**MapReduce vs. Spark: Concentration on Spark SQL**

Author: Shraddha Zingade, Unmesh Deodhar

**ABSTRACT:**

Comparing Hadoop Mapreduce and Apache Spark is a difficult task as they have many non-overlapping areas and many similarities.

There are business applications where Hadoop outperforms Spark. Spark has high speed and ease of use while MapReduce can give better results and outperform Spark in many instances.

In the paper, we have compared Map vs Reduce by analyzing of most of the features like ease of use, fault tolerance, compatibility, costs etc is done to understand the difference better.

**INTRODUCTION**

Apache Hadoop is an open source software platform for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware.[1] The functionalities provided by Hadoop are:

1. Data storage
2. Data processing
3. Data access
4. Data governance
5. Security

**HADOOP FUNCTIONALITIES**

**DATA STORAGE**

The Hadoop Distributed File System (HDFS) provides scalable, fault-tolerant, cost-efficient storage.[1] It is mainly designed to store large amount of data reliably. These data sets are streamed to the user applications at high speed. On a large system, multiple servers host the data stored in HDFS and execute user's applications. The advantage of using this approach is that, faster computation is done in less time.

**DATA PROCESSING**

**Yarn:**

Yarn is responsible for resource management, job scheduling and job monitoring for each of the divided individual tasks.

**MapReduce:**

MapReduce is a framework for writing parallel applications, that can process large amounts of structured and unstructured data in HDFS.

**Map and Reduce:** which are discussed further in the paper.

**Spark:**

Spark has capabilities like in-memory data storage, real-time processing of tasks and machine learning features that make the performance of the system faster and more efficient. It also holds intermediate results of the tasks in the memory, rather than copying to disk which reduce the number of I/O calls.

It is a more efficient and less expensive version of MapReduce.

**DATA ACCESS AND ANALYSIS**

Data access to all the files stored in the database can be done using Hive, HBase and Pig.

**Hive:**

Hive is like a traditional database code with SQL access.

This allows anyone with good knowledge of SQL to write Hive Query Language(HQL)and master Hadoop easily.

**HBase:**

HBase is Hadoop Database as it is a NoSQL database running on top of Hadoop Distributed File System(HDFS).

**MAP-REDUCE**

As mentioned before, Map-Reduce is a programming model and implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster.

It is a framework that uses large number of computer called as clusters for processing big data in parallel, on the clusters.

It splits the input data-set into independent chunks of task, which are processed by the map tasks in parallel.

The framework sorts outputs of maps, which is then input to the reduce tasks.

As the name suggests, it involves two main functions:

1. Map function.

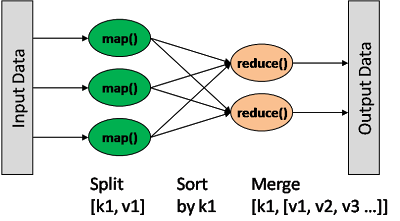
2. Reduce Function.

**1. Map function:**

The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

**2. Reduce Function:**

The Reducer processes the data that comes from the mapper. After executing, it produces a new set of output, that will be stored in HDFS.



After completion of all the tasks, the cluster reduces the data to form a result, and forwards it to the server.

**Advantages**

1. *Tasks independence:* The task are independent so, MapReduce can handle partial failure i.e. the entire nodes can fail and restart.
2. *Division of tasks is implicit:* The end-user programmer only writes map-reduce tasks but does not need to write the division of these tasks.
3. *Low overhead:* Data is written to HDFS once and then read several times. Thus, it leads to low overhead of cashing as the data is simply re-read from HDFS source.
4. *Fault tolerance:* Detecting faults and applying quick, automatic recovery is possible in MapReduce.

**Disadvantages**

1. *Multiple datasets are slow:* As there are no indexes, often the entire dataset gets copied in the process, resulting in bulky data for input.
2. *Single master:* Single master may limit scaling as it has can handle only a certain number of clusters at a time.
3. *Low Speed Access:* Managing job flow is hard as intermediate data needs to be kept on the disk.

**APACHE SPARK**

Spark is a framework used for managing big data processing requirements with a wide variety of data sets.

It is based on MapReduce but extends the MapReduce model to be used for more type of computations including interactive queries, stream processing, graph data, text data and real time data.

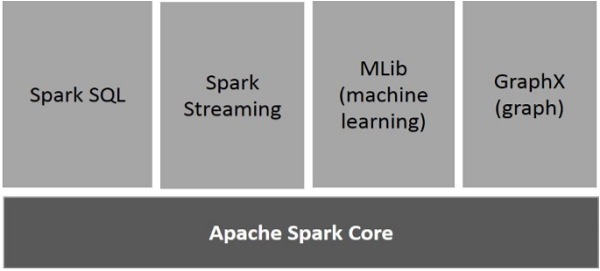
Most important feature of SPARK is **in-memory cluster computing.**

It stores the intermediate processing data in its memory instead of on the disk, thus reducing the number of read-write operations on disk. This increases the speed by 100 times when running in memory and by 10 times while running on disk.

Spark gives **flexibility to the programmers** by supporting applications in multiple languages like Java, Scala, Clojure, R or Python.

**Spark Ecosystem:**

Spark Ecosystem has the libraries and frameworks that allow Spark to achieve faster and more advanced data analytics results compared to Hadoop.



**Apache Spark Core**

Spark Core is the execution engine for spark platform that all other functionality is built upon. It provides In-Memory computing and can reference datasets in external storages.

**Spark SQL**

Spark SQL provides the feature of ETL for different data formats. It allows running SQL like queries on data thus, letting the user access data with less efforts. It allows the user to extract the data using SQL, transforms it and then loads it for any language based querying.

**Spark Streaming**

Spark Streaming can be used for real time streaming of data. It takes data in mini-batches and performs **Resilient Distributed Dataset** transformations on those data sets. It takes the data set and divides it into logical partitions. RDD is a read-only, portioned collection of records. Using this feature, operations can be executed on the real streamed data.

**Spark MLib (Machine Learning)**

MLib has a large number of machine learning frameworks and thus supports numerous machine-learning algorithms like classification, regression, clustering, collaborative filtering, dimensionality reduction and underlying optimization primitives.

**Spark GraphX**

GraphX includes a collection of graph algorithms and builders to compute graph analytics tasks.

Graph is a directed multi-graph with properties at each vertex and edge. It can have operators at the root nodes and operands at the leaf nodes. For the purpose of Graph computation, GraphX has a set of operators (e.g., subgraph, joinVertices, and aggregateMessages). It uses Pregel API for passing data from the vertex to the operators.

Thus, it provides an API for expressing graph computation.

**Advantages:**

1. Spark is 100 times faster than Hadoop because of its in-memory processing feature.

2. Spark uses the same platform for real time processing and batch processing.

3. Developed in Scala i.e. in a functional programming environment that is suitable for distributed systems which allows more functionalities to be executed in a smaller piece of code.

4. Support for streaming, SQL and machine learning operations.

**Disadvantages:**

1. It consumes a lot of memory, and issues around memory consumption and garbage collection are not handled well.

2. Cost-efficient processing of big data can lead to bottleneck because of its "in-memory" processing feature.

3. No file management system, so integration with hadoop or cloud based data platform is required.

**COMPARISON BETWEEN HADOOP MAPREDUCE AND APACHE SPARK**

**Performance:**

Spark is way faster compared to MapReduce due to its in-memory processing feature. Using this feature it delivers real-time analytics for data from machine learning, IOT sensors, security analytics, social media sites, marketing campaigns, log monitoring etc.

Mapreduce uses batch processing as it was not built with a point of view of achieving high speed execution. It was created to gather information from websites but not real-time information.

**Costs**

Spark executed processes in memory and ends up using a lot of memory. Thus, large amount of RAM is required. It keeps the temporary files created for a particular process for seven days as it can use the results for the same process if it is executed again.

MapReduce uses amounts of memory because its processing is disk-based. Thus, faster disks and lot of disk spaces are required.

Also, as the disks can also be distributed over multiple systems, time required to get I/O data from separate systems is higher than getting data from memory.

**Flexibility**

Spark is wellknown for its ease of use. It has APIs for Scala, Java, Python and Spark SQL. Also, it has an interactive mode so a user can get immediate feedback for queries.

MapReduce has no interactive mode. But using Hive and Pig makes working with MapReduce better for users.

**Data Processing**

Spark has graph based computation library called GraphX. This allows the user to view data as collections as well as graphs. User can also join or transform graphs to view the data with a different perspective.

MapReduce operates in a sequential manner.

a) Reading data from a cluster

b) Performing operation on the data

c) Writing results back to the cluster and so on.

Spark does this in single operation or step.

**Fault Tolerance**

Spark uses a Resilient Distributed Datasets that are fault-tolerant collection of elements operated in parallel. RDD can reference a dataset in an external storage system like HDFS, HBase or other data source.

Thus, if any partition in the RDD is lost, it will be recomputed by using the original transformation.

MapReduce uses TaskTracker that provides life to the JobTracker. Thus, if a heartbeat is missed, the JobTracker reschedules all the remaining jobs to another TaskTracker. But, this increases the completion time for operations that have even a single point of failure.

**Security**

MapReduce supports Access Control List as well as Kerberos authentication This ensures that the user have the right permissions to access the data.

Spark uses authentication via shared secret i.e. password authentication. As spark does not have its own database, it uses HBase or HDFS. Thus, it can enjoy Hadoop's ACL as well as Kerberos authentication.

**CONCLUSION**

MapReduce provides features that Spark does not possess. eg. distributed file systems where as Spark provides real-time, in-memory processing of datasets along with machine learning, streaming, SQL and graphX libraries.

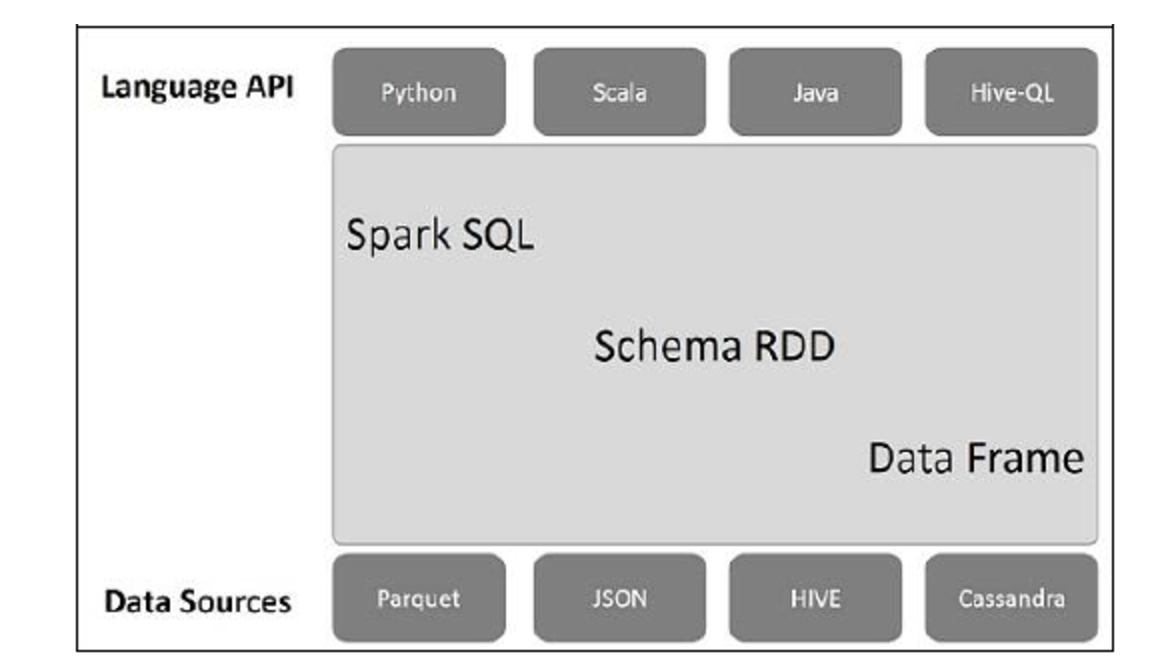
Thus, a scenario where Hadoop MapReduce and Apache Spark work together can be considered to be ideal.

**Presentation Content:**

**Spark SQL – Introduction**

Spark SQL provides the feature of ETL for different data formatsIt allows the user to extract the data using SQL, transforms it and then loads it for any language based querying.

**Spark SQL Architecture**



This architecture has three layers: Language API, Schema RDD, and Data Sources.

**1. Language API** − Spark is compatible with multiple languages. It is supported by APIs of Python, Java, Scala, HiveQL

**2. Schema RDD** − Spark Core is designed with special data structure called RDD. Generally, Spark SQL works on schemas, tables, and records. Therefore, we can use the Schema RDD as temporary table. We can call this Schema RDD as Data Frame.

**3. Data Sources** − Data Sources for Spark SQL are Parquet file, JSON document, HIVE tables and Cassandra database.

**RELATING SPARK SQL TO MYSQL**

**DataFrames (Equivalent to Tables in SQL)**

A DataFrame is a collection of data that is organized into columns. It is equivalent to relational tables in MySQL.

**Features of DataFrame**

1. Processes data in Kilobytes to Petabytes on a single node cluster.

2. Supports multiple data formats (Csv, Cassandra) and storage (HIVE, HDFS, mysql, etc)

3. Provides API for Java, Python, Scala and R Programming.

**SQLContext**

SQLContext is a class that is used for initializion of functionalities of Spark SQL. SparkContext class object (sc) is required for initializing SQLContext class object.

Example

Let us consider an example to read employee records in a JSON file named employee.json.

employee.json − Place this file in the directory where the current scala> pointer is located.

{

{"id" : "1201", "name" : "satish", "age" : "25"}

{"id" : "1202", "name" : "krishna", "age" : "28"}

{"id" : "1203", "name" : "amith", "age" : "39"}

{"id" : "1204", "name" : "javed", "age" : "23"}

{"id" : "1205", "name" : "prudvi", "age" : "23"}

}

To read the JSON Document

First, we generate a DataFrame named (dfs) and then read the JSON document named "employee.json".

scala> val dfs = sqlContext.read.json("employee.json")

Output − The arrtibutes are taken automatically from employee.json.

dfs: org.apache.spark.sql.DataFrame = [age: string, id: string, name: string]

**SHOW DATA**

If you want to see the data in the DataFrame, then use the following command.

scala> dfs.show()

<console>:22, took 0.052610 s

+----+------+--------+

|age | id | name |

+----+------+--------+

| 25 | 1201 | satish |

| 28 | 1202 | krishna|

| 39 | 1203 | amith |

| 23 | 1204 | javed |

| 23 | 1205 | prudvi |

+----+------+--------+

**SELECT OPERATION**

scala> dfs.select("name").show()

<console>:22, took 0.044023 s

+--------+

| name |

+--------+

| satish |

| krishna|

| amith |

| javed |

| prudvi |

+--------+

**WHERE OPERATION**

Use the following command for finding the employees whose age is greater than 23 (age > 23).

scala> dfs.filter(dfs("age") > 23).show()

<console>:22, took 0.078670 s

+----+------+--------+

|age | id | name |

+----+------+--------+

| 25 | 1201 | satish |

| 28 | 1202 | krishna|

| 39 | 1203 | amith |

+----+------+--------+

**GROUP BY OPERATION**

Use groupBy Method

Use the command for counting the number of employees who are of same age.

scala> dfs.groupBy("age").count().show()

<console>:22, took 5.196091 s

+----+-----+

|age |count|

+----+-----+

| 23 | 2 |

| 25 | 1 |

| 28 | 1 |

| 39 | 1 |

+----+-----+

**REFERENCES**

[1] V. Chemitiganti, "What is Apache Hadoop?," in *Business Values of Hadoop*, Hortonworks, 2016. [Online]. Available: https://hortonworks.com/apache/hadoop/. Accessed: Nov. 22, 2016.

[2] M. Rouse, "What is Hadoop? - definition from WhatIs.com," SearchCloudComputing, 2013. [Online]. Available: http://searchcloudcomputing.techtarget.com/definition/Hadoop. Accessed: Nov. 22, 2016.

[3] K. Hess, "Hadoop vs. Spark: The new age of big data," 2016. [Online]. Available: http://www.datamation.com/data-center/hadoop-vs.-spark-the-new-age-of-big-data.html. Accessed: Nov. 22, 2016.

[4] N. \*, "Apache spark vs MapReduce," in *Big Data*, Qubole, 2014. [Online]. Available: https://www.qubole.com/blog/big-data/spark-vs-mapreduce/. Accessed: Nov. 22, 2016.

[5]"Spark SQL quick guide," www.tutorialspoint.com, 2016. [Online]. Available: https://www.tutorialspoint.com/spark\_sql/spark\_sql\_quick\_guide.htm. Accessed: Nov. 29, 2016.