Apache Spark Fundamentals

Dilip K. Prasad

Outline

Introduction

Spark Fundamentals

Spark Architecture

Spark and it's Ecosystem

Spark vs Hadoop

RDD Fundamentals

Spark Transformations, Actions and Operations

Introduction

2009 AMPLab (Berkley) – resource/cluster manager

2009 No YARN

MESOS

Created framework to test Mesos – Spark was created (which is a in memory execution) – huge RAM requirement

2010/11 SPARK further development started

2012 Apache Spark -> Databricks

2015 It got traction and more support system came up

2021 INF2220 Cloud and Big Data Technology started

Spark version 0

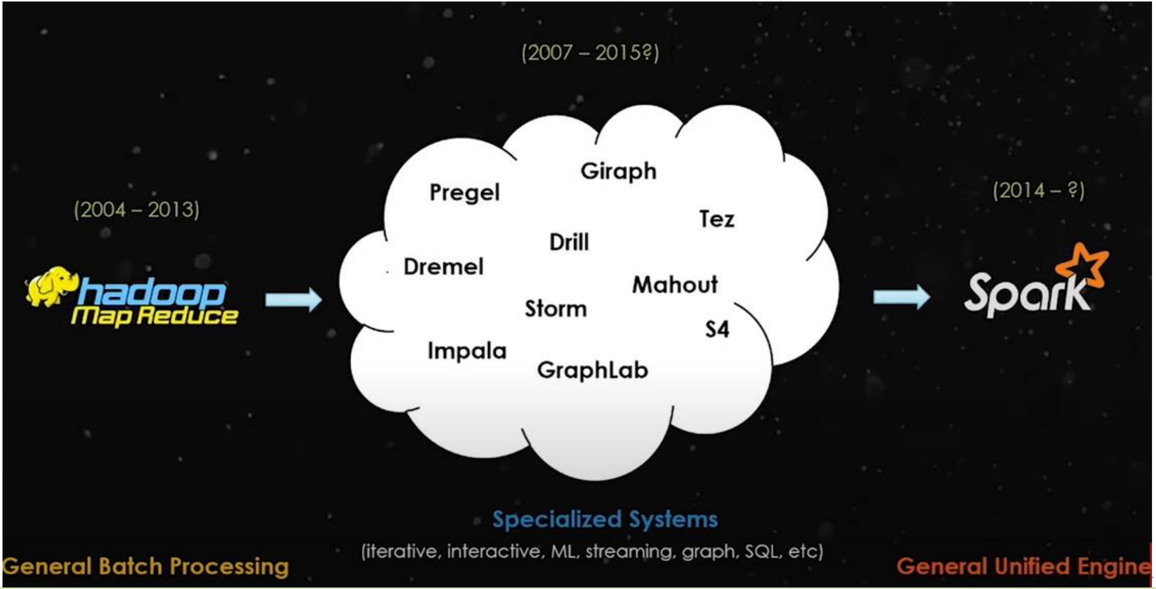
1.6

2.7

3.2.0

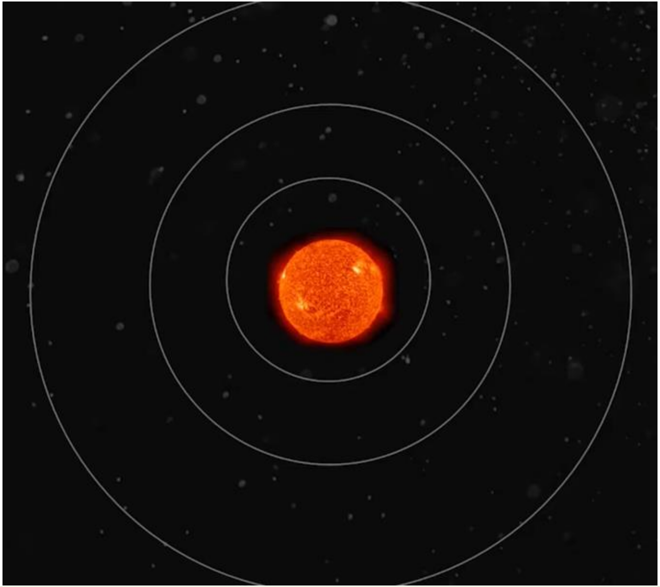
Best book – Spark the definitive guide – Mathe Zaharia (don't buy it its not text book type)

Changing Big Data World

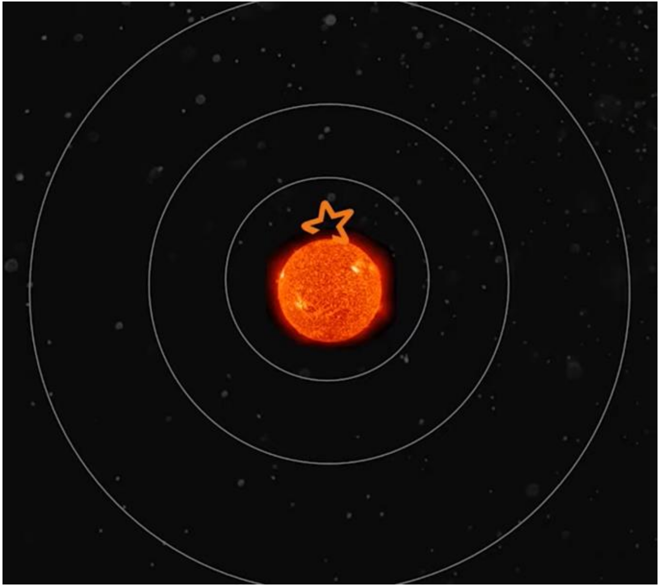


Introduction to Spark

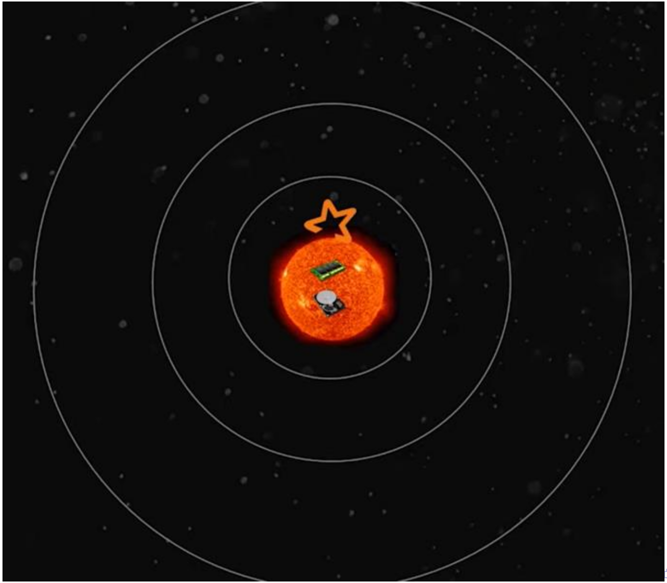
Introduction to Spark



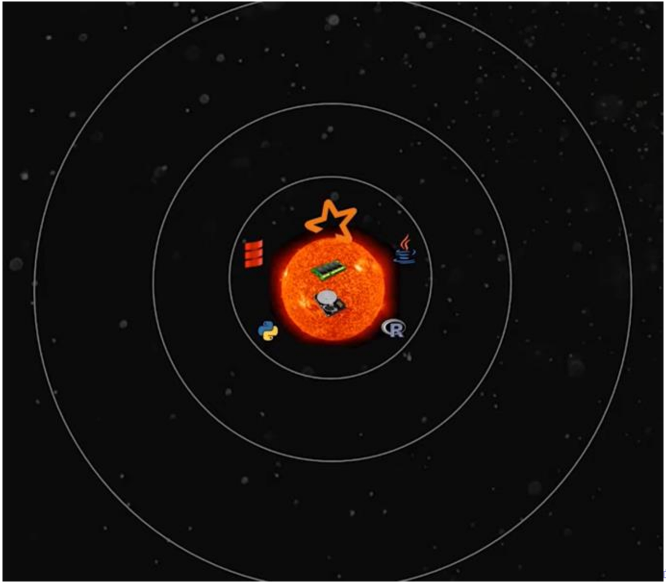
Spark Architecture & its Ecosystem



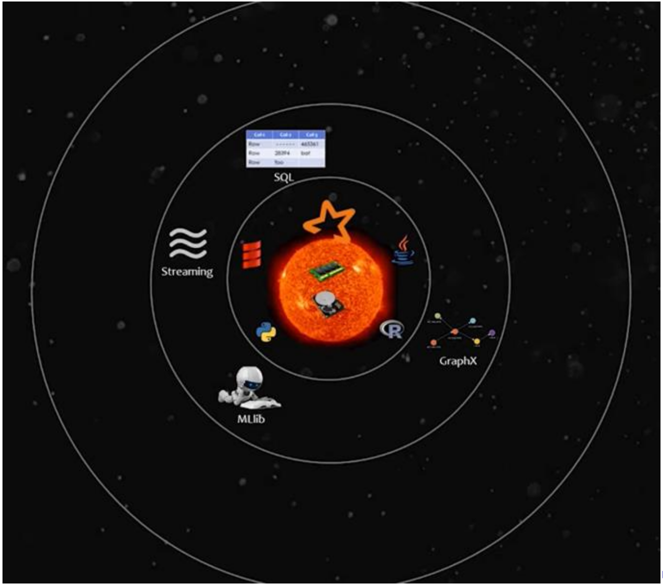
Spark Architecture & its Ecosystem



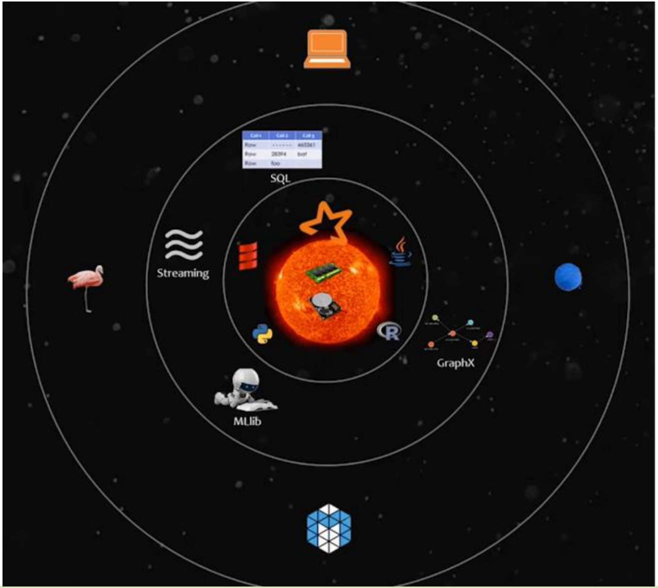
Spark Architecture & its Ecosystem



Spark Architecture & its Ecosystem

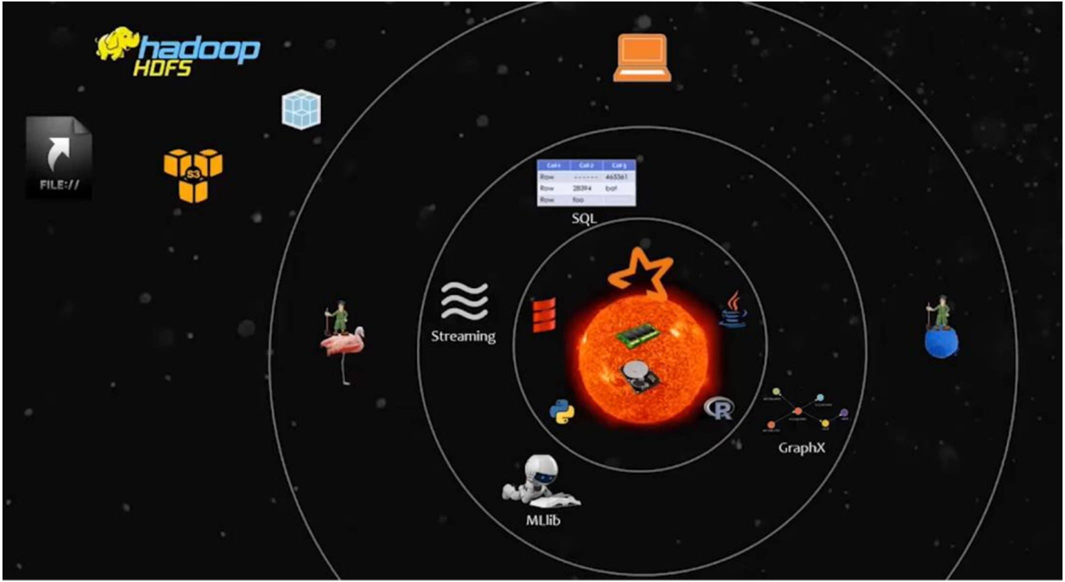


Spark Architecture & its Ecosystem

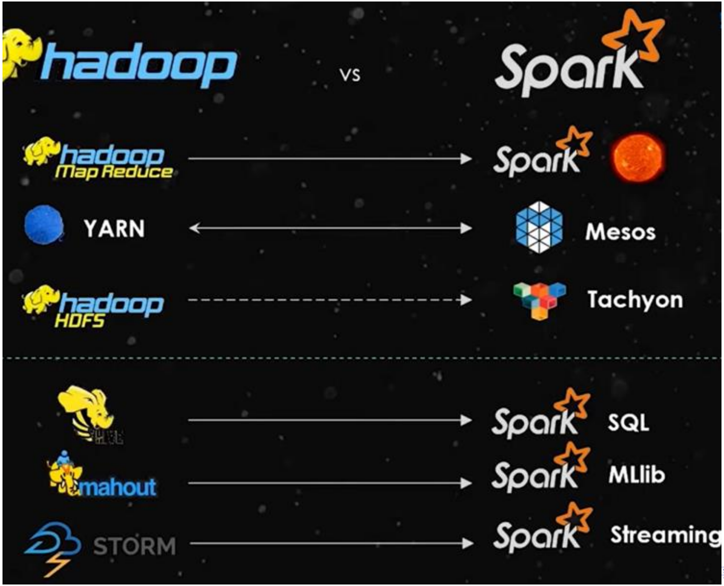


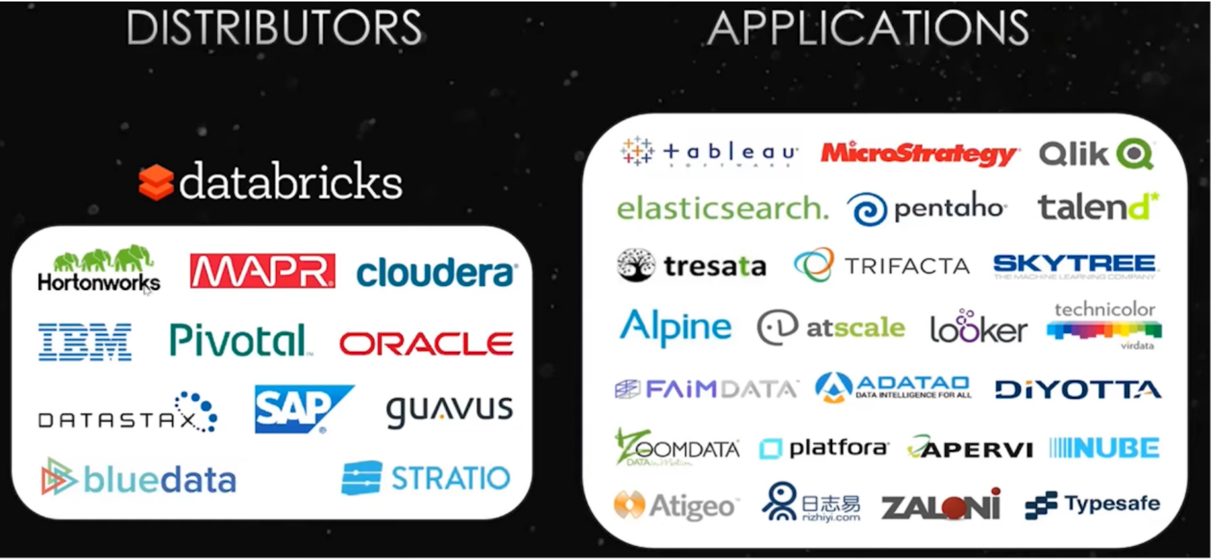
Spark Architecture & its Ecosystem

Spark Architecture & its Ecosystem

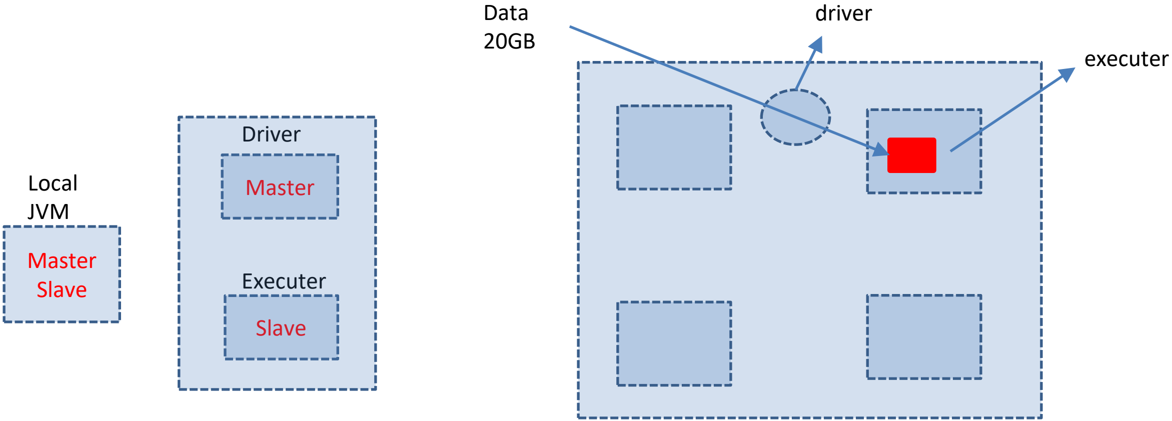


Hadoop Vs Spark




DISTRIBUTORS
APPLICATIONS
databricks
Hortonworks
MAPR
cloudera
IBM
Pivotal
ORACLE
DATASTAX
SAP
guavus
bluedata
STRATIO
tableau
MicroStrategy
Qlik
elasticsearch.
pentaho
talend
tresata
TRIFACTA
SKYTREE
Alpine
atscale
looker
technicolor
virdata
FAiMDATA
ADATAO
DATA INTELLIGENCE FOR ALL
DiYOTTA
ZOOMDATA
platfora
APERVI
NUBE
Atigeo
日志易
rizhiyi.com
ZALONI
Typesafe

How spark program get executed



Sort Competition

	Hadoop MR Record (2013)	Spark Record (2014)
Data Size	102.5 TB	100 TB
Elapsed Time	72 mins	23 mins
# Nodes	2100	206
# Cores	50400 physical	6592 virtualized
Cluster disk throughput	3150 GB/s (est.)	618 GB/s
Network	dedicated data center, 10Gbps	virtualized (EC2) 10Gbps network
Sort rate	**1.42 TB/min**	**4.27 TB/min**
Sort rate/node	**0.67 GB/min**	**20.7 GB/min**

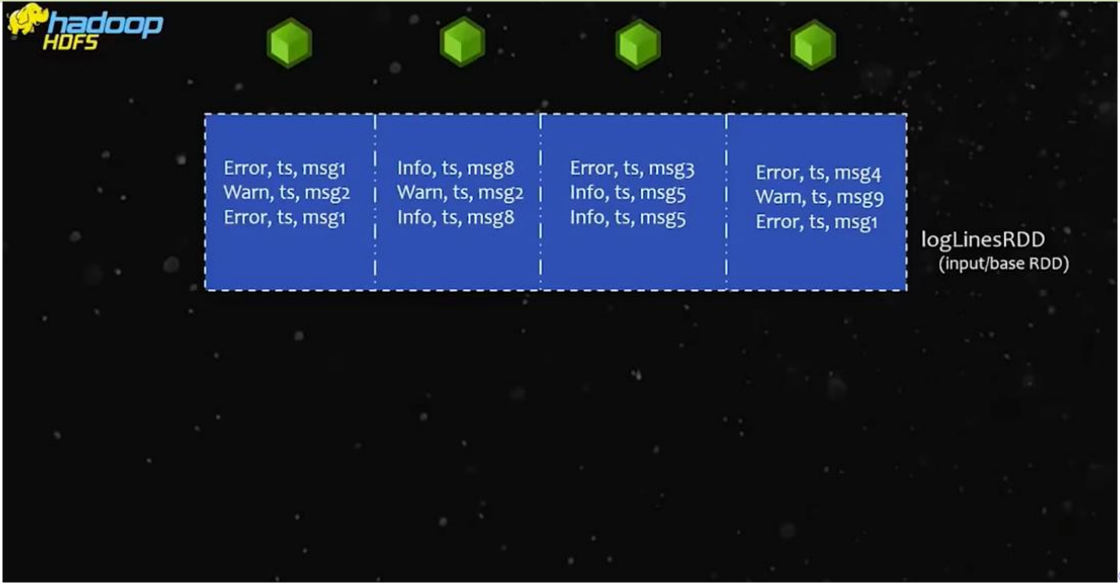
Spark, 3x faster with 1/10 the nodes

Sort benchmark, Daytona Gray: sort of 100 TB of data (1 trillion records)
http://databricks.com/blog/2014/11/05/spark-officially-sets-a-new-record-in-large-scale-sorting.html

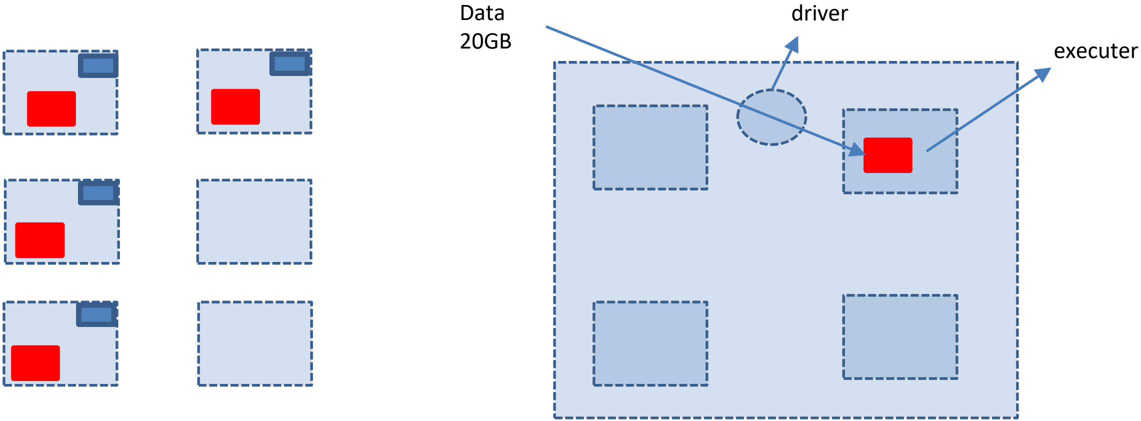
Resilient Distributed Datasets (RDDs)

- RDDs (Resilient Distributed Datasets) is Data Containers
- All the different processing components in Spark share the same abstraction called RDD
- As applications share the RDD abstraction, you can mix different kind of transformations to create new RDDs
- Created by parallelizing a collection or reading a file
- Fault tolerant

RDD


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)

RDD



DataFrames & SparkSQL

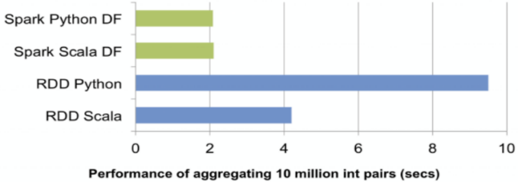
- DataFrames (DFs) is one of the other distributed datasets organized in named columns
- Similar to a relational database, Python Pandas Dataframe or R's DataTables
  - Immutable once constructed
  - Track lineage
  - Enable distributed computations
- How to construct Dataframes
  - Read from file(s)
  - Transforming an existing DFs(Spark or Pandas)
  - Parallelizing a python collection list
  - Apply transformations and actions

DataFrame example

```
// Create a new DataFrame that contains “students”
students = users.filter(users.age < 21)

//Alternatively, using Pandas-like syntax
students = users[users.age < 21]

//Count the number of students users by gender
students.groupBy("gender").count()

// Join young students with another DataFrame called logs
students.join(logs, logs.userId == users.userId,
“left_outer")
```

RDDs vs. DataFrames

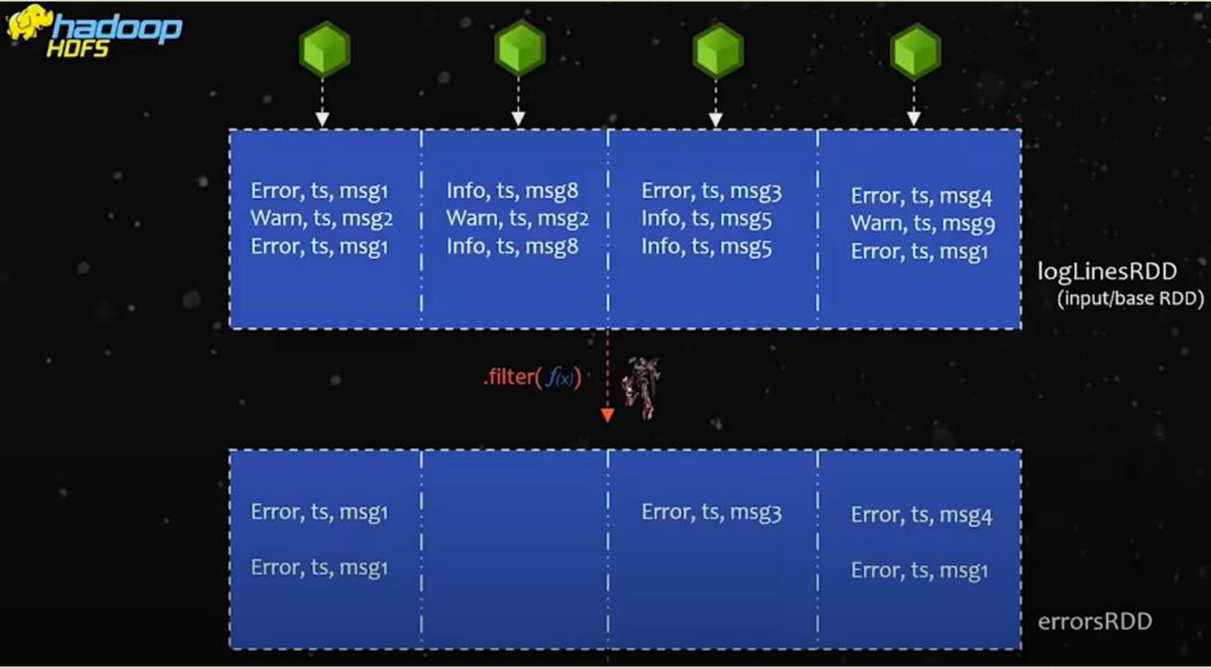
- RDDs provide a low level interface into Spark
- DataFrames have a schema
- DataFrames are cached and optimized by Spark
- DataFrames are built on top of the RDDs and the core Spark API



Example: performance

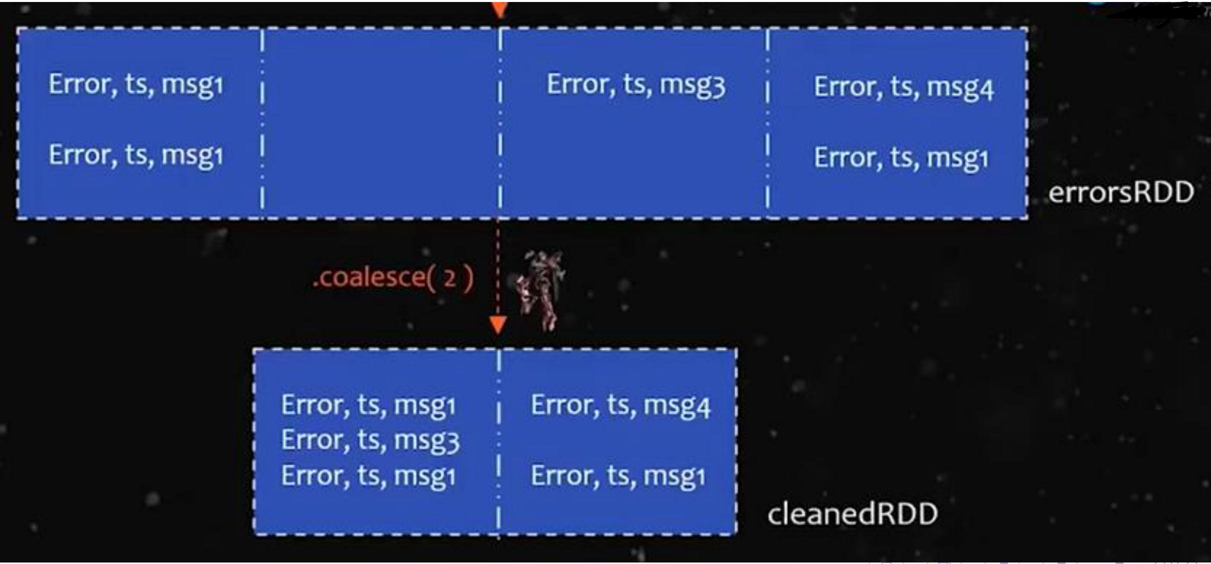
Spark Operations

Transformations (create a new RDD)	map filter sample groupByKey reduceByKey sortByKey intersection	flatMap union join cogroup cross mapValues reduceByKey
Actions (return results to driver program)	collect Reduce Count takeSample take lookupKey	first take takeOrdered countByKey save foreach



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

Transformation

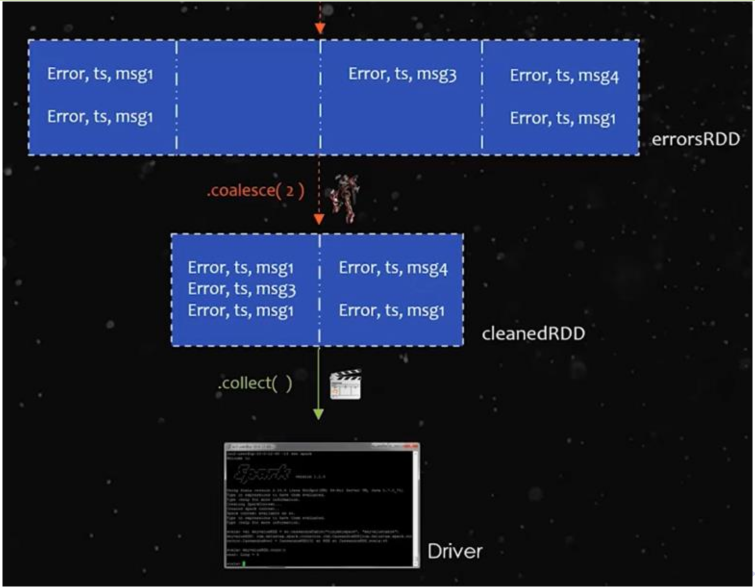


Actions

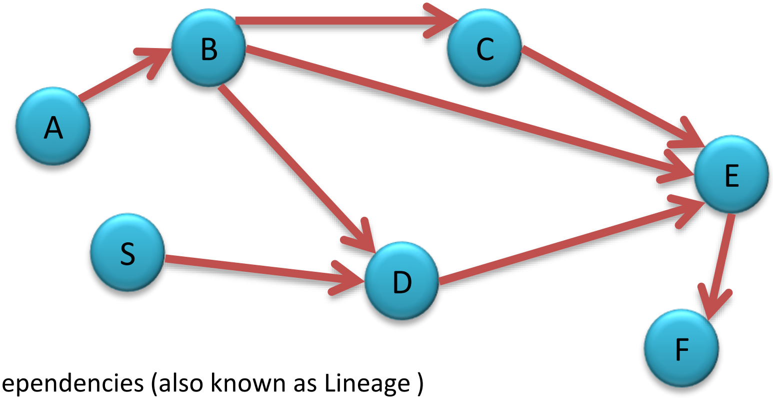
What is an action?

- The final stage of the workflow
- Triggers the execution of the DAG
- Returns the results to the driver
- Or writes the data to HDFS or to a file

Spark Actions



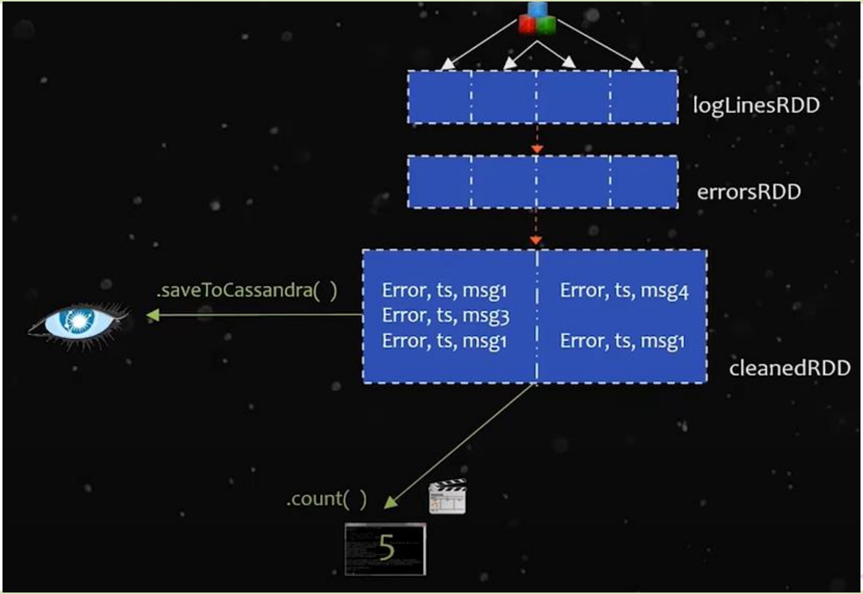
Directed Acyclic Graphs (DAG)



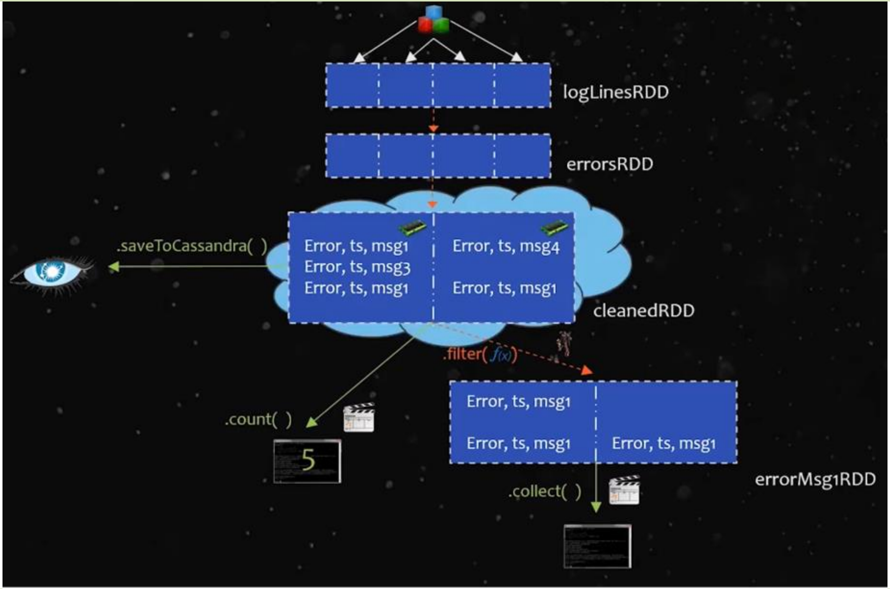
DAGs track dependencies (also known as Lineage)

- nodes are RDDs
- arrows are Transformations

Spark DAG

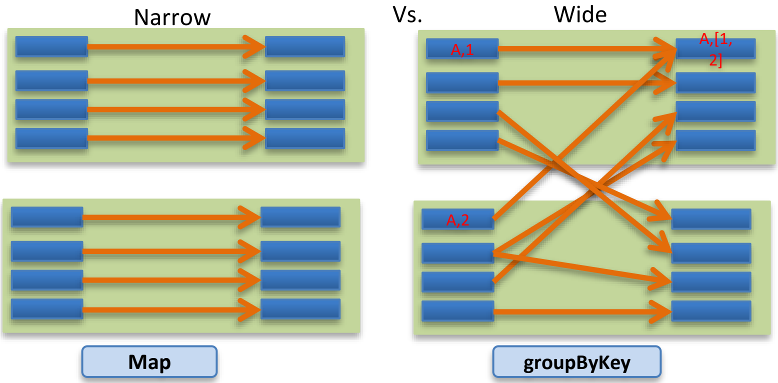

logLinesRDD
errorsRDD
cleanedRDD
Error, ts, msg1
Error, ts, msg3
Error, ts, msg1
Error, ts, msg4
Error, ts, msg1
.saveToCassandra( )
.count( )
5

Spark DAG

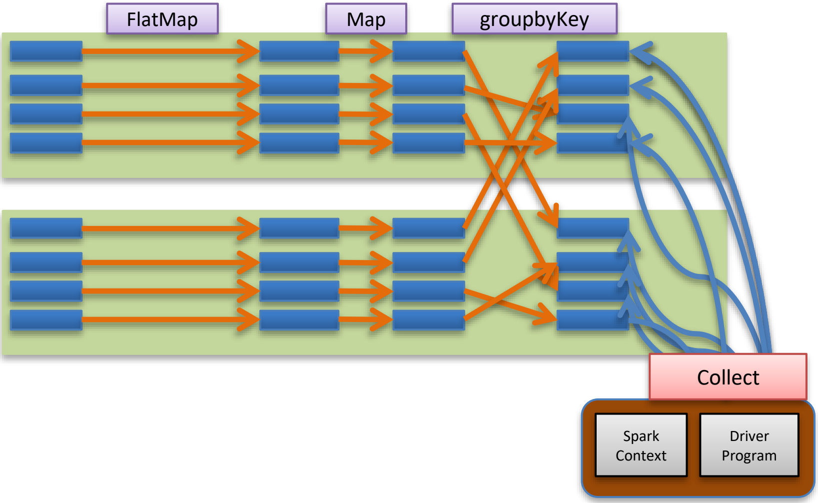


logLinesRDD
errorsRDD
.saveToCassandra( )
Error, ts, msg1
Error, ts, msg3
Error, ts, msg1
Error, ts, msg4
Error, ts, msg1
cleanedRDD
.filter(f(x))
.count( )
5
Error, ts, msg1
Error, ts, msg1
Error, ts, msg1
errorMsg1RDD
.collect( )

Narrow Vs. Wide transformation



Spark Workflow



Python RDD API Examples

Word count

```
text_file = sc.textFile("hdfs://usr/godil/text/book.txt")
counts = text_file.flatMap(lambda line: line.split(" ")) \
      .map(lambda word: (word, 1)) \
     .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("hdfs://usr/godil/output/wordCount.txt")
```

Logistic Regression

```
# Every record of this DataFrame contains the label and
# features represented by a vector.
df = sqlContext.createDataFrame(data, ["label", "features"])
# Set parameters for the algorithm.
# Here, we limit the number of iterations to 10.
lr = LogisticRegression(maxIter=10)
# Fit the model to the data.
model = lr.fit(df)
# Given a dataset, predict each point's label, and show the results.
model.transform(df).show()
```

Examples from http://spark.apache.org/

RDD Persistence and Removal

RDD Persistence

RDD.persist()

Storage level:

MEMORY_ONLY, MEMORY_AND_DISK, MEMORY_ONLY_SER, DISK_ONLY,…….

RDD Removal

RDD.unpersist()

Broadcast Variables and Accumulators (Shared Variables)

- Broadcast variables allow the programmer to keep a read-only variable cached on each node, rather than sending a copy of it with tasks

```
>broadcastV1 = sc.broadcast([1, 2, 3,4,5,6])
>broadcastV1.value
[1,2,3,4,5,6]
```

- Accumulators are variables that are only “added” to through an associative operation and can be efficiently supported in parallel

```
accum = sc.accumulator(0)
accum.add(x)
accum.value
```

Spark’s Main Use Cases

- Streaming Data
- Machine Learning
- Interactive Analysis
- Data Warehousing
- Batch Processing
- Exploratory Data Analysis
- Graph Data Analysis
- Spatial (GIS) Data Analysis
- And many more

Spark in the Real World (I)

Uber – the online taxi company gathers terabytes of event data from its mobile users every day.

- By using Kafka, Spark Streaming, and HDFS, to build a continuous ETL pipeline
- Convert raw unstructured event data into structured data as it is collected
- Uses it further for more complex analytics and optimization of operations

Pinterest – Uses a Spark ETL pipeline

- Leverages Spark Streaming to gain immediate insight into how users all over the world are engaging with Pins—in real time.
- Can make more relevant recommendations as people navigate the site
- Recommends related Pins
- Determine which products to buy, or destinations to visit

Spark in the Real World (II)

Here are Few other Real World Use Cases:

Conviva – 4 million video feeds per month

This streaming video company is second only to YouTube.

Uses Spark to reduce customer churn by optimizing video streams and managing live video traffic

Maintains a consistently smooth, high quality viewing experience.

Capital One – is using Spark and data science algorithms to understand customers in a better way.

Developing next generation of financial products and services

Find attributes and patterns of increased probability for fraud

Netflix – leveraging Spark for insights of user viewing habits and then recommends movies to them.

User data is also used for content creation

Spark: when not to use

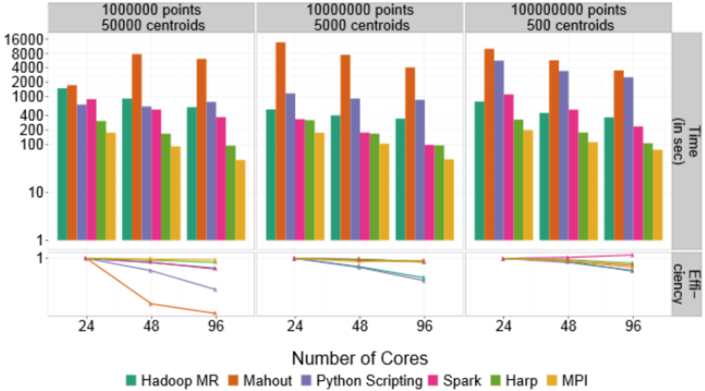
Even though Spark is versatile, that doesn't mean Spark's in-memory capabilities are the best fit for all use cases:

- For many simple use cases Apache MapReduce and Hive might be a more appropriate choice
- Spark was not designed as a multi-user environment
- Spark users are required to know that memory they have is sufficient for a dataset
- Adding more users adds complications, since the users will have to coordinate memory usage to run code

HPC and Big Data Convergence

- Clouds and supercomputers are collections of computers networked together in a datacenter
- Clouds have different networking, I/O, CPU and cost trade-offs than supercomputers
- Cloud workloads are data oriented vs. computation oriented and are less closely coupled than supercomputers
- Principles of parallel computing same on both
- Apache Hadoop and Spark vs. Open MPI

HPC and Big Data K-Means example



MPI definitely outpaces Hadoop, but can be boosted using a hybrid approach of other technologies that blend HPC and big data, including Spark and HARP. Dr. Geoffrey Fox, Indiana University. (http://arxiv.org/pdf/1403.1528.pdf)

PGAS Vs MPI vs openMP

	Thread Count	Memory Count	Nonlocal Access
Traditional	1	1	N/A
OpenMP	Either 1 or p	1	N/A
MPI	p	p	No. Message required.
C+CUDA	1+p	2 (Host/device)	No. DMA required.
UPC, CAF, pMatlab	p	p	Supported.
X10, Asynchronous PGAS	p	q	Supported.