



An Overview of Apache Spark

Carol McDonald



Find my presentation and other related resources here:

<http://events.mapr.com/JacksonvilleJava>

(you can find this link in the event's page at meetup.com)



Today's Presentation



Free On-Demand Training



Whiteboard & demo
videos



Free eBooks



Free Hadoop Sandbox



And more...

MAPR[®]



Agenda

- MapReduce Refresher
- What is Spark?
- The Difference with Spark
- Examples and Resources





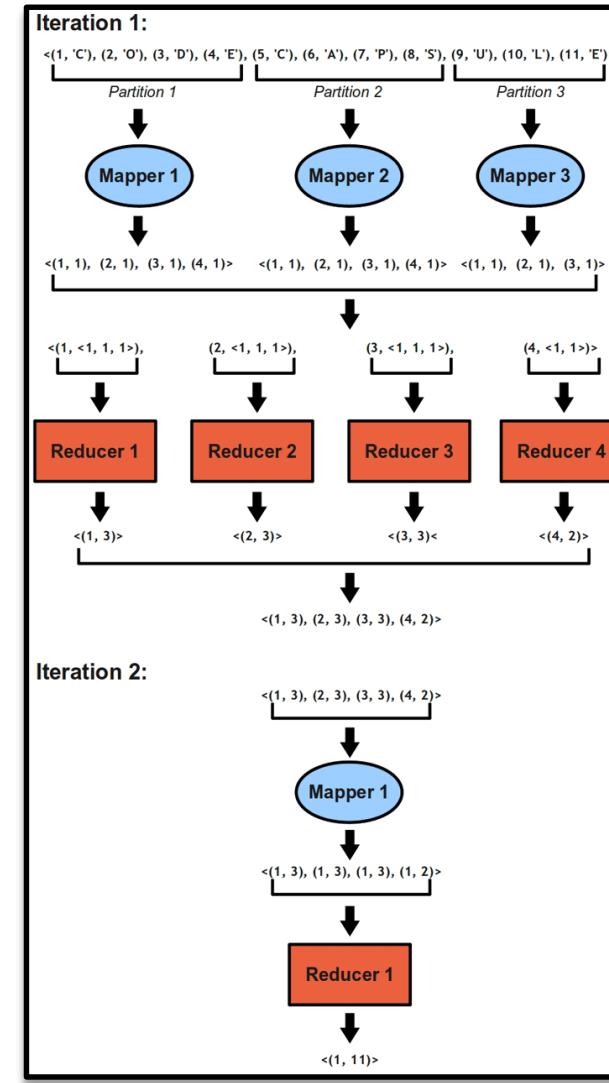
MapReduce Refresher





MapReduce: A Programming Model

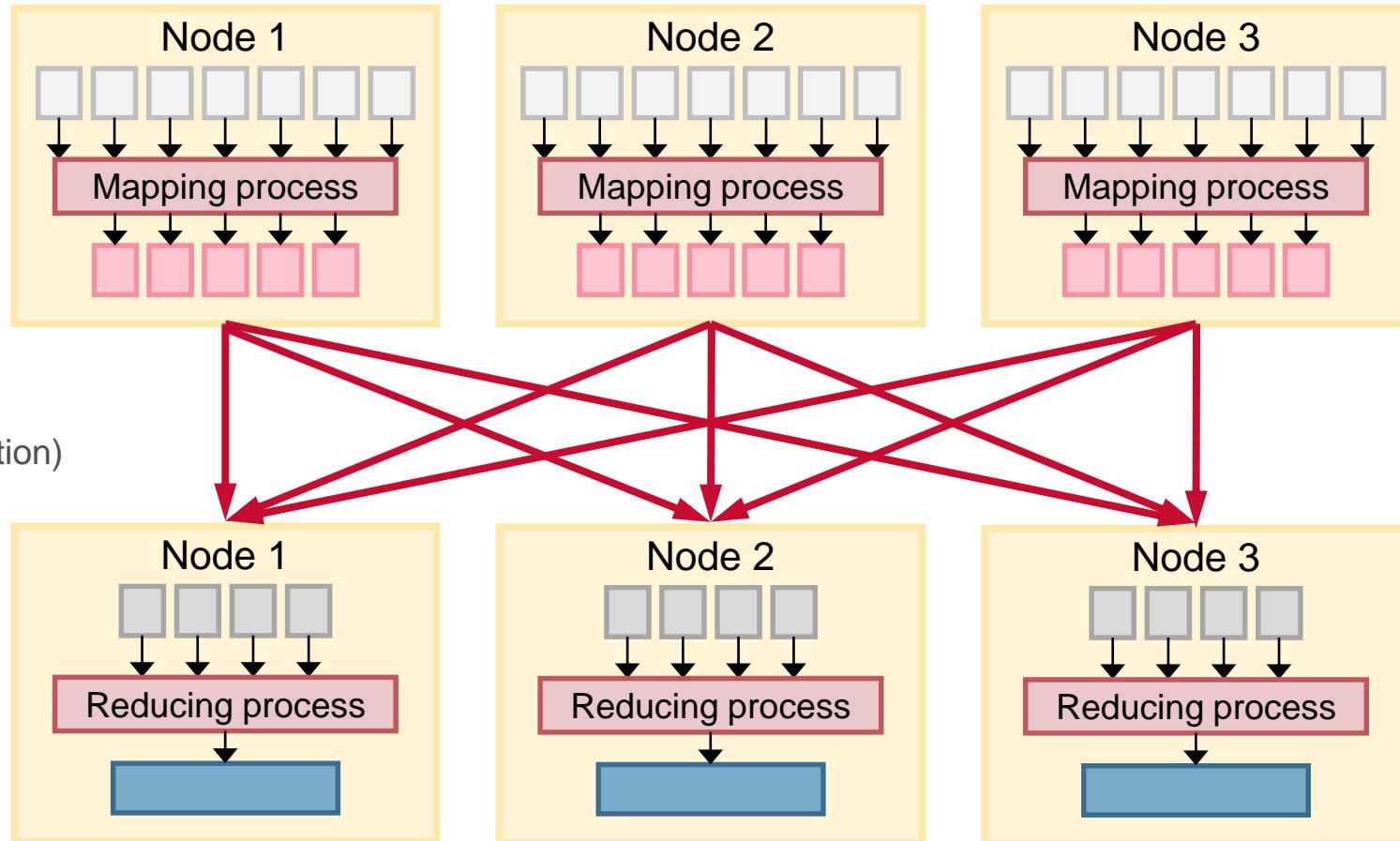
- **MapReduce:**
Simplified Data Processing on Large Clusters
(published 2004)
- **Parallel and Distributed Algorithm:**
 - Data Locality
 - Fault Tolerance
 - Linear Scalability





The Hadoop Strategy

Distribute data
(share nothing)

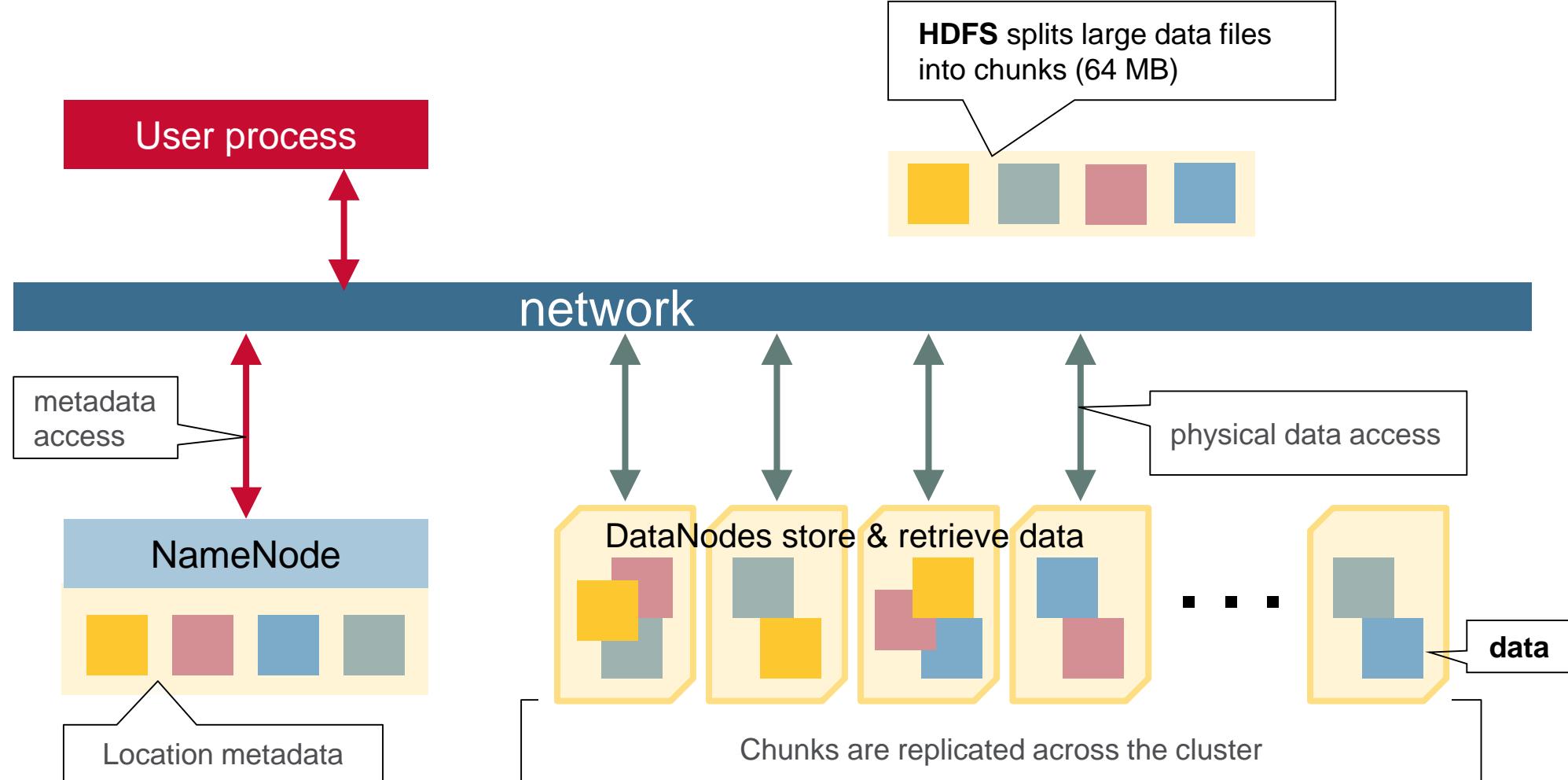


<http://developer.yahoo.com/hadoop/tutorial/module4.html>



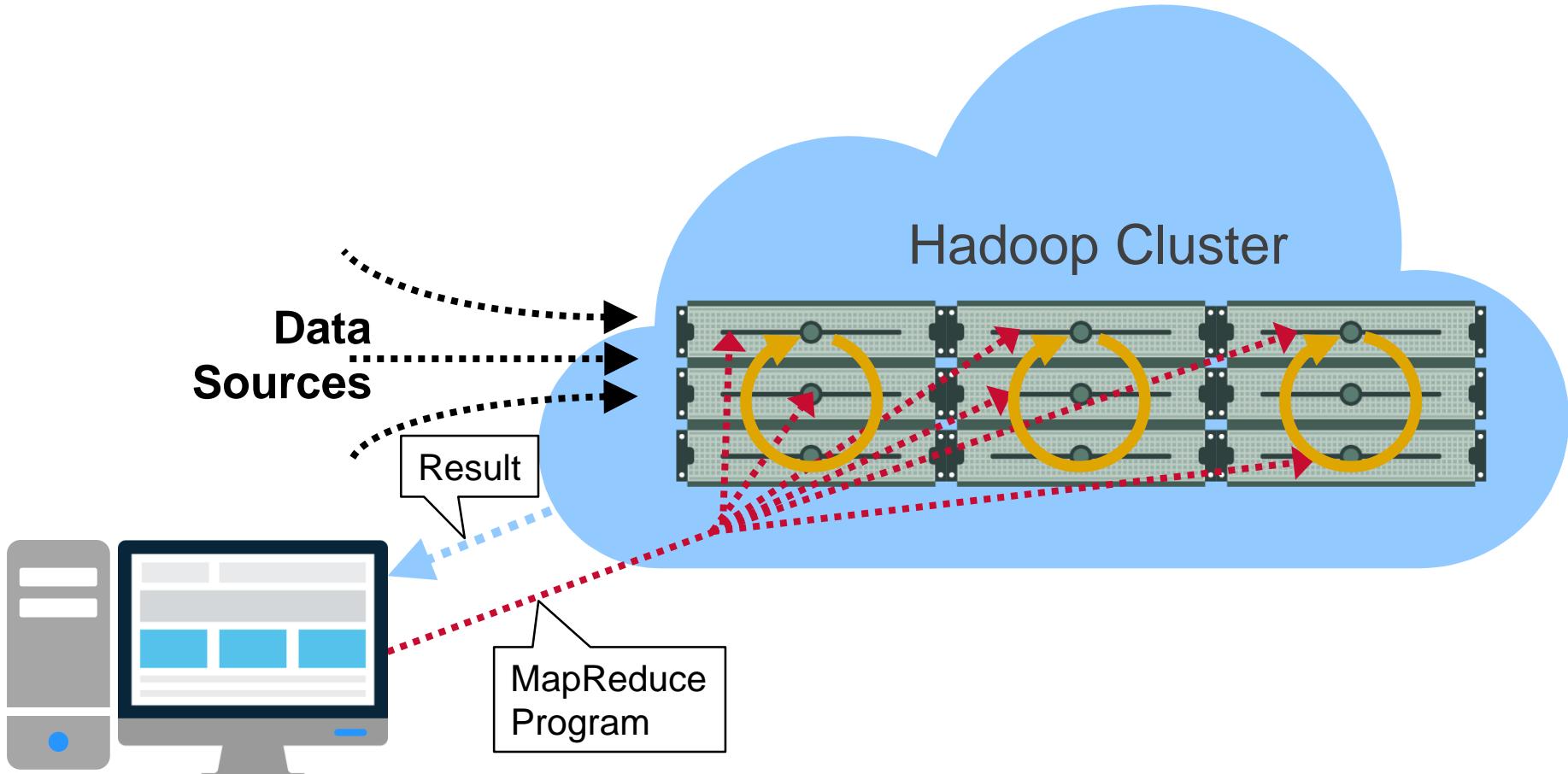


Distribute Data: HDFS





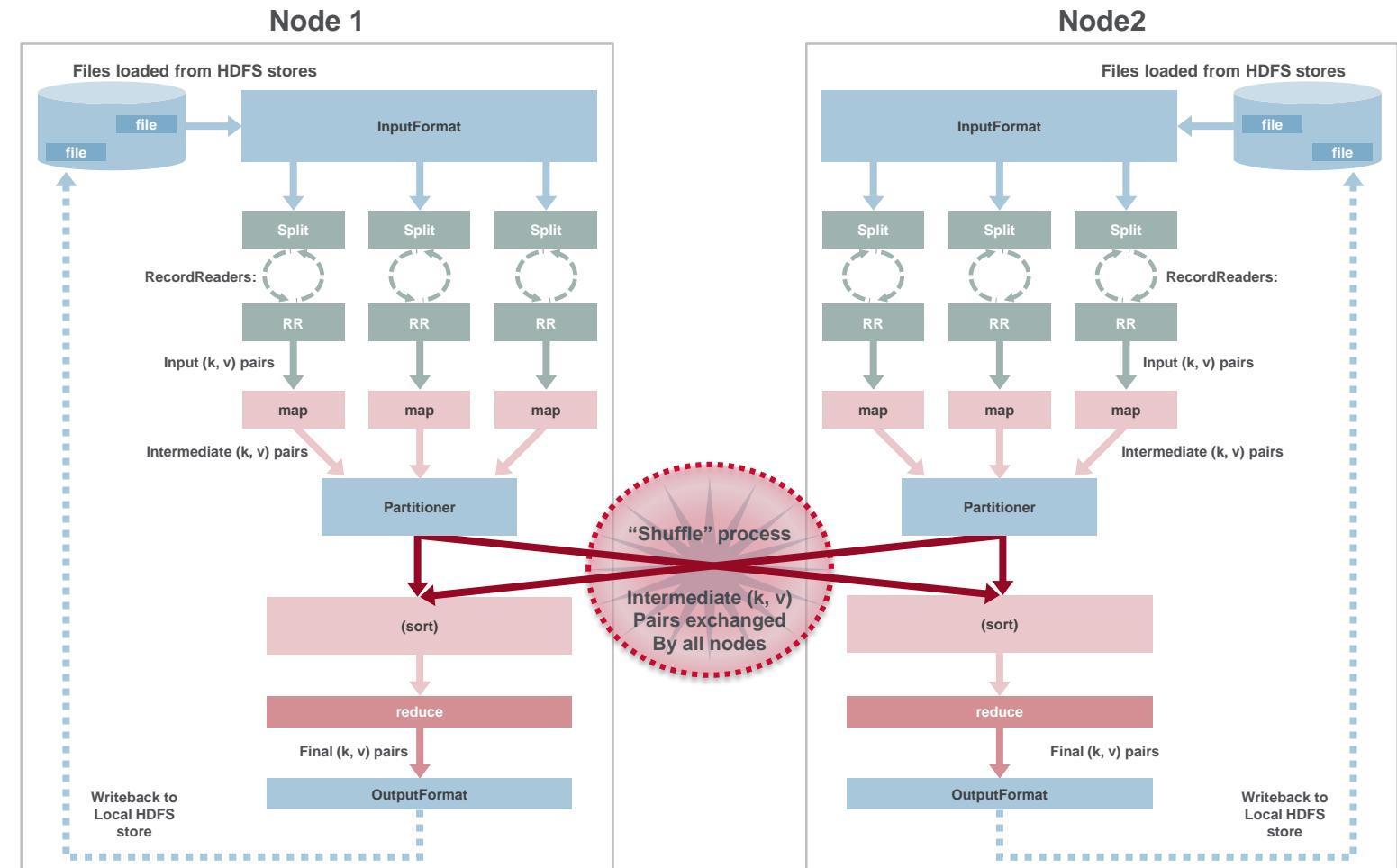
Distribute Computation





MapReduce Execution and Data Flow

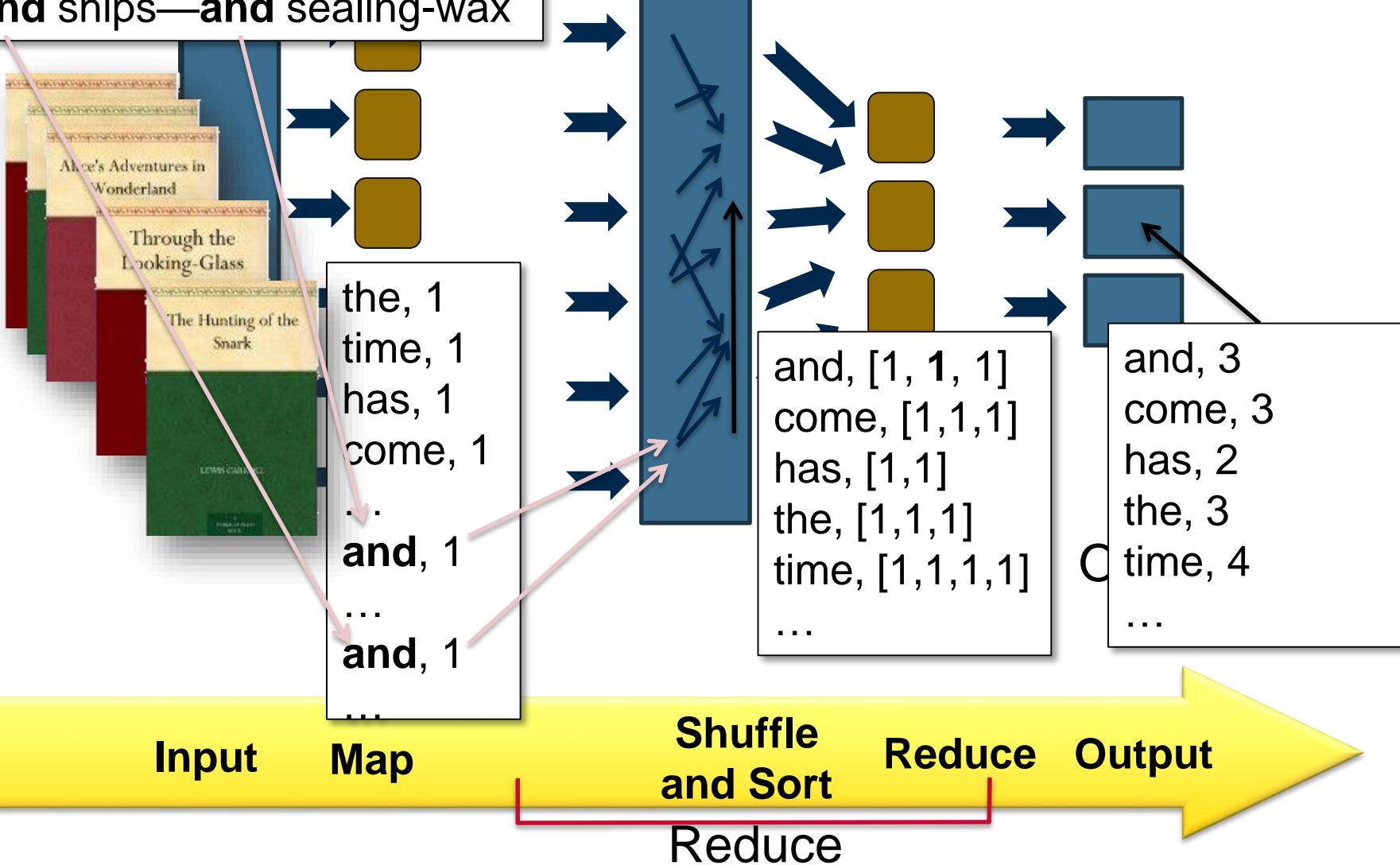
- Map
 - Loading data and **defining a key values**
- Shuffle
 - Sorts, collects key-based data
- Reduce
 - Receives **(key, list)**'s to process and output





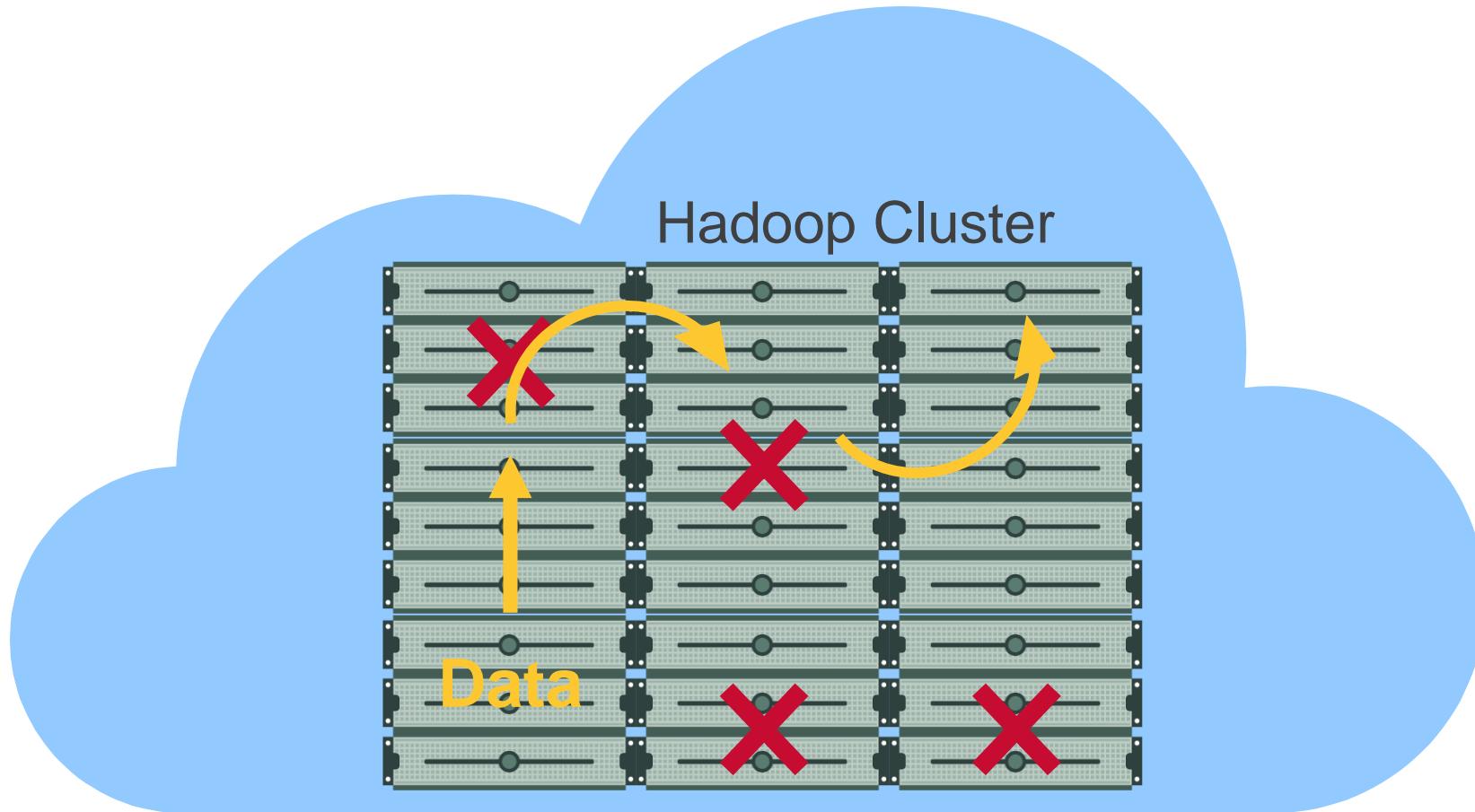
MapReduce Example: Word Count

"The time has come," the Walrus said,
"To talk of many things:
Of shoes—and ships—and sealing-wax





Tolerate Failures



Failures are expected & managed gracefully

DataNode fails -> name node will locate replica

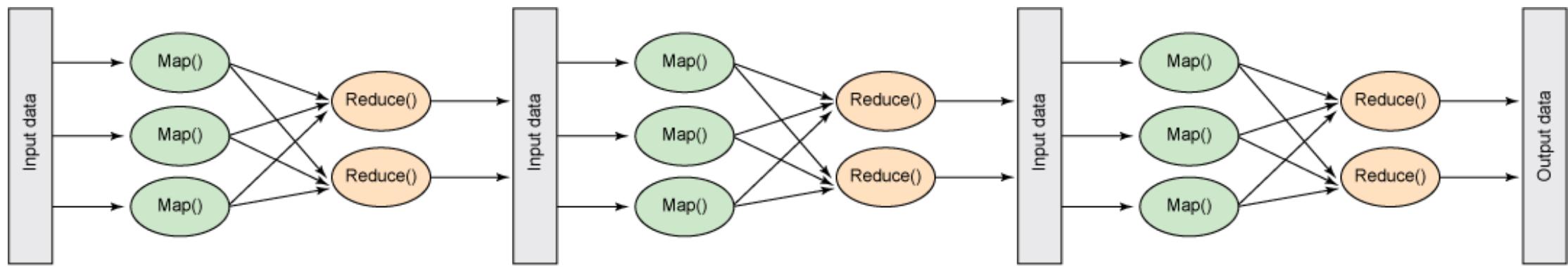
MapReduce task fails -> job tracker will schedule another one





MapReduce Processing Model

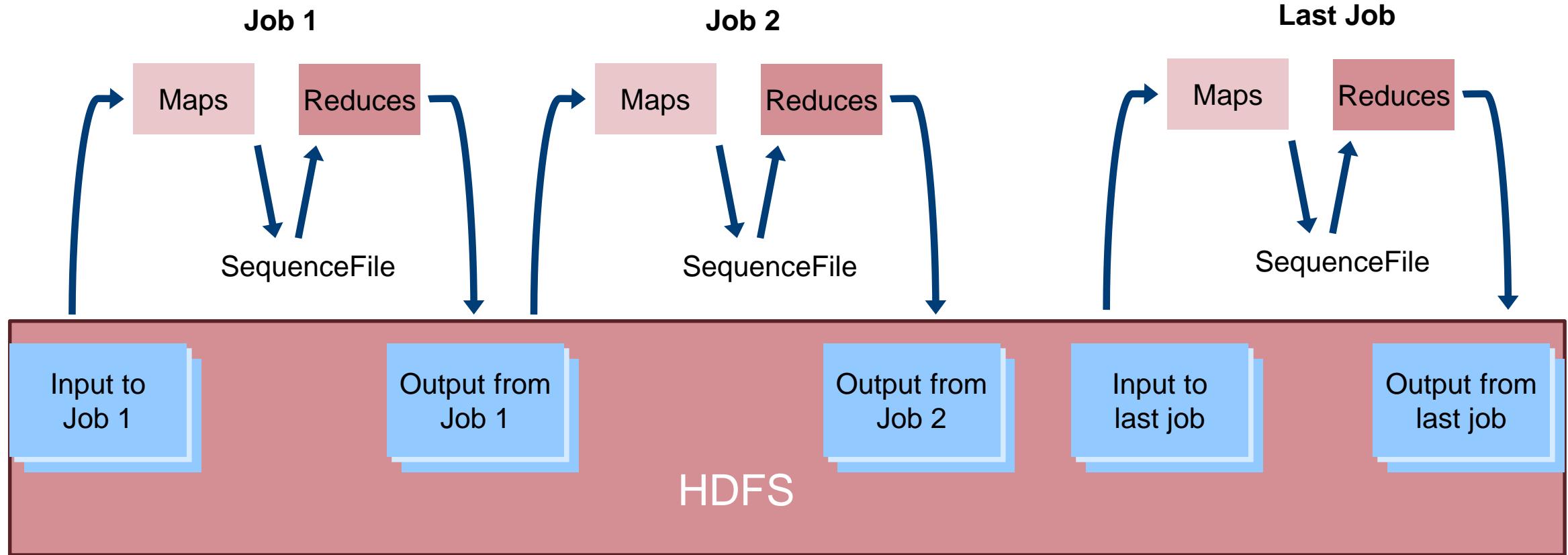
- For complex work, chain jobs together
 - Use a higher level language or DSL that does this for you





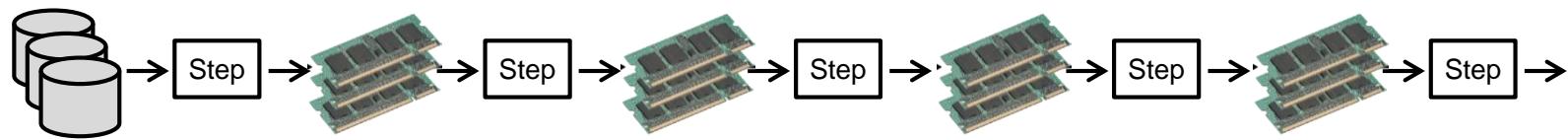
Typical MapReduce Workflows

Iteration is slow because it writes/reads data to disk





Iterations



In-memory Caching

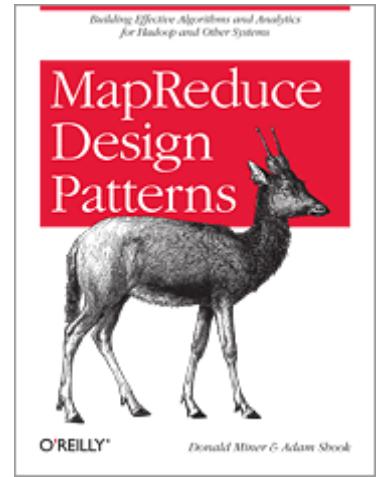
- Data Partitions read from RAM instead of disk





MapReduce Design Patterns

- Summarization
 - Inverted index, counting
- Filtering
 - Top ten, distinct
- Aggregation
- Data Organization
 - partitioning
- Join
 - Join data sets
- Metapattern
 - Job chaining





Inverted Index Example

alice.txt

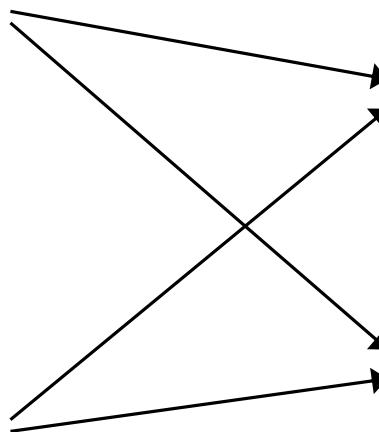
"The time has
come," the
Walrus said

time, alice.txt
has, alice.txt
come, alice.txt
..

macbeth.txt

tis time to do it

tis, macbeth.txt
time, macbeth.txt
do, macbeth.txt
...



come, (alice.txt)
do, (macbeth.txt)
has, (alice.txt)
time, (alice.txt, macbeth.txt)
...





MapReduce: The Good

- Optimized to Read and write large data files quickly
 - Process data on node
 - Minimize movement of disk head
- Scalable
- Built in fault tolerance
- Developer focuses on Map/Reduce, not infrastructure
- simple? API





MapReduce: The Bad

- Optimized for **disk IO**
 - Doesn't leverage memory well
 - **Iterative** algorithms go through disk IO path again and again
- Primitive API
 - simple abstraction
 - Key/Value in/out





Free Hadoop MapReduce On Demand Training

- <https://www.mapr.com/services/mapr-academy/big-data-hadoop-online-training>

DEV 301 - Developing Hadoop Applications

This course teaches developers, with lectures and hands-on lab exercises, how to write Hadoop Applications using MapReduce and YARN in Java. The course extensively covers MapReduce programming, debugging, managing jobs, improving performance, working managing workflows, and using other programming languages for MapReduce.

[Learn More !\[\]\(0e60d2d9b679b4cf53dbe1e685ee345d_img.jpg\)](#)





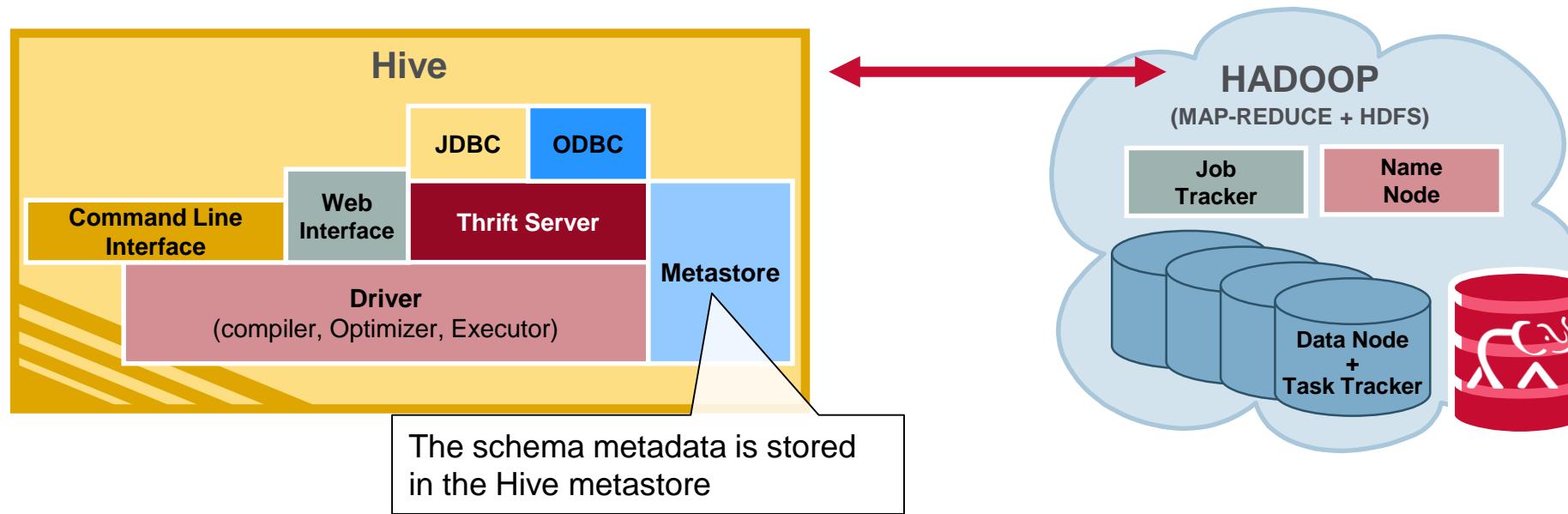
What is Hive?

- Data Warehouse on top of Hadoop
 - analytical querying of data
 - without programming
- SQL like execution for Hadoop
 - SQL evaluates to **MapReduce** code
 - Submits jobs to your cluster





Hive Metastore



Hive Table definition

key string	price bigint	vol bigint

HBase trades_tall Table

key	cf1:price	cf1:vol
AMZN_986186008	12.34	1000
AMZN_986186007	12.00	50

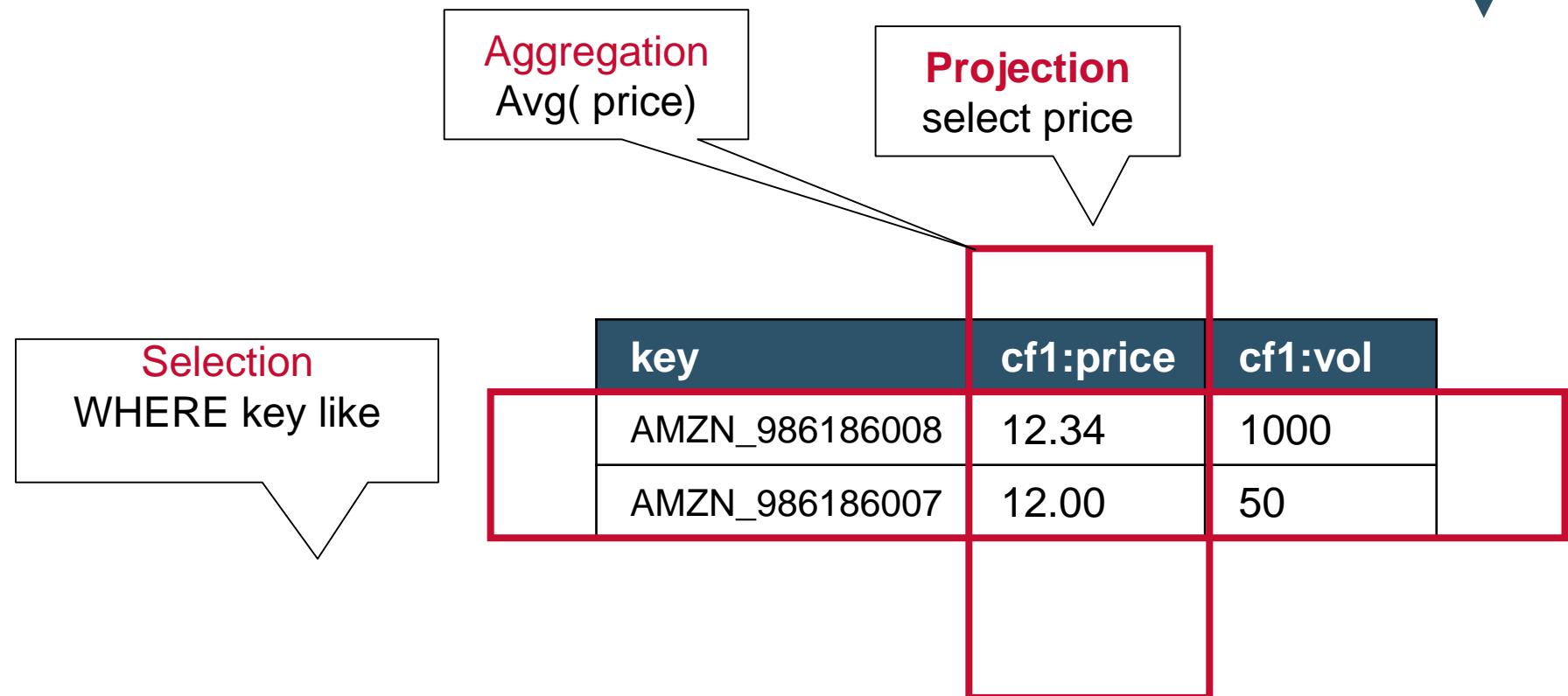




Hive HBase – External Table

SQL evaluates to MapReduce code

```
SELECT AVG(price) FROM trades WHERE key LIKE "AMZN" ;
```

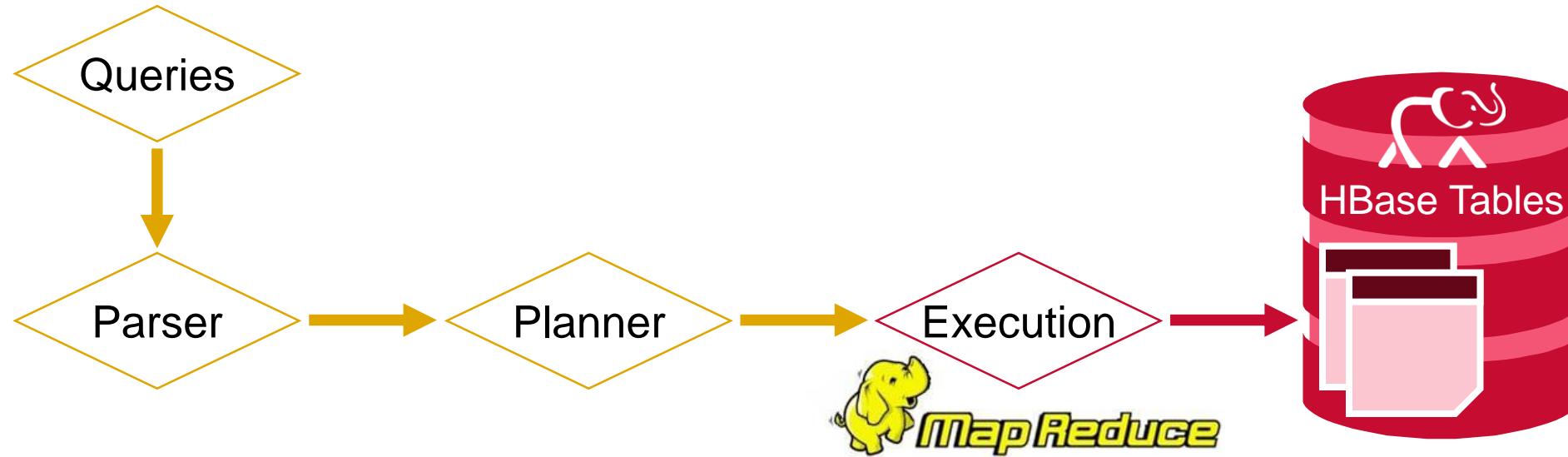




Hive HBase – Hive Query

SQL evaluates to MapReduce code

```
SELECT AVG(price) FROM trades WHERE key LIKE "GOOG" ;
```





Hive Query Plan

- EXPLAIN SELECT AVG(price) FROM trades WHERE key LIKE "GOOG%";

STAGE PLANS:

Stage: Stage-1

Map Reduce

Map Operator Tree:

TableScan

Filter Operator

 predicate: (key like 'GOOG%') (type: boolean)

Select Operator

 Group By Operator

Reduce Operator Tree:

 Group By Operator

 Select Operator

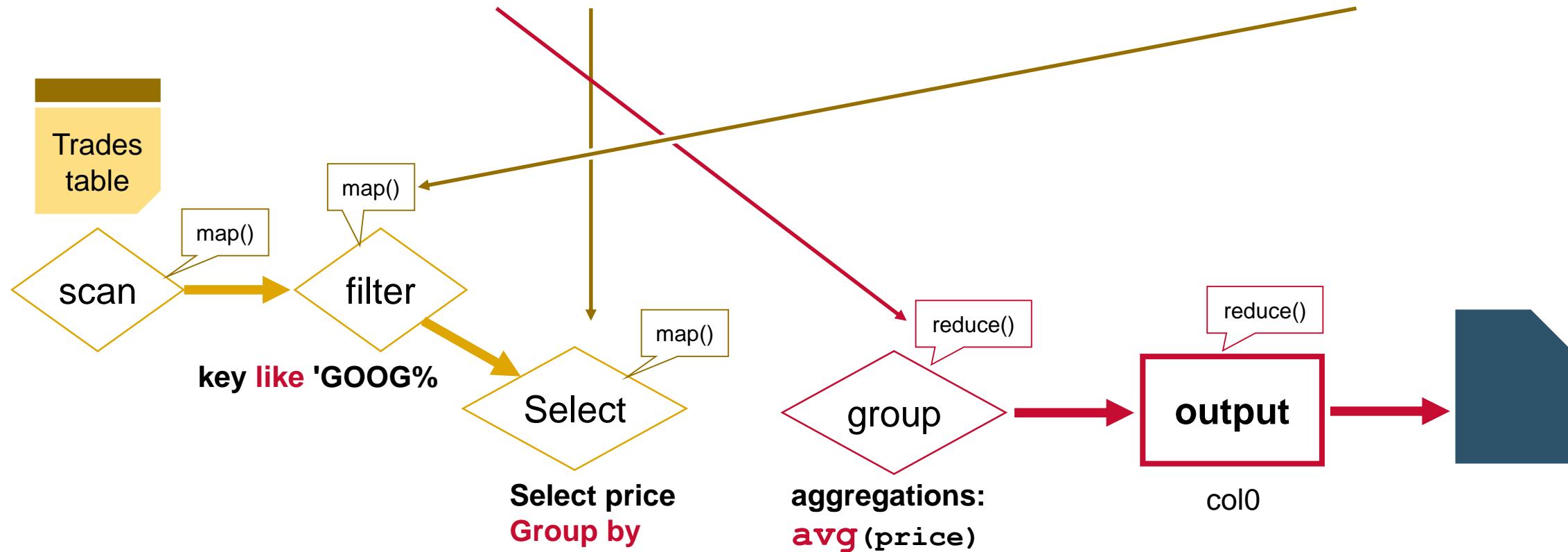
 File Output Operator





Hive Query Plan – (2)

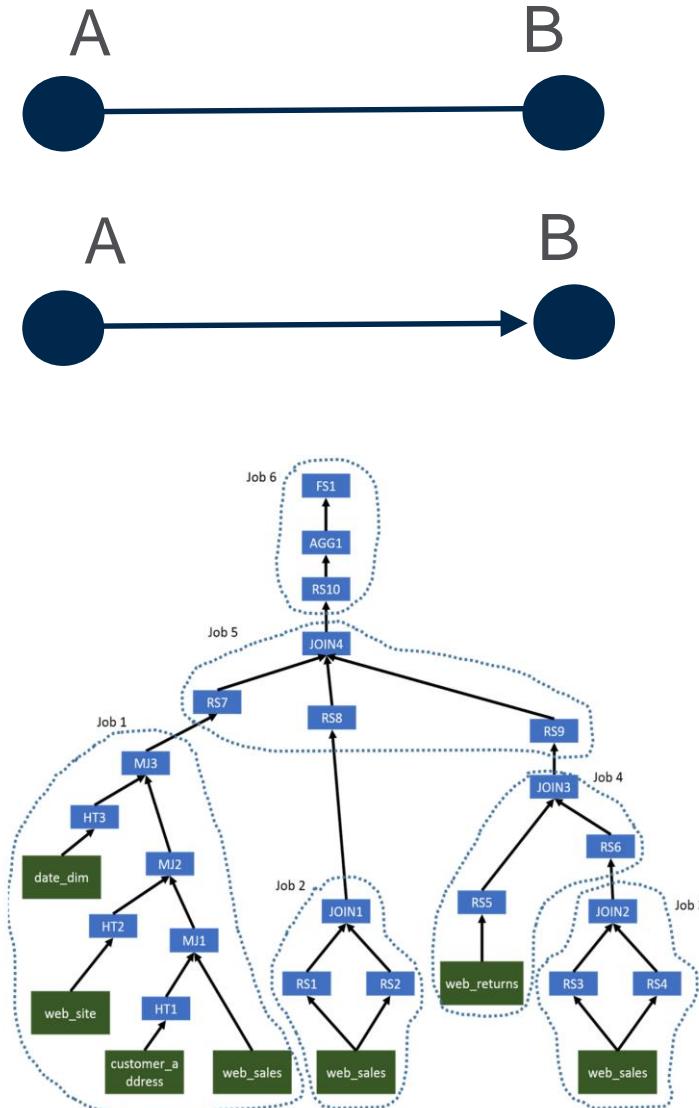
```
hive> SELECT AVG(price) FROM trades WHERE key LIKE "GOOG%";
```





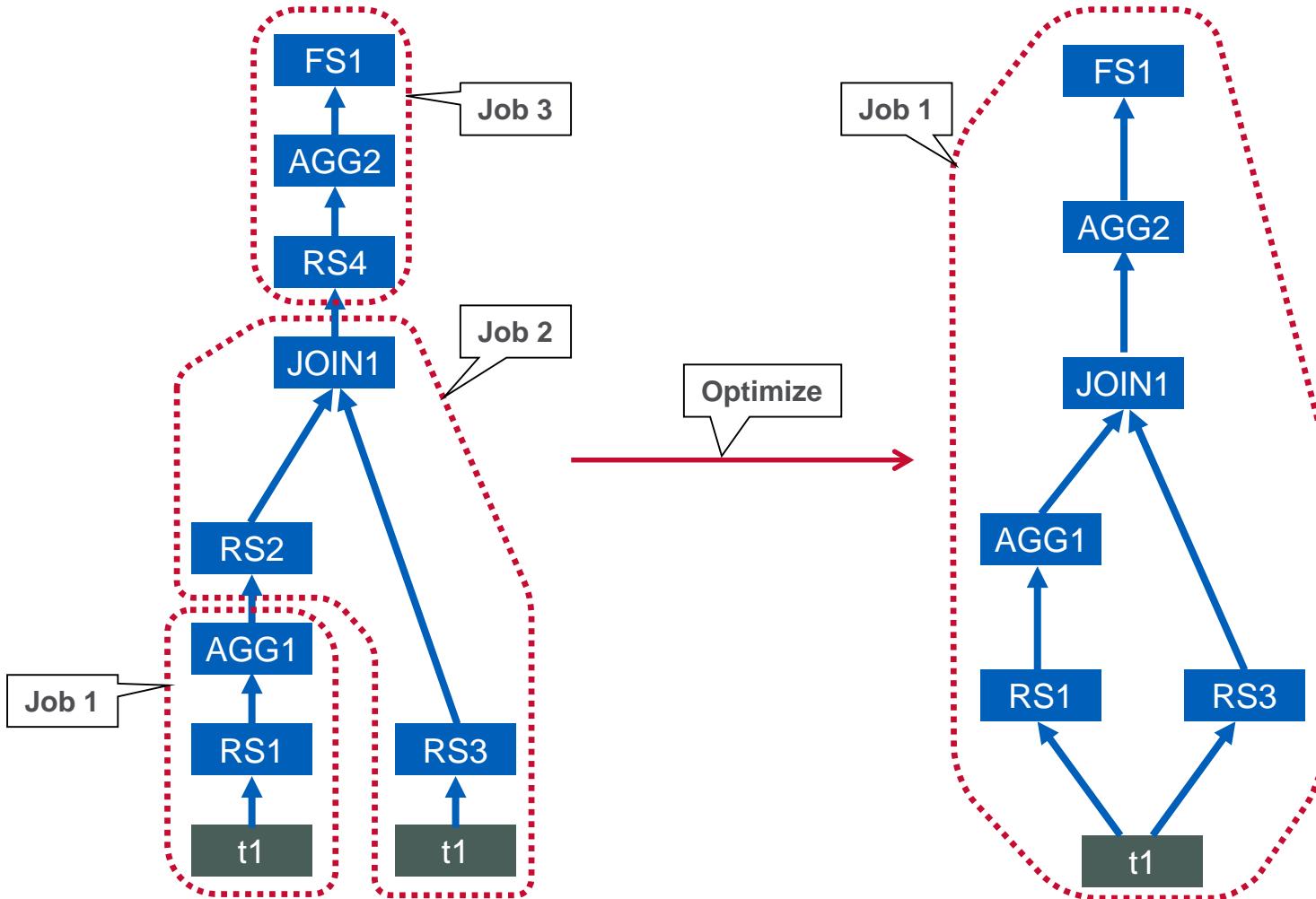
What is a Directed Acyclic Graph (DAG) ?

- Graph
 - **vertices** (points) and **edges** (lines)
- Directed
 - Only in a single direction
- Acyclic
 - No looping





Hive Query Plan Map Reduce Execution





Free HBase On Demand Training

(includes Hive and MapReduce with HBase)

- <https://www.mapr.com/services/mapr-academy/big-data-hadoop-online-training>

DEV 320 - HBase Data Model and Architecture

This course is intended for data analysts, data architects and application developers. DEV 320 provides you with a thorough understanding of the HBase data model and architecture, which is required before going on to designing HBase schema applications.

[Learn More](#)

DEV 325 - HBase Schema Design

Targeted towards data analysts, data architects and application developers, the goal of this course is to enable you to design schemas based on design guidelines. You will learn about the various elements of schema design and how to design them. It offers an in-depth look at designing row keys, avoiding hot-spotting and designing column families. It discusses how to map data to an HBase model. You will learn the differences between tall tables and wide tables. Concepts are conveyed through a series of scenarios.

[Learn More](#)

DEV 330 - Developing HBase Applications: Basics

NEW!

Targeted towards data architects and application developers who have experience with Java, the goal of this course is to teach you how to develop Java programs using Hadoop as a distributed NoSQL datastore.

© 2014 MapR Technologies

MAPR

35

Apache Spark





Unified Platform

Spark SQL

Spark Streaming
(Streaming)

MLlib
(Machine learning)

GraphX (*Graph computation*)

Spark (*General execution engine*)

Mesos

Hadoop YARN

Distributed File System (HDFS, MapR-FS, S3, ...)





Why Apache Spark?

- Distributed Parallel Cluster computing
 - Scalable , Fault tolerant
- Programming Framework
 - APIs in Java, Scala, Python
 - Less code than map reduce
- Fast
 - Runs computations in memory
 - Tasks are threads





- **Iterative Algorithms on large amounts of data**
- Some Example Use Cases:
 - Anomaly detection
 - Classification
 - Predictions
 - Recommendations



Why Iterative Algorithms

- Algorithms that need **iterations**
 - Clustering (K-Means, Canopy, ...)
 - Gradient descent (e.g., Logistic Regression, Matrix Factorization)
 - Graph Algorithms (e.g., PageRank, Line-Rank, components, paths, reachability, centrality,)
 - **Alternating Least Squares ALS**
 - Graph communities / dense sub-components
 - Inference (believe propagation)
 - ...





Data Sources

- Local Files
- S3
- Hadoop Distributed Filesystem
 - any Hadoop InputFormat
- HBase
- other NoSQL data stores

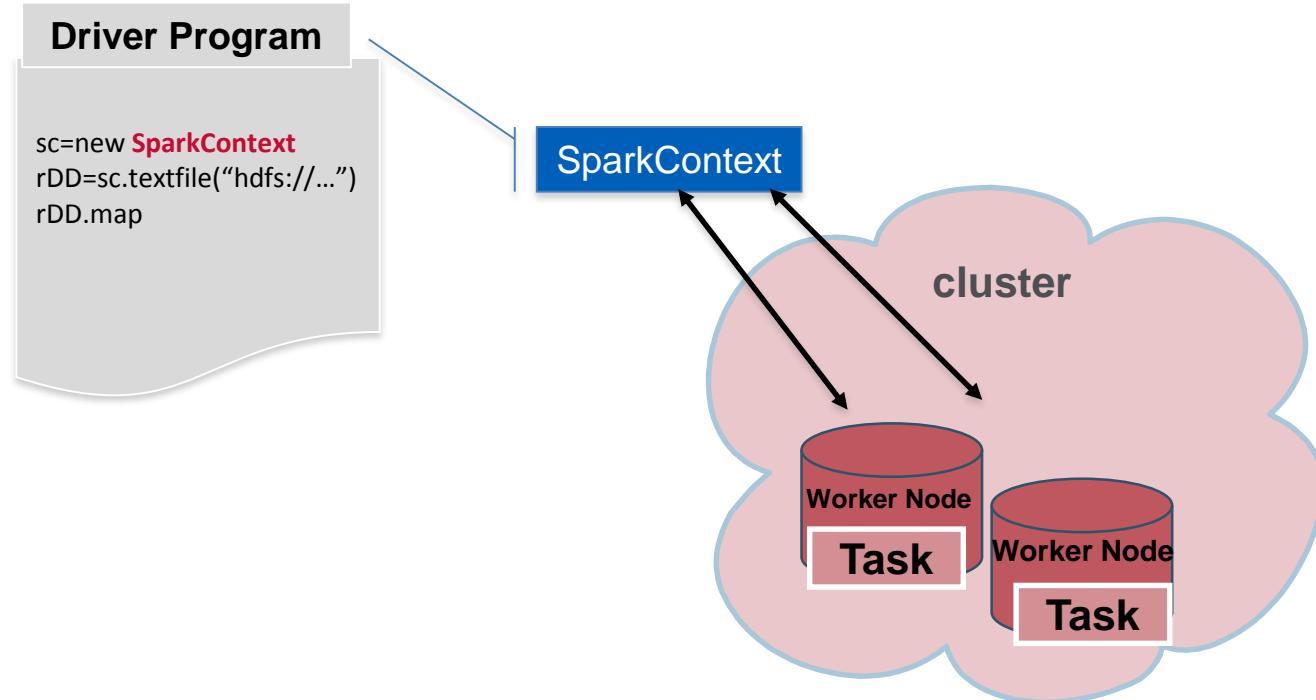


How Spark Works





Spark Programming Model

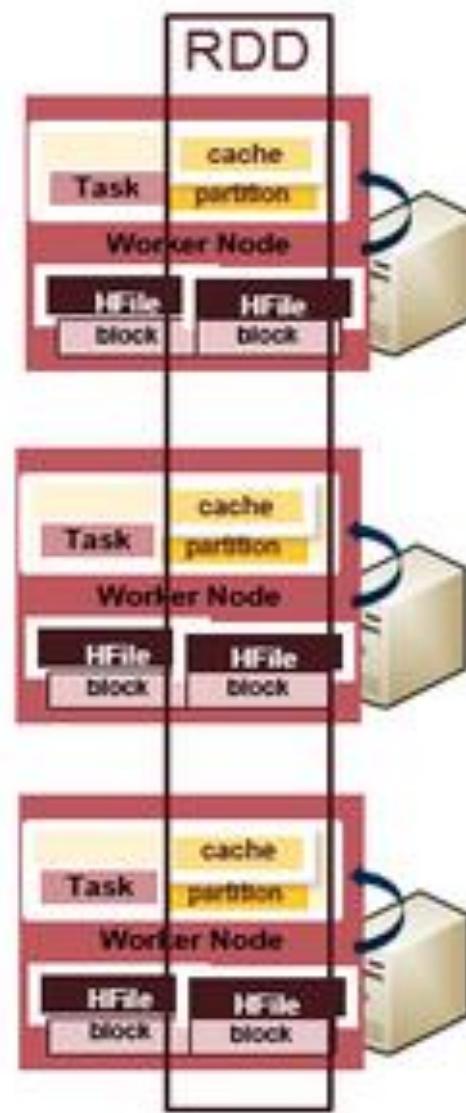




Resilient Distributed Datasets (RDD)

Spark revolves around RDDs

- read only collection of elements

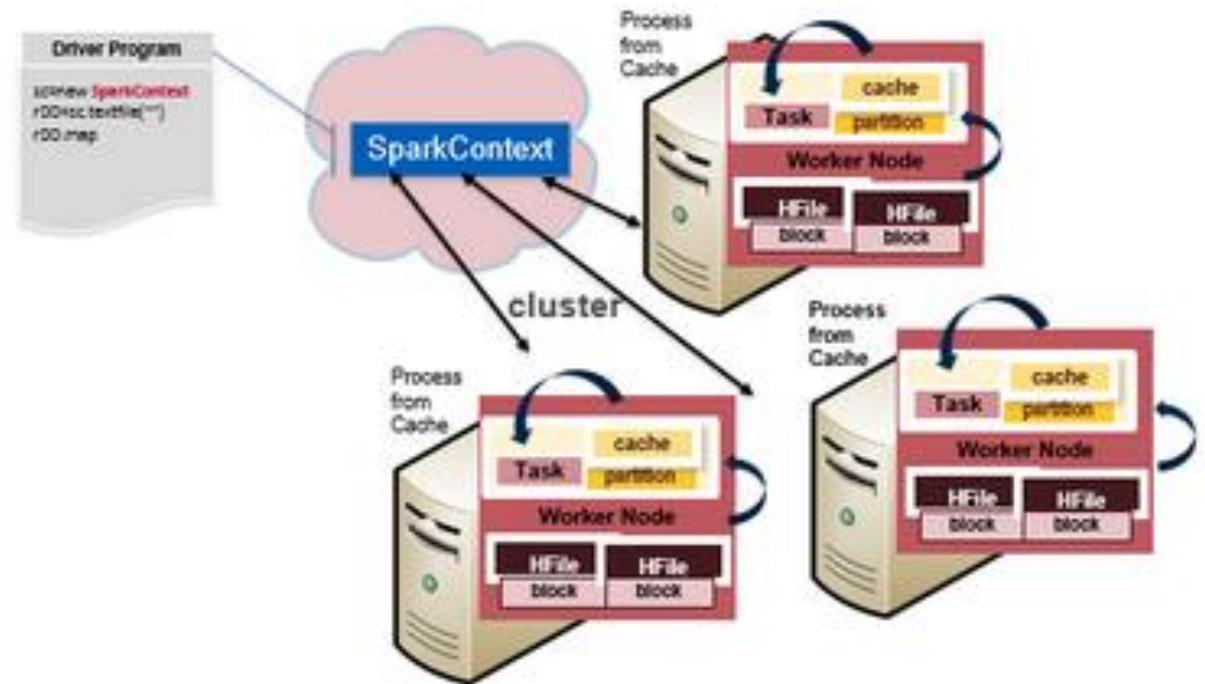




Resilient Distributed Datasets (RDD)

Spark revolves around RDDs

- read only **collection** of elements
- operated on in **parallel**
- **Cached** in memory
 - Or on disk
- Fault tolerant





Working With RDDs

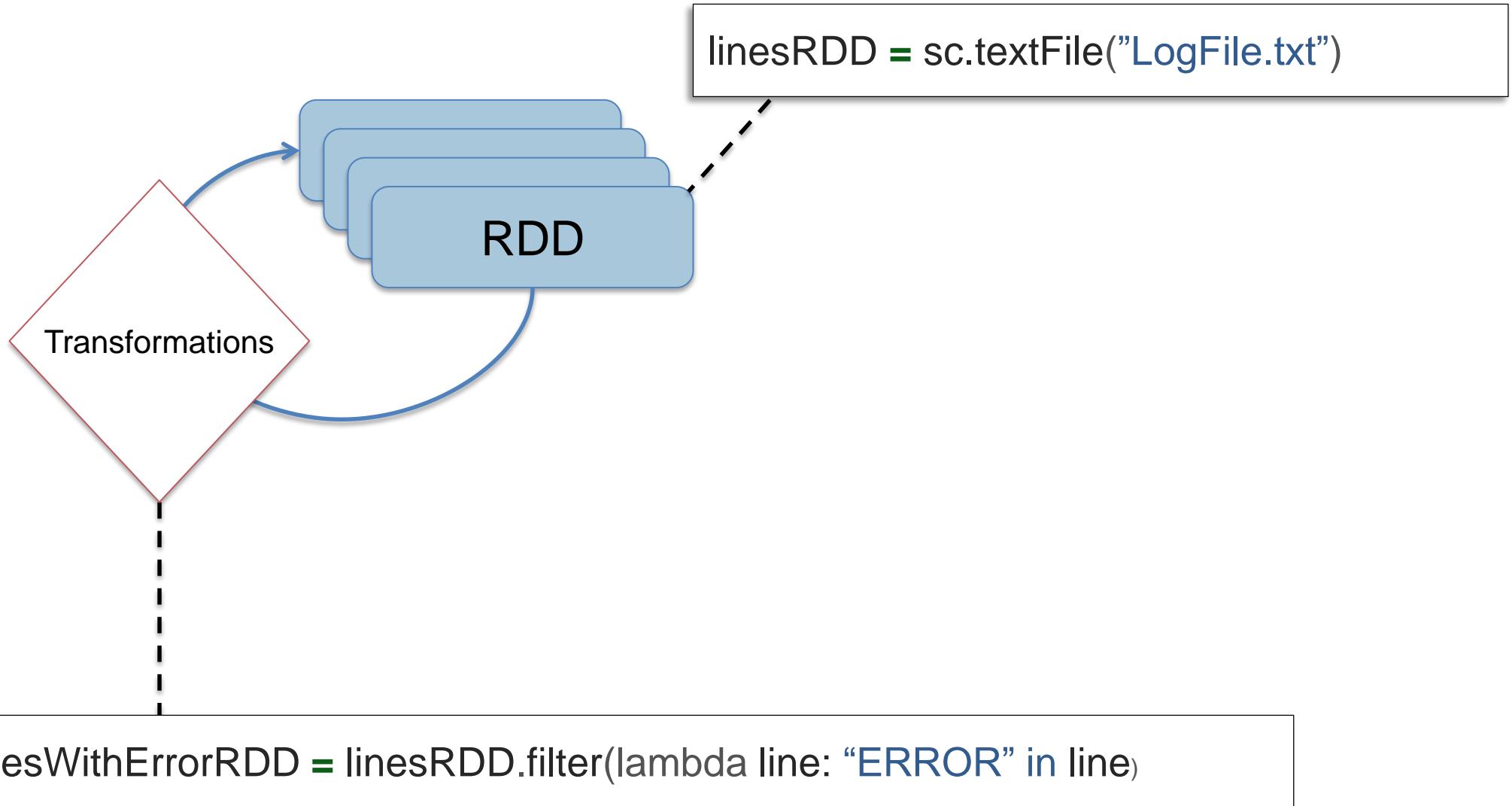
RDD

```
textFile = sc.textFile("SomeFile.txt")
```



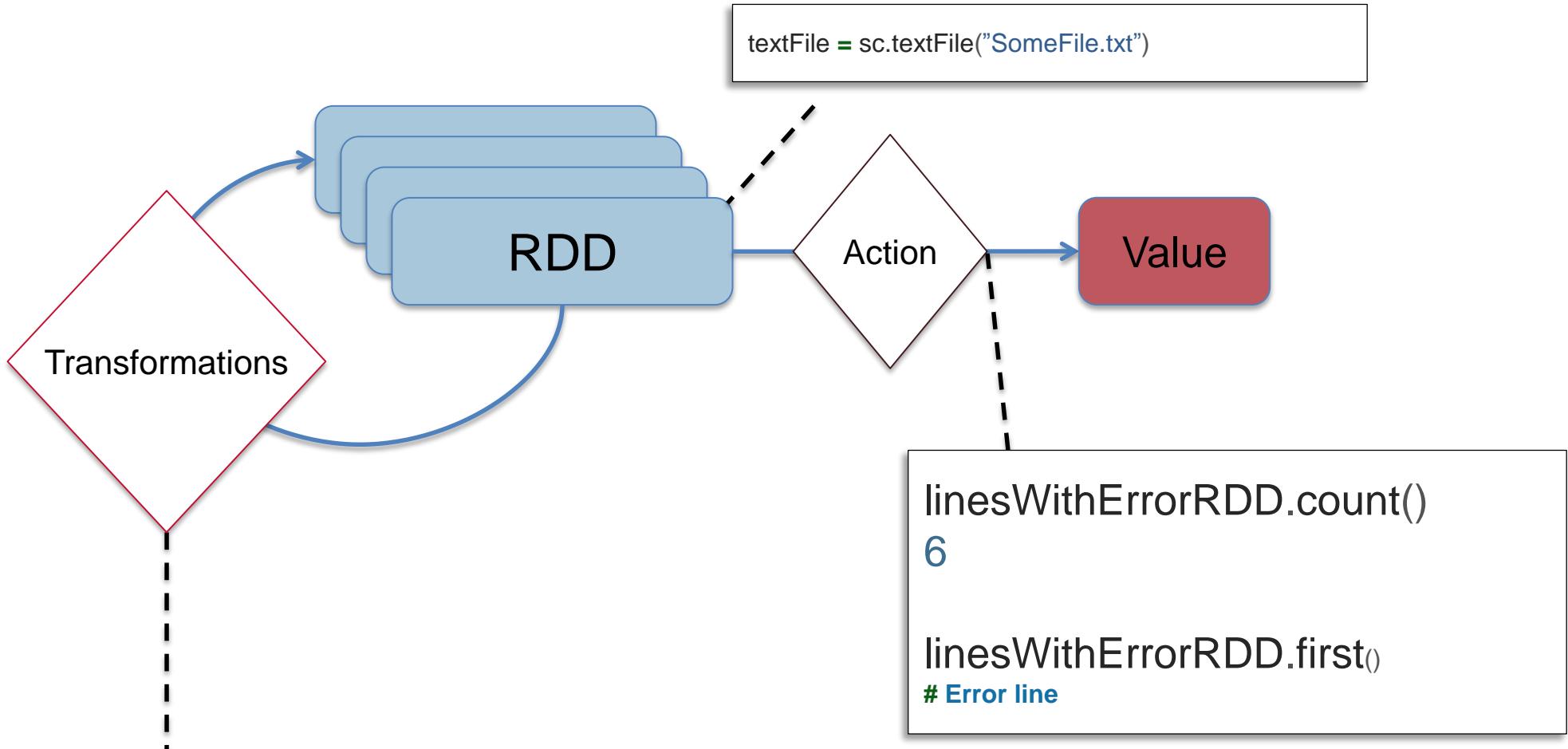


Working With RDDs





Working With RDDs





Example Spark Word Count in Scala

```
// Load our input data.
```

```
val input = sc.textFile(inputFile)
```

```
// Split it up into words.
```

```
val words = input.flatMap(line => line.split(" "))
```

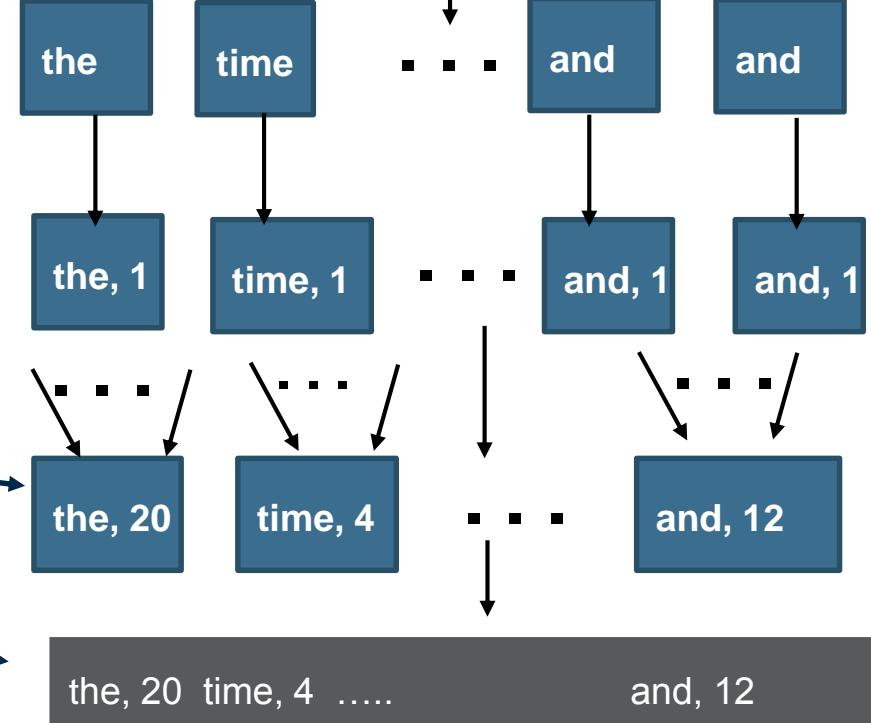
```
// Transform into pairs and count.
```

```
val counts = words
  .map(word => (word, 1))
  .reduceByKey{case (x, y) => x + y}
```

```
// Save the word count back out to a text file,
```

```
counts.saveAsTextFile(outputFile)
```

"The time has come," the Walrus said,
"To talk of many things:
Of shoes—and ships—and sealing-wax





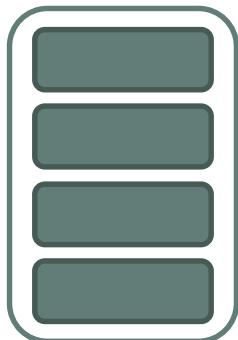
Example Spark Word Count in Scala

```
// Load input data.
```

```
val input = sc.textFile(inputFile)
```

HadoopRDD

MapPartitionsRDD



textFile

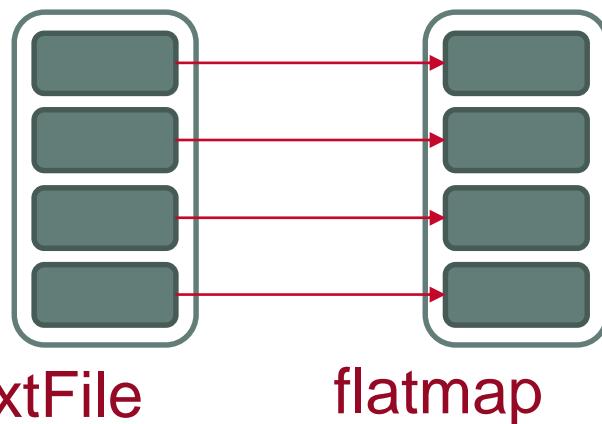




Example Spark Word Count in Scala

```
// Load our input data.  
val input = sc.textFile(inputFile)  
// Split it up into words.  
val words = input.flatMap(line => line.split(" "))
```

HadoopRDD
MapPartitionsRDD
MapPartitionsRDD

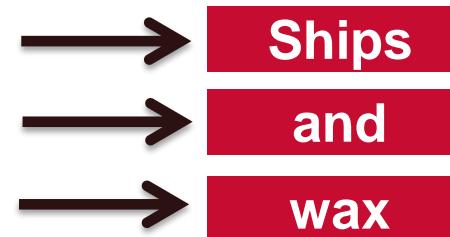




FlatMap

flatMap
→
`line => line.split(" ")`

1 to many mapping



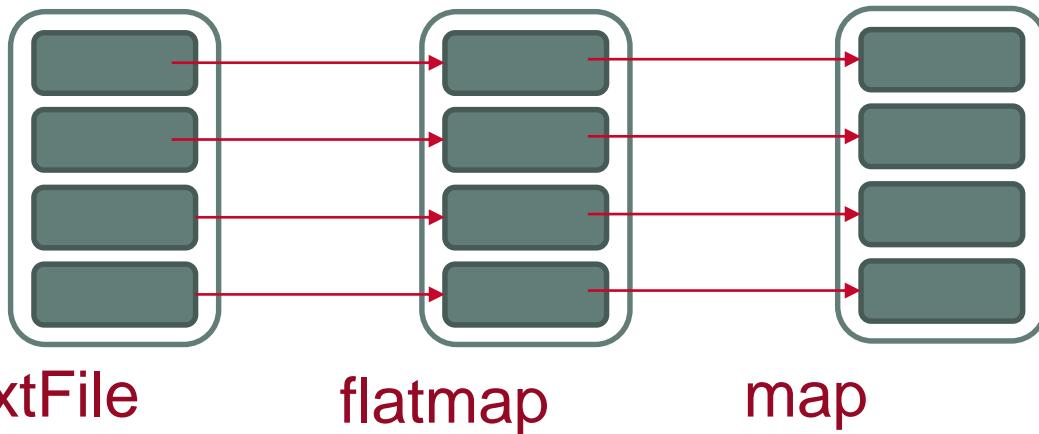
RDD<String> wordsRDD





Example Spark Word Count in Scala

```
val input = sc.textFile(inputFile)                                HadoopRDD  
val words = input.flatMap(line => line.split(" "))           MapPartitionsRDD  
// Transform into pairs                                         MapPartitionsRDD  
val counts = words.map(word => (word, 1))                      MapPartitionsRDD
```





Map

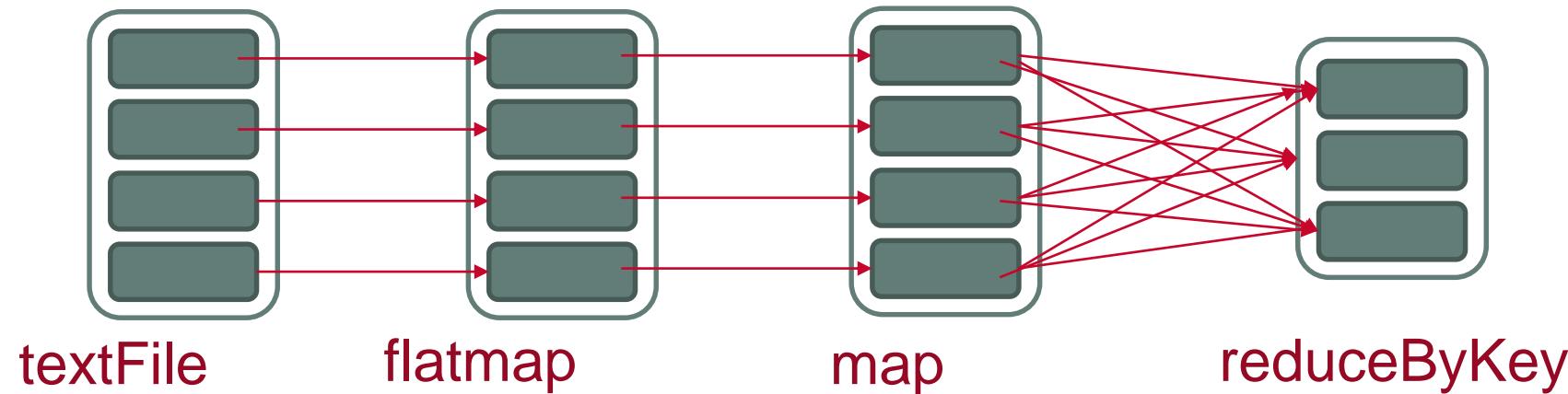
1 to 1 mapping





Example Spark Word Count in Scala

```
val input = sc.textFile(inputFile)                                HadoopRDD  
val words = input.flatMap(line => line.split(" "))              MapPartitionsRDD  
val counts = words  
    .map(word => (word, 1))                                      MapPartitionsRDD  
    .reduceByKey{case (x, y) => x + y}                            ShuffledRDD
```

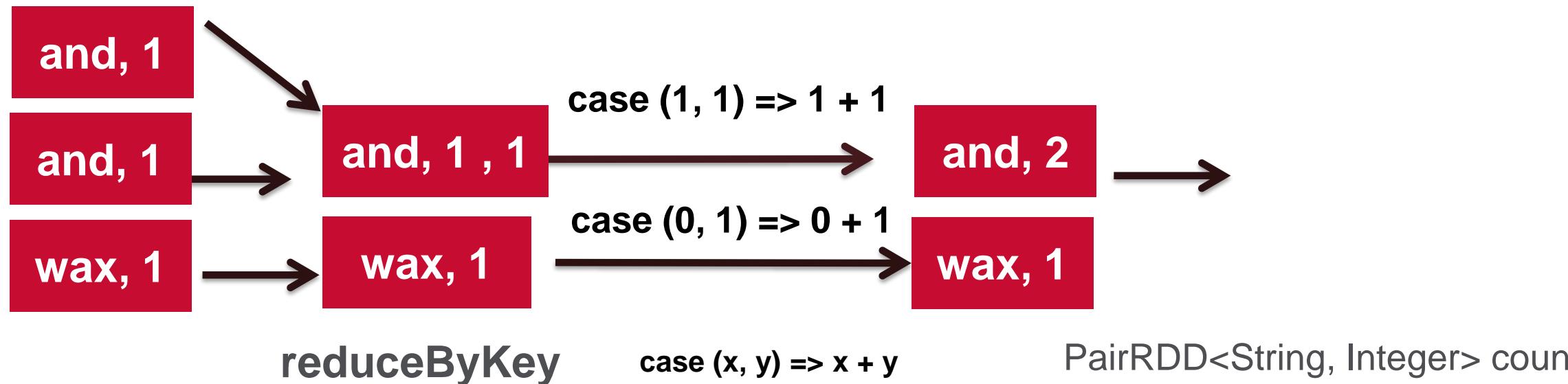




reduceByKey

PairRDD Function :

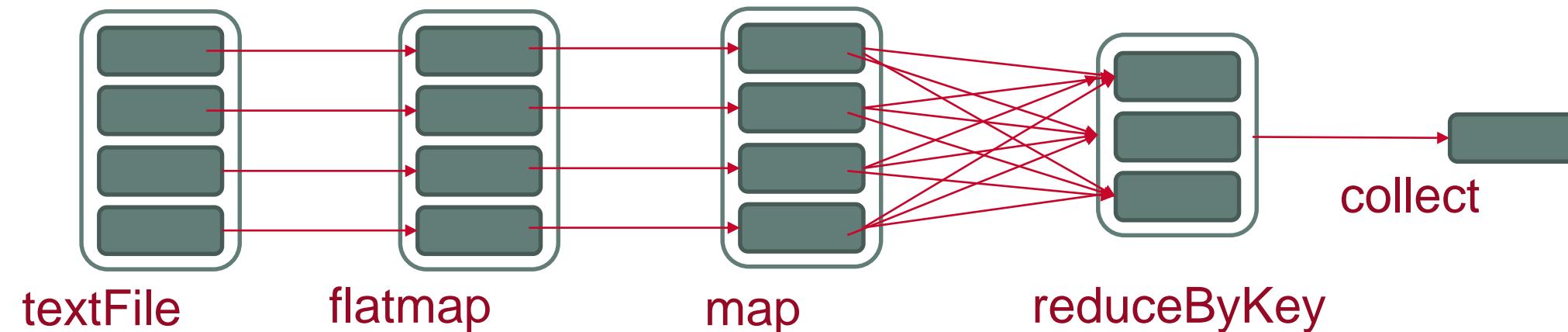
`reduceByKey(func: (value, value) => value): RDD[(key, value)]`





Example Spark Word Count in Scala

```
val input = sc.textFile(inputFile)                                HadoopRDD  
val words = input.flatMap(line => line.split(" "))                MapPartitionsRDD  
val counts = words  
    .map(word => (word, 1))                                         MapPartitionsRDD  
    .reduceByKey{case (x, y) => x + y}                               ShuffledRDD  
  
val countArray = counts.collect()                                    Array
```





Demo Interactive Shell

- Iterative Development
 - Cache those RDDs
 - Open the shell and ask questions
 - We have all wished we could do this with MapReduce
 - Compile / save your code for scheduled jobs later
 - Scala – spark-shell
 - Python – pyspark



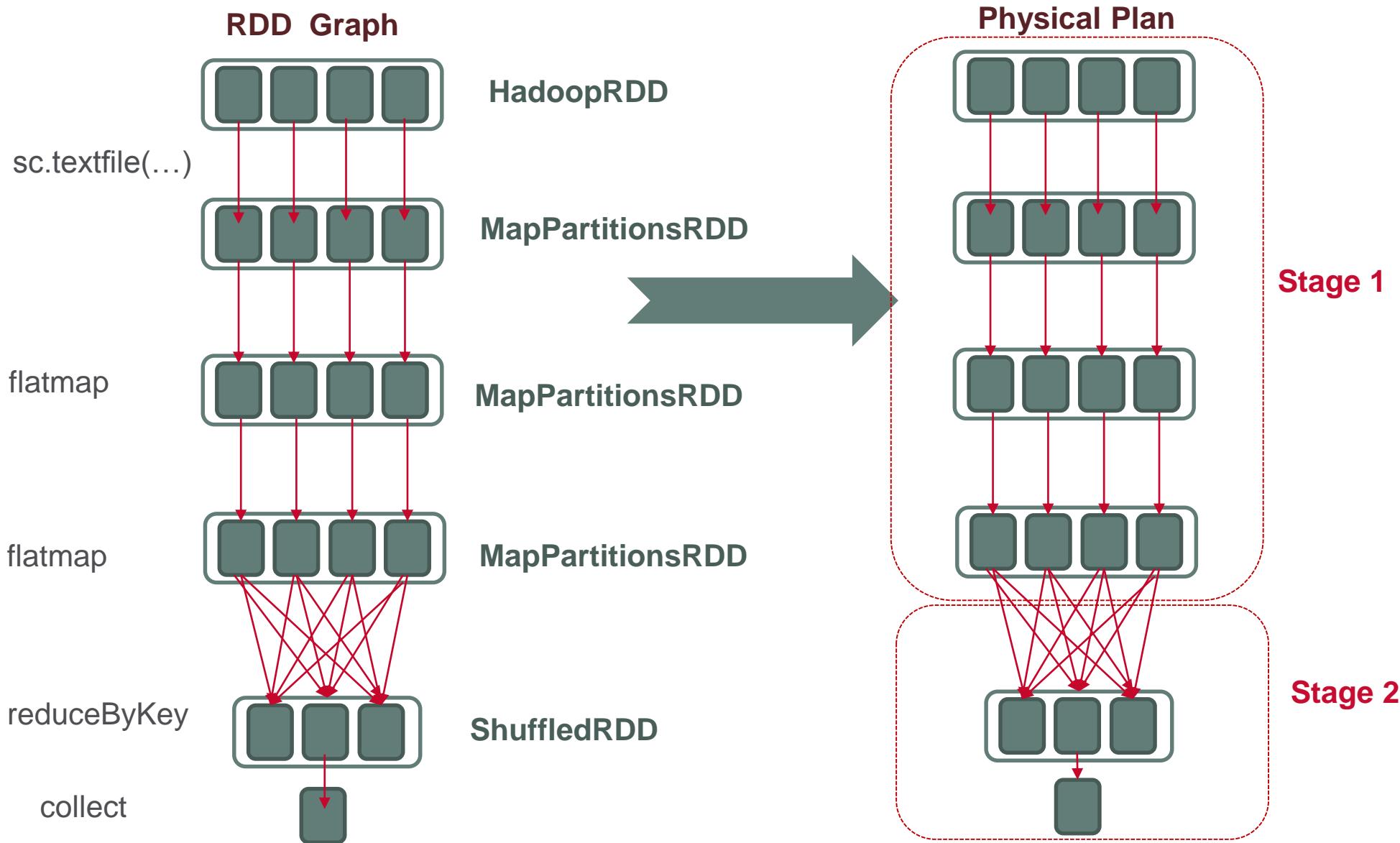


Components Of Execution



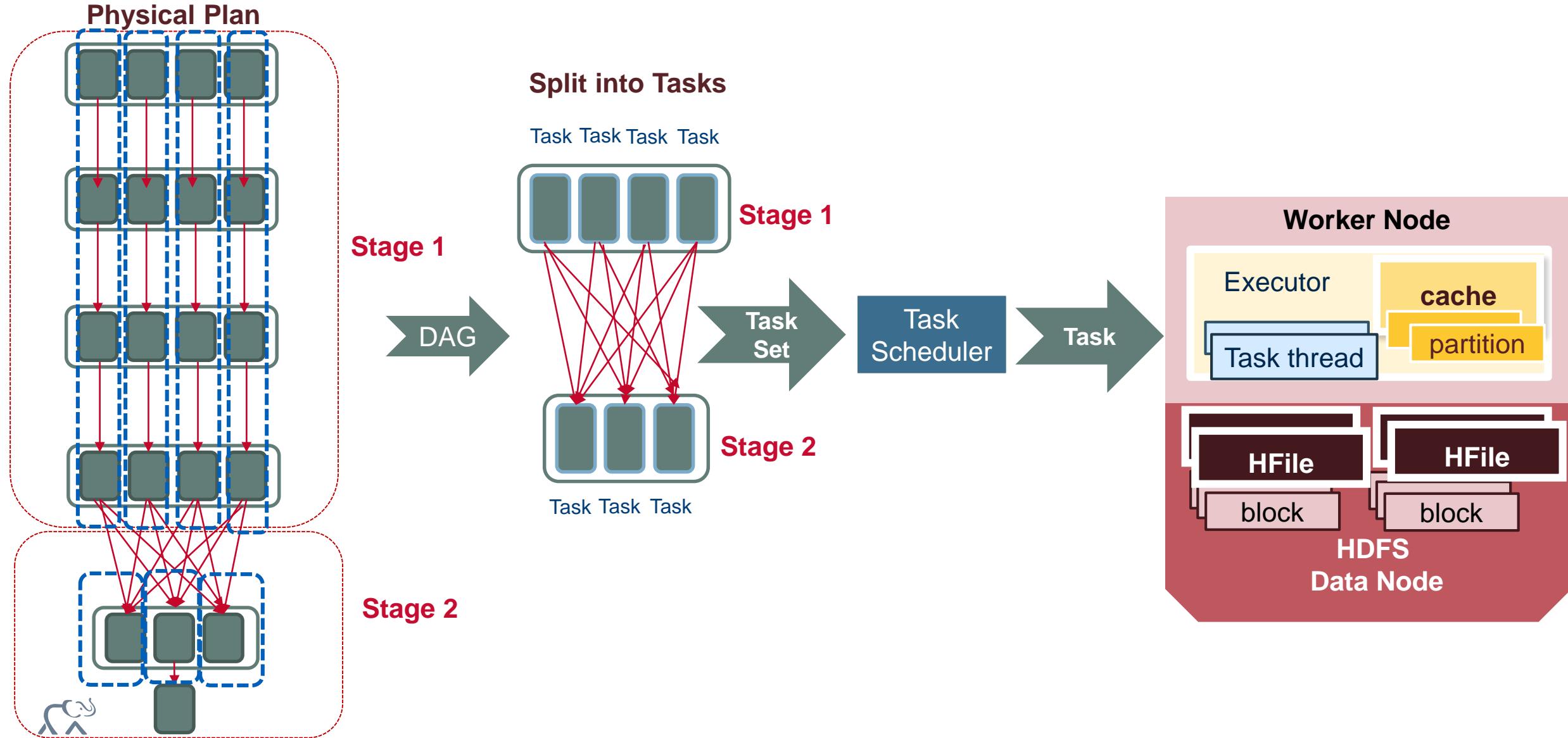


Spark RDD DAG -> Physical Execution plan





Physical Execution plan -> Stages and Tasks





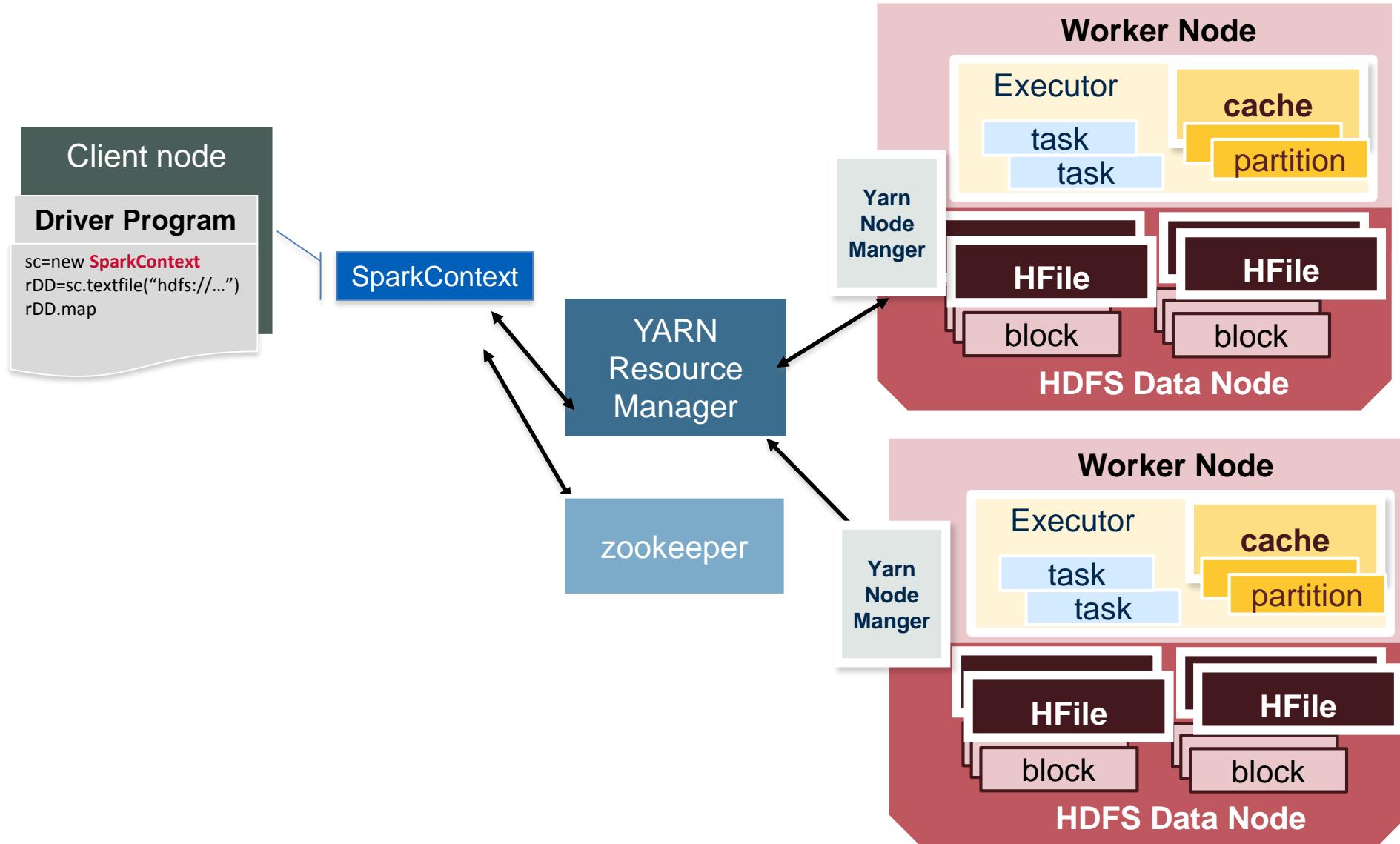
Summary of Components

- Task : unit of execution
- Stage: **Group** of Tasks
 - Base on **partitions** of RDD
 - Tasks run in **parallel**
- DAG : Logical Graph of RDD operations
- RDD : **Parallel dataset** with partitions





How Spark Application runs on a Hadoop cluster





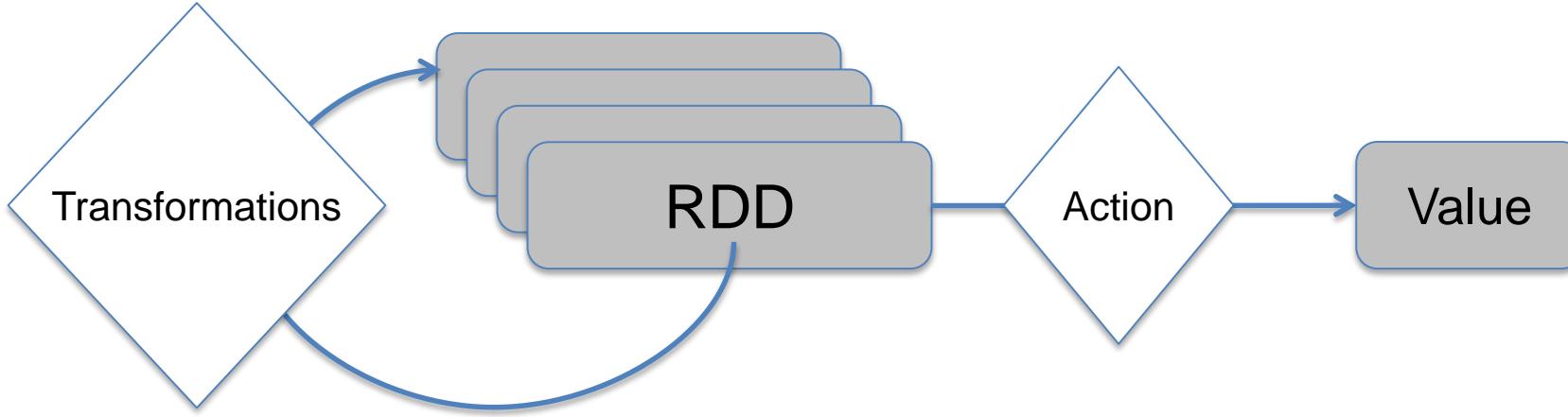
Deploying Spark – Cluster Manager Types

- Standalone mode
- Mesos
- YARN
- EC2
- GCE





RDD Transformations and Actions



Transformations
(define a new RDD)

map
filter
sample
union
groupByKey
reduceByKey
join
cache

Actions
(return a value)

reduce
collect
count
save
lookupKey
...



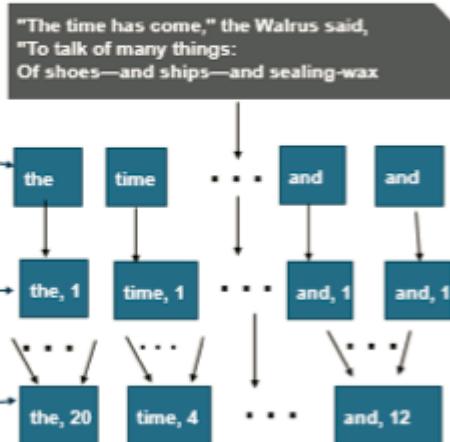
MapR Tutorial: Getting Started with Spark on MapR Sandbox

- <https://www.mapr.com/products/mapr-sandbox-hadoop/tutorials/spark-tutorial>

Next, we will look at how to write, compile, and run a Standalone Spark word count application step through the following word count application in Java.

Example Word Count App in Java

```
JavaRDD<String> input = sc.textFile(inputFile);
// Split each line into words
JavaRDD<String> words = input.flatMap(
    new FlatMapFunction<String, String>() {
        public Iterable<String> call(String x) {
            return Arrays.asList(x.split(" "));
        }
    });
// Turn the words into (word, 1) pairs
JavaPairRDD<String, Integer> word1s = words.mapToPair(
    new PairFunction<String, String, Integer>(){
        public Tuple2<String, Integer> call(String x){
            return new Tuple2(x, 1);
        }
    });
// reduce add the pairs by key to produce counts
JavaPairRDD<String, Integer> counts = word1s.reduceByKey(
    new Function2<Integer, Integer, Integer>(){
        public Integer call(Integer x, Integer y){
            return x + y;
        }
    });
}
```

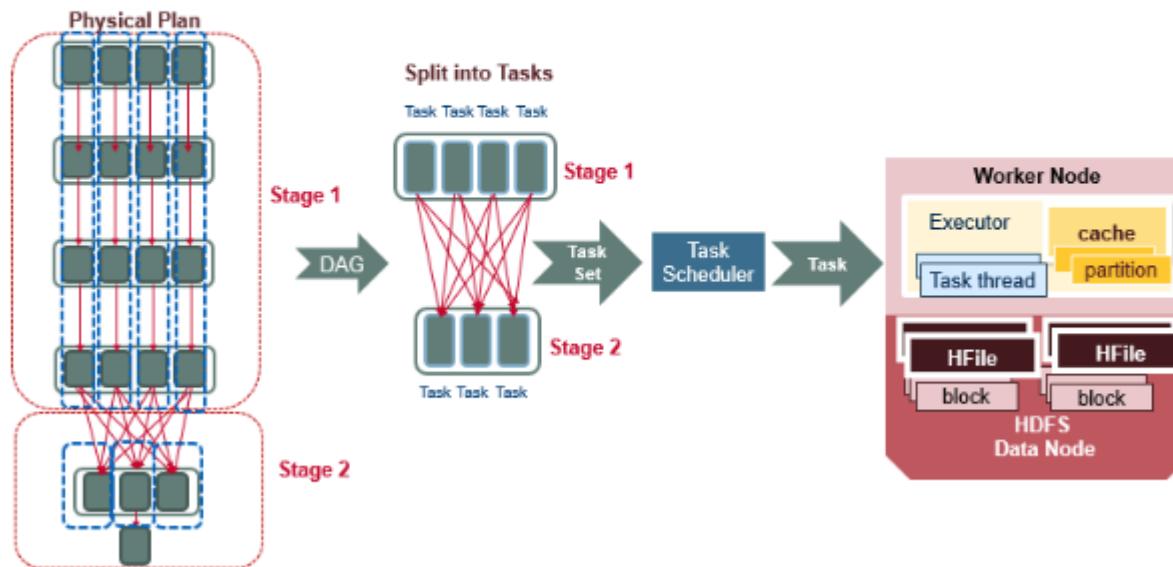




MapR Blog: Getting Started with the Spark Web UI

- <https://www.mapr.com/blog/getting-started-spark-web-ui>

The scheduler splits the RDD graph into stages, based on the transformations. The narrow transformations (transformations without data movement) will be grouped (pipe-lined) together into a single stage. This physical plan has two stages, with everything before ShuffledRDD in the first stage.





Example: Log Mining





Example: Log Mining

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

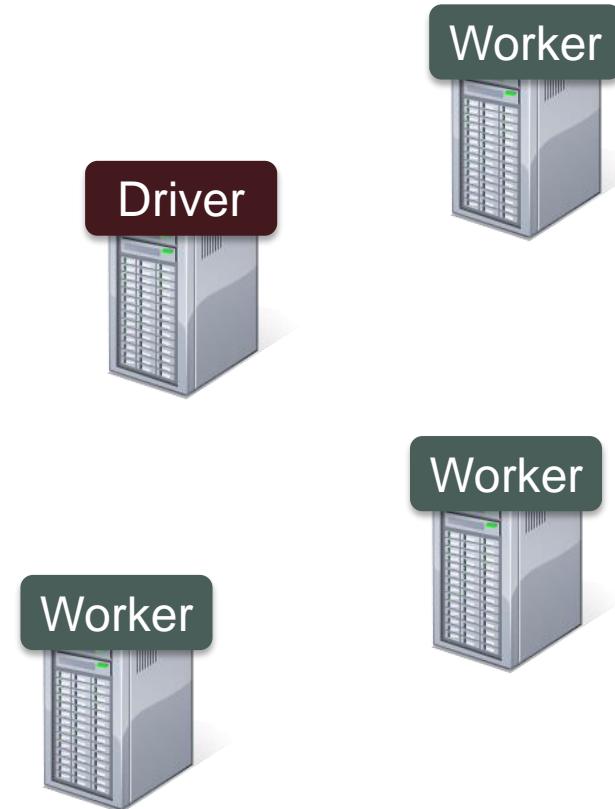
Based on slides from Pat McDonough at  DATABRICKS





Example: Log Mining

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





Example: Log Mining

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

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





Example: Log Mining

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

Base RDD

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

Worker

Driver

Worker

Worker





Example: Log Mining

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

```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))
```





Example: Log Mining

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

Transformed RDD

```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))
```





Example: Log Mining

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

```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()
```



```
messages.filter(lambda s: "mysql" in s).count()
```





Example: Log Mining

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

```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()
```

```
messages.filter(lambda s: "mysql" in s).count()
```





Example: Log Mining

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

```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()
```



```
messages.filter(lambda s: "mysql" in s).count()
```



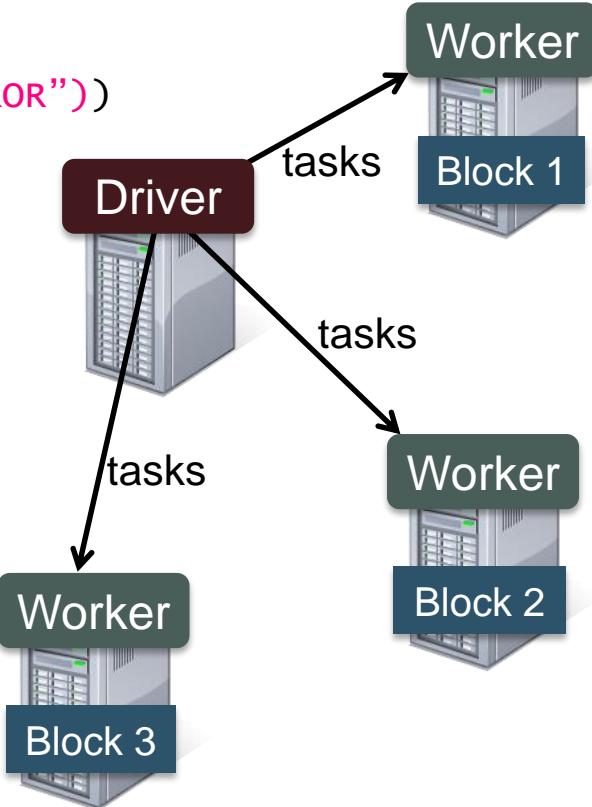


Example: Log Mining

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

```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()
```

```
messages.filter(lambda s: "mysql" in s).count()
```



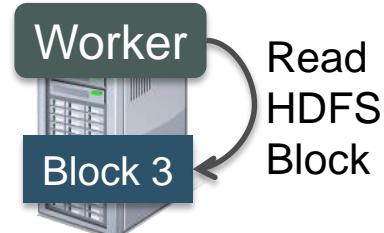
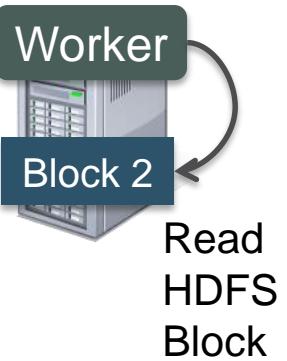
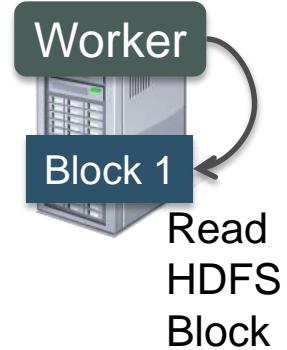


Example: Log Mining

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

```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()
```

```
messages.filter(lambda s: "mysql" in s).count()
```



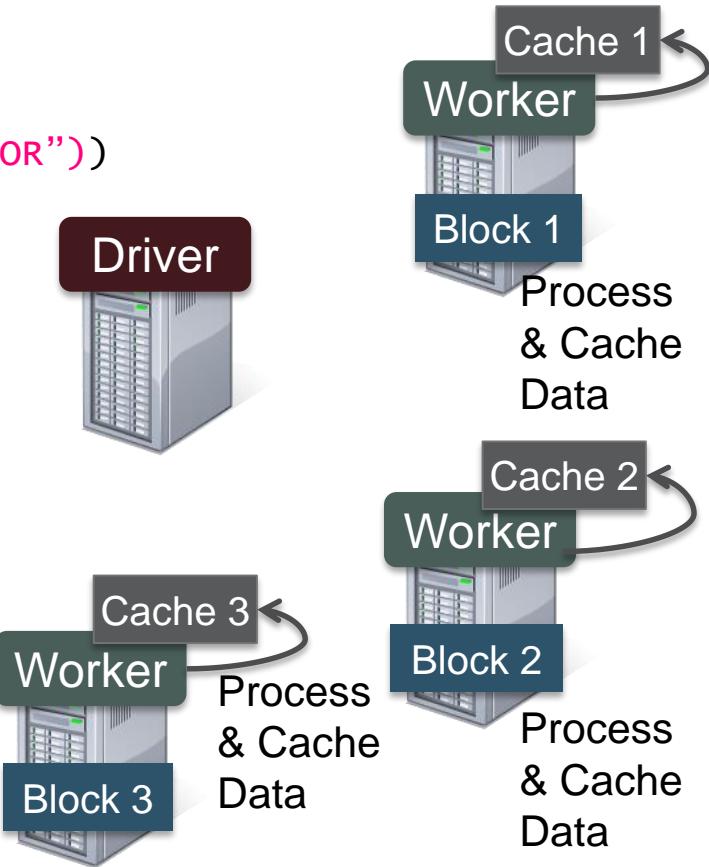


Example: Log Mining

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

```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()
```

```
messages.filter(lambda s: "mysql" in s).count()
```



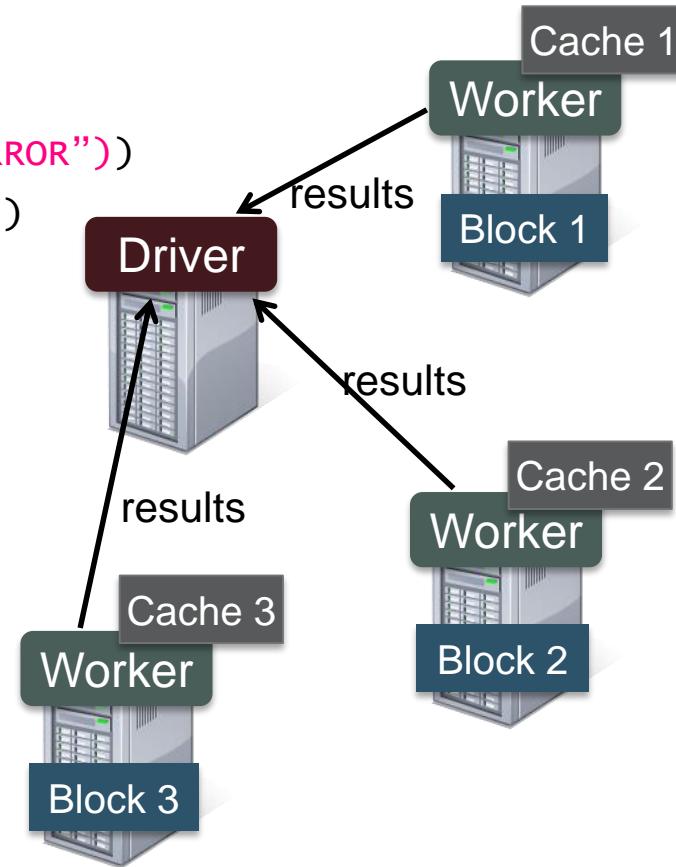


Example: Log Mining

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

```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()
```

```
messages.filter(lambda s: "mysql" in s).count()
```





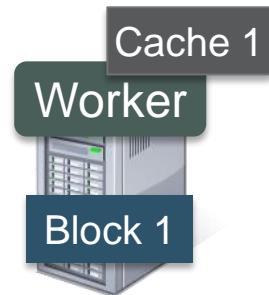
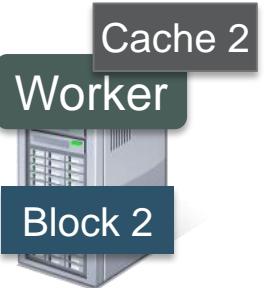
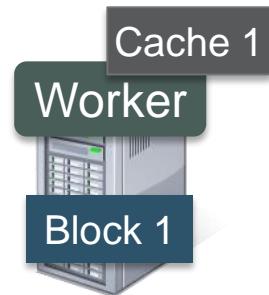
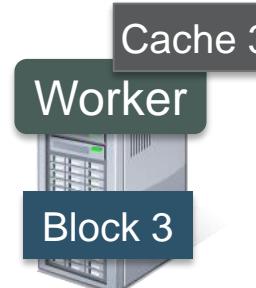
Example: Log Mining

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

```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()
```



```
messages.filter(lambda s: "mysql" in s).count()  
messages.filter(lambda s: "php" in s).count()
```



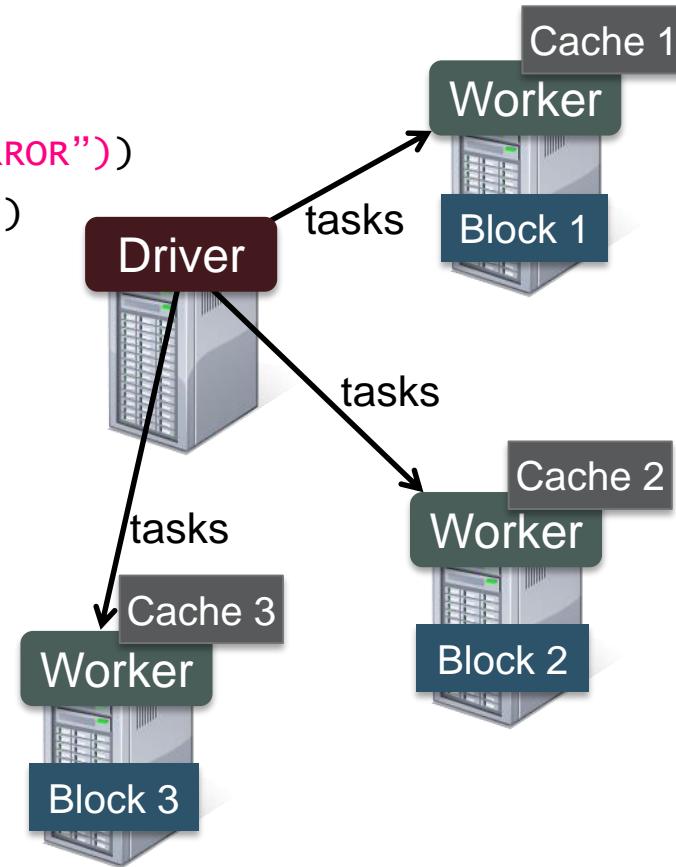


Example: Log Mining

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

```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()
```

```
messages.filter(lambda s: "mysql" in s).count()  
messages.filter(lambda s: "php" in s).count()
```

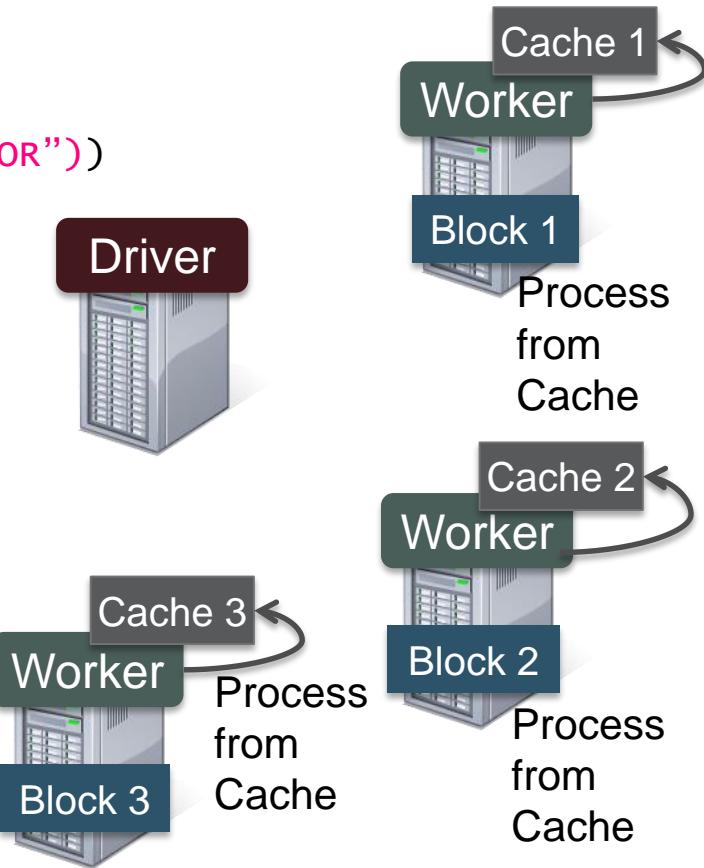




Example: Log Mining

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

```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()  
  
messages.filter(lambda s: "mysql" in s).count()  
messages.filter(lambda s: "php" in s).count()
```



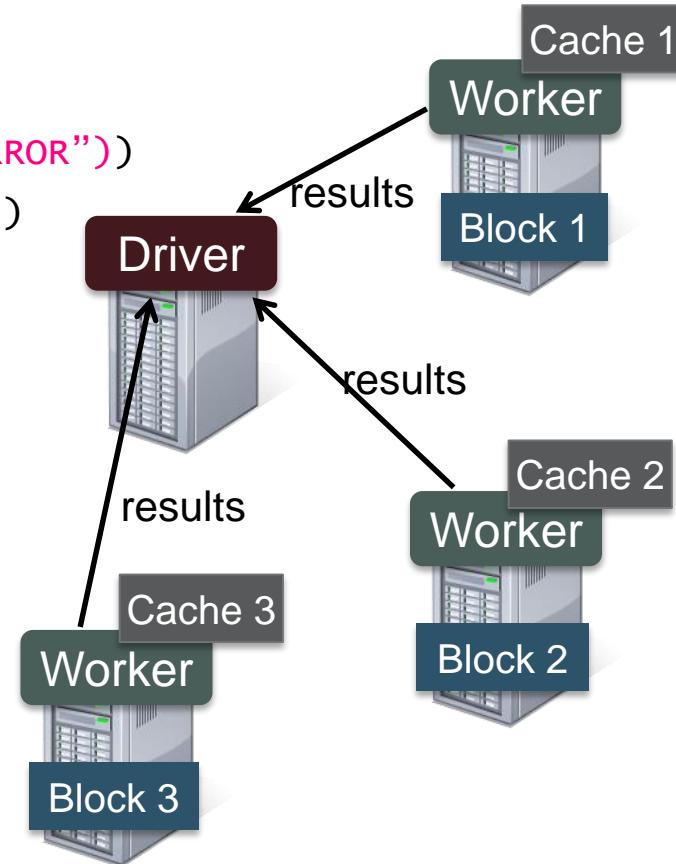


Example: Log Mining

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

```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()
```

```
messages.filter(lambda s: "mysql" in s).count()  
messages.filter(lambda s: "php" in s).count()
```





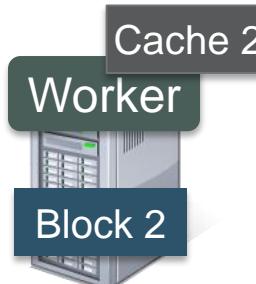
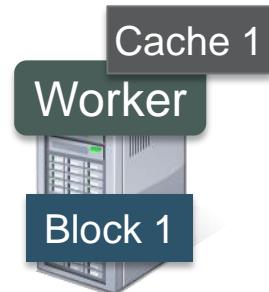
Example: Log Mining

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

```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()
```

```
messages.filter(lambda s: "mysql" in s).count()  
messages.filter(lambda s: "php" in s).count()
```

Cache your data → Faster Results





Dataframes





Scenario: Online Auction Data

AuctionID	Bid	Bid Time	Bidder	Bidder Rate	Open Bid	Price	Item	Days To Live
8213034705	95	2.927373	jake7870	0	95	117.5	xbox	3
8213034705	115	2.943484	davidbresler2	1	95	117.5	xbox	3
8213034705	100	2.951285	gladimacowgirl	58	95	117.5	xbox	3
8213034705	117.5	2.998947	daysrus	95	95	117.5	xbox	3





Creating DataFrames

// Define schema using a Case class

```
case class Auction(auctionid: String, bid: Float,  
bidtime: Float, bidder: String, bidderrate: Int,  
openbid: Float, finprice: Float, itemtype: String,  
dtl: Int)
```





Creating DataFrames

// Create the RDD

```
val auctionRDD=sc.textFile("/user/user01/data/ebay.csv")
    .map(_.split(","))
```

// Map the data to the Auctions class

```
val auctions=auctionRDD.map(a=>Auction(a(0),
a(1).toFloat, a(2).toFloat, a(3), a(4).toInt,
a(5).toFloat, a(6).toFloat, a(7), a(8).toInt))
```





Creating DataFrames

```
// Convert RDD to DataFrame
```

```
val auctionsDF=auctions.toDF()
```

```
//To see the data
```

```
auctionsDF.show()
```

auctionid	bid	bidtime	bidder	bidderrate	openbid	price	item	daystolive
8213034705	95.0	2.927373	jake7870	0	95.0	117.5	xbox	3
8213034705	115.0	2.943484	davidbresler2	1	95.0	117.5	xbox	3 ...





Inspecting Data: Examples

//To see the schema

```
auctionsDF.printSchema()  
root  
|-- auctionid: string (nullable = true)  
|-- bid: float (nullable = false)  
|-- bidtime: float (nullable = false)  
|-- bidder: string (nullable = true)  
|-- bidderrate: integer (nullable = true)  
|-- openbid: float (nullable = false)  
|-- price: float (nullable = false)  
|-- item: string (nullable = true)  
|-- daystolive: integer (nullable = true)
```





Inspecting Data: Examples

//Total number of bids

```
val totbids=auctionsDF.count()
```

//How many bids per item?

```
auction.groupBy("auctionid", "item").count.show
```

auctionid	item	count
3016429446	palm	10
8211851222	xbox	28
3014480955	palm	12
8214279576	xbox	4
3014844871	palm	18
3014012355	palm	35
1641457876	cartier	2





Creating DataFrames

```
// Register the DF as a table
```

```
auctionsDF.registerTempTable("auctionsDF")
```

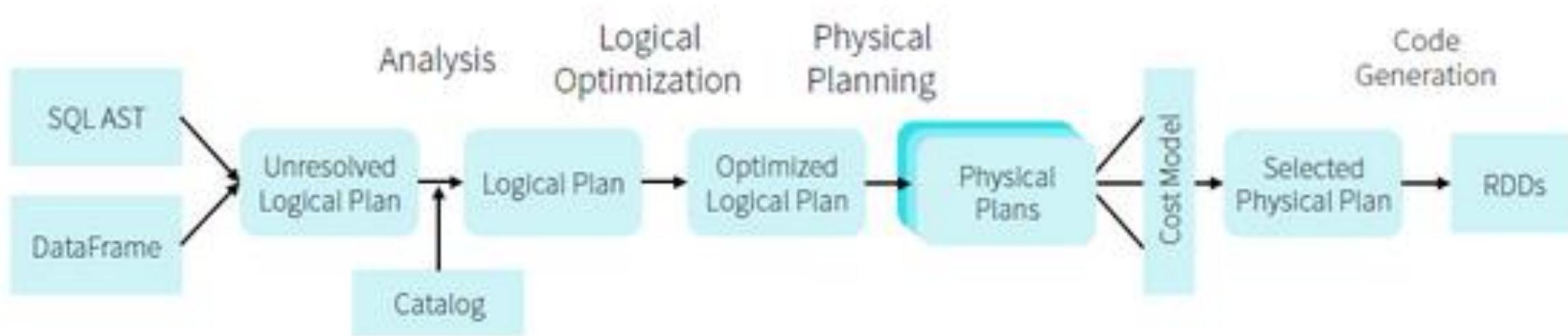
```
val results =sqlContext.sql("SELECT auctionid, MAX(price)  
as maxprice FROM auction GROUP BY item,auctionid")  
results.show()
```

```
auctionid  maxprice  
3019326300 207.5  
8213060420 120.0 . . .
```





The physical plan for DataFrames





DataFrame Execution plan

```
// Print the physical plan to the console
auction.select("auctionid").distinct.explain()

== Physical Plan ==
Distinct false
Exchange (HashPartitioning [auctionid#0], 200)
  Distinct true
  Project [auctionid#0]
    PhysicalRDD
      [auctionid#0,bid#1,bidtime#2,bidder#3,
       bidderrate#4,openbid#5,price#6,item#7,daystolive#8],
      MapPartitionsRDD[11] at mapPartitions at
      ExistingRDD.scala:37
```

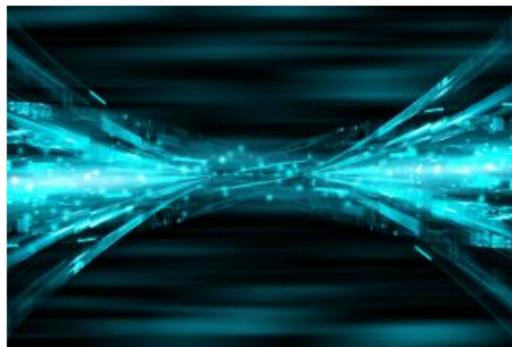




MapR Blog: Using Apache Spark DataFrames for Processing of Tabular Data

- <https://www.mapr.com/blog/using-apache-spark-dataframes-processing-tabular-data>

Using Apache Spark DataFrames for Processing of Tabular Data



🕒 June 24, 2015

Carol McDonald

This post will help you get started using Apache Spark DataFrames with Scala on the MapR Sandbox. The new [Spark DataFrames API](#) is designed to make big data processing on tabular data easier. A Spark DataFrame is a distributed collection of data organized into named columns that provides operations to filter, group, or compute aggregates, and can be used with Spark SQL. DataFrames can be constructed from structured data files, existing RDDs, tables in Hive, or



Machine Learning



Collaborative Filtering with Spark

- Recommend Items
 - (filtering)
- Based on User preferences data
 - (collaborative)

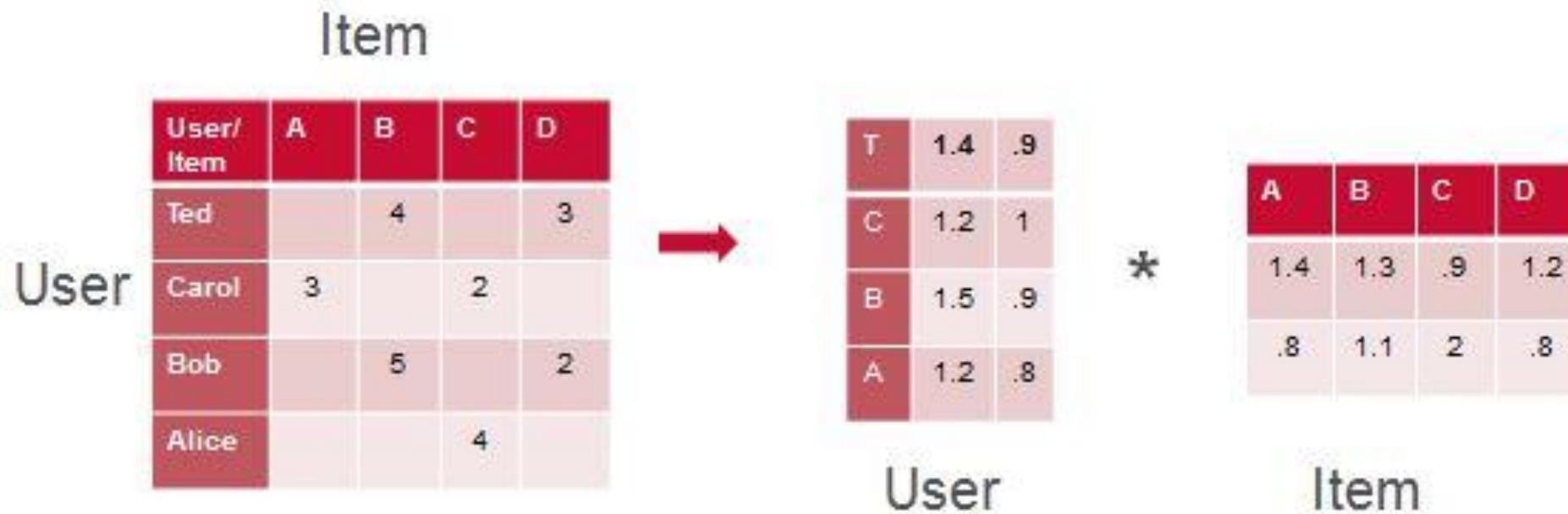
Users Item Rating Matrix



	REDFORD FONDA ELECTRIC	THE WAY WE WERE	THREE DAYS OF THE CONDOR
Ted	4	5	5
Carol		5	5
Bob		5	?

Alternating Least Squares

- approximates sparse user item rating matrix
- as product of two dense matrices, User and Item factor matrices





Parse Input

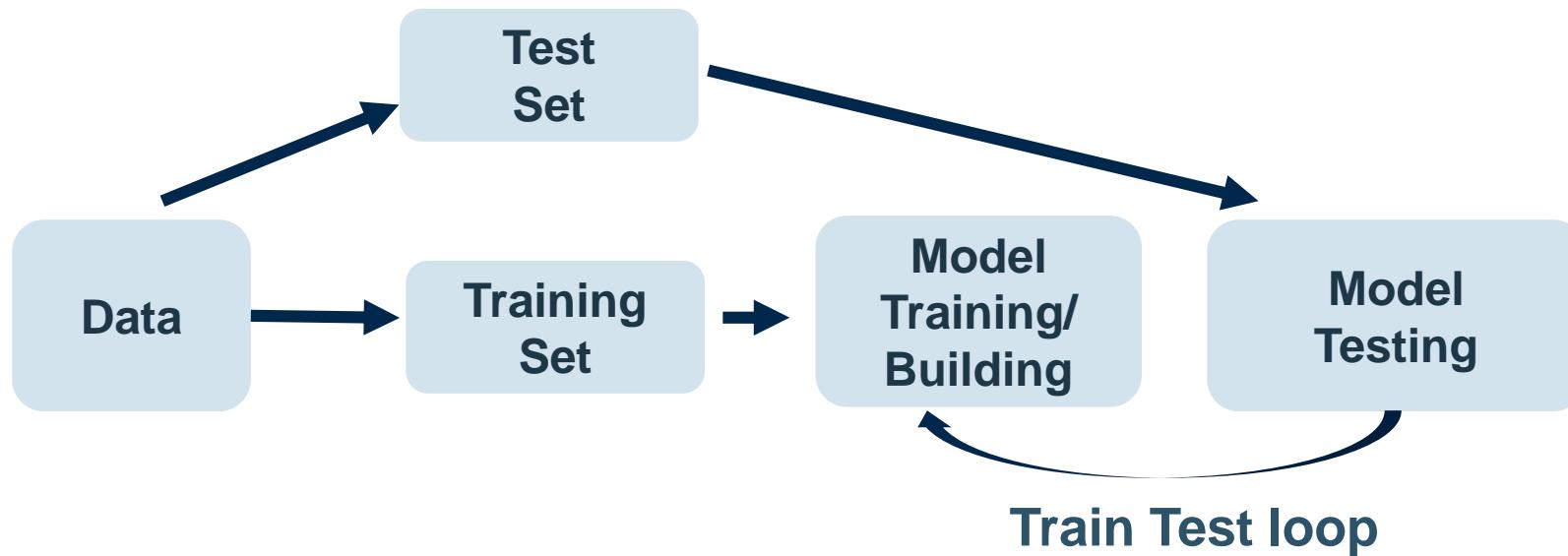
```
// parse input UserID::MovieID::Rating
def parseRating(str: String): Rating= {
    val fields = str.split("::")
    Rating(fields(0).toInt, fields(1).toInt,
fields(2).toDouble)
}

// create an RDD of Ratings objects
val ratingsRDD = ratingText.map(parseRating).cache()
```





ML Cross Validation Process





Create Model

```
// Randomly split ratings RDD into training data RDD (80%)
// and test data RDD (20%)

val splits = ratingsRDD.randomSplit(Array(0.8, 0.2), 0L)

val trainingRatingsRDD = splits(0).cache()
val testRatingsRDD = splits(1).cache()

// build a ALS user product matrix model with rank=20,
// iterations=10

val model = (new
ALS().setRank(20).setIterations(10).run(trainingRatingsRDD))
```





Get predictions

```
// Get the top 5 movie predictions for user 4169
val topRecsForUser = model.recommendProducts(4169, 5)

// get (user,product) pairs
val testUserProductRDD = testRatingsRDD.map {
  case Rating(user, product, rating) => (user, product)
}

// get predictions for (user,product) pairs
val predictionsForTestRDD = model.predict(testUserProductRDD)
```





Test Model

```
// prepare predictions for comparison
val predictionsKeyedByUserProductRDD = predictionsForTestRDD.map{
  case Rating(user, product, rating) => ((user, product), rating)
}

// prepare test for comparison
val testKeyedByUserProductRDD = testRatingsRDD.map{
  case Rating(user, product, rating) => ((user, product), rating)
}

//Join the test with predictions
val testAndPredictionsJoinedRDD =
  testKeyedByUserProductRDD.join(predictionsKeyedByUserProductRDD)
```





Test Model

```
val falsePositives =(testAndPredictionsJoinedRDD.filter{  
    case ((user, product), (ratingT, ratingP)) =>  
        (ratingT <= 1 && ratingP >=4)  
    })  
falsePositives.take(2)
```

```
Array[((Int, Int), (Double, Double))] =  
((3842,2858),(1.0,4.106488210964762)),  
((6031,3194),(1.0,4.790778049100913))
```





```
//Evaluate the model using Mean Absolute Error (MAE) between  
test and predictions
```

```
val meanAbsoluteError = testAndPredictionsJoinedRDD.map {  
    case ((user, product), (testRating, predRating)) =>  
        val err = (testRating - predRating)  
        Math.abs(err)  
} .mean()
```

```
meanAbsoluteError: Double = 0.7244940545944053
```





Soon to Come

- Spark On Demand Training
 - <https://www.mapr.com/services/mapr-academy/>
- Blogs and Tutorials:
 - Movie Recommendations with Collaborative Filtering
 - Spark Streaming





- <https://www.mapr.com/blog/parallel-and-iterative-processing-machine-learning-recommendations-spark>



Parallel and Iterative Processing for Machine Learning Recommendations with Spark





Examples and Resources





Coming soon Free Spark On Demand Training

- <https://www.mapr.com/services/mapr-academy/>

DEV 301 - Developing Hadoop Applications

This course teaches developers, with lectures and hands-on lab exercises, how to write Hadoop Applications using MapReduce and YARN in Java. The course extensively covers MapReduce programming, debugging, managing jobs, improving performance, working managing workflows, and using other programming languages for MapReduce.

[Learn More !\[\]\(4ecd4ba3a40eeca88dd72b4557f923ce_img.jpg\)](#)



Find my presentation and other related resources here:

<http://events.mapr.com/JacksonvilleJava>

(you can find this link in the event's page at meetup.com)



Today's Presentation



Free On-Demand Training



Whiteboard & demo
videos



Free eBooks



Free Hadoop Sandbox



And more...

MAPR[®]

Spark on MapR

- Certified Spark Distribution
- Fully supported and packaged by MapR in partnership with Databricks
 - mapr-spark package with Spark, Shark, Spark Streaming today
 - Spark-python, GraphX and MLLib soon
- YARN integration
 - Spark can then allocate resources from cluster when needed



References

- Spark web site: <http://spark.apache.org/>
- <https://databricks.com/>
- Spark on MapR:
 - <http://www.mapr.com/products/apache-spark>
- [Spark SQL and DataFrame Guide](#)
- [Apache Spark vs. MapReduce – Whiteboard Walkthrough](#)
- [Learning Spark - O'Reilly Book](#)
- [Apache Spark](#)



Q & A

Engage with us!

@mapr



maprtech

mapr-technologies



MapR

kbotzum@mapr.com



maprtech

