Big data: architectures and data analytics

Introduction to Spark
Spark

- Apache Spark™ is a fast and general-purpose engine for large-scale data processing
- Spark aims at achieving the following goals in the Big data context
  - Generality: diverse workloads, operators, job sizes
  - Low latency: sub-second
  - Fault tolerance: faults are the norm, not the exception
  - Simplicity: often comes from generality

Spark History

- Originally developed at the University of California - Berkeley's AMPLab

  - 2002: MapReduce @ Google
  - 2004: MapReduce paper
  - 2006: Hadoop Summit
  - 2008: Apache Spark top-level
  - 2010: Spark paper
  - 2014: Apache Spark top-level
Spark: Motivations

MapReduce and Iterative Jobs

- Iterative jobs, with MapReduce, involve a lot of disk I/O for each iteration and stage

![Diagram showing Mappers and Reducers in iterative jobs with stages 1 and 2.](image-url)
MapReduce and Iterative Jobs

- Disk I/O is very slow (even if it is local I/O)

![Diagram of MapReduce Stages]

Apache Spark: Motivation and Opportunity

- Motivation
  - Using MapReduce for complex iterative jobs or multiple jobs on the same data involves lots of disk I/O

- Opportunity
  - The cost of main memory decreased
    - Hence, large main memories are available in each server

- Solution
  - Keep more data in main memory
    - Basic idea of Spark
From MapReduce to Spark

- MapReduce: Iterative job
  - Input
  - HDFS read
  - Iteration 1
  - HDFS write
  - Iteration 2
  - ... 

From MapReduce to Spark

- Spark: Iterative job
  - Input
  - HDFS read
  - Iteration 1
  - Iteration 2
  - ... 
  - Data are shared between the iterations by using the main memory
    - Or at least part of them
    - 10 to 100 times faster than disk
From MapReduce to Spark

- MapReduce: Multiple analyses of the same data

```
Input
  HDFS read
  HDFS read
  HDFS read
  HDFS read

query1 result1
query2 result2
query3 result3
...
```

From MapReduce to Spark

- Spark: Multiple analyses of the same data

```
Input
  Distributed memory

HDFS read

query1 result1
query2 result2
query3 result3
```

- Data are read only once from HDFS and stored in main memory
  - Split of the data across the main memory of each server
**Spark: Resilient Distributed Data sets (RDDs)**

- Data are represented as Resilient Distributed Datasets (RDDs)
  - Partitioned/Distributed collections of objects spread across the nodes of a clusters
  - Stored in main memory (when it is possible) or on local disk
- Spark programs are written in terms of operations on resilient distributed data sets

---

**Spark: Resilient Distributed Data sets (RDDs)**

- RDDs are built and manipulated through a set of parallel
  - Transformations
    - map, filter, join, ...
  - Actions
    - count, collect, save, ...
- RDDs are automatically rebuilt on machine failure
Spark Computing Framework

- Provides a programming abstraction (based on RDDs) and transparent mechanisms to execute code in parallel on RDDs
  - Hides complexities of fault-tolerance and slow machines
  - Manages scheduling and synchronization of the jobs

MapReduce vs Spark

<table>
<thead>
<tr>
<th></th>
<th>Hadoop MapReduce</th>
<th>Spark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage</td>
<td>Disk only</td>
<td>In-memory or on disk</td>
</tr>
<tr>
<td>Operations</td>
<td>Map and Reduce</td>
<td>Map, Reduce, Join, Sample, etc…</td>
</tr>
<tr>
<td>Execution model</td>
<td>Batch</td>
<td>Batch, interactive, streaming</td>
</tr>
<tr>
<td>Programming environments</td>
<td>Java</td>
<td>Scala, Java, Python, and R</td>
</tr>
</tbody>
</table>
MapReduce vs Spark

- Lower overhead for starting jobs
- Less expensive shuffles

In-Memory RDDs Can Make a Big Difference

- Two iterative Machine Learning algorithms:
  - K-means Clustering
  - Logistic Regression
## Petabyte Sort Challenge

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

**Daytona Gray**

100 TB sort benchmark record (tied for 1st place)

---

## Spark: Main components
Spark is based on a basic component (the Spark Core component) that is exploited by all the high-level data analytics components:

- This solution provides a more uniform and efficient solution with respect to Hadoop where many non-integrated tools are available.
- When the efficiency of the core component is increased also the efficiency of the other high-level components increases.
Spark Components

- Spark Core
  - Contains the basic functionalities of Spark exploited by all components
    - Task scheduling
    - Memory management
    - Fault recovery
    - ...
  - Provides the APIs that are used to create RDDs and applies transformations and actions on them

Spark Components

- Spark SQL structured data
  - This component is used to interact with structured datasets by means of the SQL language
  - It supports also
    - Hive Query Language (HQL)
  - It interacts with many data sources
    - Hive Tables
    - Parquet
    - JSON
Spark Components

- Spark Streaming real-time
  - It is used to process live streams of data in real-time
  - The APIs of the Streaming real-time components operated on RDDs and are similar to the ones used to process standard RDDs associated with “static” data sources

Spark Components

- MLlib
  - It is a machine learning/data mining library
  - It can be used to apply the parallel versions of some machine learning/data mining algorithms
    - Data preprocessing and dimensional reduction
    - Classification algorithms
    - Clustering algorithms
    - Itemset mining
    - ....
Spark Components

- GraphX
  - A graph processing library
  - Provides many algorithms for manipulating graphs
    - Subgraph searching
    - PageRank
    - ....

Spark Schedulers

- Spark can exploit many schedulers to execute its applications
  - Hadoop YARN
    - Standard scheduler of Hadoop
  - Mesos cluster
    - Another popular scheduler
  - Standalone Spark Scheduler
    - A simple cluster scheduler included in Spark