Amirkabir University of Technology
(Tehran Polytechnic)

Cloud Computing

Introduction to Apache Spark

Seyyed Ahmad Javadi

sajavadi@aut.ac.ir

Fall 2021

Apache Spark 101

Lance Co Ting Keh

Senior Software Engineer, Machine Learning @ Box

Outline

I. About me
II. Distributed Computing at a High Level
III. Disk versus Memory based Systems
IV. Spark Core
    I. Brief background
    II. Benchmarks and Comparisons
    III. What is an RDD
    IV. RDD Actions and Transformations
    V. Caching and Serialization
    VI. Anatomy of a Program
    VII. The Spark Family

Why Distributed Computing?

Divide and Conquer

Problem Single machine cannot complete the computation at hand

Solution Parallelize the job and distribute work among a network of machines

Issues Arise in Distributed Computing

View the world from the eyes of a single worker

- How do I **distribute** an algorithm?
- How do I **partition** my dataset?
- How do I maintain **a single consistent view** of a shared state?
- How do I recover from **machine failures**?
- How do I **allocate cluster resources**?
- …..

Finding majority element in a single machine

Think distributed

List(20, 18, 20, 18, 20)

Finding majority element in a distributed dataset

Think distributed

List(1, 18, 1, 18, 1)

List(2, 18, 2, 18, 2)

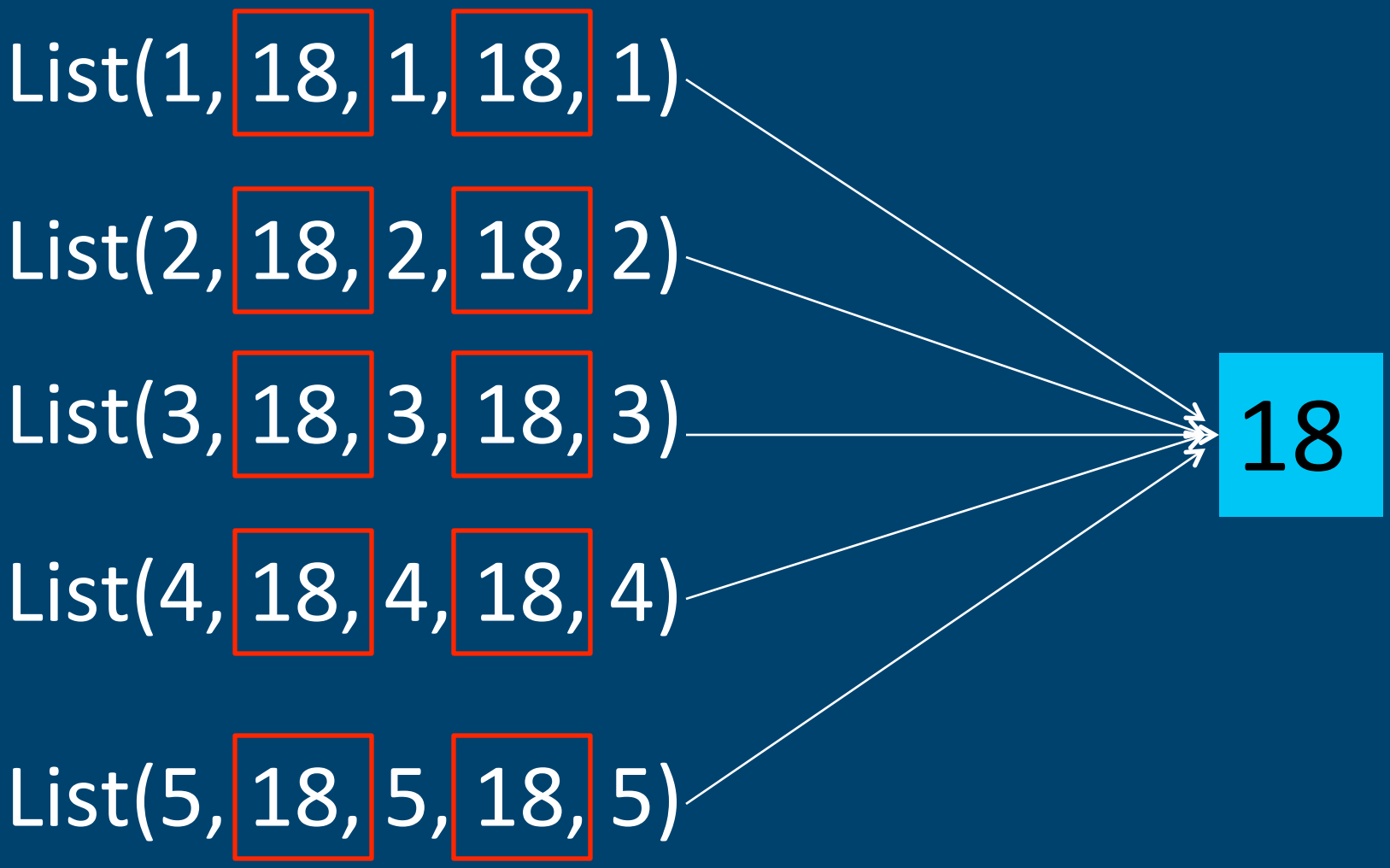
List(3, 18, 3, 18, 3)

List(4, 18, 4, 18, 4)

List(5, 18, 5, 18, 5)

Finding majority element in a distributed dataset

Think distributed



Disk Based vs Memory Based Frameworks

Acyclic data flow

- Disk Based Frameworks
  - Persists intermediate results to disk
  - Data is reloaded from disk with every query
  - Easy failure recovery
  - Best for ETL like work-loads
  - Examples: Hadoop, Dryad

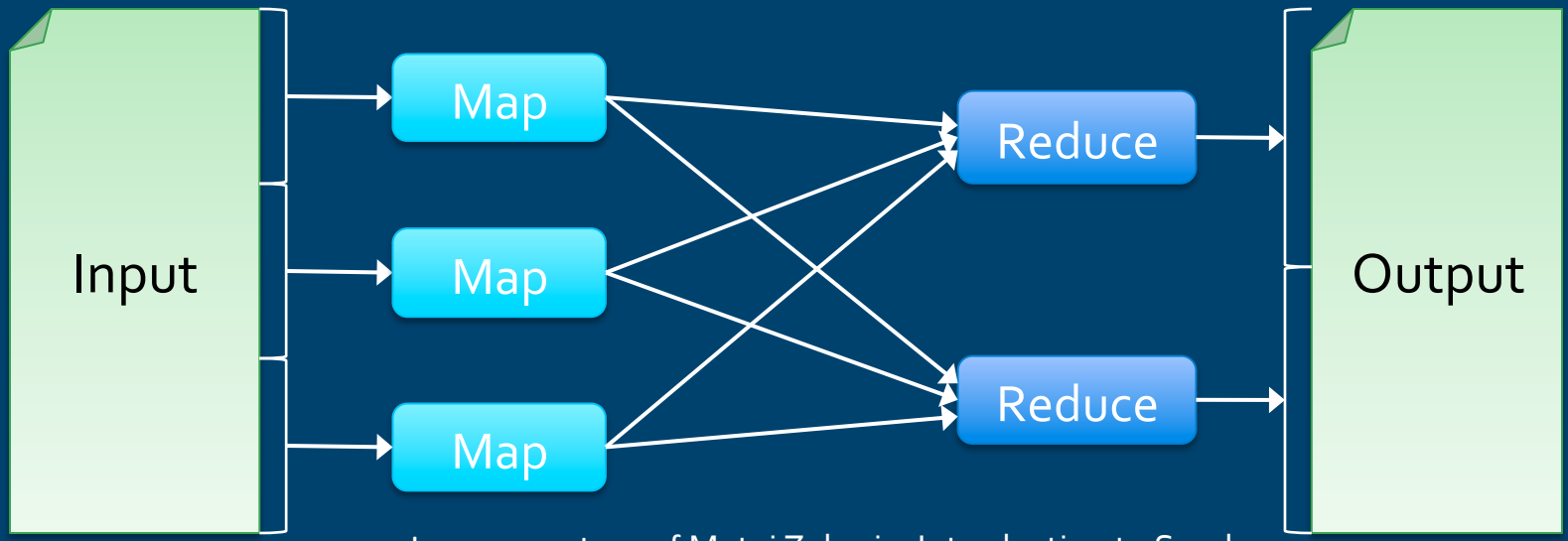


Image courtesy of Matei Zaharia, Introduction to Spark

Disk Based vs Memory Based Frameworks

Reuse working data set in memory

- Memory Based Frameworks
  - Circumvents heavy cost of I/O by keeping intermediate results in memory
  - Sensitive to availability of memory
  - Remembers operationsapplied to dataset
  - Best for iterative workloads
  - Examples: Spark, Flink

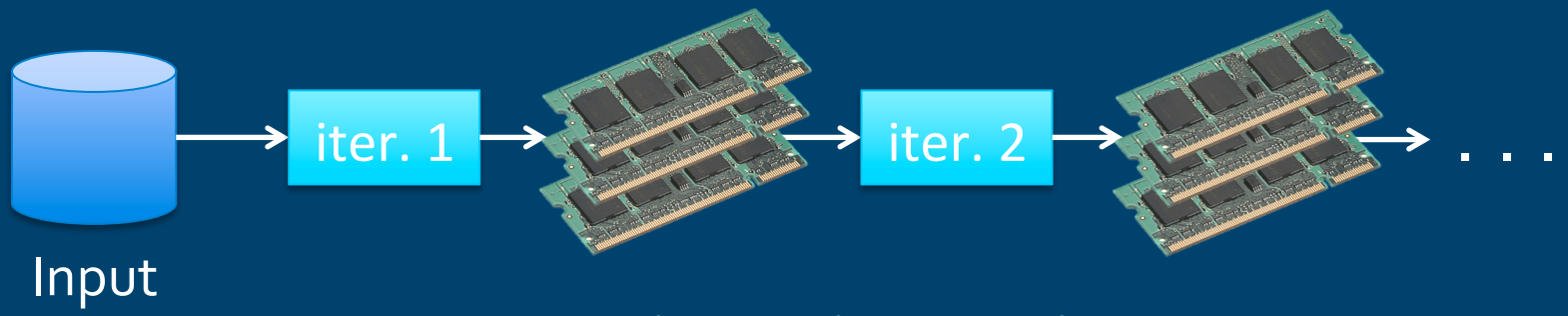


Image courtesy of Matei Zaharia, Introduction to Spark

The rest of the talk

I. Spark Core
   I. Brief background
   II. Benchmarks and Comparisons
   III. What is an RDD
   IV. RDD Actions and Transformations
   V. Spark Cluster
   VI. Anatomy of a Program
   VII. The Spark Family

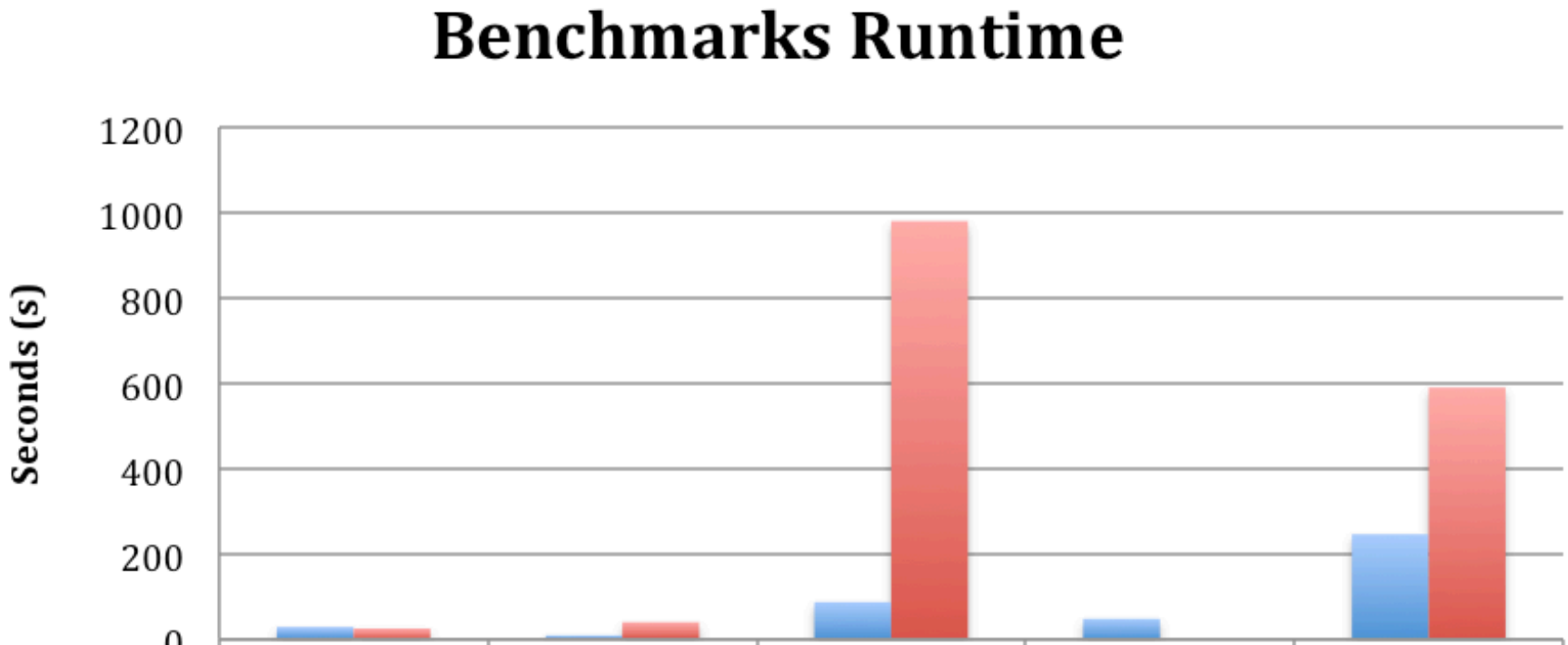
Spark Background

Arose from an academic setting

- Amplab UC Berkley
- Project Lead: Dr. Matei Zaharia
- First paper published on RDD's was in 2012
- Open sourced from day one, growing number of contributors
- Released its 1.0 version May 2014. Currently in 1.2.1
- *Databricks* company established to support Spark and all its related technologies. Matei currently sits as its CTO
- Amazon, Alibaba, Baidu, eBay, Groupon, Ooyala, OpenTable, Box, Shopify, TechBase, Yahoo!

Spark versus Scalding (Hadoop)

Clear win for iterative applications



	Generating Datapoints	Kmeans (per iteration)	Twitter	Wikipedia	Jaccard
Spark	29	9	87	48	246
Scalding	25	40	980		590

Resilient Distributed Datasets (RDDs)

- Main object in Spark's universe
- Think of it as representing the data at that stage in the operation
- Allows for *coarse-grained* transformations (e.g. map, group-by, join)
- Allows for efficient fault recovery using *lineage*
  - Log *one* operation to apply to many elements
  - Recompute lost partitions of dataset on failure
  - No cost if nothing fails

RDD Actions and Transformations

Transformations are realized when an action is called

- Transformations
  - Lazy operations applied on an RDD
  - Creates a new RDD from an existing RDD
  - Allows Spark to perform optimizations
  - e.g. map, filter, flatMap, union, intersection, distinct, reduceByKey, groupByKey
- Actions
  - Returns a value to the driver program after computation
  - e.g. reduce, collect, count, first, take, saveAsFile

RDD Representation

- Simple common interface:
  - Set of partitions
  - Preferred locations for each partition
  - List of parent RDDs
  - Function to compute a partition given parents
  - Optional partitioning info
- Allows capturing wide range of transformations

Slide courtesy of Matei Zaharia, Introduction to Spark

Spark Cluster

Driver

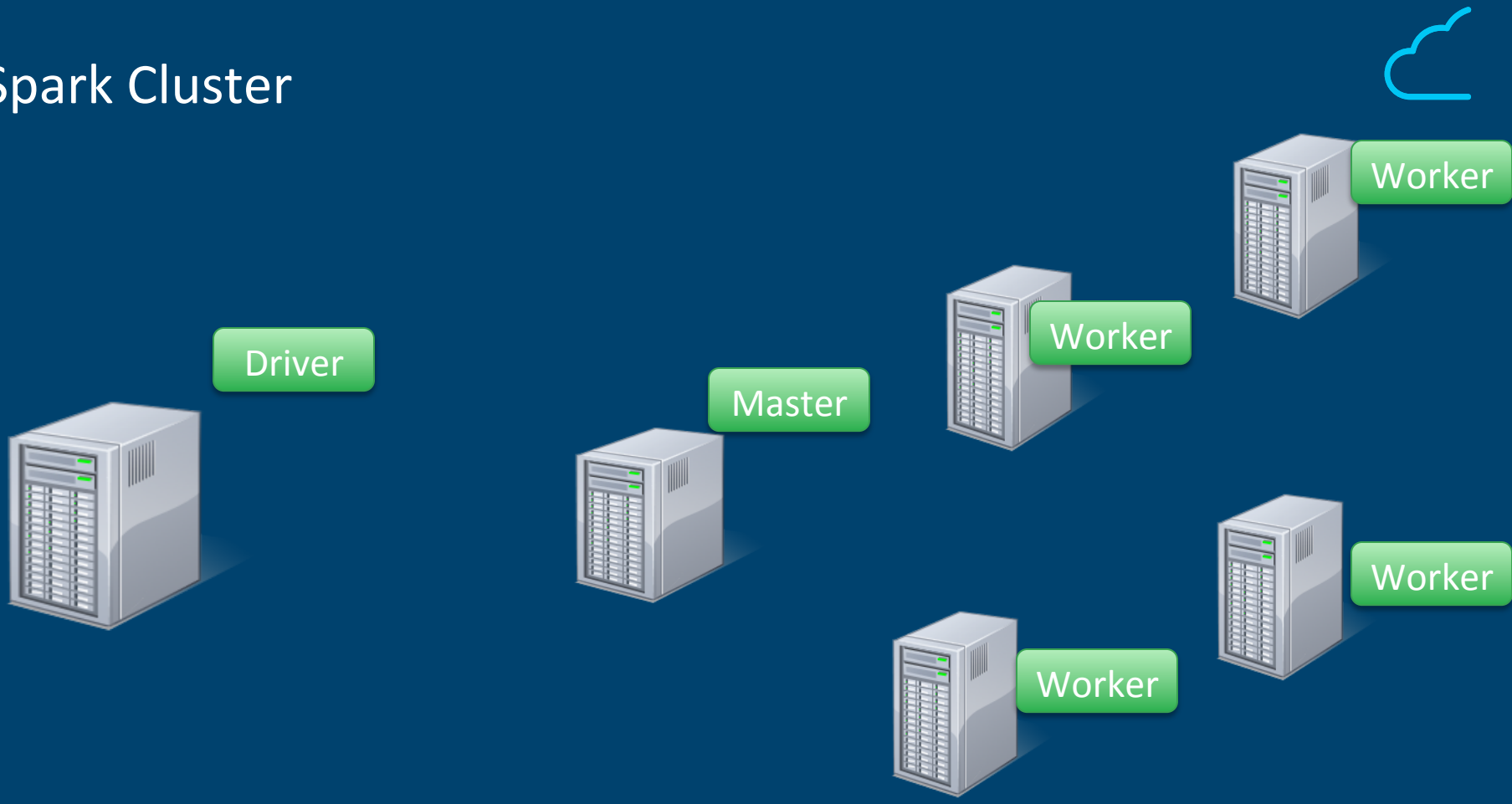
- Entry point of Spark application
- Main Spark application is ran here
- Results of “reduce” operations are aggregated here

Spark Cluster

Master

- Distributed coordination of Spark workers including:
  - Health checking workers
  - Reassignment of failed tasks
  - Entry point for job and cluster metrics

Spark Cluster



Worker

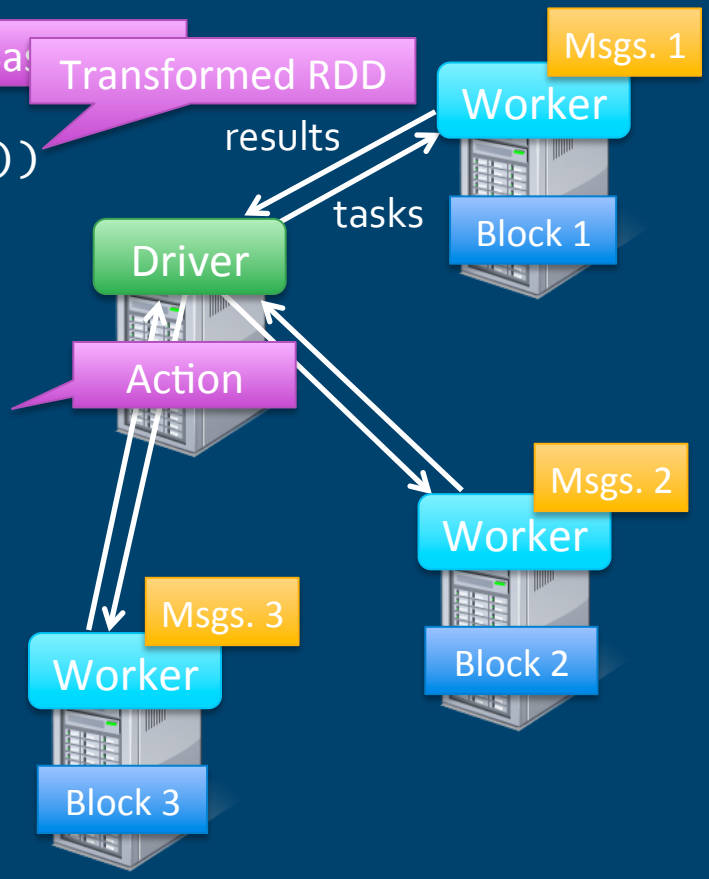
- Spawns executors to perform tasks on partitions of data

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(_.startsWith(“ERROR”))
messages = errors.map(_.split(‘\t’)(2))
messages.persist()

messages.filter(_.contains(“foo”)).count
messages.filter(_.contains(“bar”)).count
. . .
```



Result: scaled to 1 TB data in 5-7 sec
(vs 170 sec for on-disk data)

Slide courtesy of Matei Zaharia, Introduction to Spark

The Spark Family

Cheaper by the dozen

- Aside from its performance and API, the diverse tool set available in Spark is the reason for its wide adoption
  1. Spark Streaming
  2. Spark SQL
  3. MLlib
  4. GraphX