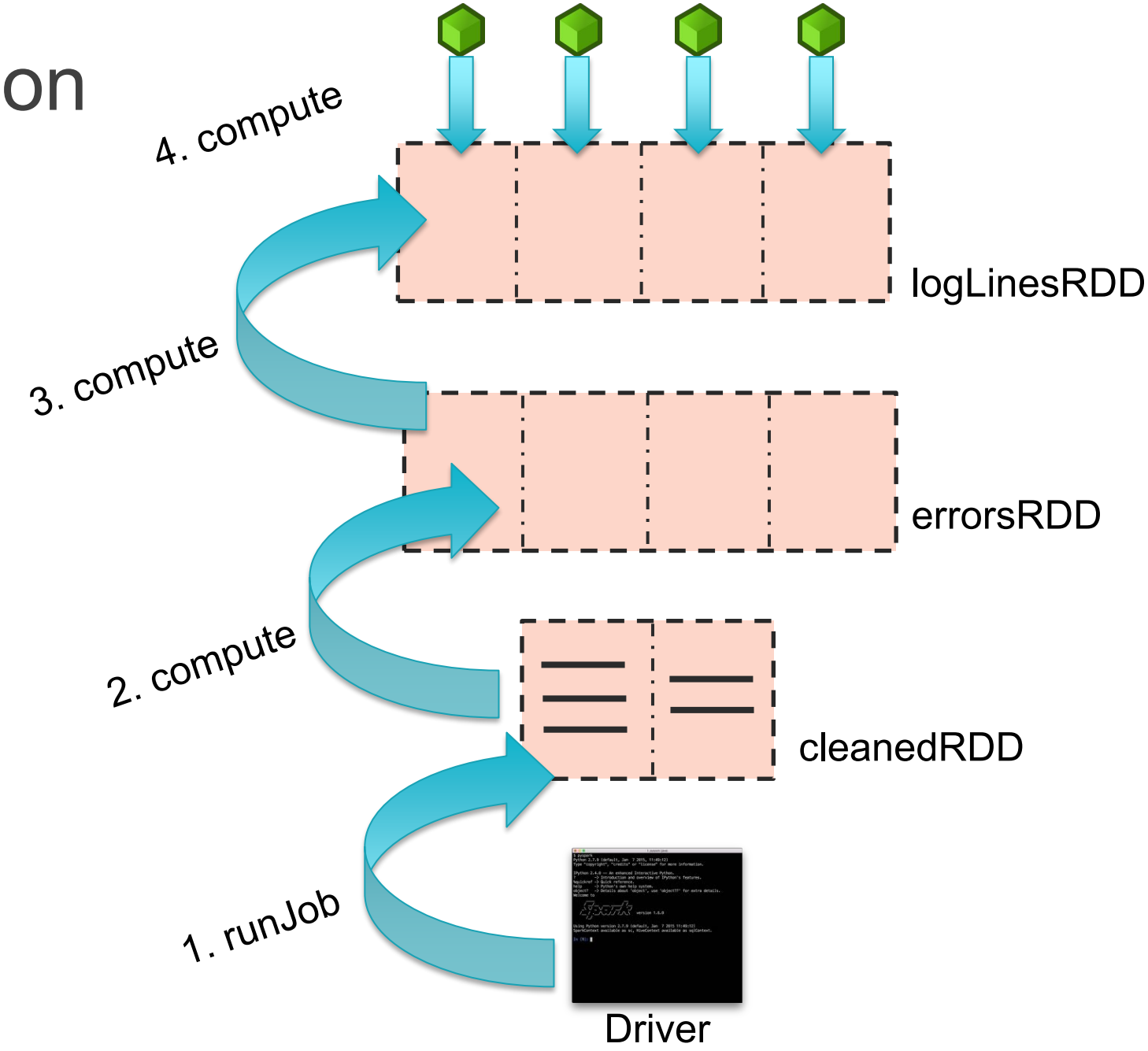

hadoop
HDFS
Error, ts, msg1
Warn, ts, msg2
Error, ts, msg1
Info, ts, msg8
Warn, ts, msg2
Info, ts, msg8
Error, ts, msg3
Info, ts, msg5
Info, ts, msg5
Error, ts, msg4
Warn, ts, msg9
Error, ts, msg1
logLinesRDD
(input/base RDD)
.filter(λ)
Error, ts, msg1
Error, ts, msg1
Error, ts, msg3
Error, ts, msg4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

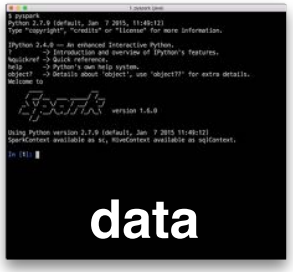

Error, ts, msg1
Error, ts, msg1
Error, ts, msg3
Error, ts, msg4
Error, ts, msg1
errorsRDD
.coalesce(2)
Error, ts, msg1
Error, ts, msg3
Error, ts, msg1
Error, ts, msg4
Error, ts, msg1
cleanedRDD
.collect( )
Driver

DAG


logLinesRDD
.filter(λ)
errorsRDD
.coalesce(2)
cleanedRDD
.collect()
Driver

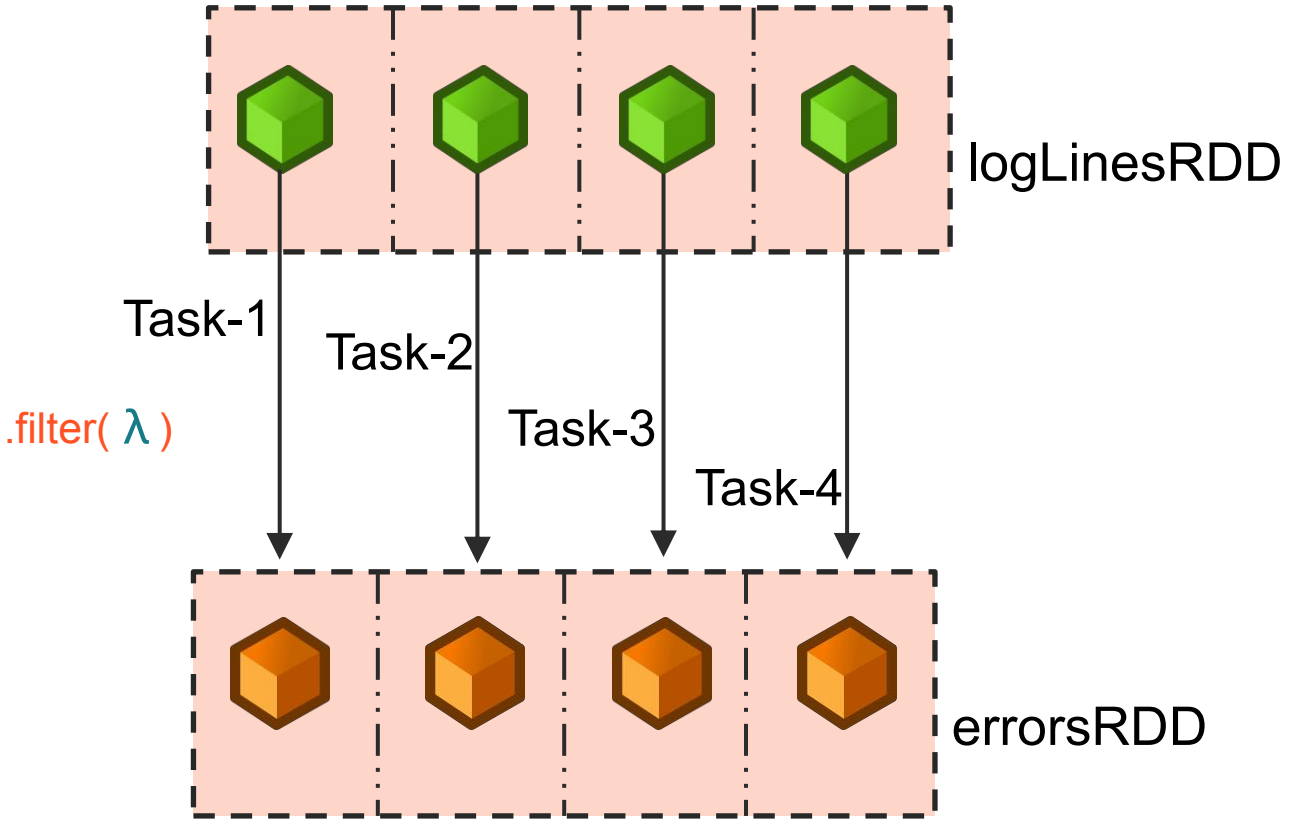
Execution


4. compute
logLinesRDD
3. compute
errorsRDD
2. compute
cleanedRDD
1. runJob
Driver



Driver

Partition >>> Task >>> Partition



Lifecycle of an RDD-based Spark Program

1) Create base RDD

2) Chain together transformations

3) Cache intermediate RDDs

4) Perform actions

Transformations

```
map()                intersection()   zipWithIndex()

flatMap()            distinct()       pipe()

filter()             groupByKey()     coalesce()

mapPartitions()      reduceByKey()    ...
```

Actions

```
reduce()

collect()

count()

first()
```

```
take()

takeOrdered()

saveAsTextFile()

...
```

RDDs vs DataFrames

- RDDs provide a low-level interface into Apache Spark
- DataFrames have a schema
- DataFrames are cached using Tungsten format
- DataFrames are optimized via Catalyst
- DataFrames are built on top of the RDD and core APIs