Introduction to Apache Spark

May 2016 HRUG

Doug Forrest

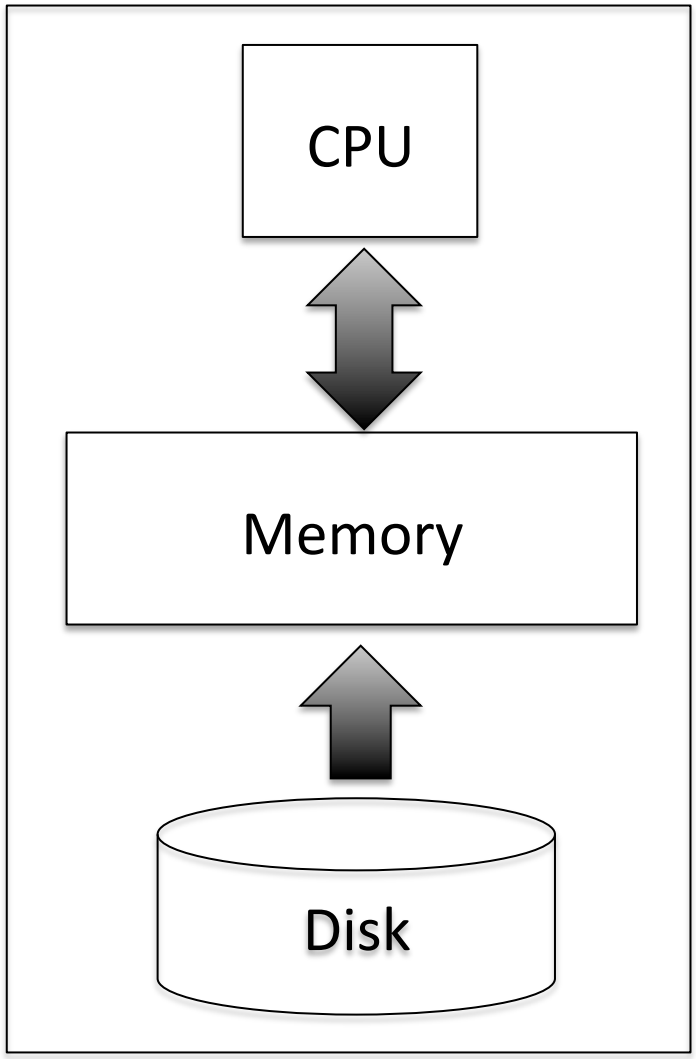
Outline

- Cluster Computing
- Spark Overview
- Getting Started
- Pyspark API
- Example
- Q&A

What is Big Data?

- Data sets that are so large or complex that traditional data processing applications are inadequate - wikipedia
- Data size is a moving target
- Characterized by:
  - Volume – amount of data
  - Velocity – speed of input / output
  - Variety – data types and sources

Typical Workflow



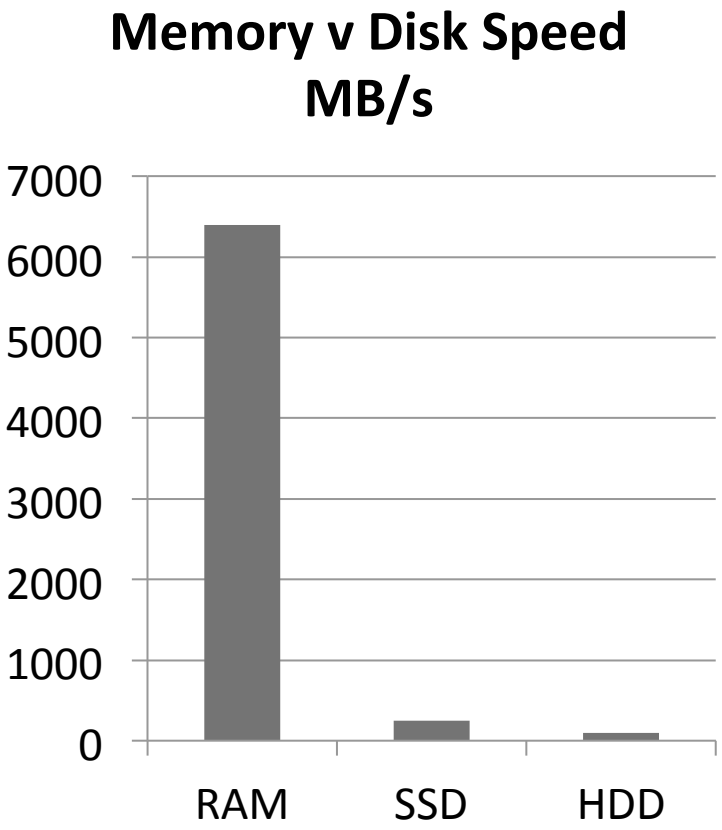
Single Node Architecture

```
# read from disk into memory
df = read.csv('dataset.csv')
# query data in memory
summary(df)
```

- Takes advantage of fast reads from memory

What happens when the dataset is too large to load into memory?

The Problem: Disk Bottleneck

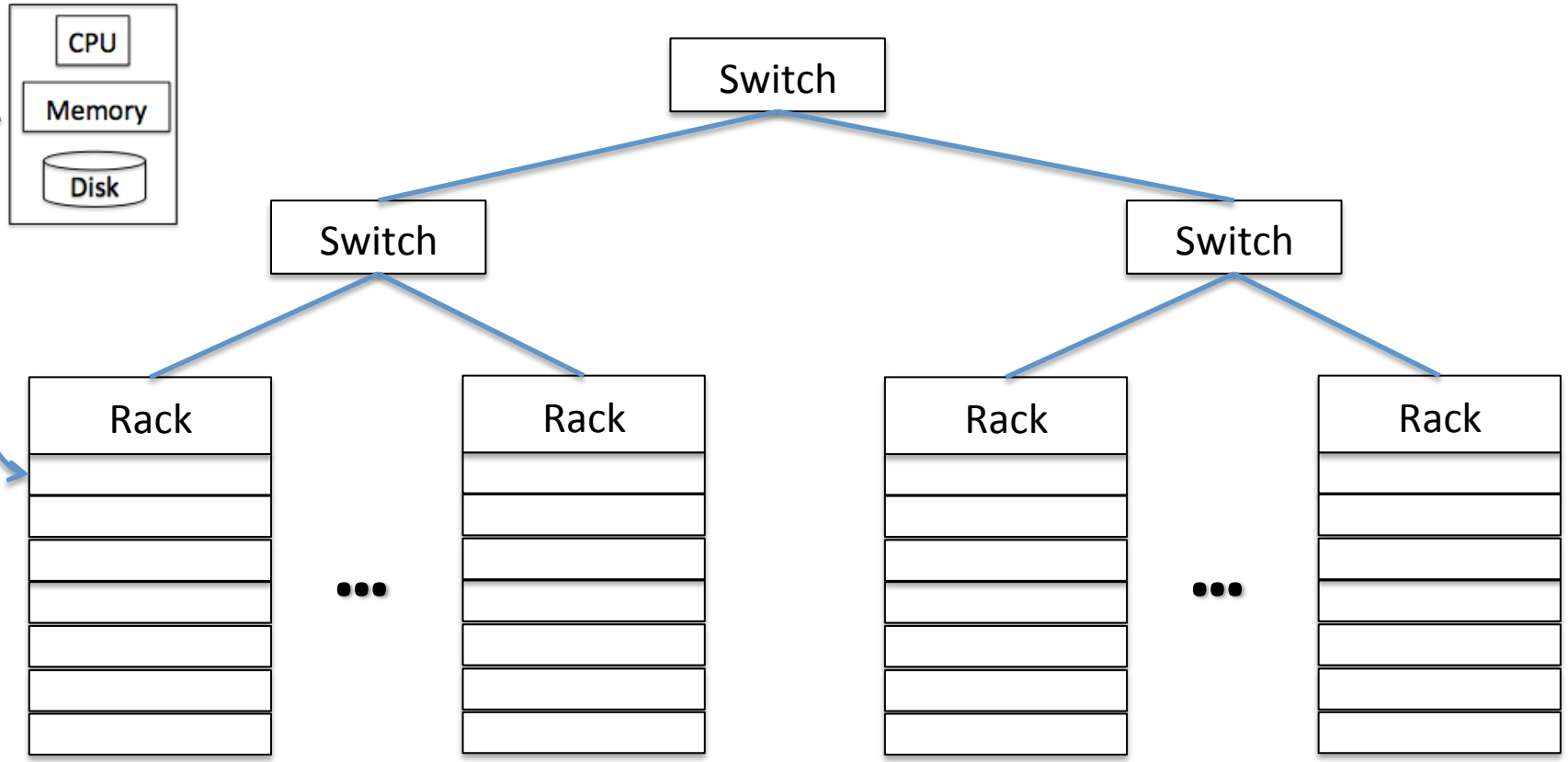


Data Size	Magnetic Disk (100 MB/s)	SSD (250 MB/s)
500 GB	1 Hour, 23 Minutes	33 Minutes
1 TB	3 Hours	1 Hour
50 TB	6 Days	2.3 Days
1 PB	116 Days	46 Days

Disk read time, single pass

- Most useful applications are iterative
- Single machine storage is limited
- Need to distribute data over multiple nodes

Cluster Architecture



- Each rack contains 16-64 commodity nodes
- Data center may have 50,000 – 100,000 nodes

Cluster Computing Complications

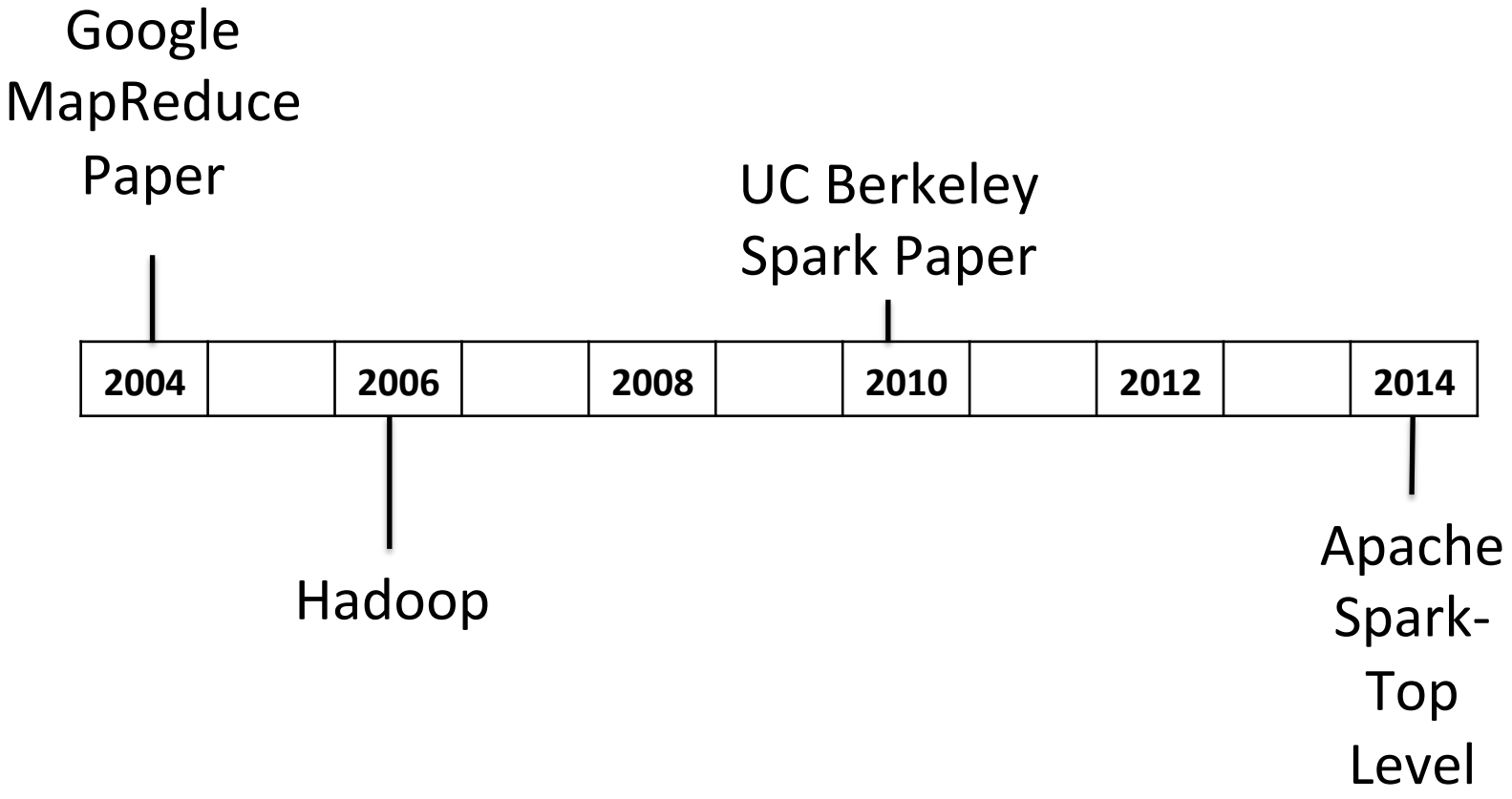
- Node failures
  - Files must be stored redundantly
    - Distributed File Systems (DFS): Files are divided into chunks and replicated across nodes
  - Computations divided into tasks
    - Cluster Manager to manage nodes and tasks
    - Tasks can be sent to different nodes if failure or long running
- Rack failures and data center failures
- Network speed
- Data Locality

Spark Overview

spark.apache.org

- Software for large scale data processing on a cluster of computers
- Developed at UC Berkeley, now top level Apache Project
- Free and Open Source (Apache License 2.0)
- Written in Scala, and runs on the Java Virtual Machine
- Active Development, one of the most active open source big data projects

Timeline


Google MapReduce Paper
UC Berkeley Spark Paper
2004
2006
2008
2010
2012
2014
Hadoop
Apache Spark-Top Level

Why Spark?

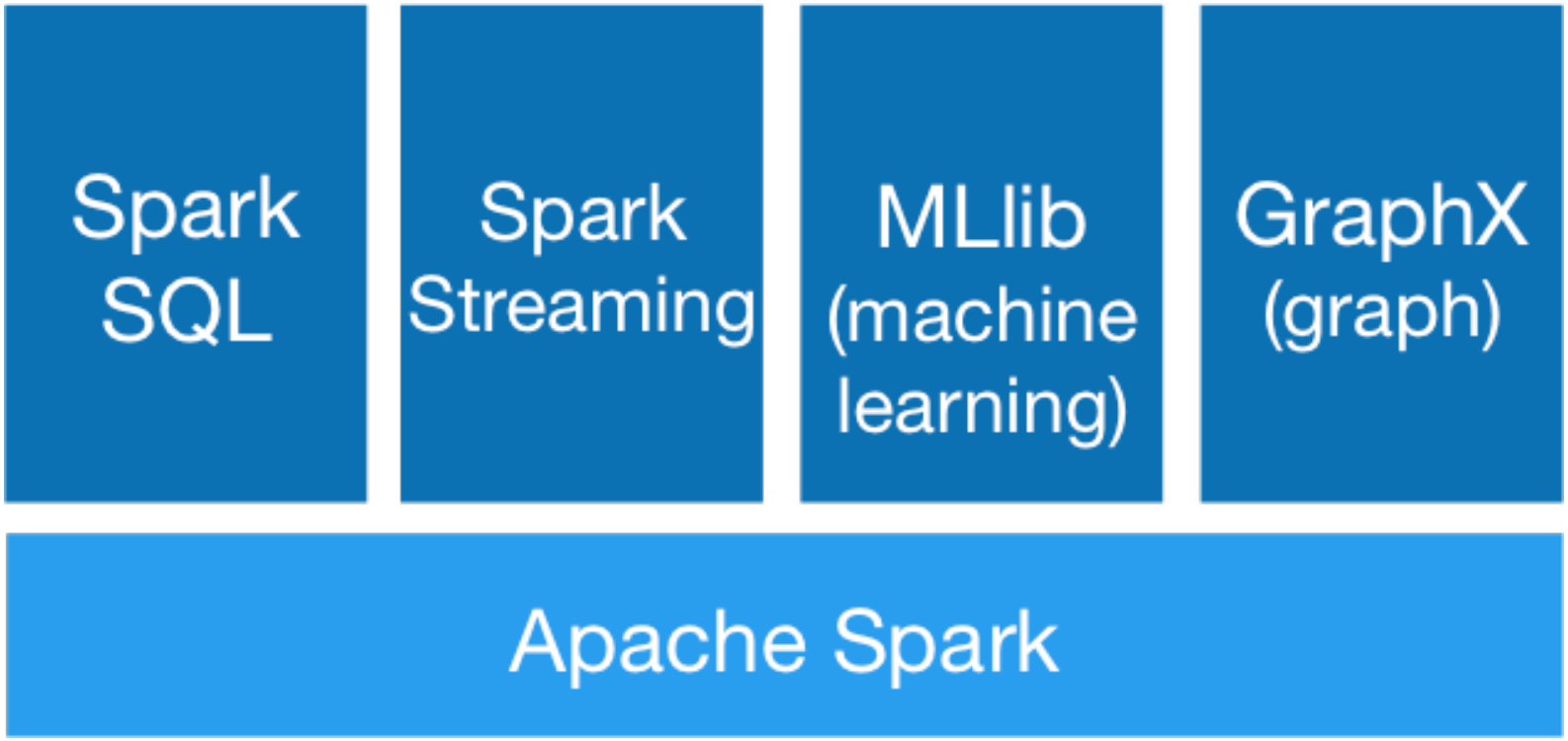
spark.apache.org

- Ease of use
- Multipurpose
- Compatibility
- Speed

Easy to use

- Hides complexities
- APIs in Java, Python and R
- High level of abstraction
- Interactive Shell
- Works with many data sources and cluster managers

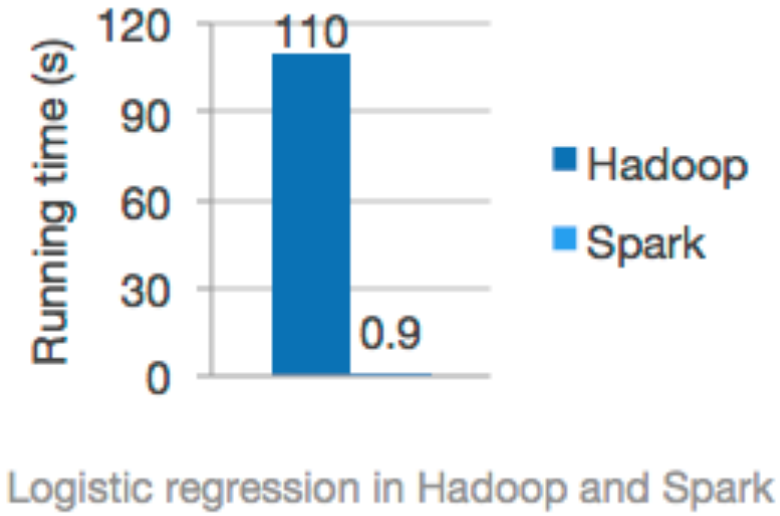
Multifunction



spark.apache.org

- Supports iterative, batch and stream processing
- Many third party packages

Fast



Logistic regression in Hadoop and Spark

- In-memory computing
- DAG execution

spark.apache.org

10x faster on disk and 100x faster in memory than Hadoop

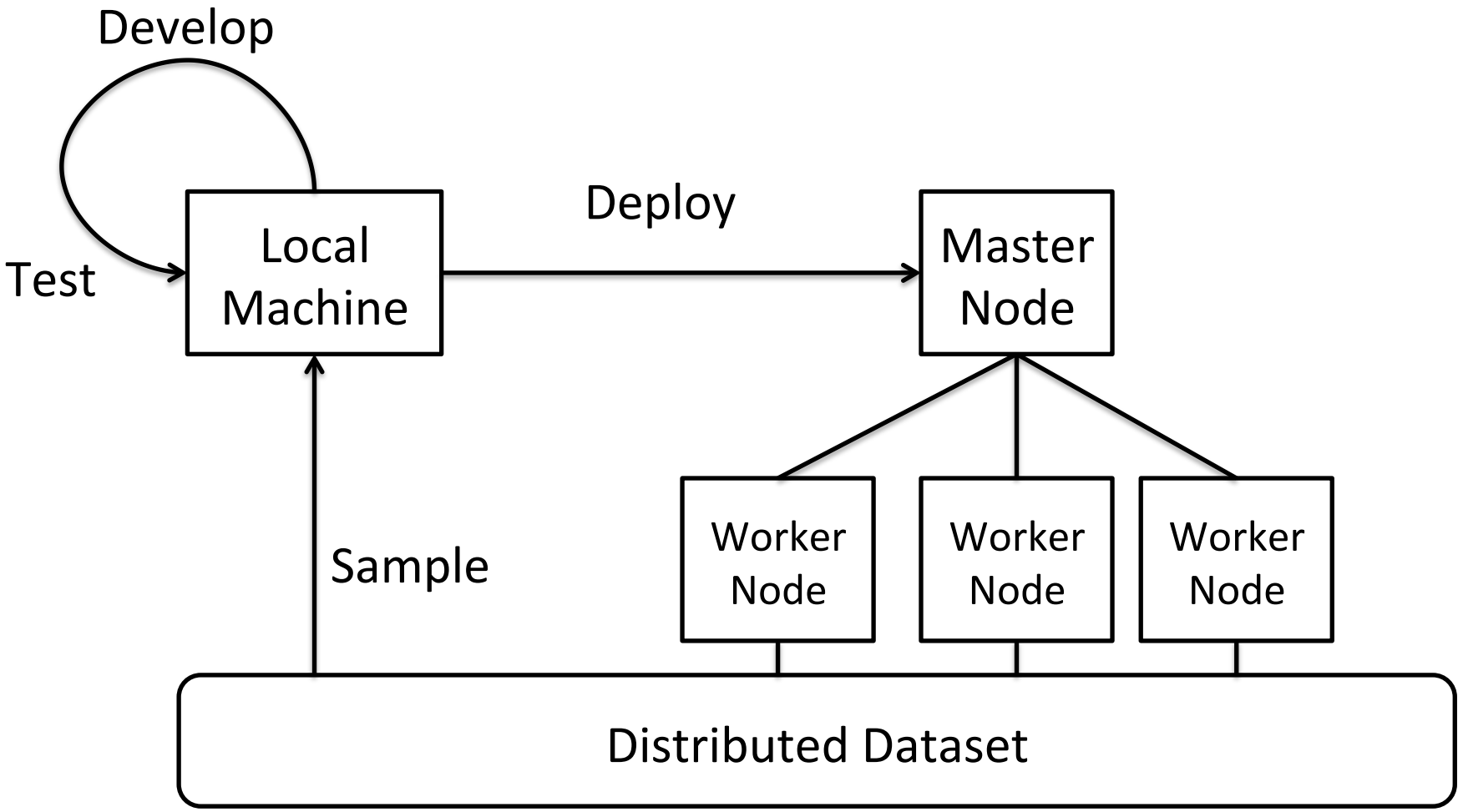
Example Use Cases

- Data Analysis / Analytics
- Extract Transform Load
- Yahoo
  - Recommendation engine for news (MLib)
  - Use existing BI tools to query advertising data in Hadoop (Spark SQL)
- ClearStory (data analytics)
  - Merge their internal data sources with external sources (Spark Core and Spark SQL)

    datanami.com/2014/03/06/apache_spark_3_real-world_use_cases
- Hearst Corporation (media)
  - Real time dashboard of article performance (Spark Streaming)

    aws.amazon.com/elasticmapreduce/details/spark/

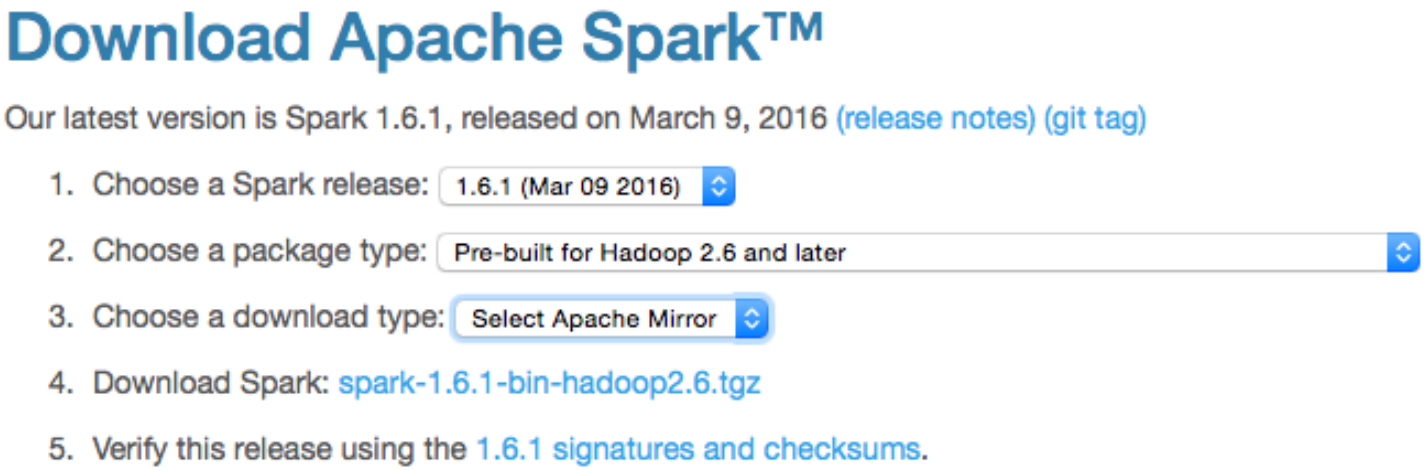
Example Data Analysis Workflow



Getting Started Locally

#Verify Java installation version >= 7
$ java -version

# Verify Python installation
$ python --version



spark.apache.org/downloads.html

# Unpack the download
$ tar –xf spark-1.6.1-bin-hadoop2.6.tgz

Running Locally

Using Python shell

$ cd spark-1.6.1-bin-hadoop2.6

$ bin/pyspark

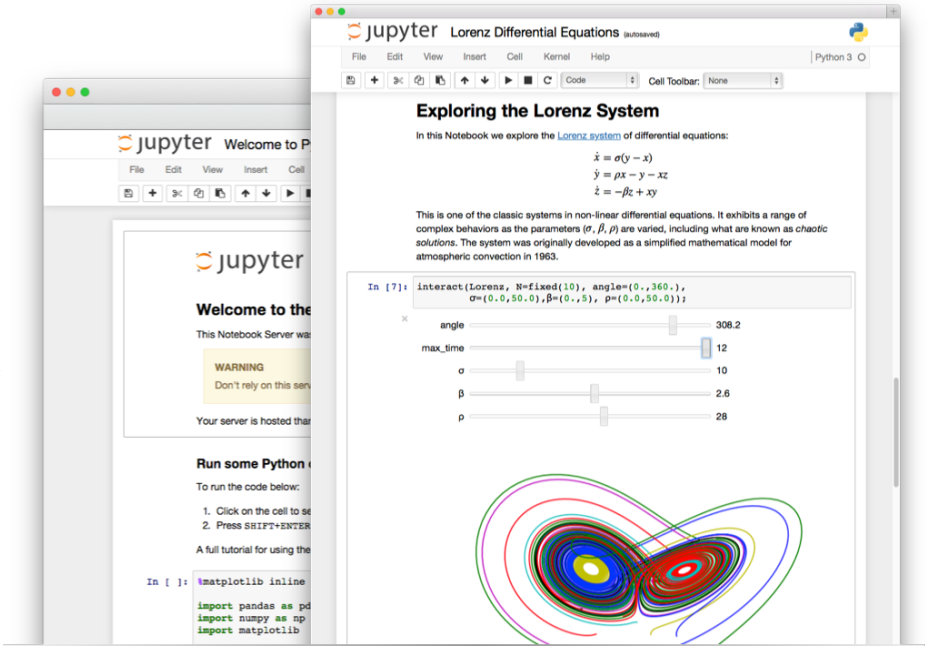
```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.4.0
      /_/

Using Python version 2.7.9 (default, Jan  7 2015 11:49:12)
SparkContext available as sc, HiveContext available as sqlContext.
>>>
```

Using IPython

$ IPYTHON=1 ./bin/pyspark

Using Jupyter Notebook

$ IPYTHON_OPTS="notebook" ./bin/pyspark



jupyter.org

Running Locally - Windows

Using Python shell

$ cd spark-1.6.1-bin-hadoop2.6

$ bin\pyspark

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.4.0
      /_/

Using Python version 2.7.9 (default, Jan  7 2015 11:49:12)
SparkContext available as sc, HiveContext available as sqlContext.
>>>
```

Using Ipython

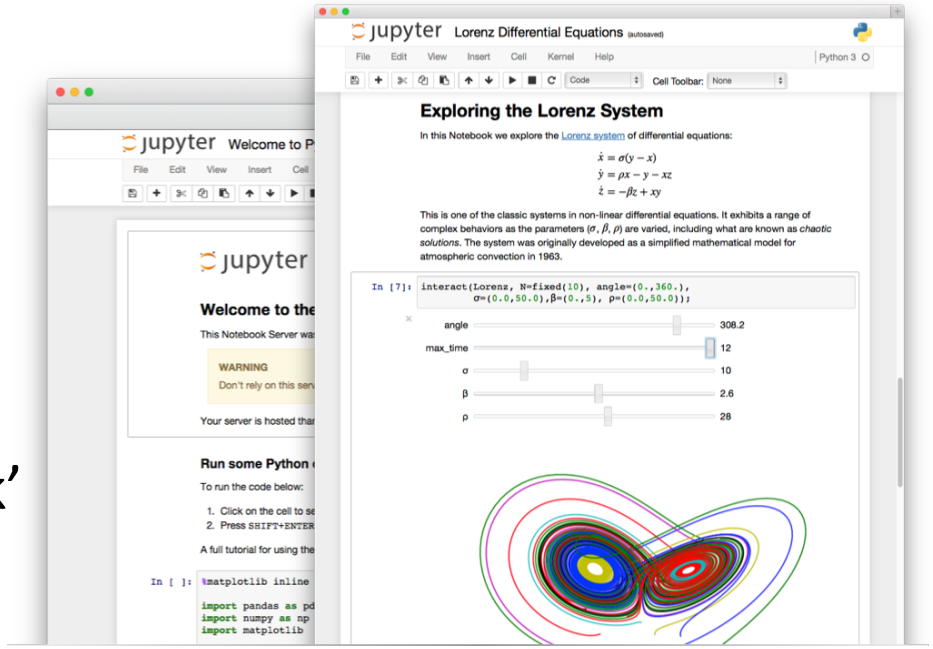
$ set IPYTHON=1

$ bin\pyspark

Using Jupyter Notebook

$ set IPYTHON_OPTS='notebook'

$ bin\pyspark



jupyter.org

Deployment

Deployment options

- Stand alone Spark (in-house)
- Amazon EC2 or EMR
- Microsoft Azure, Google Cloud, Databricks, …

Cluster Manager Options

- Spark Manager
- Apache Mesos
- Hadoop Yarn

Batch submit

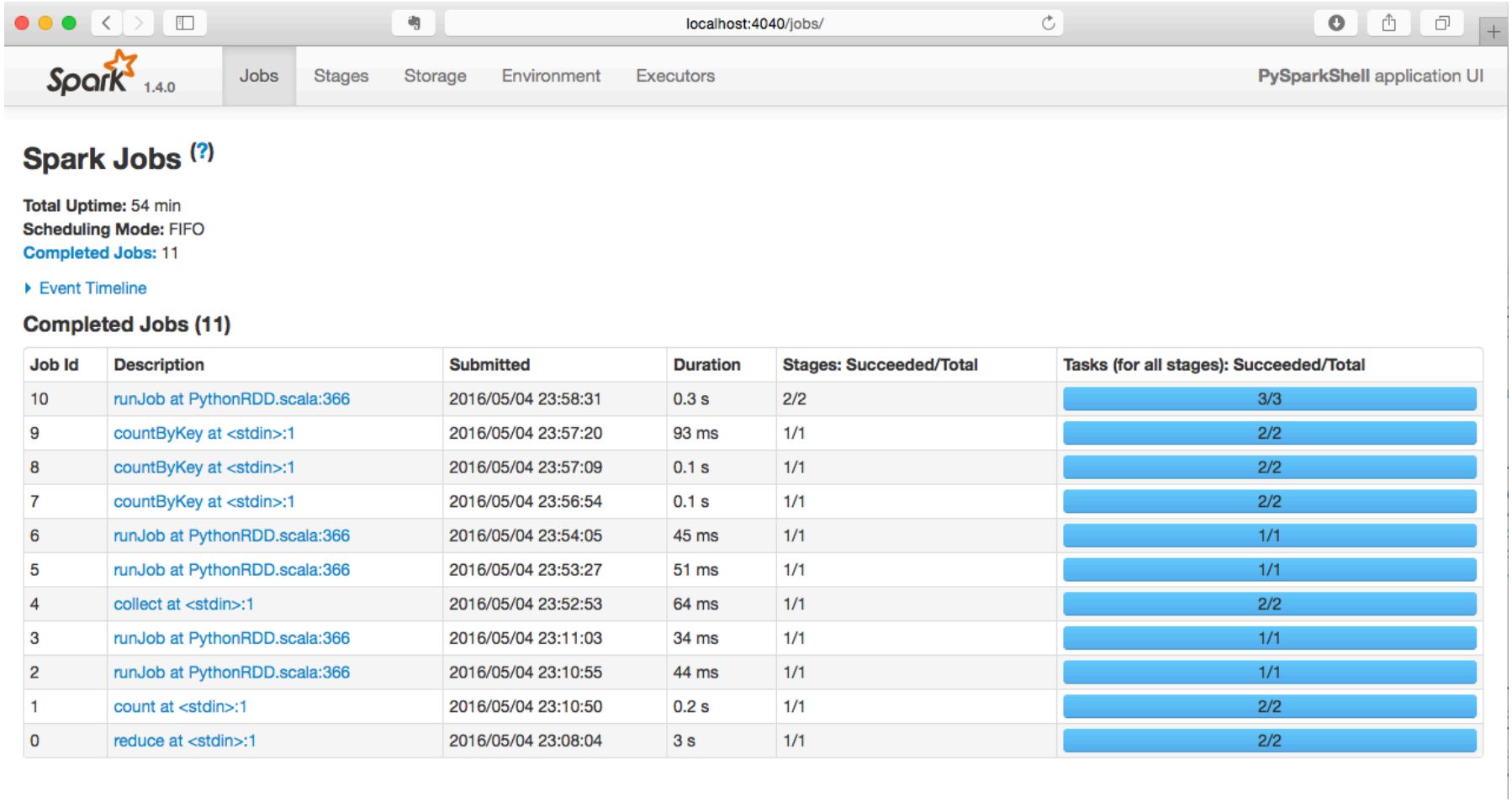
```
$ ./bin/spark-submit  --master spark://HOST:PORT
python_script.py
```

Or run interactively

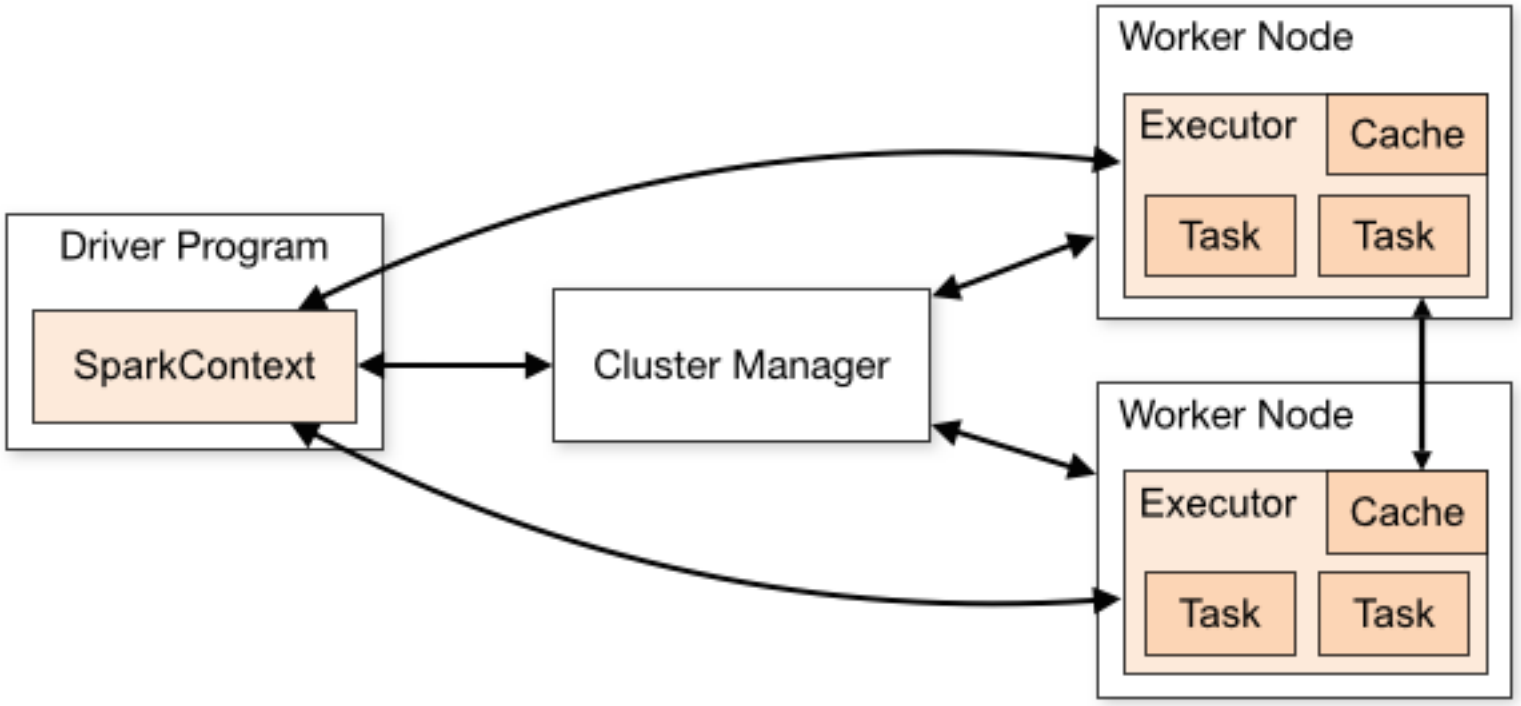
```
$ ./bin/pyspark --master spark://HOST:PORT
```

Built-in Monitoring

http://<driver-node>:4040 in browser or localhost:4040



Distributed Processing with Spark



spark.apache.org/docs/1.6.1/cluster-overview

Spark Context

- Main entry point for Spark functionality
- Represents the connection to a Spark cluster
- Create RDDs
- Automatically created in shell

```
# Initialize Spark in Python
from pyspark import SparkConf, SparkContext

conf = SparkConf().setMaster("local").setAppName("My App")
sc = SparkContext(conf = conf)
```

Resilient Distributed Dataset (RDD)

- Spark's abstraction for distributed files
  - Allows for parallel operations
  - Can persist in memory
  - Lazy evaluation
- Created by:
  - Parallelizing an existing Python collection (list)
  - Referencing external data
  - Transforming an existing RDD

Create RDD

```
# Distribute a local Python collection to form an RDD
>>> sc.parallelize([1, 2, 3, 4, 5])

# Read a text file from HDFS, local or URI
>>> sc.textFile(filename)
```

RDD Operations

Transformations

create a new dataset from an existing

>>> map(func)

>>> filter(func)

>>> distinct()

…

Actions

return a value to the driver

>>> reduce(func)

>>> take(n)

>>> collect()

…

RDD example

```
>>> sc
<pyspark.context.SparkContext object at 0x1019cc290>

# Create an RDD from a python list
>>> rdd = sc.parallelize([1,2,3,4,5])

# Apply a map transformation
>>> rdd1 = rdd.map(lambda x: x + 1)

# Collect all of the rdd items in driver
>>> rdd1.collect()
[2, 3, 4, 5, 6]

# Apply reduce action
>>> rdd1.reduce(lambda x, y: x + y)
20
```

Simplified Example

```
>>> rdd = sc.parallelize([1,2,3,4,5])
>>>(rdd
     .map(lambda x: x + 1)
     .reduce(lambda x, y: x +y))
20
```

Spark SQL

- Structured Data
- SQL or Dataframe Operations
- Construct from
  - structured data files (Parquet, JSON)
  - external databases
  - existing RDDs
- Connect to Hive and JDBC / ODBC
- Efficient and Fast

Dataframe Example

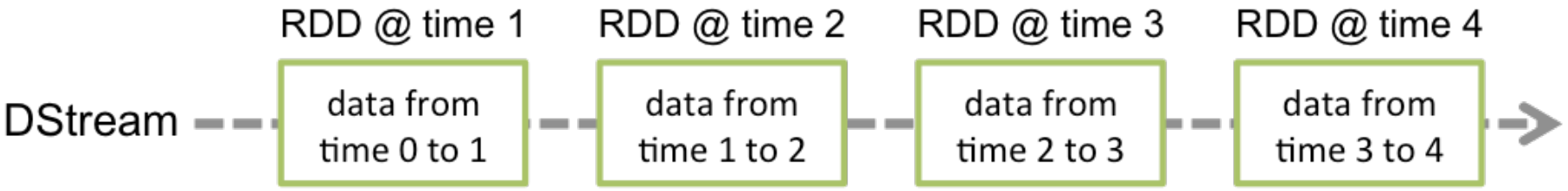
```
sqlContext = SQLContext(sc)
# Create the DataFrame
df = sqlContext.read.json("people.json")

df.show() # Show the contents of the DataFrame
+---+-------+
|age|   name|
+---+-------+
| 10|Michael|
| 30|   Andy|
| 19| Justin|
+---+-------+

df.describe().show() # ~ summary(df)
+-------+------------------+
|summary|               age|
+-------+------------------+
|  count|                 3|
|   mean|19.666666666666668|
| stddev|10.016652800877813|
|    min|                10|
|    max|                30|
+-------+------------------+
```

Spark Machine Learning

- spark.mllib and spark.ml
- Algorithms:
  - Linear regression (L1, L2, and elastic-net)
  - Logistic regression
  - Support vector machine (SVM)
  - Classification and regression tree
  - Random forest and gradient-boosted trees
  - Alternating least squares (ALS)
  - Clustering (k-means, bisecting k-means, Gaussian mixtures (GMM), and power iteration clustering)
  - Singular value decomposition (SVD) and QR decomposition
  - Principal component analysis (PCA)
  - Naive Bayes

Spark Streaming

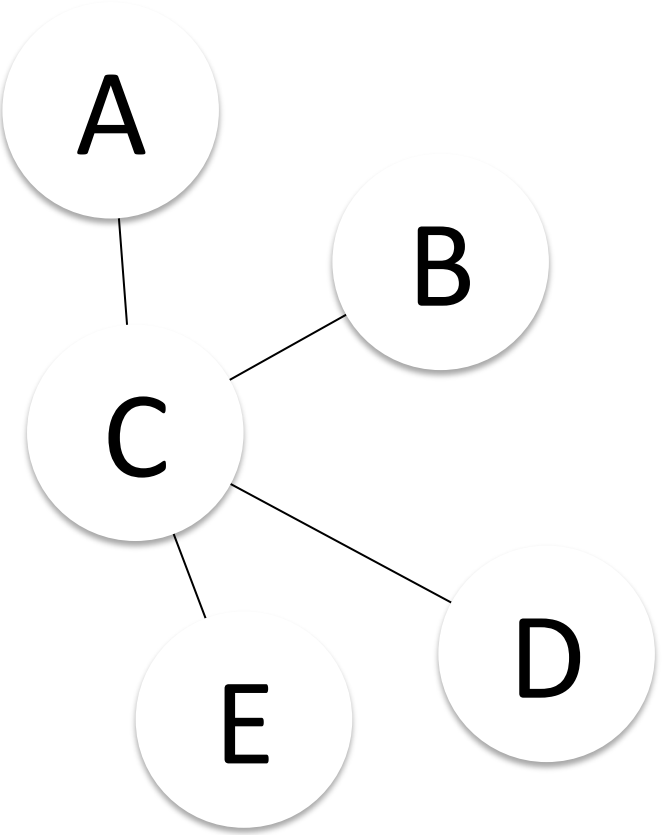
- Input data from live stream
- Output to file system, database, or dashboard
- Discretized Stream (DStream)
  - Input stream of data mapped to series of RDDs
  - Perform operations on underlying RDDs



spark.apache.org/docs/latest/streaming-programming-guide

Spark GraphX

- Integrate graphs with Spark workflow
- Algorithms
  - PageRank
  - Connected components
  - Label propagation
  - SVD++
  - Strongly connected components
  - Triangle count



Jupyter Notebook Example

Questions?