Outline

- Growth of big datasets
- Introduction to Apache Hadoop and Spark for developing applications
- Components of Hadoop, HDFS, MapReduce and HBase
- Capabilities of Spark and the differences from a typical MapReduce solution
- Some Spark use cases for data analysis

Growth of Big Datasets

- Internet/Online Data
  - Clicks
  - Searches
  - Server requests
  - Web logs
  - Cell phone logs
  - Mobile GPS locations
  - User generated content
  - Entertainment (YouTube, Netflix, Spotify, …)
- Healthcare and Scientific Computations
  - Genomics, medical images, healthcare data, billing data
- Graph data
  - Telecommunications network
  - Social networks (Facebook, Twitter, LinkedIn, …)
  - Computer networks
- Internet of Things
  - RFID
  - Sensors
- Financial data

Data

- The Large Hadron Collider produces about 30 petabytes of data per year
- Facebook’s data is growing at 8 petabytes per month
- The New York stock exchange generates about 4 terabyte of data per day
- YouTube had around 80 petabytes of storage in 2012
- Internet Archive stores around 19 petabytes of data
- Boeing 787 produces half a terabyte of data per flight

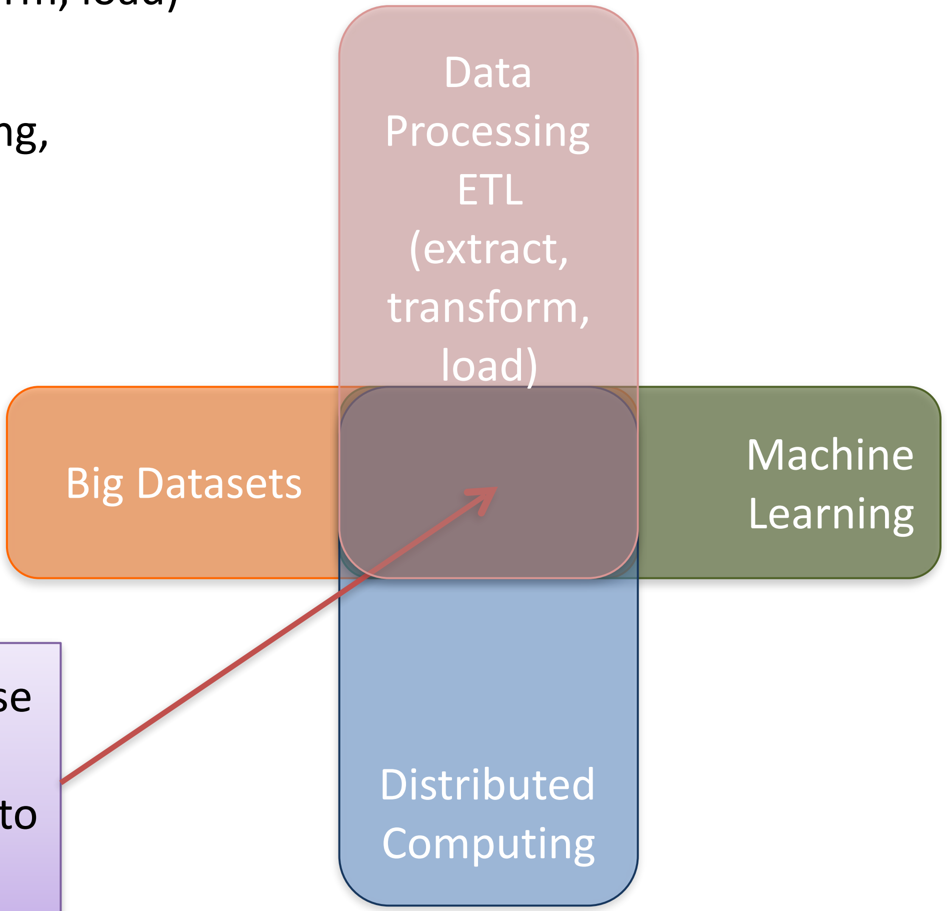
Cloud and Distributed Computing

- The second trend is pervasiveness of cloud based storage and computational resources
  - For processing of these big datasets
- Cloud characteristics
  - Provide a scalable standard environment
  - On-demand computing
  - Pay as you need
  - Dynamically scalable
  - Cheaper

Data Processing and Machine learning Methods

- Data processing (third trend)
  - Traditional ETL (extract, transform, load)
  - Data Stores (HBase, ……..)
  - Tools for processing of streaming, multimedia & batch data
- Machine Learning (fourth trend)
  - Classification
  - Regression
  - Clustering
  - Collaborative filtering

Working at the Intersection of these four trends is very exciting and challenging and require new ways to store and process **Big Data**

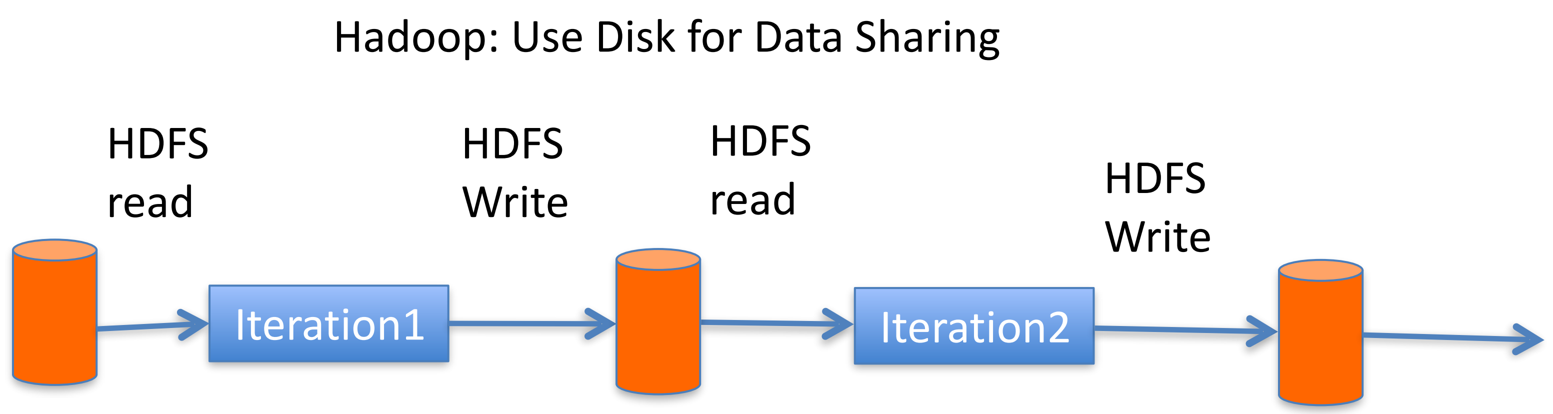


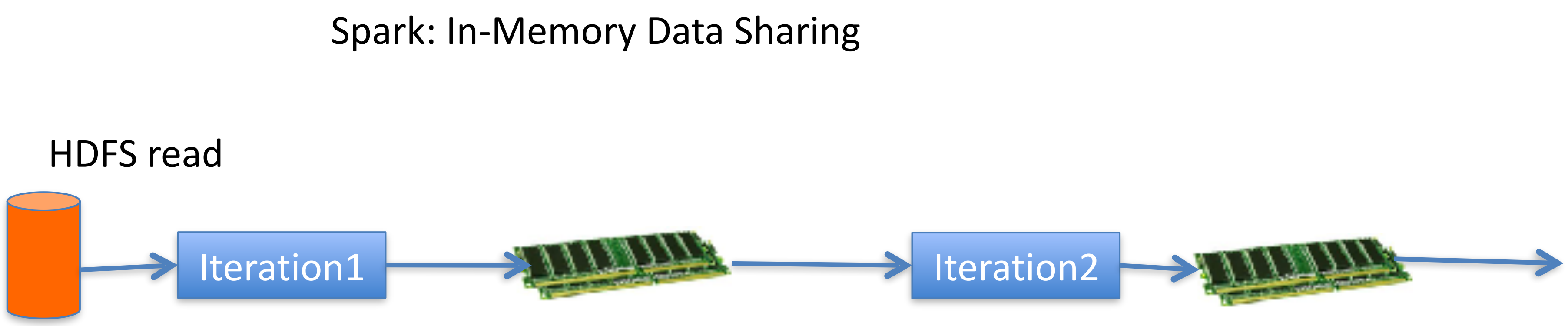
One Solution is Apache Spark

- A new general framework, which solves many of the short comings of MapReduce
- Capable of leveraging the Hadoop ecosystem, e.g. HDFS, YARN, HBase, S3, …
- Has many other workflows, i.e. join, filter, flatMapdistinct, groupByKey, reduceByKey, sortByKey, collect, count, first…
  - (around 30 efficient distributed operations)
- In-memory caching of data (for iterative, graph, and machine learning algorithms, etc.)
- Native Scala, Java, Python, and R support
- Supports interactive shells for exploratory data analysis
- Spark API is extremely simple to use
- Developed at AMPLab UC Berkeley, now by databricks.com

Matei Zahari

Spark Uses Memory instead of Disk





Sort competition

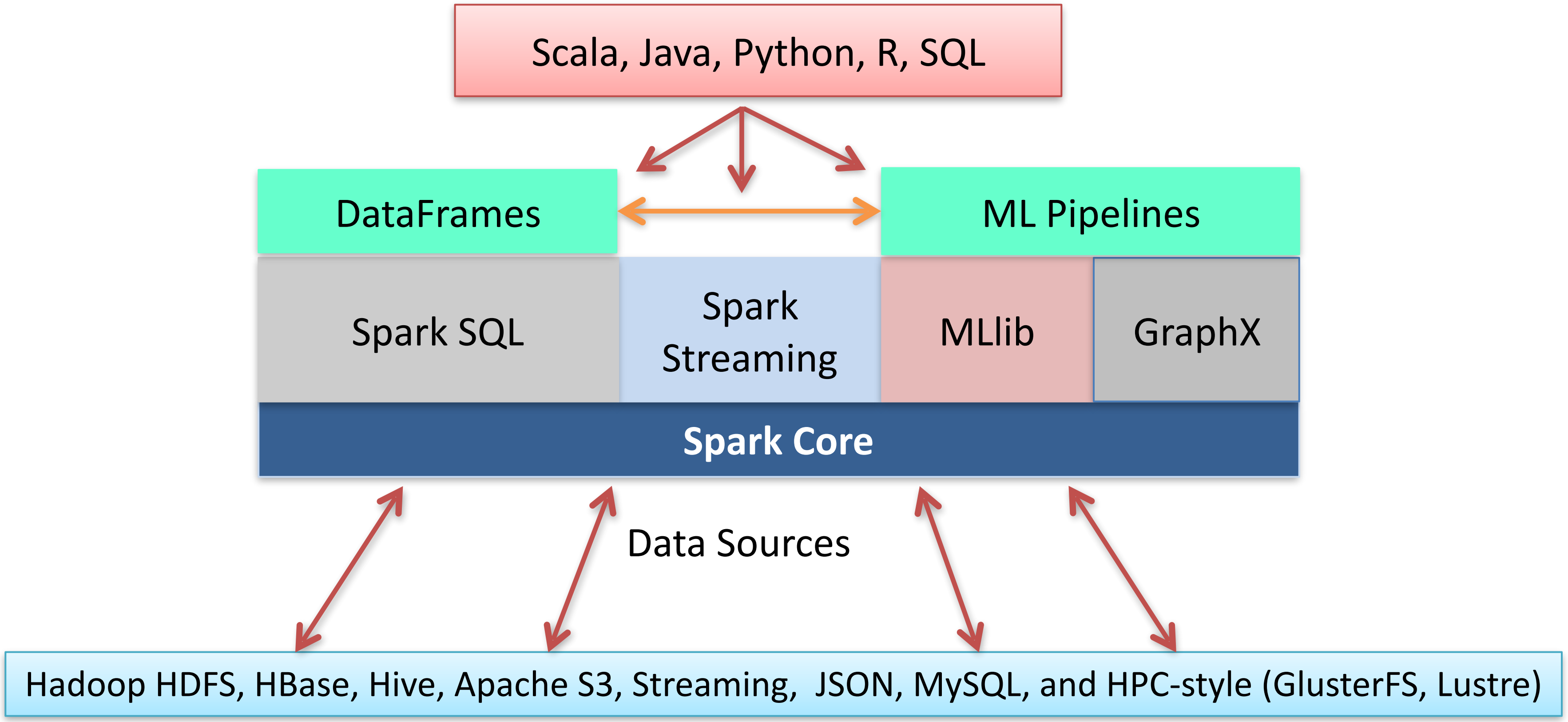
	Hadoop MR Record (2013)	**Spark Record (2014)**
Data Size	102.5 TB	100 TB
Elapsed Time	72 mins	23 mins
# Nodes	2100	206
# Cores	50400 physical	6592 virtualized
Cluster disk throughput	3150 GB/s (est.)	618 GB/s
Network	dedicated data center, 10Gbps	virtualized (EC2) 10Gbps network
Sort rate	**1.42 TB/min**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

Apache Spark

Apache Spark supports data analysis, machine learning, graphs, streaming data, etc. It can read/write from a range of data types and allows development in multiple languages.


Scala, Java, Python, R, SQL
DataFrames
ML Pipelines
Spark SQL
Spark Streaming
MLlib
GraphX
Spark Core
Data Sources
Hadoop HDFS, HBase, Hive, Apache S3, Streaming, JSON, MySQL, and HPC-style (GlusterFS, Lustre)

Resilient Distributed Datasets (RDDs)

- RDDs (Resilient Distributed Datasets) are 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, immutable and distributed

DataFrames & SparkSQL

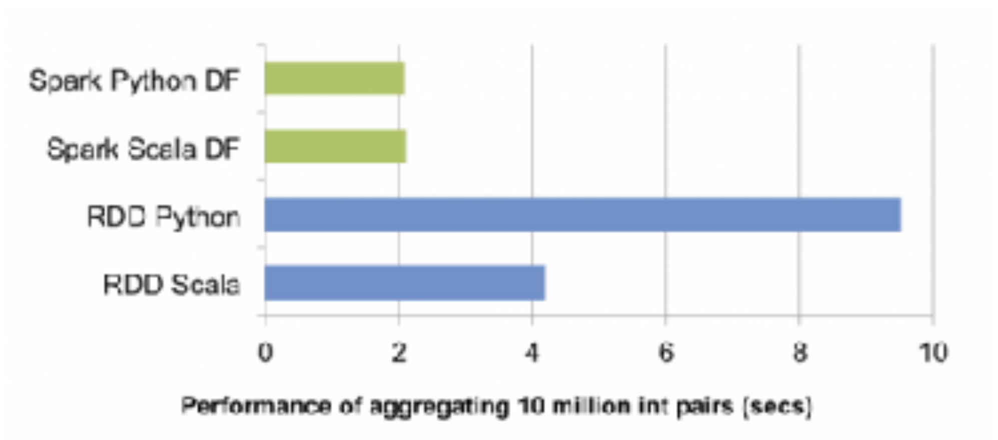
- DataFrames (DFs) is one of the other distributed datasets organized in named columns
- Similar to a relational database, Python Pandas Dataframe or R’s DataTables
  - Immutable once constructed
  - Track lineage
  - Enable distributed computations
- How to construct Dataframes
  - Read from file(s)
  - 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

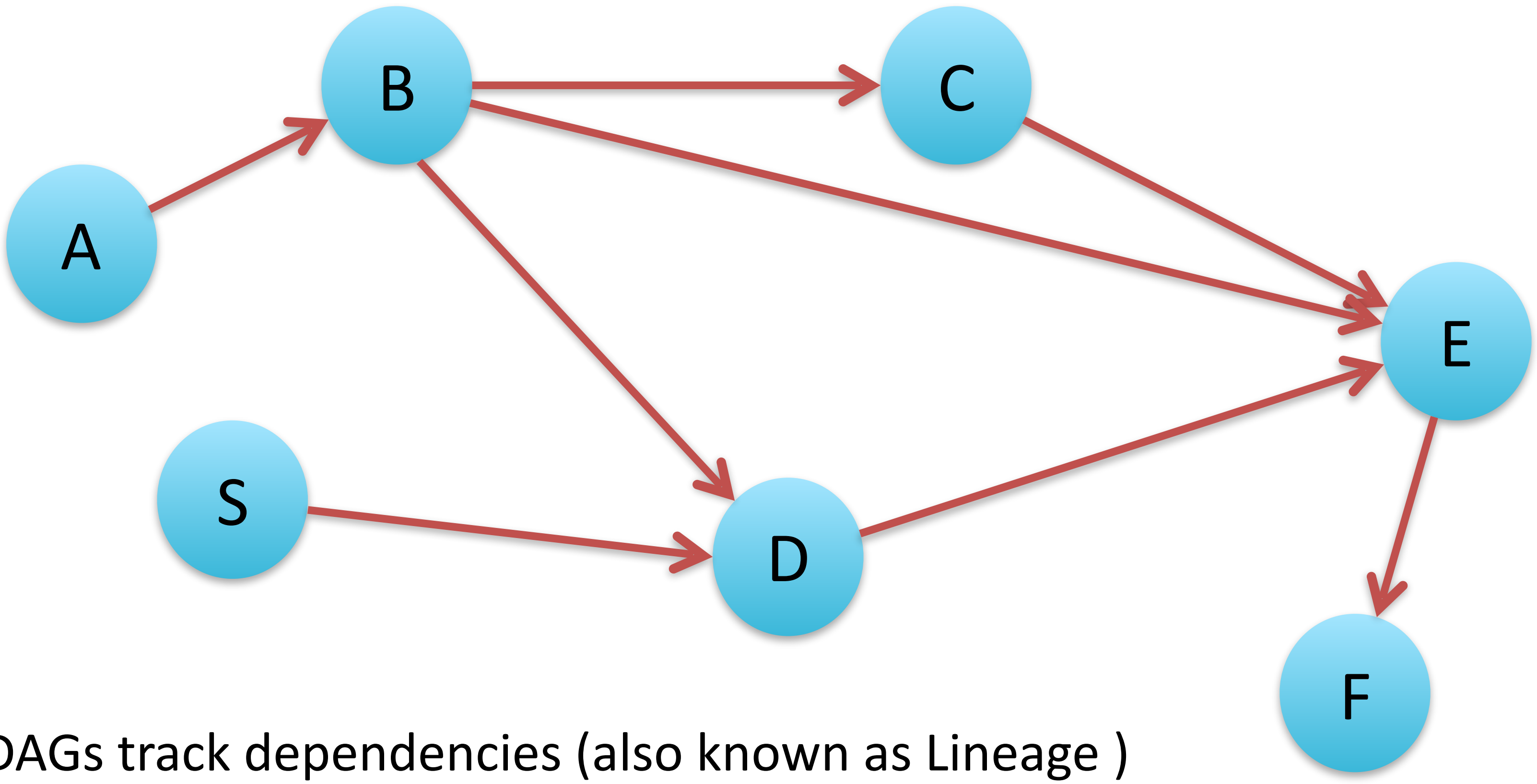


Example: performance

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

Directed Acyclic Graphs (DAG)


$$\begin{tikzcd}
 & B \arrow[rr] \arrow[dr] \arrow[rrr] & & C \arrow[r] & E \arrow[d] \\
A \arrow[ur] & S \arrow[r] & D \arrow[urr] & & F
\end{tikzcd}$$

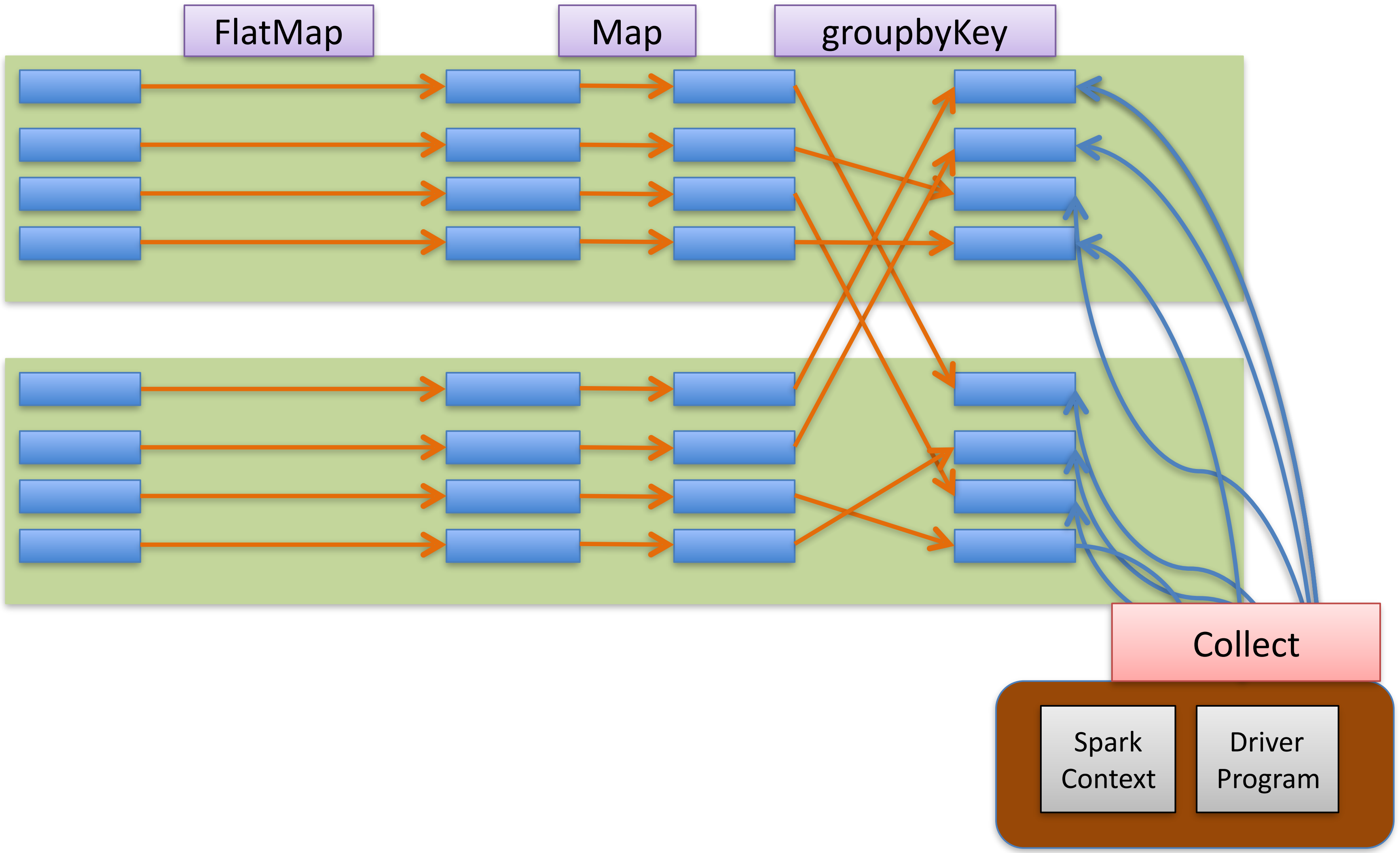
DAGs track dependencies (also known as Lineage)

- nodes are RDDs
- arrows are Transformations

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 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 cache : don’t overdo it

- 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