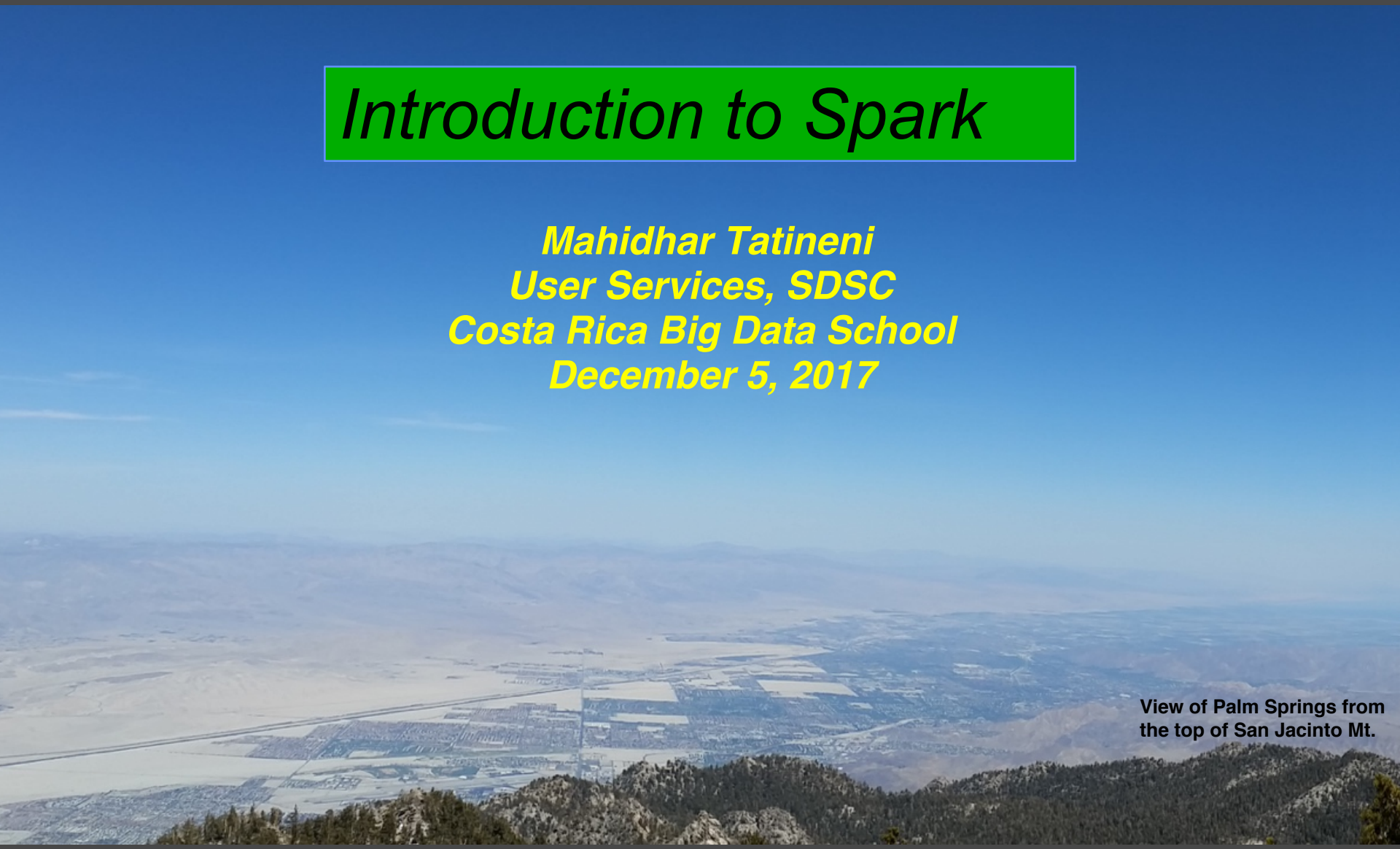
Introduction to Spark

Mahidhar Tatineni
User Services, SDSC
Costa Rica Big Data School
December 5, 2017

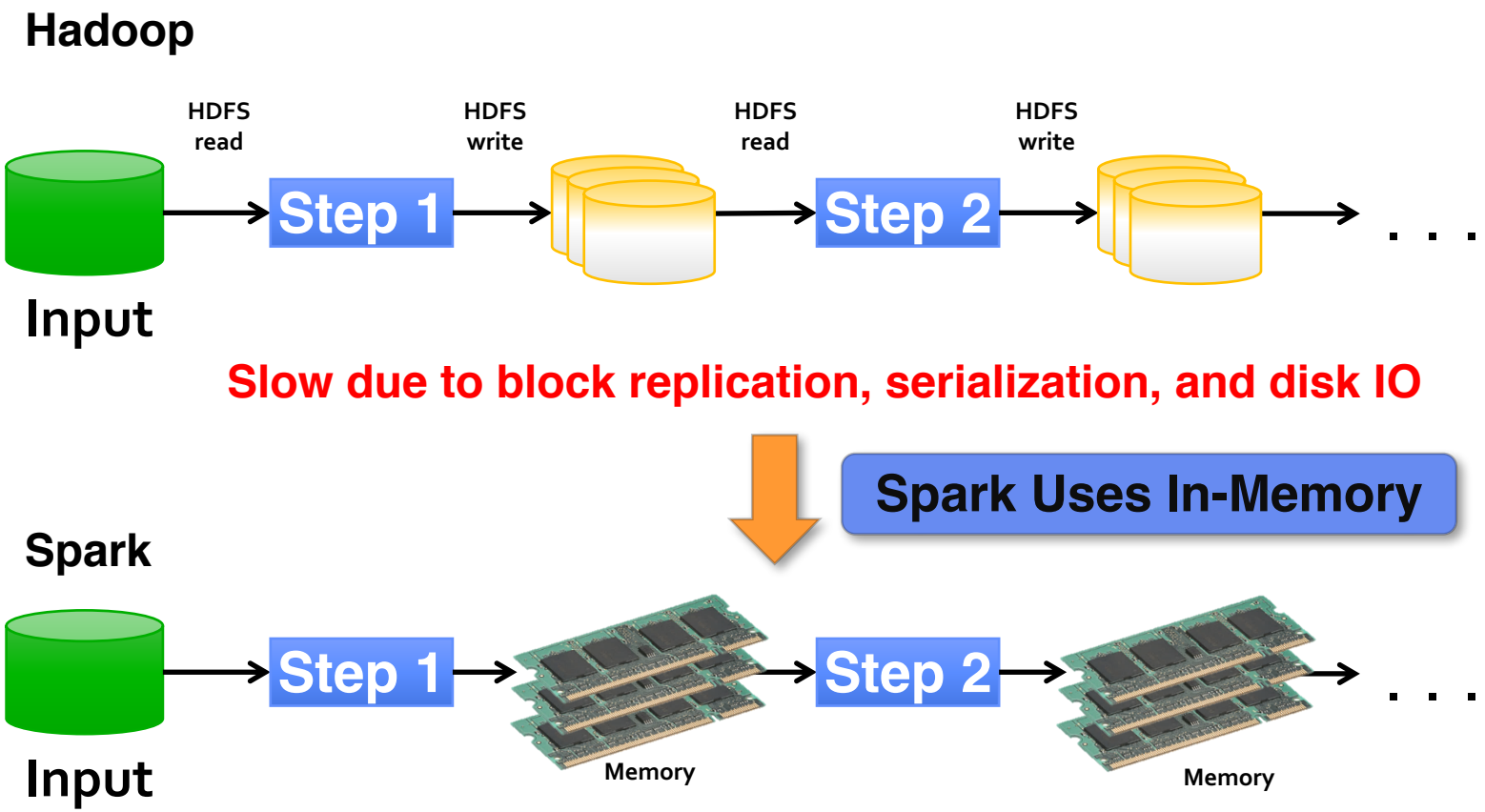


View of Palm Springs from the top of San Jacinto Mt.

Overview

- **A distributed data analysis and computing framework**
- **Designed for performance**
  - Uses RAM as much as possible
  - Lazy execution - can be smart about the computations based on the knowledge of steps needed for result
  - Resilient to node failures
- **Easy to use**
  - Python, Scala, Java interfaces
  - Interactive shells
  - Many distributed primitives
- **Many paradigms available**
  - SparkSQL
  - Streaming
  - Mlib
  - Graphx

Efficiency of using RAM for Data Sharing



Keeping required data in memory can be 10-100× faster than going to disk/network + allows for optimizations with RDD approach

Figure credit: Xiaoyi Lu, "How to Accelerate Your Big Data Applications with Hadoop and Spark?", PEARC17

Spark RDD Approach

- Create or Load RDD - from data in storage (HDFS, HBase, JSON, S3 etc)
- Can also create by transforming another RDD
- Transform RDDs - filtering, unique entries, sampling, union, intersection etc
- Spark framework will not compute these immediately - only when results are needed. This allows for optimization of the whole process (input to first result).
- Only perform actions to get results.
- Automatically rebuilt on failure - rebuilt if a partition is lost.

Benefits of RDD Model

- **Consistency (helped by immutability)**
- **Inexpensive fault tolerance (log lineage rather than replicating/checkpointing data)**
- **Locality-aware scheduling of tasks on partitions**
- **Model applicable to a broad variety of applications**
- **Easy Programming**
  - Java, R, Python, Scala interfaces
- **High-Performance**
  - Memory speeds are much higher than disk speeds!
- **Scalable**

Example RDD Operations

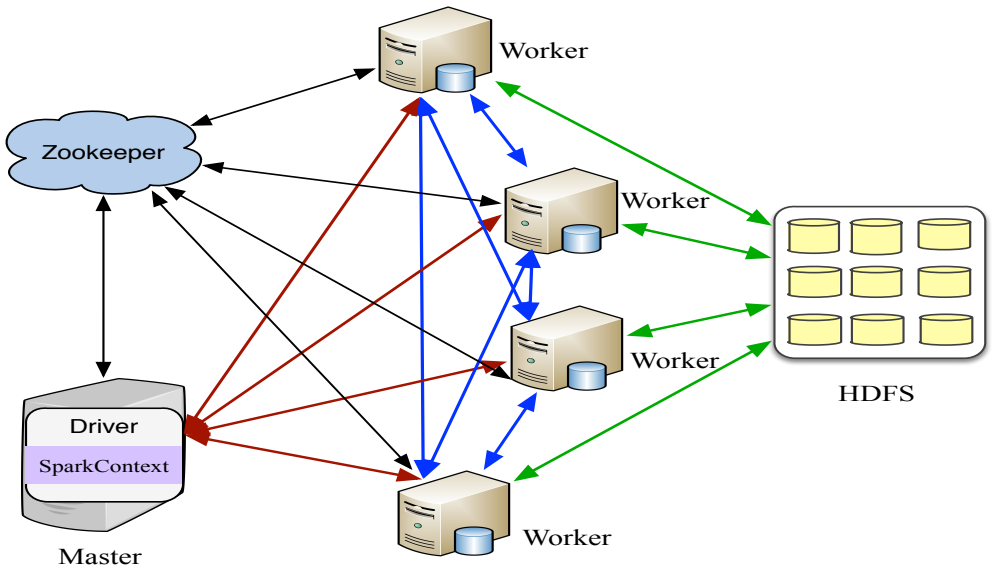
Transformations (define a new RDD)	Actions (return a result to driver)
map	reduce
filter	collect
sample	count
union	first
groupByKey	Take
reduceByKey	countByKey
sortByKey	saveAsTextFile
join	saveAsSequenceFile
…	…

More Information:

- https://spark.apache.org/docs/latest/programming-guide.html#transformations
- https://spark.apache.org/docs/latest/programming-guide.html#actions

Spark Architecture Overview

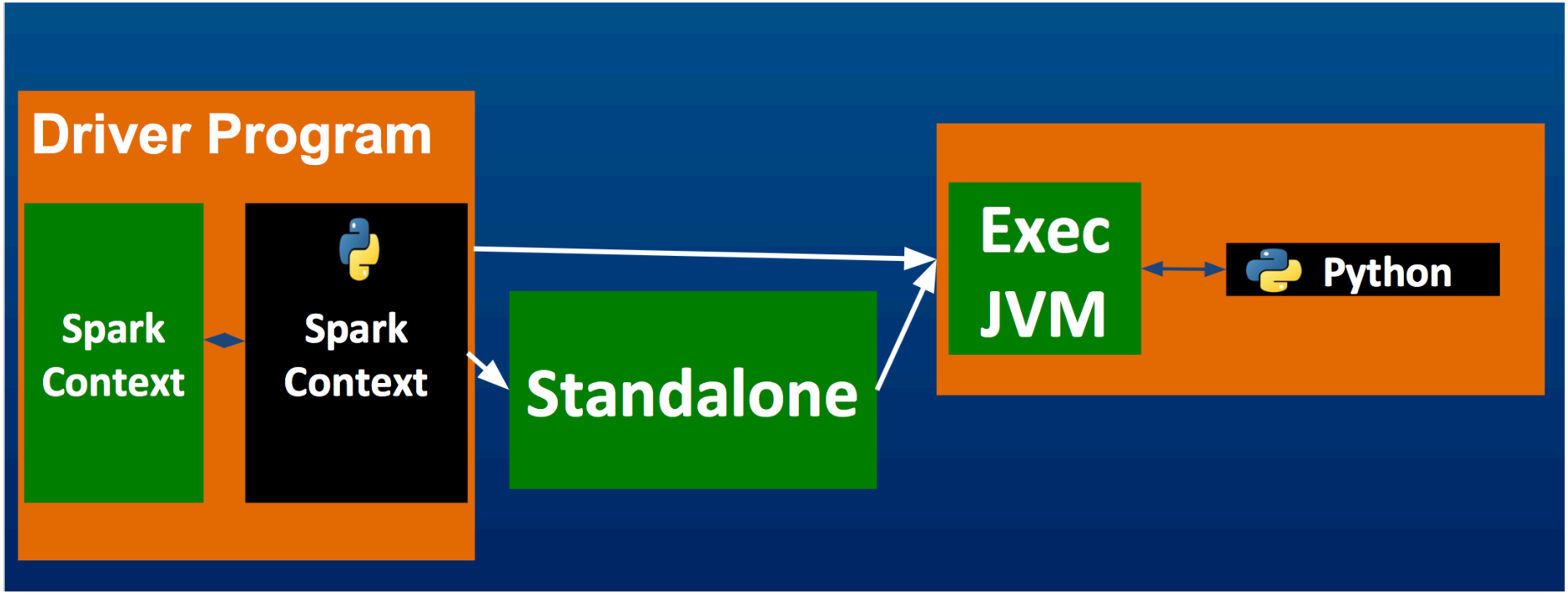
- **An in-memory data-processing framework**
  - Iterative machine learning jobs
  - Interactive data analytics
  - Scala based Implementation
  - Standalone, YARN, Mesos
- **Scalable and communication intensive**
  - Wide dependencies between Resilient Distributed Datasets (RDDs)
  - MapReduce-like shuffle operations to repartition RDDs
  - Sockets based communication



http://spark.apache.org

Slide credit: Xiaoyi Lu, "How to Accelerate Your Big Data Applications with Hadoop and Spark?", PEARC17

Standalone Mode

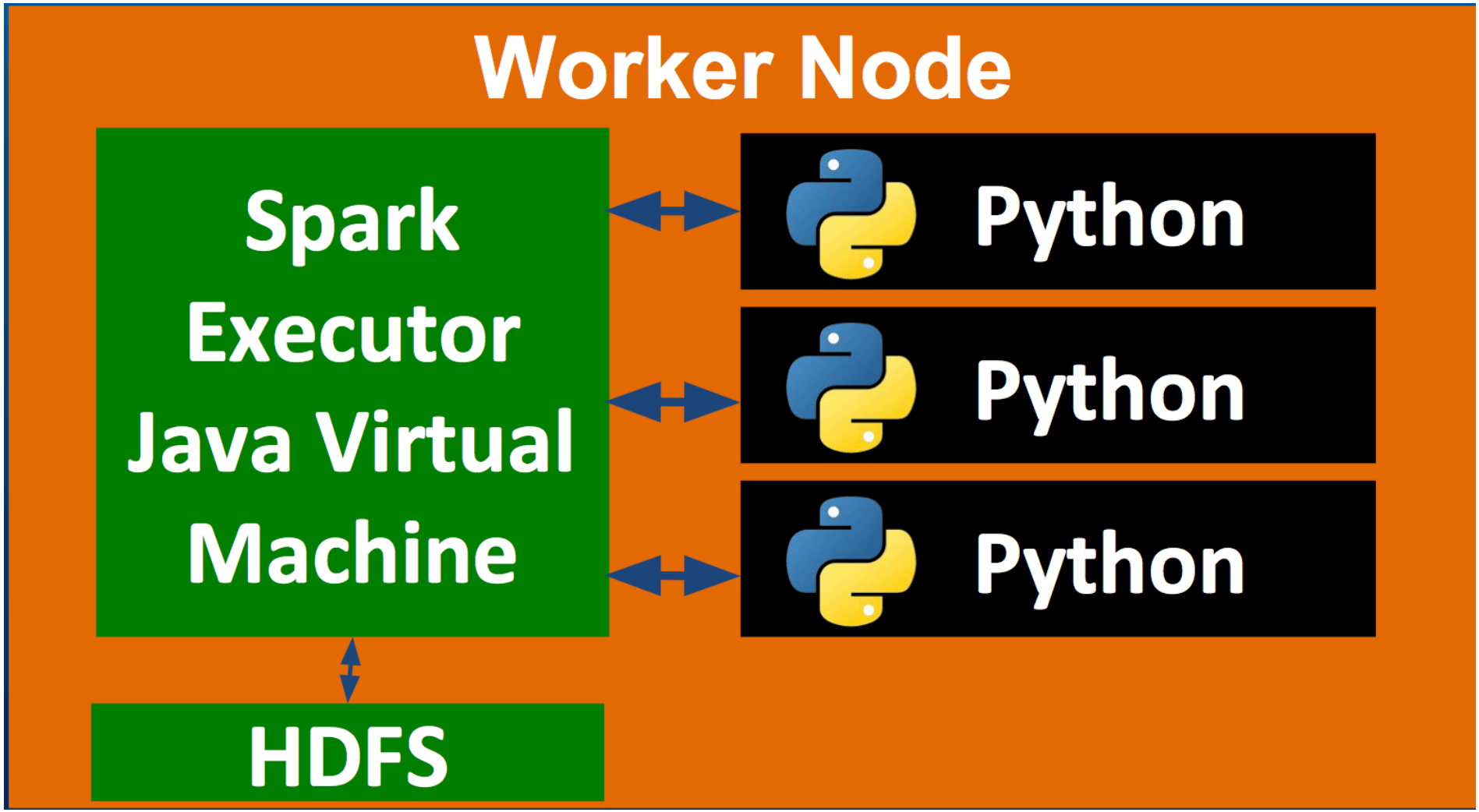


Spark Connection Objects

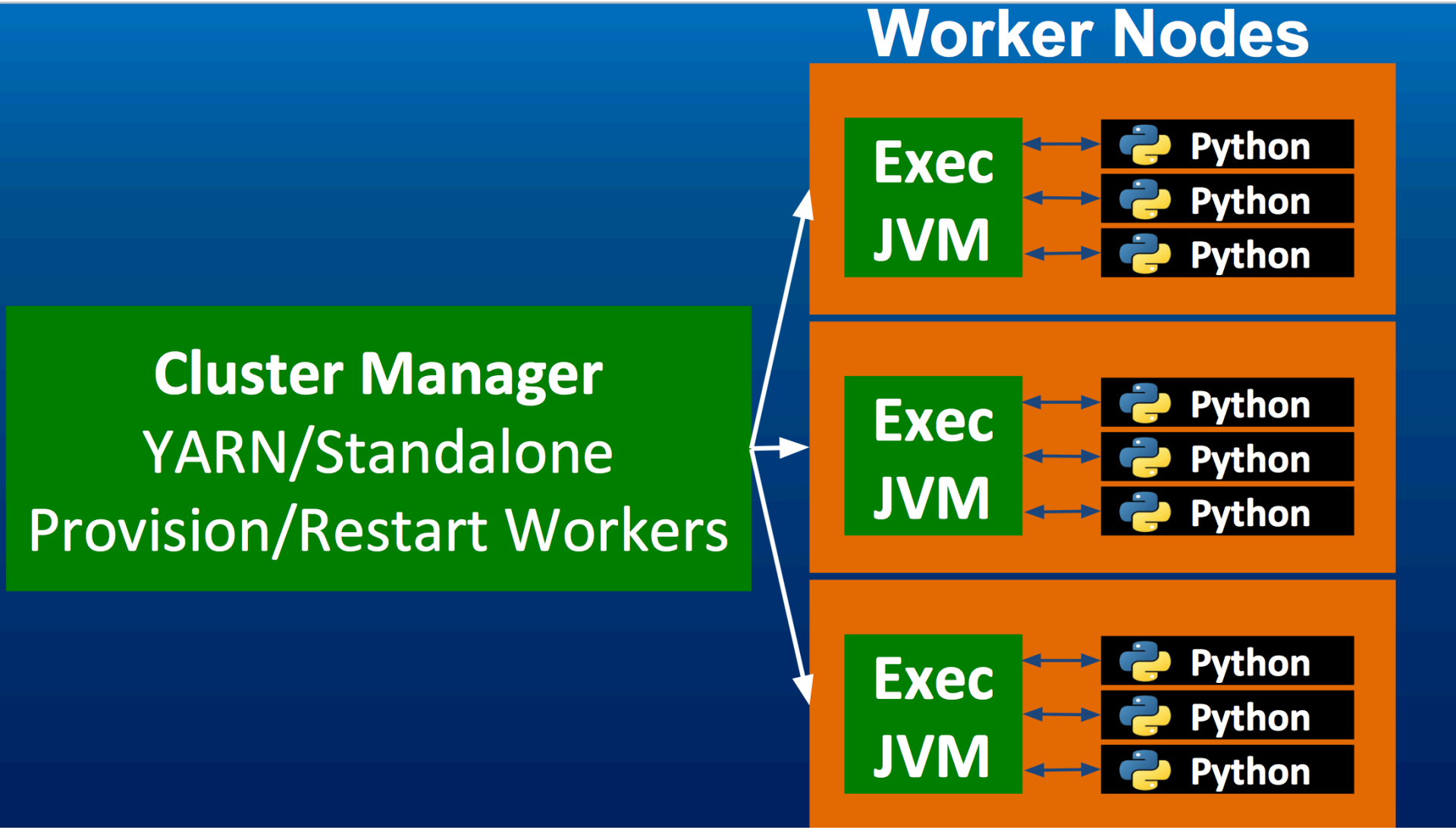
- **Prior to Spark 2.0.0**
  - SparkContext - connection to Spark execution environment
  - SQLContext - connection to SparkSQL
  - HiveContext

- **After Spark 2.0.0 - in addition to above:**
  - Datasets/DataFrames introduced, SparkSession is the entry point to a Spark execution environment
  - org.apache.spark.sql.SparkSession, pyspark.sql.sparkSession

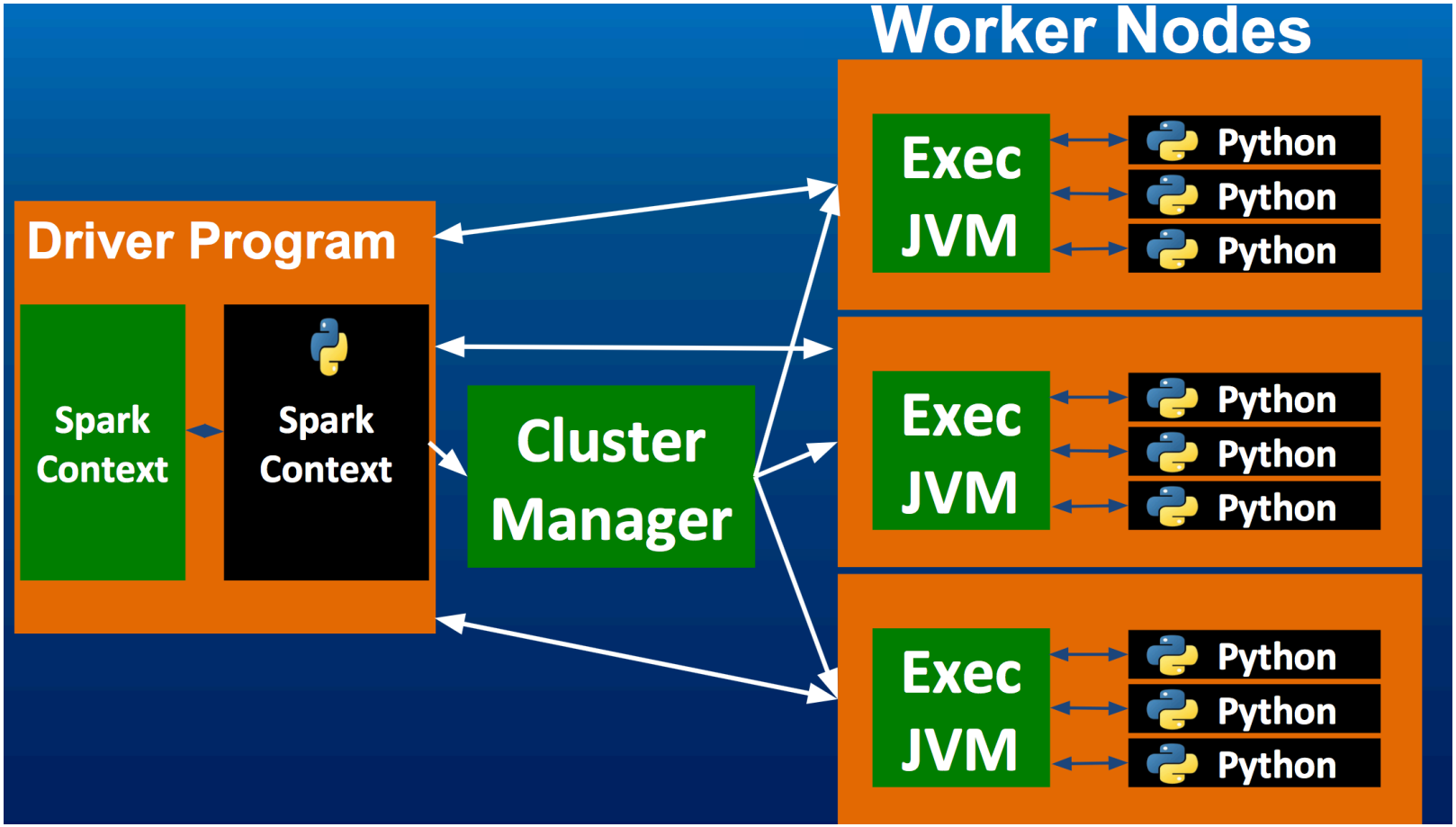
Spark Worker Node Setup



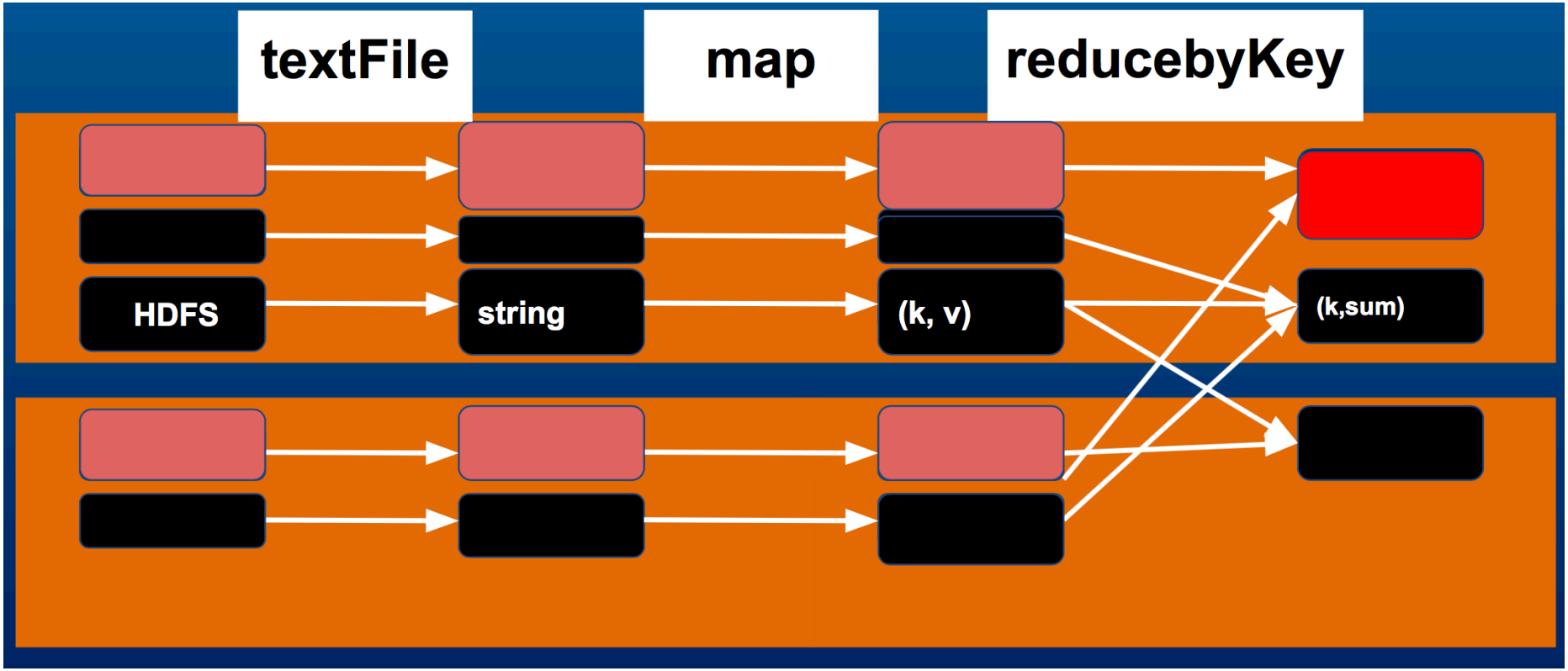
Spark Cluster Setup



Spark Driver Setup



Example Spark Graph (details in next tutorial)



Spark Optimizations

- **Persistence**
  - Lazy evaluation => RDD dependencies are known
  - Data that might be used multiple times can be persisted
  - Memory only, memory and disk, disk only
- **Pair RDDs**
  - Spark operations for RDDs with key/value pairs.
  - E.g. reduceByKey, groupByKey(), keys() etc
- **Partitioning**
  - Custom partitioners with persist()

Hands On

- Clone the repository:

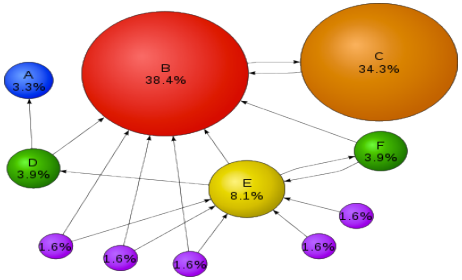
https://github.com/mahidhar/Spark_handson

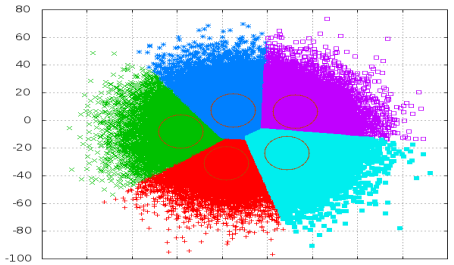
- Command:

git clone https://github.com/mahidhar/Spark_handson

Spark Benchmarks

- **GroupByTest/SortByTest**
  - A commonly used micro-benchmark that uses the groupByKey (sortByKey) operation, which groups the values for each key (sorts the pairs alphabetically) in the RDD into a single sequence and results in data shuffle
- **PageRank**
  - Counts the number and quality of links to a page to determine a rough estimate of how important the website is. A typical search engine benchmark
- **K-Means**
  - Partitions the input objects to k clusters by calculating the nearest mean cluster of each object belongs to. A typical social network benchmark
- **Sort/Grep/WordCount**
  - Spark-based implementations for micro-benchmarks





Slide credit: Xiaoyi Lu, "How to Accelerate Your Big Data Applications with Hadoop and Spark?", PEARC17

Spark Performance

	Hadoop MR Record	Spark Record	Spark 1 PB
Data Size	102.5 TB	100 TB	1000 TB
Elapsed Time	72 mins	23 mins	234 mins
# Nodes	2100	206	190
# Cores	50400 physical	6592 virtualized	6080 virtualized
Cluster disk throughput	3150 GB/s (est.)	618 GB/s	570 GB/s
Sort Benchmark Daytona Rules	Yes	Yes	No
Network	dedicated data center, 10Gbps	virtualized (EC2) 10Gbps network	virtualized (EC2) 10Gbps network
Sort rate	**1.42 TB/min**	**4.27 TB/min**	**4.27 TB/min**
Sort rate/node	**0.67 GB/min**	**20.7 GB/min**	**22.5 GB/min**

Reference: https://databricks.com/blog/2014/10/10/spark-petabyte-sort.html

Thank You!

Questions