INTRODUCTION TO SPARK WITH SCALA

Introduction to Spark

Agenda

- Introduction to Spark
- Core Concepts and Architecture

Introduction to Apache Spark

"Big data" is moving beyond one-pass batch jobs to low latency applications that need data sharing

- Matei Zaharai

Introduction to Apache Spark

The leading candidate for “successor to MapReduce” is Apache Spark

- - Mike Olson
- Cloudera Chief Strategy Officer
- 12/30/2013

http://vision.cloudera.com/mapreduce-spark/

Introduction to Apache Spark

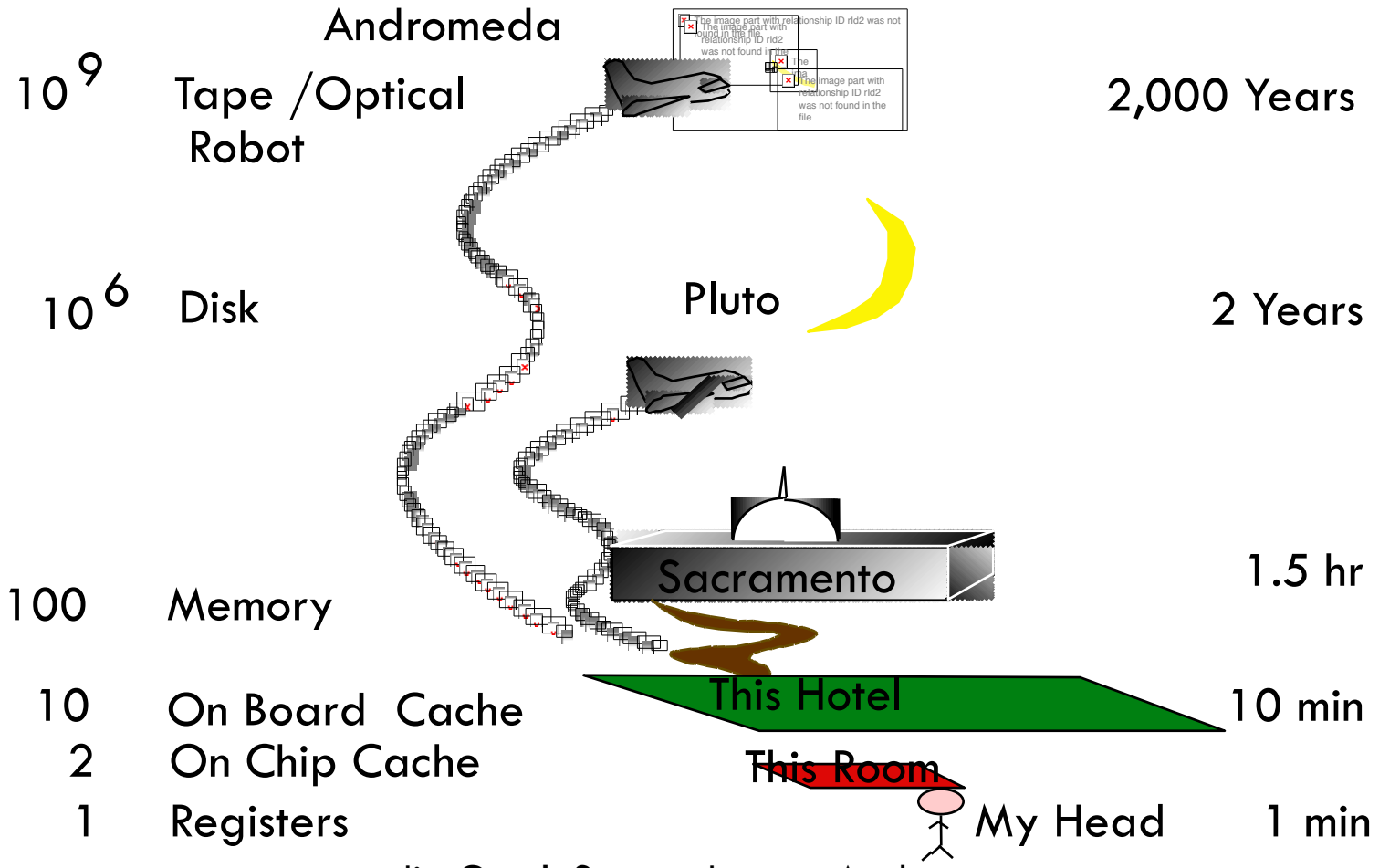
In-memory compute and machine-learning logic are the key to unlocking the value of Big Data

- - Christopher Nguyen
  - CEO Adatao

http://blog.adatao.com/adatao-why-and-how-we-decided-on-apache-spark/

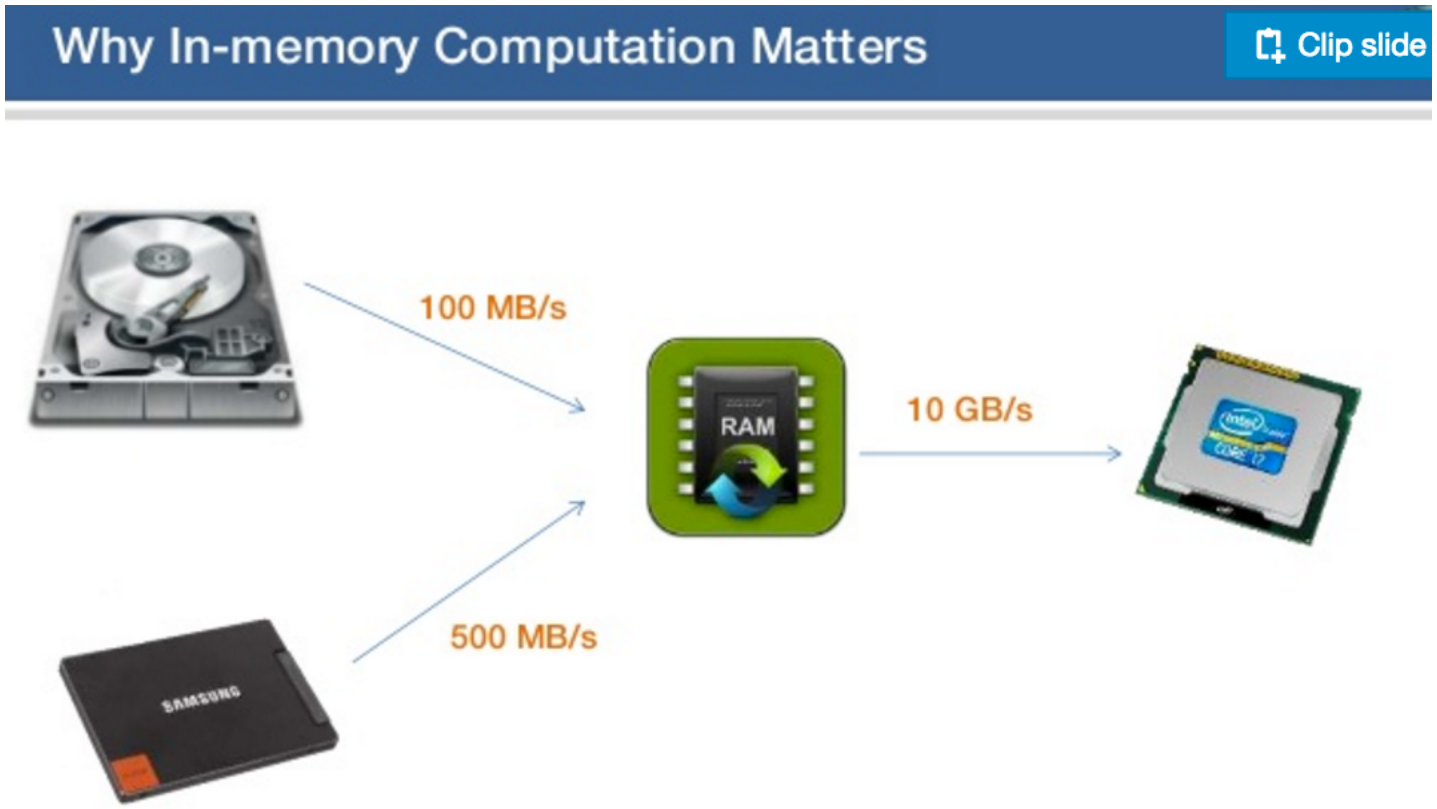
Introduction to Apache Spark

How Far Away is the Data?

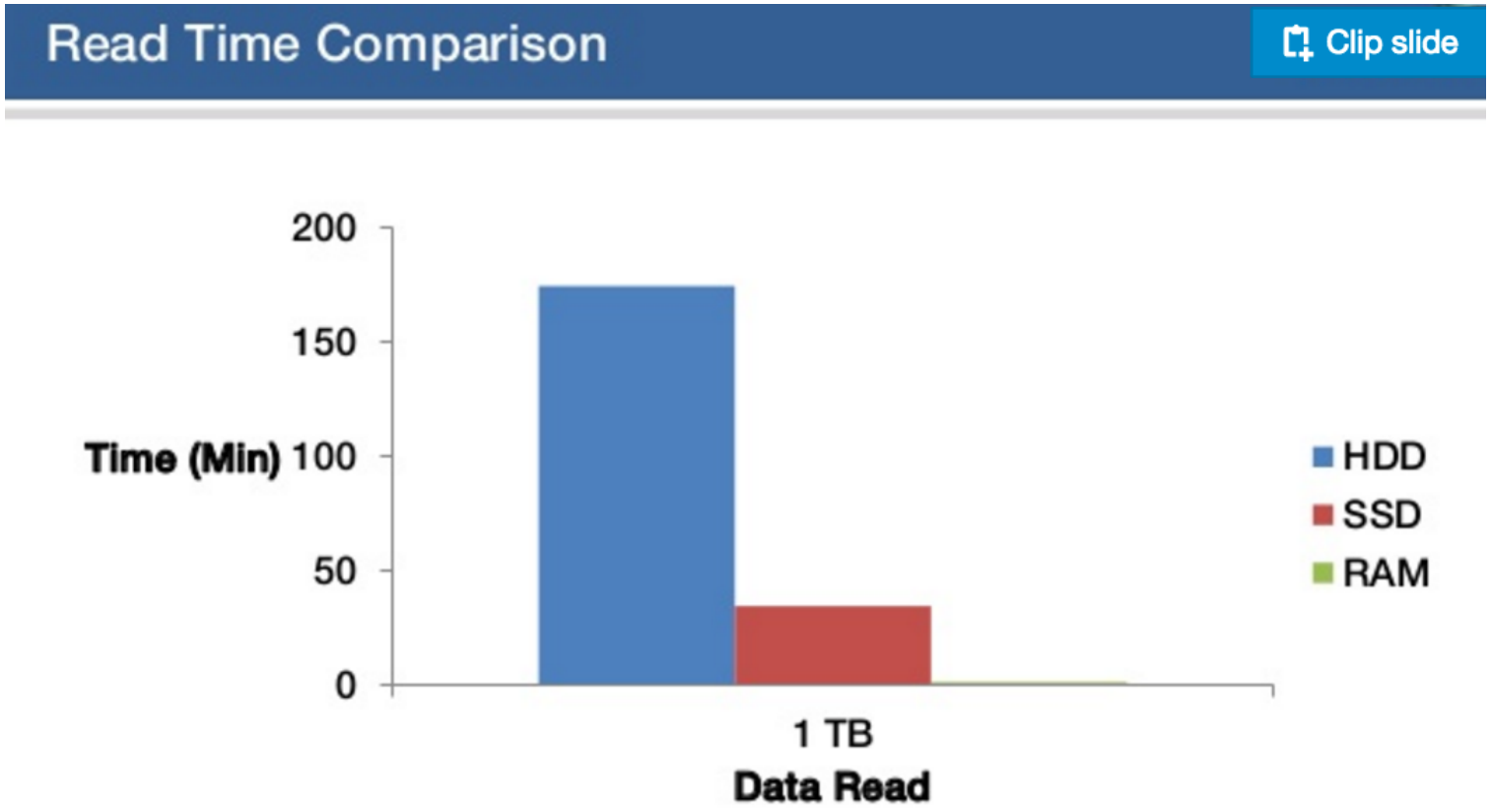


Jim Gray's Storage Latency Analogy

Introduction to Apache Spark



Introduction to Apache Spark



Introduction to Apache Spark

Batch Processing	Interactive, iterative, streaming, graph	Unified Engine
	Dremel Impala Drill Girah Impala Presto Storm Samza Tez	
2004-2013	2007-2015	2014-?

Introduction to Apache Spark

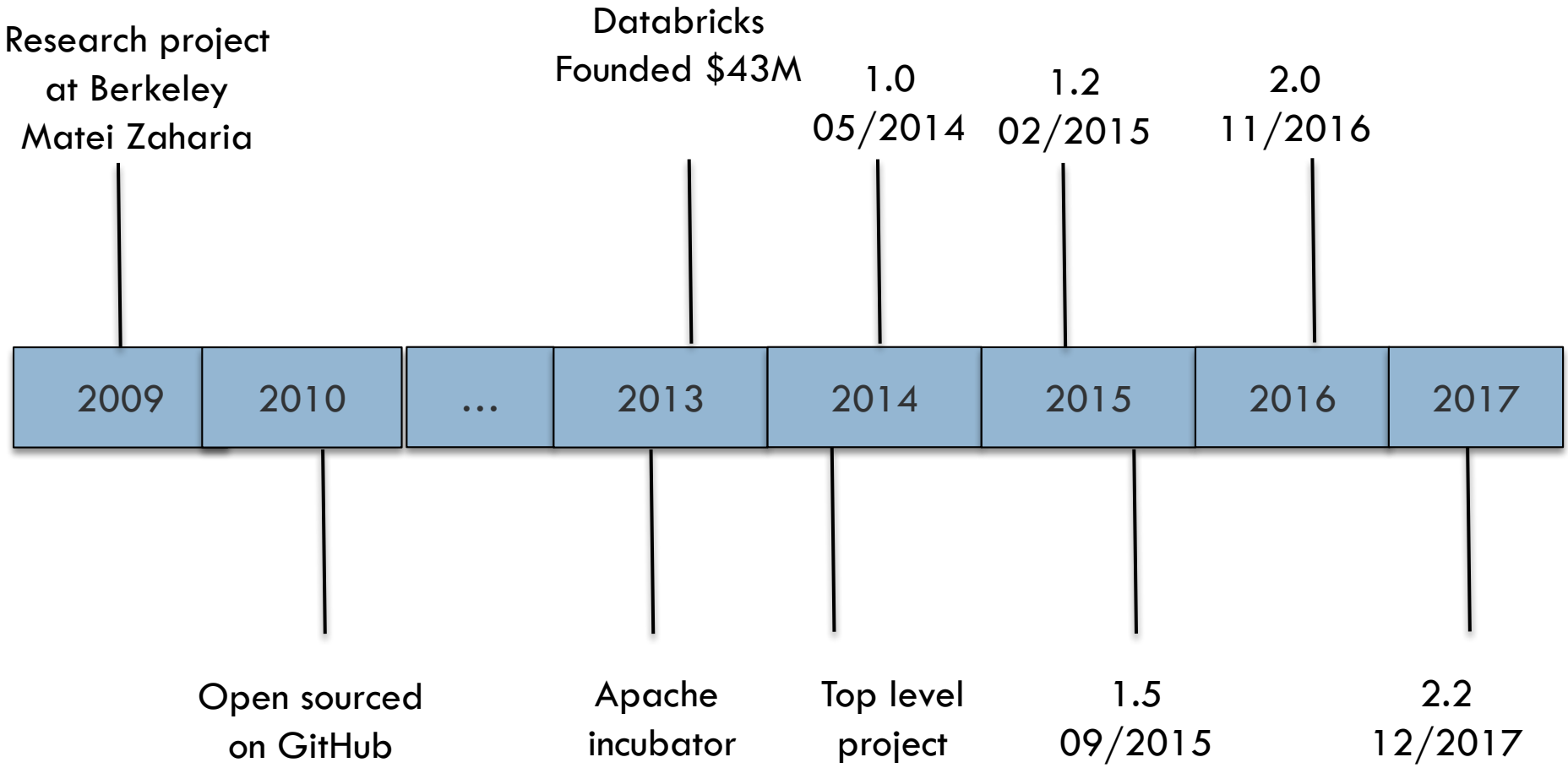
In recent days…… 2010

Spark: Cluster Computing with Working Sets
people.csail.mit.edu/matei/papers/2010/hotcloud_spark.pdf

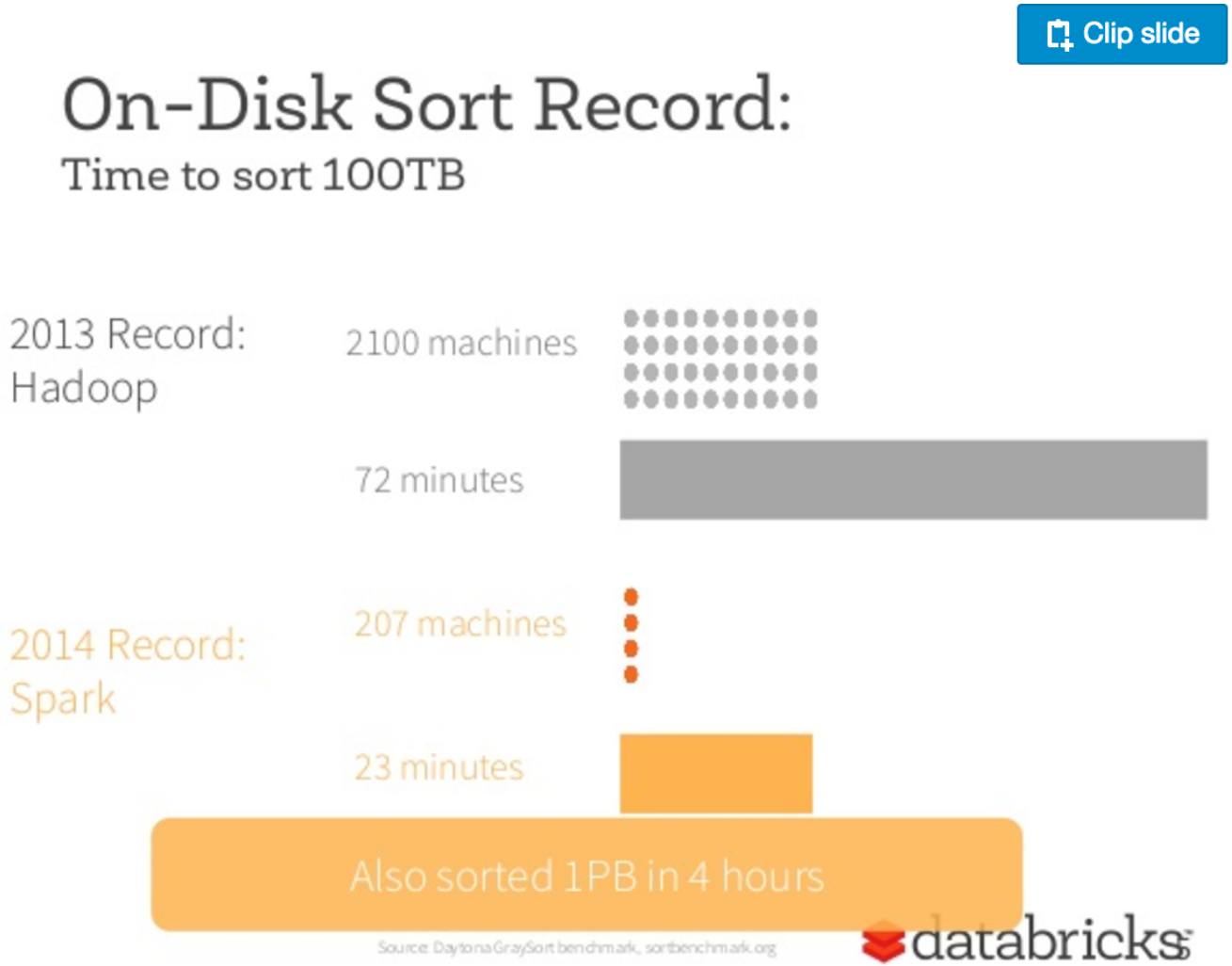
Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing
usenix.org/system/files/conference/nsdi12/nsdi12-final138.pdf

Lightning-fast cluster computing

Introduction to Apache Spark



Introduction to Apache Spark



Introduction to Apache Spark

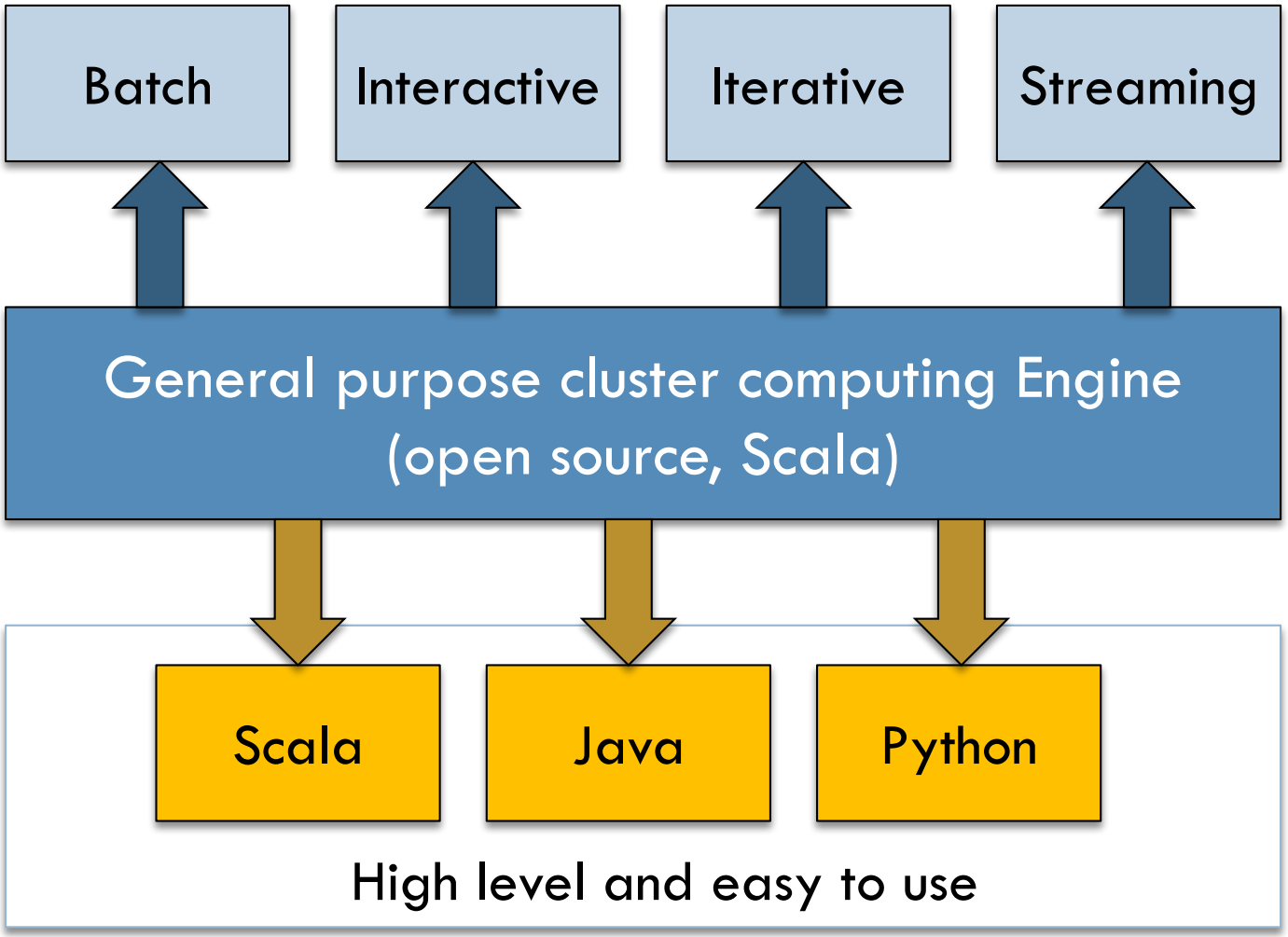
2014 Daytona GraySort Contest

	Hadoop MR Record	Spark Record
Data Size	100TB	100TB
Elapsed Time	72 mins	**23 mins**
# Nodes	2100	**206**
# Cores	50400	**6592**
Sort Rate	1.42 TB/min	**4.27 TB/min**
Sort Rate/node	0.67 GB/min	**20.7 GB/min**

3X faster and 10X less resources – Not using in-memory cache

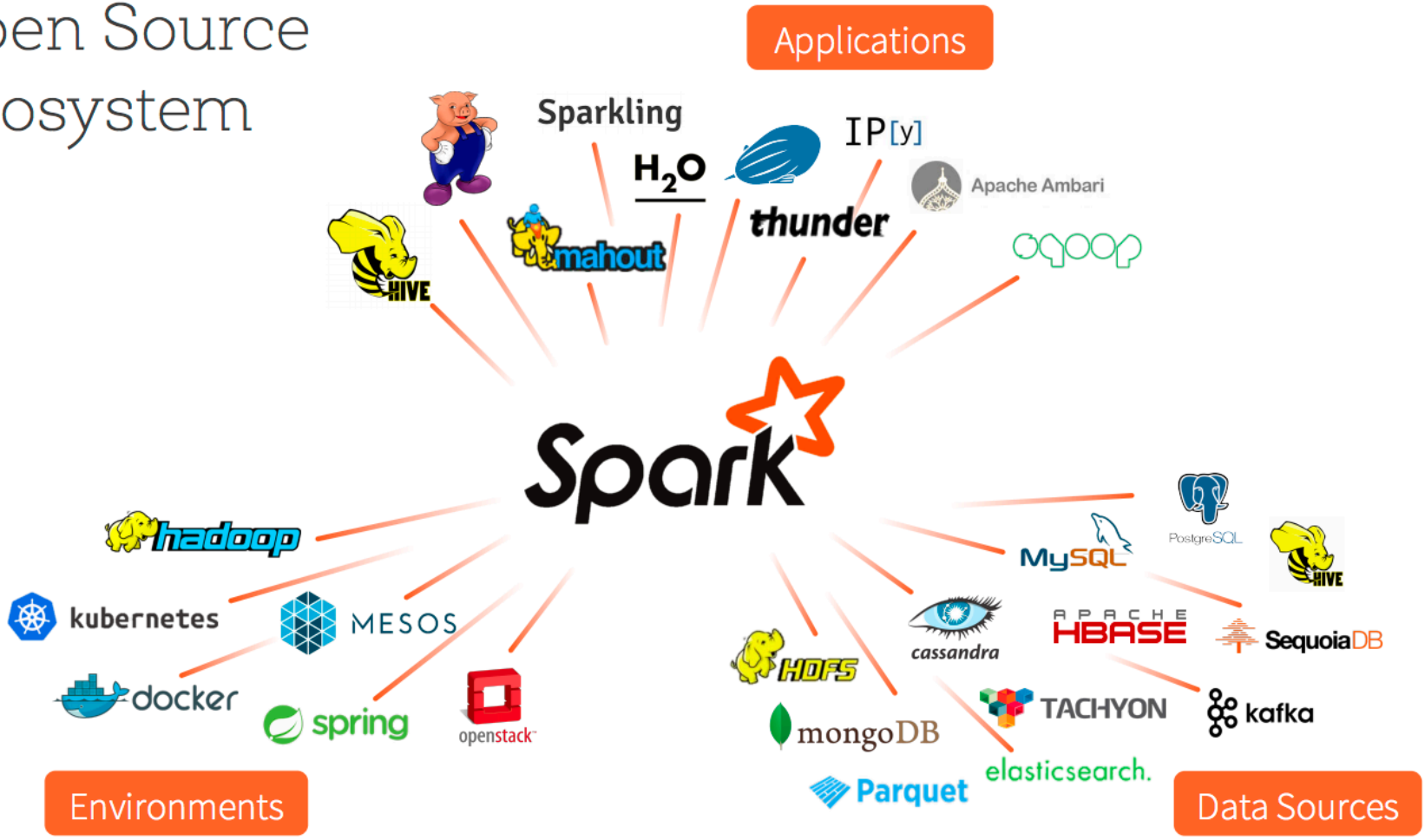
https://databricks.com/blog/2014/10/10/spark-petabyte-sort.html

Introduction to Apache Spark



Introduction to Apache Spark

Open Source Ecosystem



http://bit.ly/1VVXMh0

Introduction to Apache Spark

Adoptions

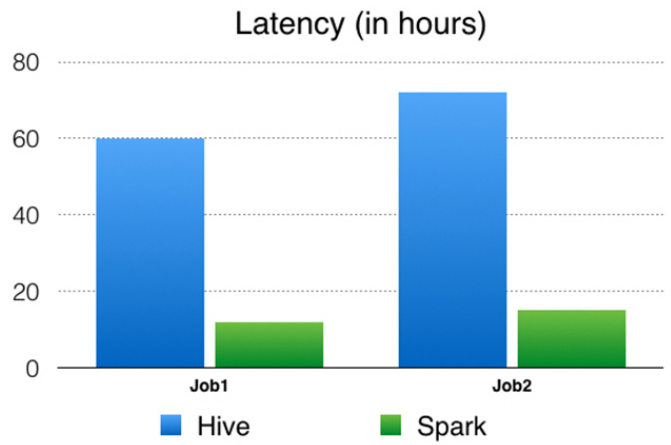
http://bit.ly/1VVXMh0

Introduction to Apache Spark

Notable Use Cases

Introduction to Apache Spark

- FB
  - Large-scale language model training
    - Converted Hive pipeline to Spark
    - 2.6x faster
  - A 60 TB+ production use case
    - Feature preparation for entity ranking



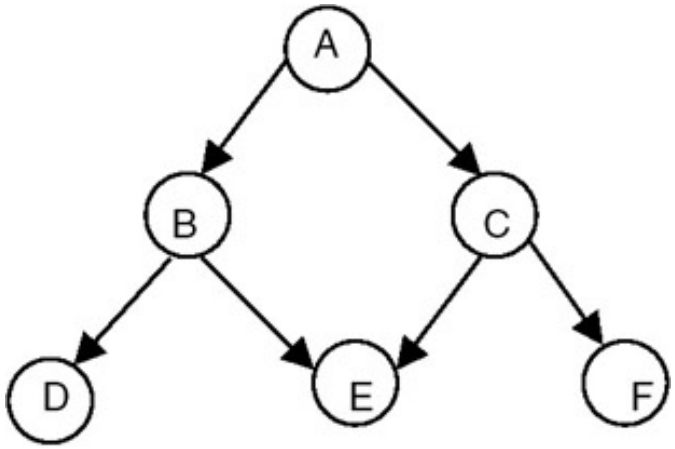
Introduction to Apache Spark

- Salesforce
  - Power insights on Sales Emails
    - Recommending hot sales lead to follow up
    - Recommending next actions based on email interactions
  - Power Einstein platform

Introduction to Apache Spark

Resilient Distributed Dataset

In-memory computing primitives

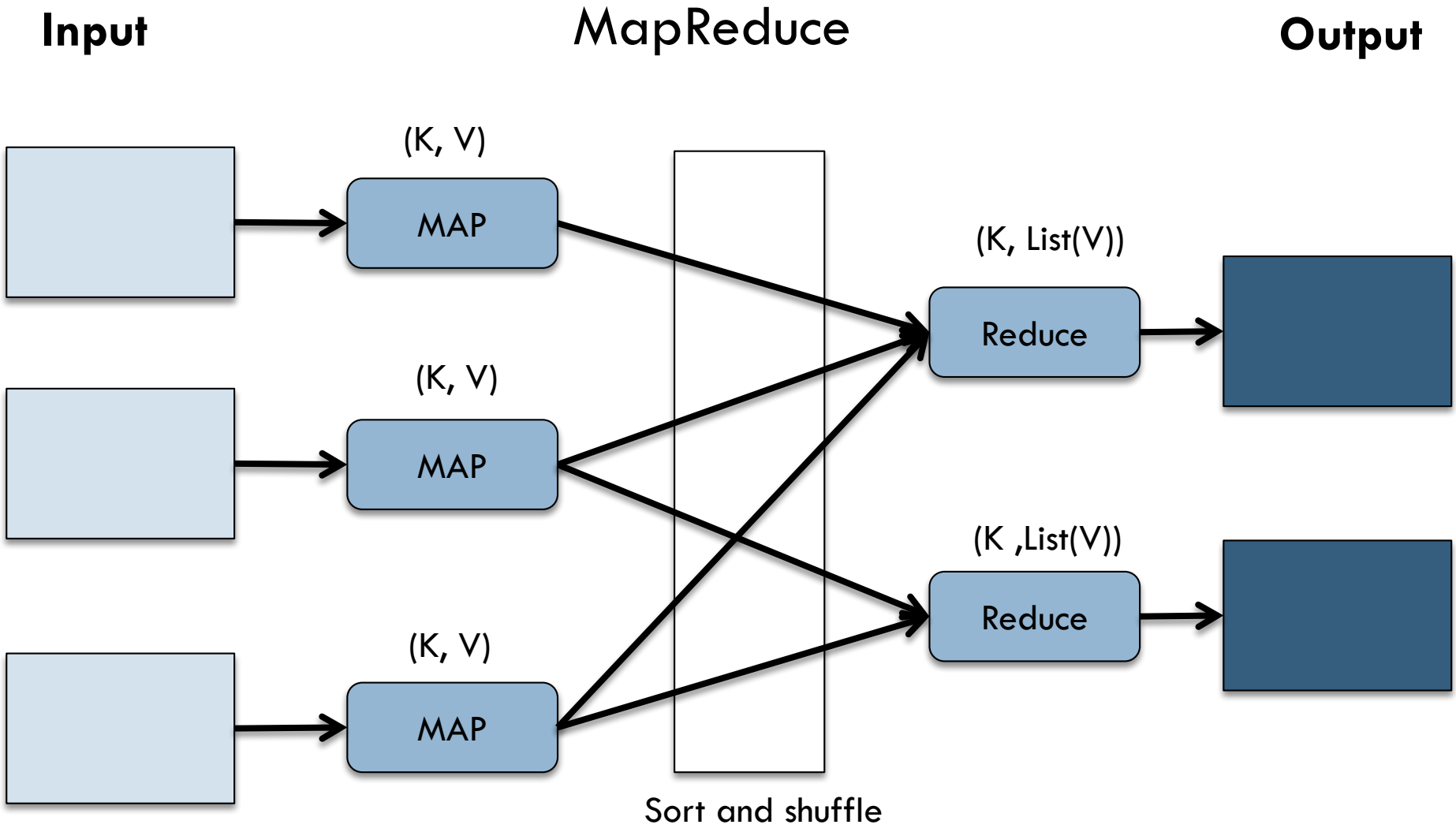


DAG execution engine

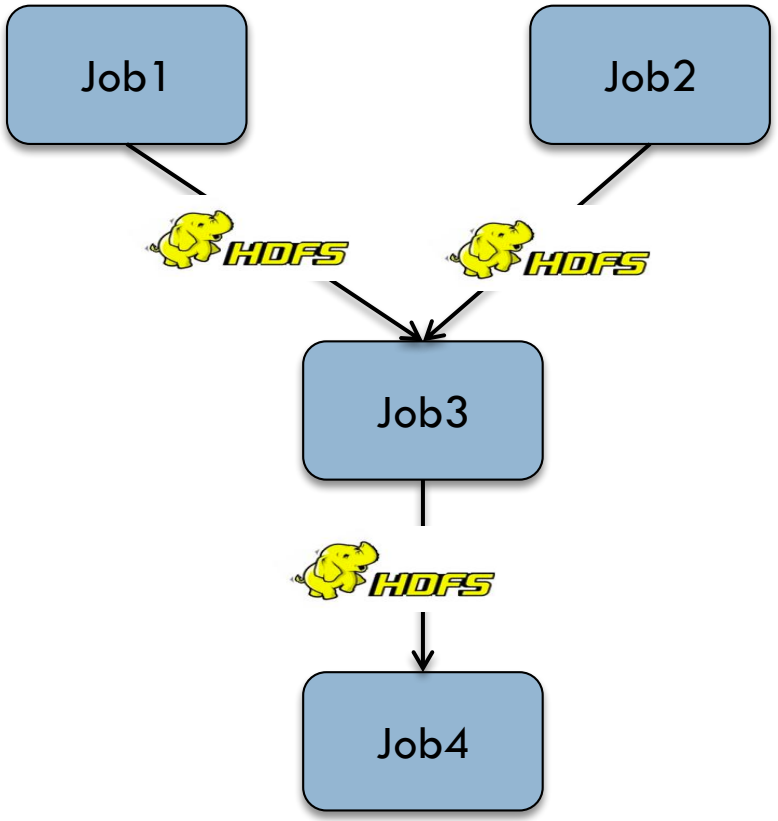
Introduction to Apache Spark

- MapReduce limitations
  - Difficult and tedious to program
    - Low level
    - Assembly language for data processing
  - Performance bottlenecks
  - Not easily used for non-batch use cases
    - Iterative, interactive, graph processing, streaming
  - Need to string together a series of MapReduce jobs

Introduction to Apache Spark



Introduction to Apache Spark



MapReduce doesn't support multi-stage jobs

Introduction to Apache Spark

- Main abstraction is the relation
- Provide operators to manipulate the relations

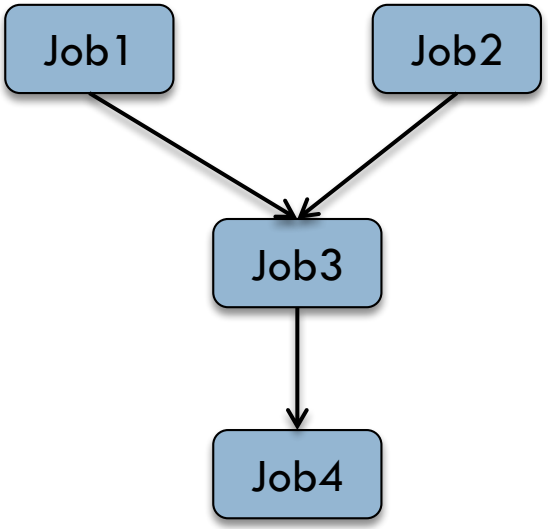
```
// loading data file
input_lines = LOAD '/dataset' AS (line:chararray);

// transformation
- filter, group, join, order, union
```

```
member_account = LOAD
'/data/databases/MEMBE
R2/MEMBER_ACCOUNT/#LAT
EST' using
LiAvroStorage();

member_account_filtere
d = FILTER
member_account BY
COUNTRY_CODE == 'us'
AND POSTAL_CODE is not
null;
```

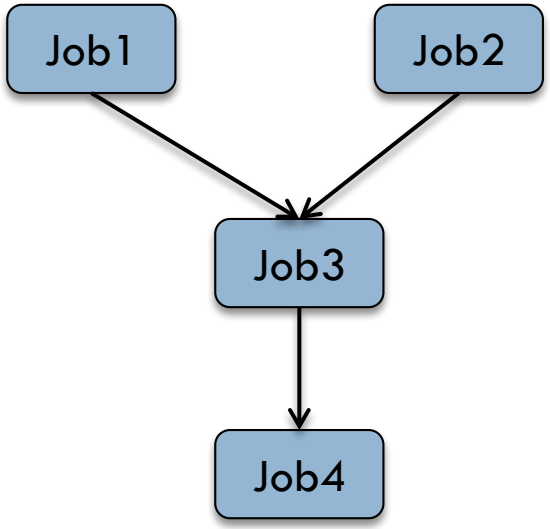
Generates



Introduction to Apache Spark

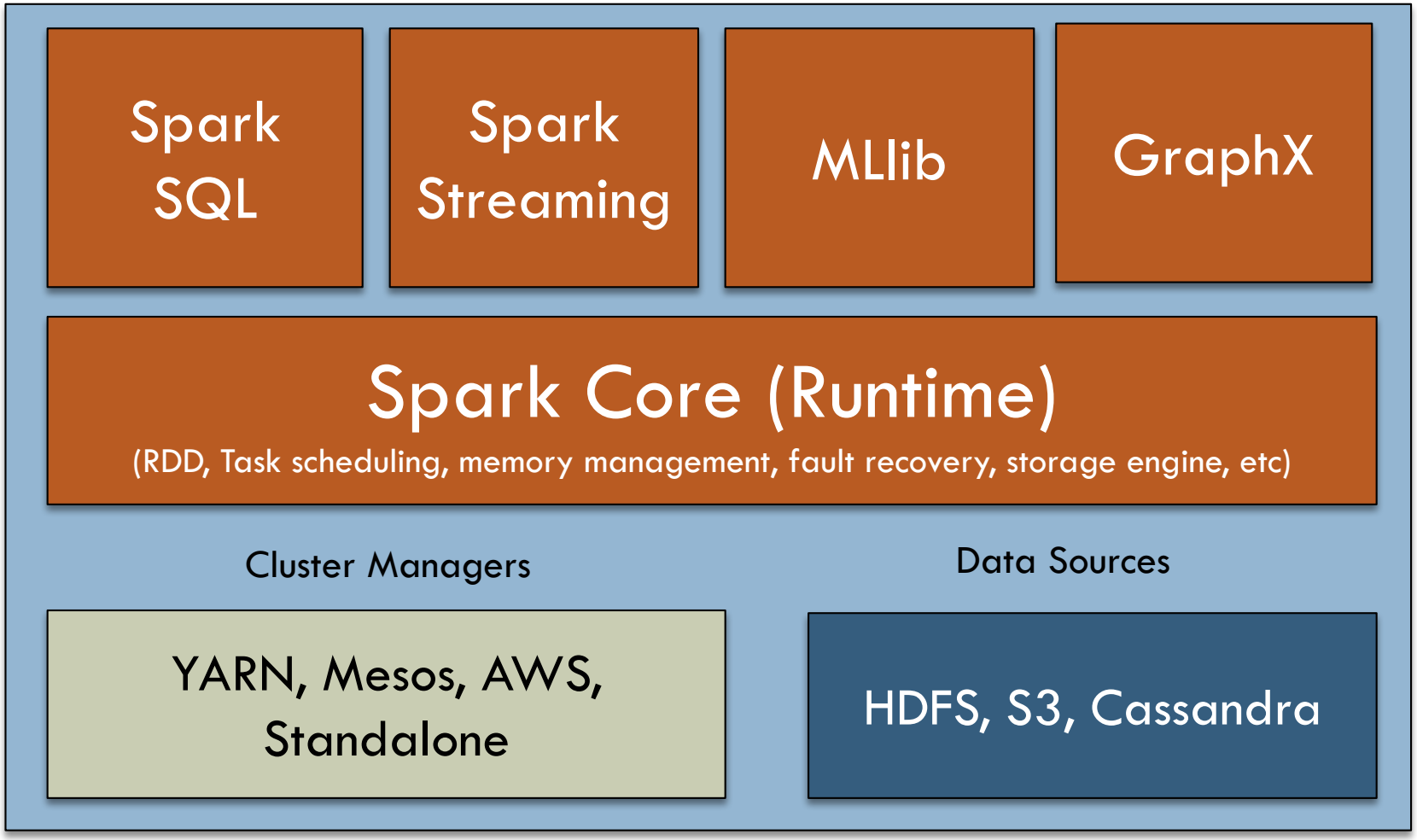
- Main abstraction is the table
- Provide SQL to manipulate the tables

```
select contry,
count(*) from
employee group
by country
having size >
100
```



Core Concepts and Architecture

Apache Spark Unified Stack

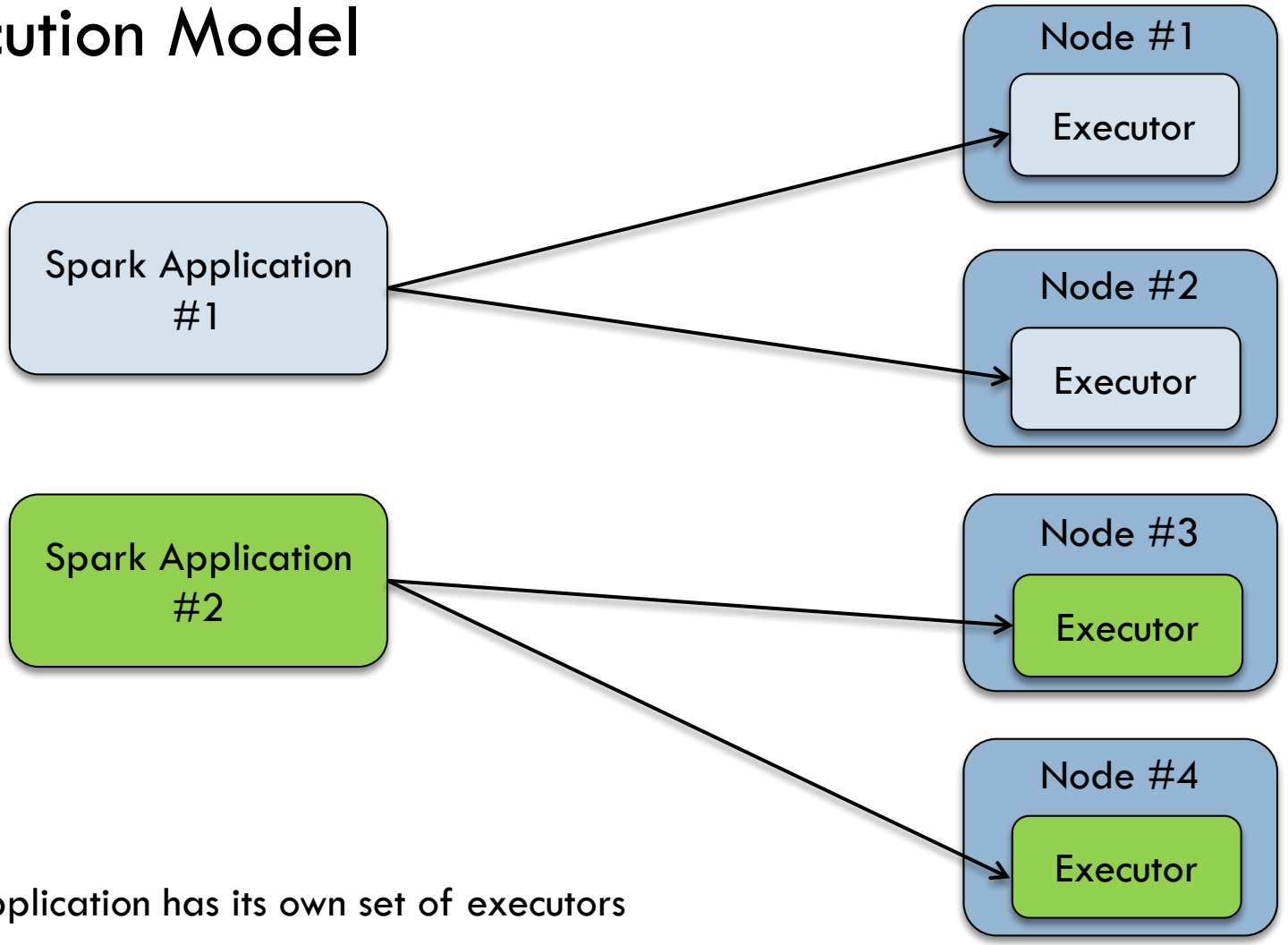


Core Concepts and Architecture

- Spark doesn't come with a storage layer
  - Unlike Hadoop
- Can easily integrate with HDFS and others
- Apache Alluxio (fka Tachyon)
  - Memory-centric distributed storage system
  - Reliable data-sharing at memory speed

Apache Spark Overview

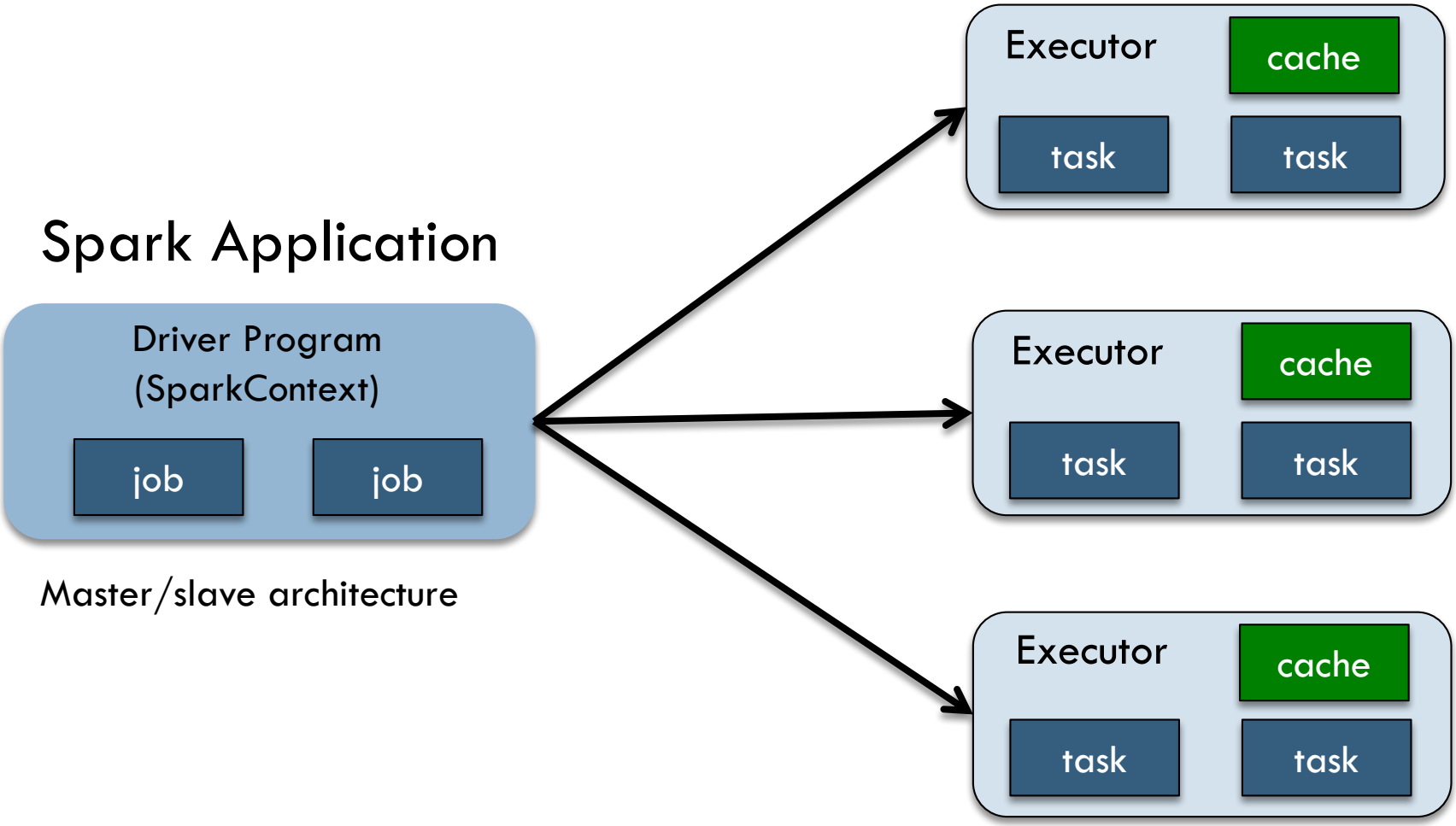
Execution Model



Each application has its own set of executors

Core Concepts and Architecture

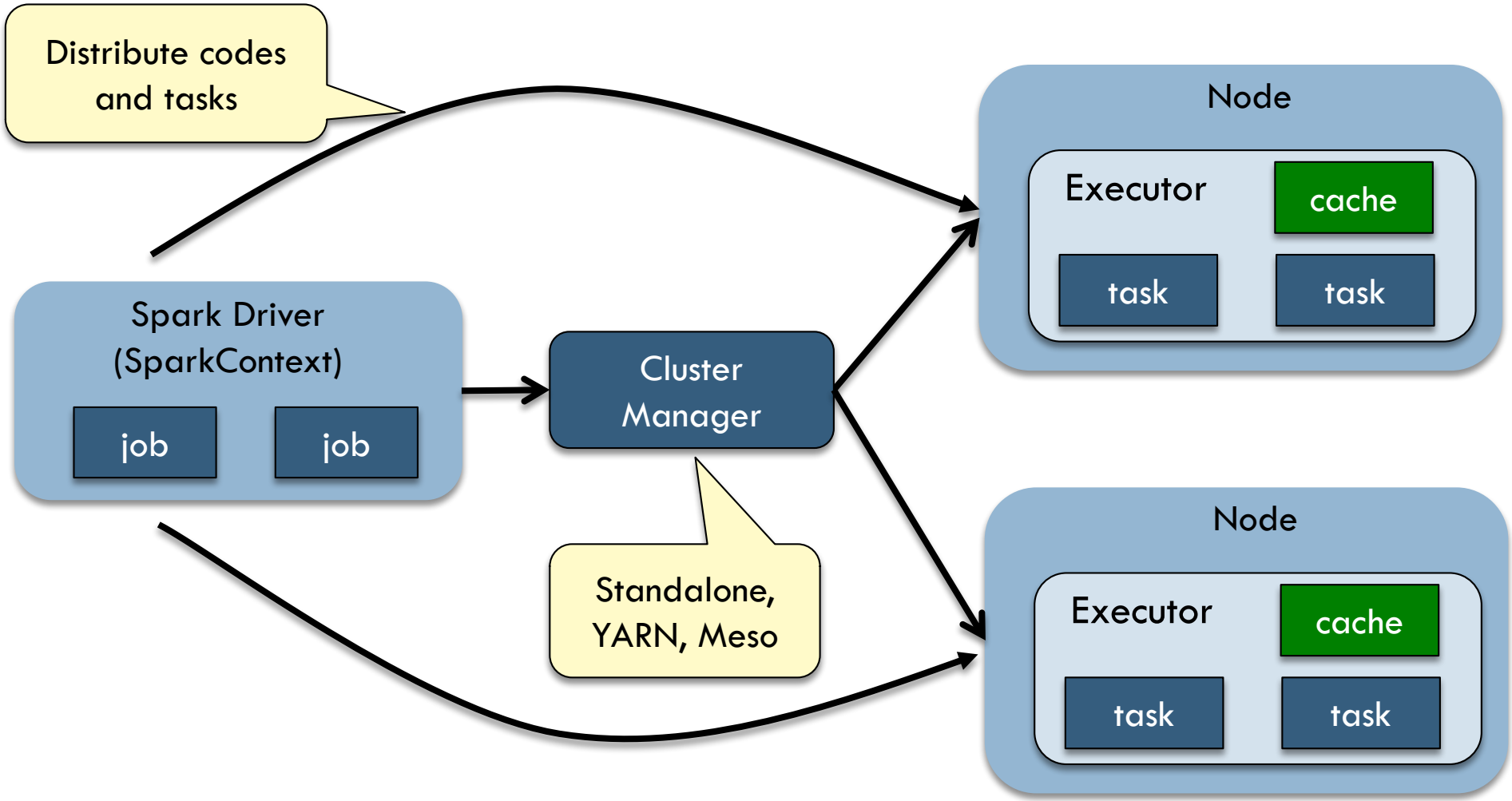
Spark Application Execution Model



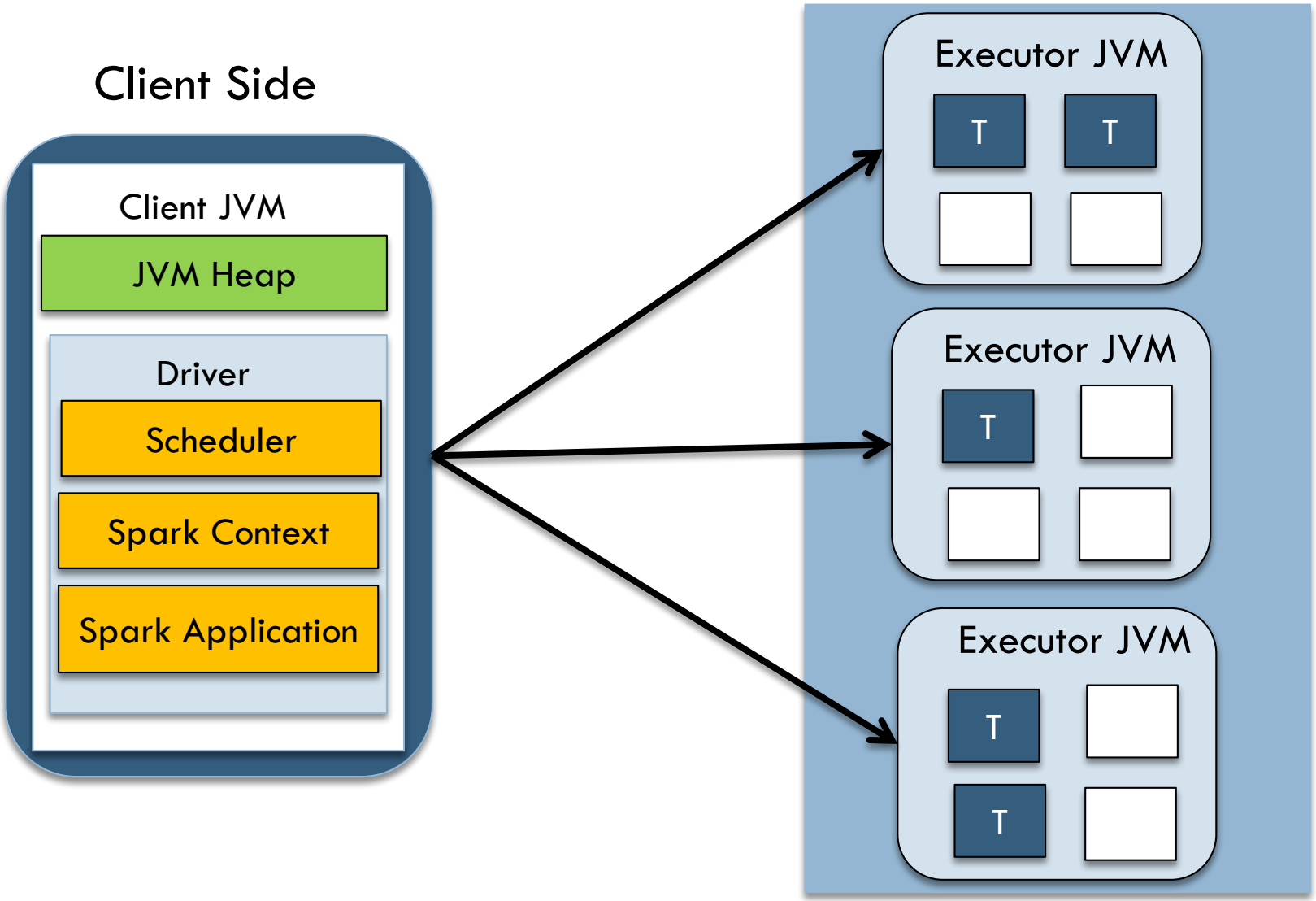
Master/slave architecture

Core Concepts and Architecture

Spark Application Execution Model



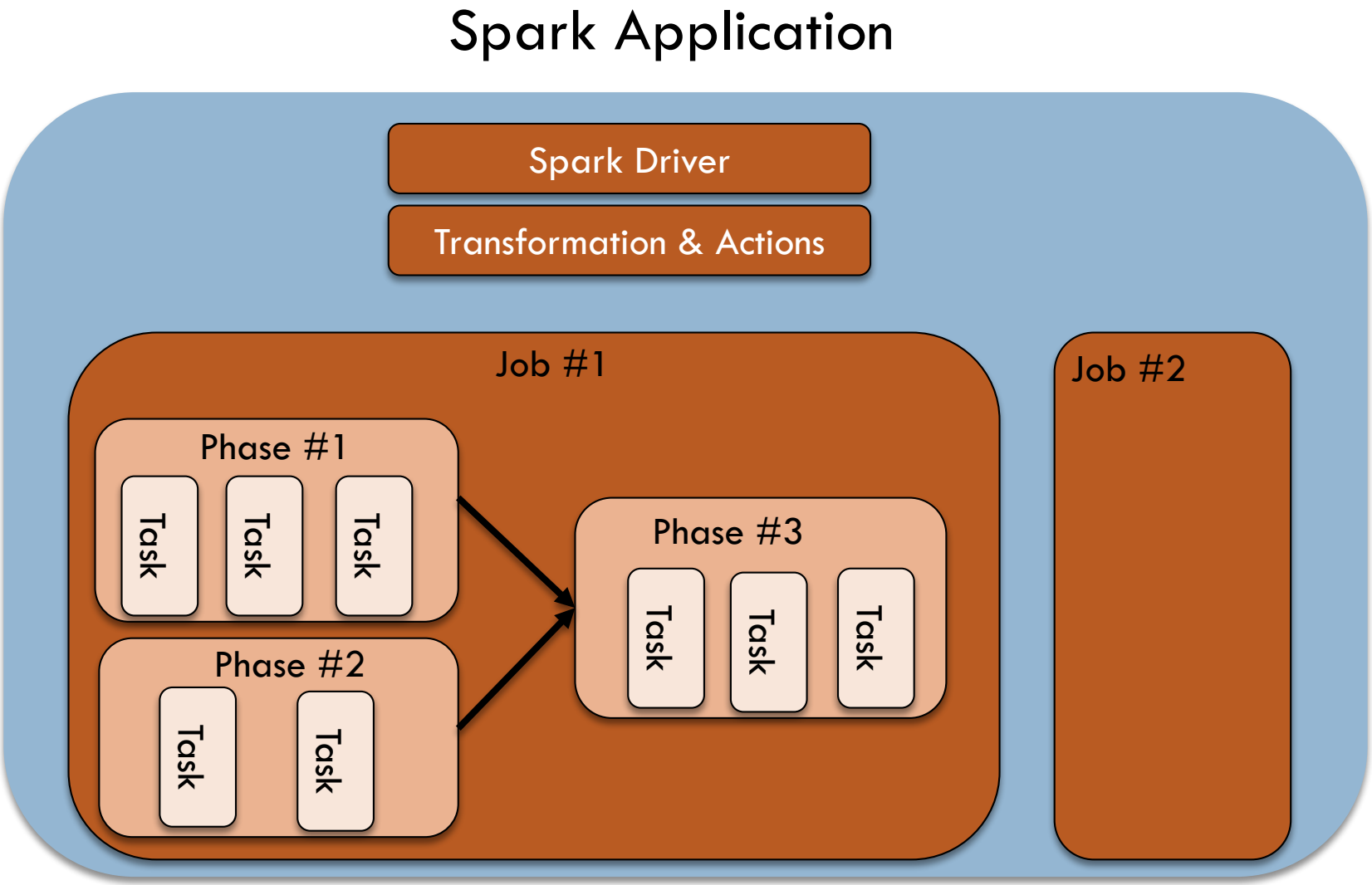
Core Concepts and Architecture



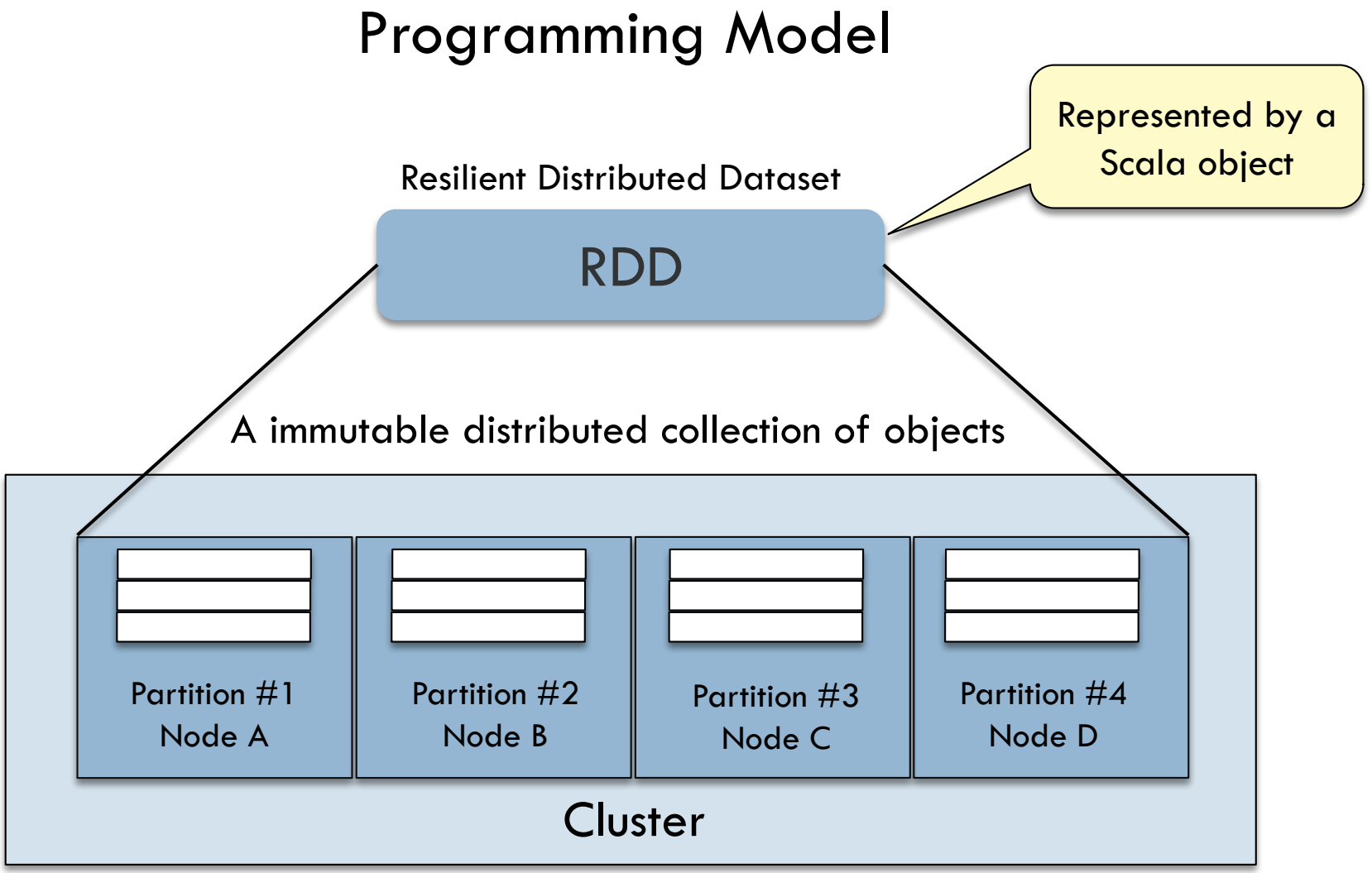
Core Concepts and Architecture

- Driver
  - Central coordinator
  - Optimization
    - Pipelining
  - Convert logic into set of stages
  - Each stage consists of multiple tasks
  - Tasks are smallest unit of work
  - Tasks are packaged and prepared to send to executors
  - Tasks are coordinated and scheduled to run on executors

Core Concepts and Architecture



Apache Spark Overview



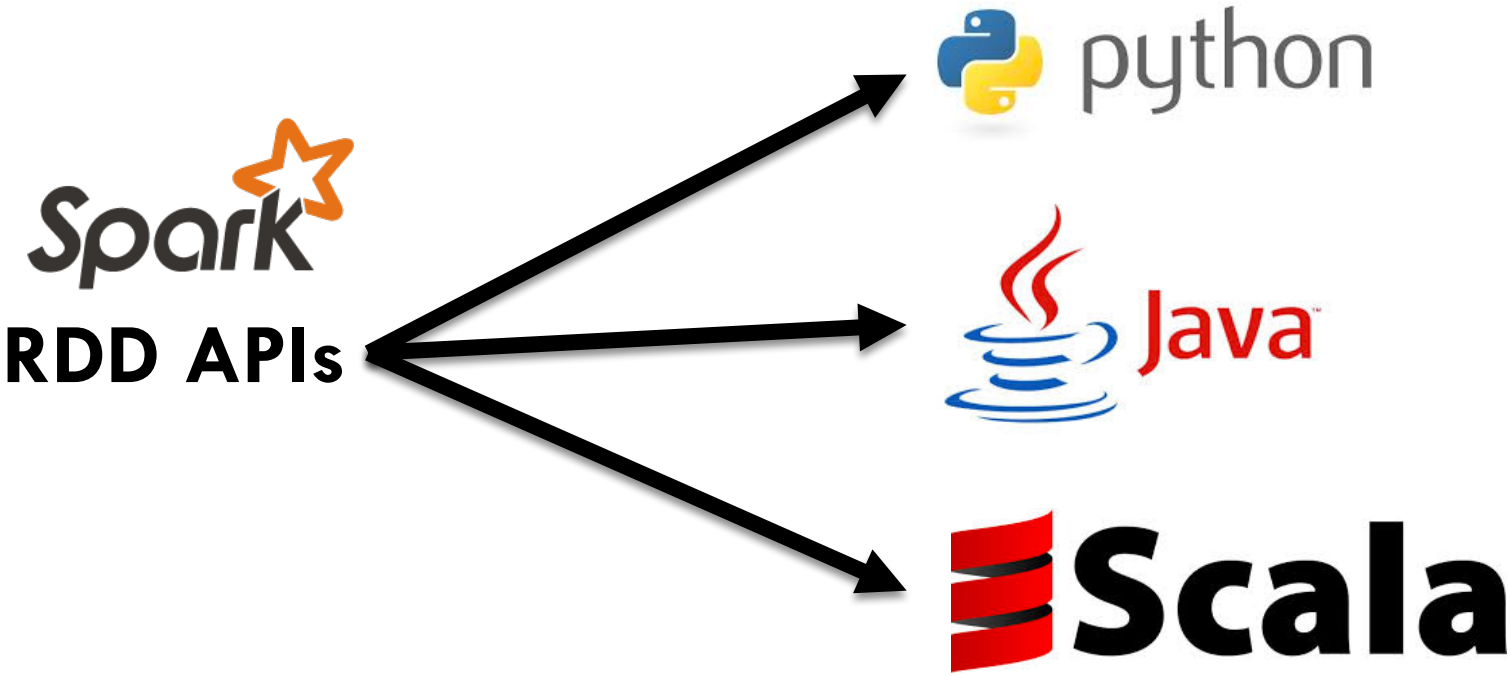
Core Concepts and Architecture

- RDD (Resilient Distributed Dataset)
  - Dataset
    - Initial data can from a file or created programmatically
  - Distributed
    - Store across nodes in the cluster
  - Resilient
    - Automatic rebuilt on failure (through lineage)
- Able to persist intermediate results in memory
  - Spill to disk if not enough RAM
  - Controllable persistence

Core Concepts and Architecture

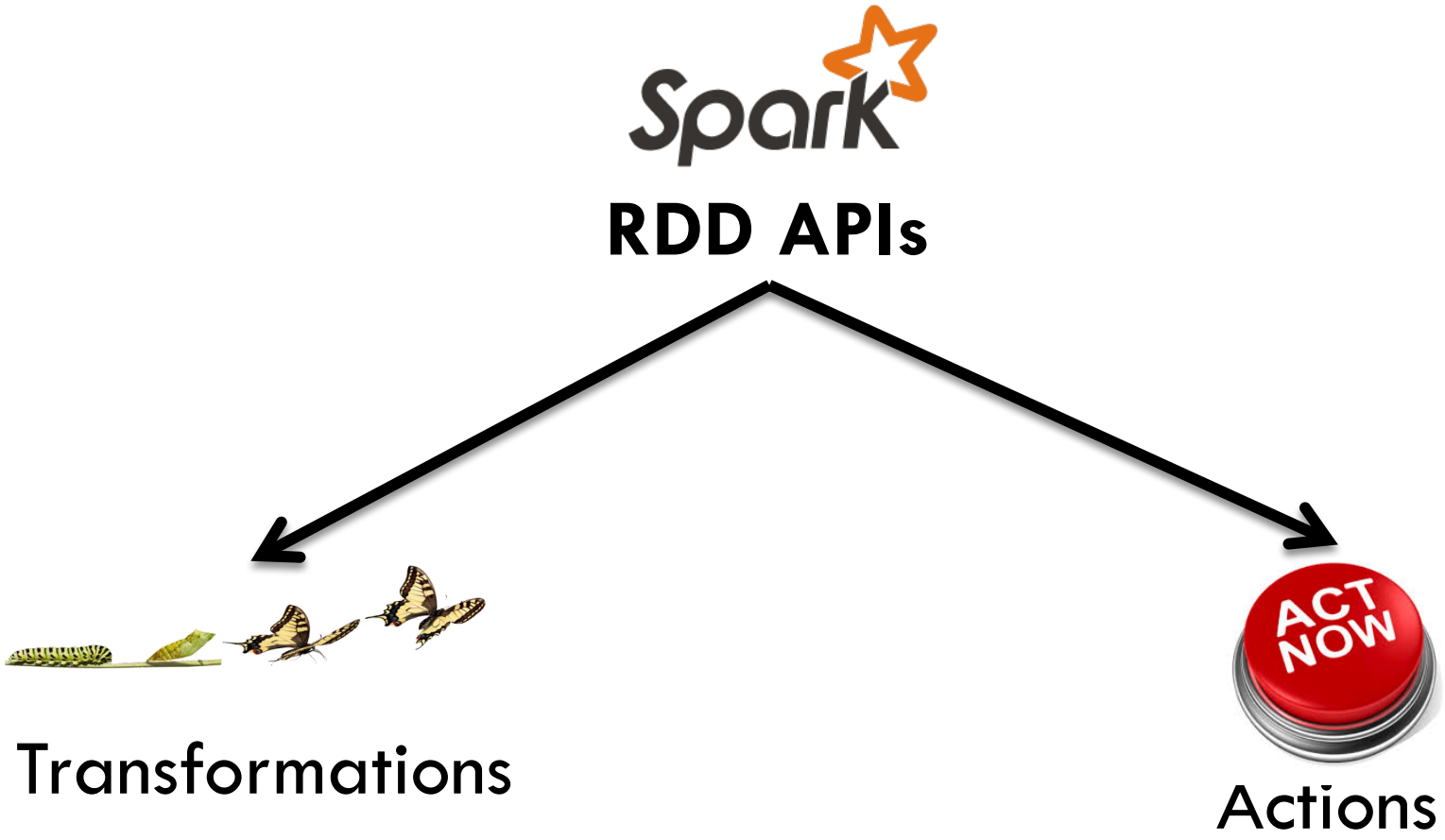
Power of Spark

(Rich APIs and interactive shell)



Similar to Scala collection APIs, but for distributed data computation

Core Concepts and Architecture



Core Concepts and Architecture

- RDD programming interface
  - Transformations (e.g map, filter, groupBy, join)
    - Operations on RDD
    - **Create** new RDD from previous one
    - **Lazy** evaluation (when use an action), why?
  - Actions (e.g count, collect, save)
    - Return a result or write to storage
    - **Force** evaluation of the transformations
  - Restricted share variables
    - Broadcast and accumulators

Core Concepts and Architecture

- RDD programming interface
  - Broadcast variables
    - For a large read-only piece of data (lookup table)
    - Distribute and cache on each executor (machine)
      - Rather than tasks
    - Reduce communication costs
  - Accumulators
    - Can only “add” to using an associative operation
    - Only driver can read
    - Can be used to implement like counters in MapReduce

Core Concepts and Architecture

- RDD Persistence
  - One of the most important capabilities
  - Reading data from memory is extremely fast
  - Good interactive use cases and iterative algorithms
  - Support different storage level
    - Memory
    - Disk
    - Serialized or unserialized
  - LRU for eviction policy

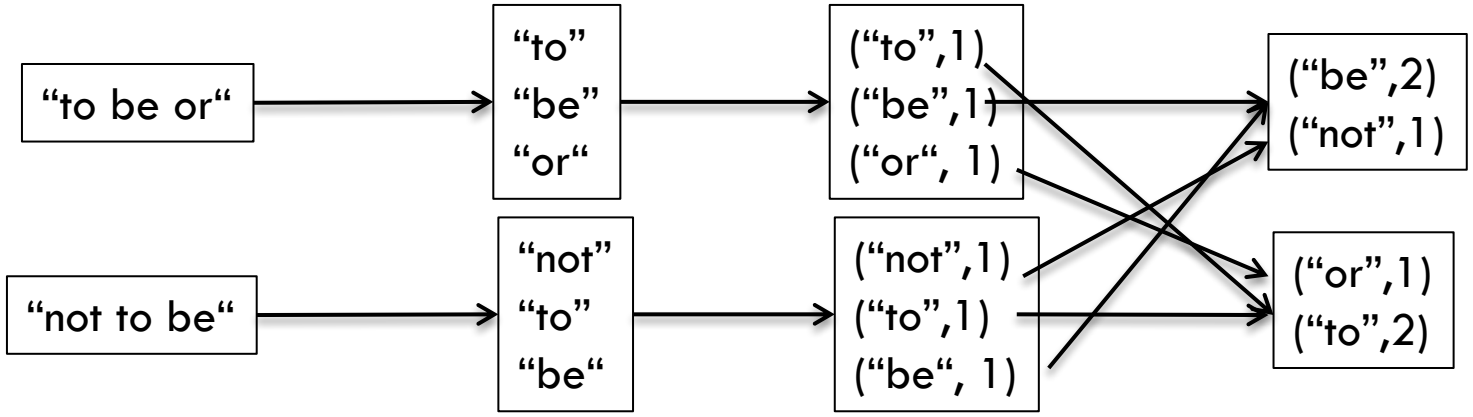
Core Concepts and Architecture

Word count example

```
// loading data file
val file = sc.textFile("README.md")

val words = file.flatMap(l => l.split(" "))
val wordCountPairs = words.map(word => (word, 1))
val wordCountSum = wordCountPairs.reduceByKey(_ + _)

// store output
wordCountSum.saveAsTextFile("/tmp/output")
```



Core Concepts and Architecture


```
// loading data file
val file = sc.textFile("README.md")

val words = file.flatMap(l => l.split(" "))
val wordCountPairs = words.map(word => (word, 1))
val wordCountSum = wordCountPairs.reduceByKey(_ + _)
// store output
wordCountSum.saveAsTextFile("/output")
```