

# Why Parallel Processing?



Divide and Conquer

<u>Problem</u> Single machine cannot complete the computation at hand

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



### Issues Arise in Parallel Processing



View the world from the eyes of a single worker

- How do I parallelize an algorithm?
- How do | 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?
- •



Think distributed





Think distributed

### Finding majority element in a distributed dataset



Think distributed

### Spark Background

#### Arose from an academic setting

- Amplab UC Berkeley
- 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 1.4.1
- 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!

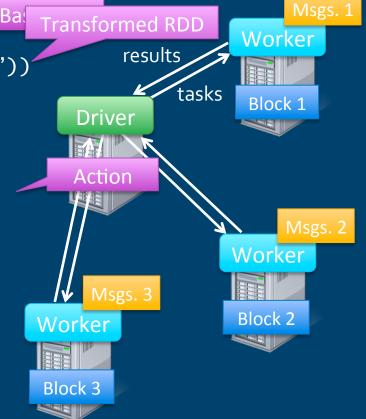
# Example: Log Mining

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)



### 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

### Spark Cluster





#### Driver

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

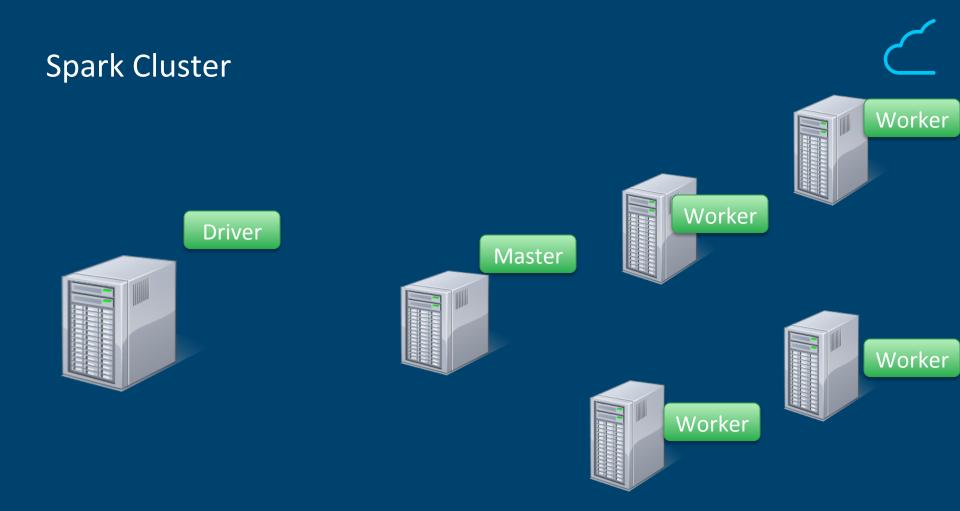
### Spark Cluster





#### Master

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



#### Worker

Spawns executors to perform tasks on partitions of data

## The Spark Family

#### Cheaper by the dozen

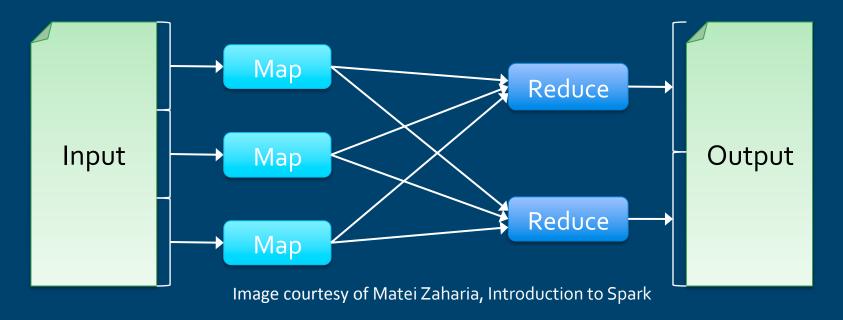
- Aside from its performance and API, the diverse tool set available in Spark is the reason for its wide adoption
  - 1. Spark SQL
  - 2. Spark Streaming
  - 3. MLlib
  - 4. GraphX

### 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



### 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

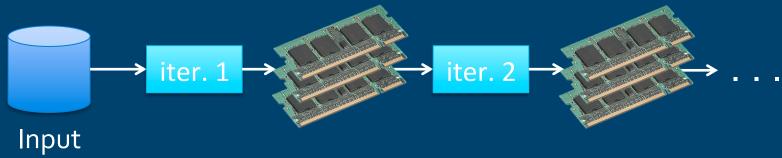
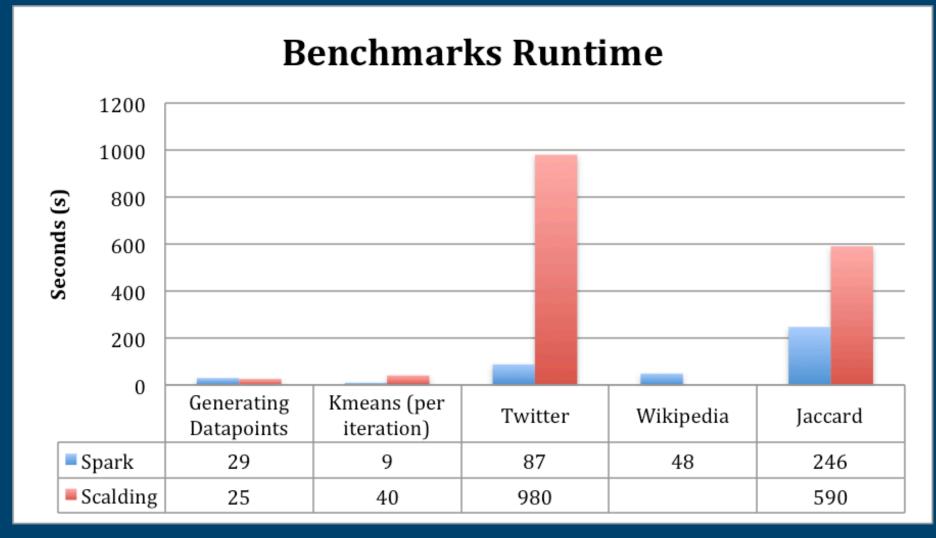


Image courtesy of Matei Zaharia, Introduction to Spark





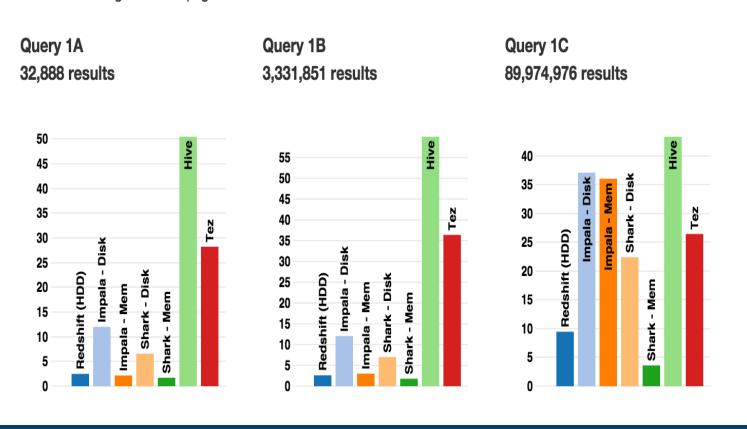
Clear win for iterative applications



### Ad-hoc batch queries



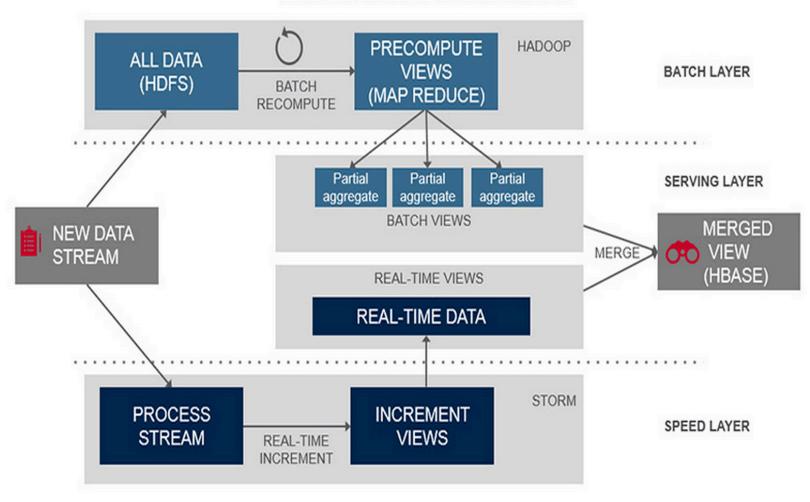
SELECT pageURL, pageRank FROM rankings WHERE pageRank > X



#### Lambda Architecture



### Lambda Architecture







**Unified Framework** 

### Lambda Architecture

