Hadoop/MapReduce (1)

Eslam Montaser Roushdi
Facultad de Informática
Universidad Complutense de Madrid
Grupo G-Tec UCM
www.tecnologiaUCM.es

February, 2014

UNIVERSIDAD COMPLUTENSE MADRID

Our aim

- Study how to handle the Big Data by using Hadoop/MapReduce, in order to analysis this data by using RHadoop.

Outlines:

UNIVERSIDAD COMPLUTENSE MADRID

1) What is Big Data?

Large datasets

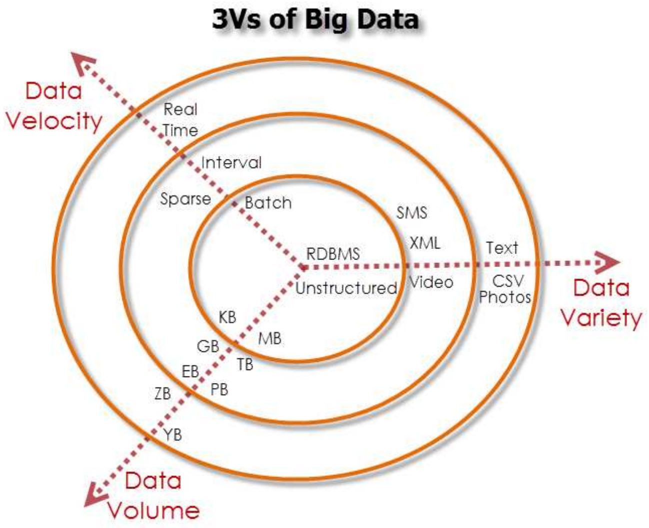
- Big Data has to deal with large and complex datasets that can be structured, semi-structured, or unstructured.
- Five exabytes of data are generated every days!! "Eric Schmidt".
  How to explore, analyze such large datasets?

Facebook

- $>$ 800 million active users in Facebook.
- interacting with $>$ 1 billion objects.
- 2.5 petabytes of userdata per day!

UNIVERSIDAD COMPLUTENSE MADRID

1) What is Big Data?



2) Introduction to Hadoop.

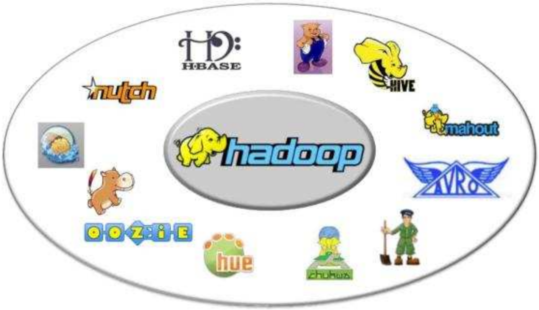
What is Hadoop?

- Hadoop was created by **Doug Cutting** and **Mike Cafarella** in 2005.
- Apache Hadoop is an open source Java framework for processing and querying vast amounts of data on large clusters of commodity hardware.
- Supports applications under a free license.
- Operates on unstructured and structured data.
- A large and active ecosystem.
- Handles thousands of nodes and petabytes of data.
- Hadoop enables scalable, cost-effective, flexible, fault-tolerant solutions.

2) Introduction to Hadoop.

Studying Hadoop components

- Mahout.
- Pig.
- Hive.
- HBase.
- Sqoop.
- ZooKeper.
- Ambari.

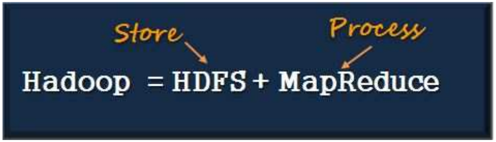


UNIVERSIDAD COMPLUTENSE MADRID

2) Introduction to Hadoop.

Exploring Hadoop features

- **Hadoop Common**: common utilities package.
- **HDFS**: Hadoop Distributed File System with high throughput access to application data.
- **MapReduce**: A software framework for distributed processing of large data sets on computer clusters.
- **Hadoop YARN**: a resource-management platform responsible for managing compute resources in clusters and using them for scheduling the applications of users.



3) Hadoop Distributed File System.

Hadoop Distributed File System (HFDS)

Is a distributed, scalable, and portable file-system written in Java for the Hadoop framework.

The characteristics of HDFS:

- Fault tolerant.
- Runs with commodity hardware.
- Able to handle large datasets.
- Master slave paradigm.
- Write once file access only.

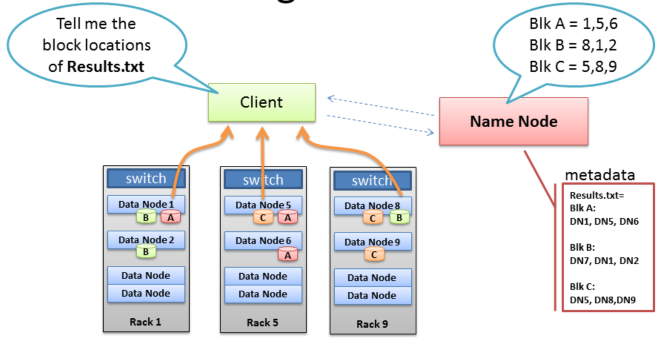
HDFS components

- NameNode.
- DataNode.
- Secondary NameNode.

3) Hadoop Distributed File System.

HDFS architecture

Client reading files from HDFS



- Client receives Data Node list for each block
- Client picks first Data Node for each block
- Client reads blocks sequentially

4) MapReduce and Hadoop MapReduce.

MapReduce

- Is derived from Google MapReduce.
- Is a programming model for processing large datasets distributed on a large cluster.
- MapReduce is the heart of Hadoop.

MapReduce paradigm

- **Map phase**: Once divided, datasets are assigned to the task tracker to perform the Map phase. The data functional operation will be performed over the data, emitting the mapped key and value pairs as the output of the Map phase, (i.e data processing).

- **Reduce phase**: The master node then collects the answers to all the subproblems and combines them in some way to form the output; the answer to the problem it was originally trying to solve, (i.e data collection and digesting).

UNIVERSIDAD COMPLUTENSE MADRID

4) MapReduce and Hadoop MapReduce.

MapReduce Goals

1. **Scalability**to large data volumes:
   - Scan 100 TB on 1 node @ 50 MB/s = 24 days.
   - Scan on 1000-node cluster = 35 minutes.
2. **Cost-efficiency**:
   - Commodity nodes (cheap, but unreliable).
   - Commodity network.
   - Automatic fault-tolerance (fewer admins).
   - Easy to use (fewer programmers).

Challenges

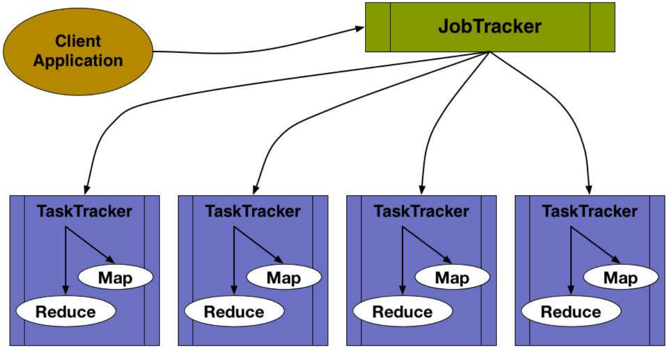
1. **Cheap nodes fail, especially if you have many**:
   - Mean time between failures for 1 node = 3 years.
   - MTBF for 1000 nodes = 1 day.

   Solution: Build fault-tolerance into system.
2. **Programming distributed systems is hard**:
   Solution: Users write data-parallel "map" and "reduce" functions, system handles work distribution and faults.

4) MapReduce and Hadoop MapReduce.

MapReduce components

- **JobTracker**.
- **TaskTracker**.



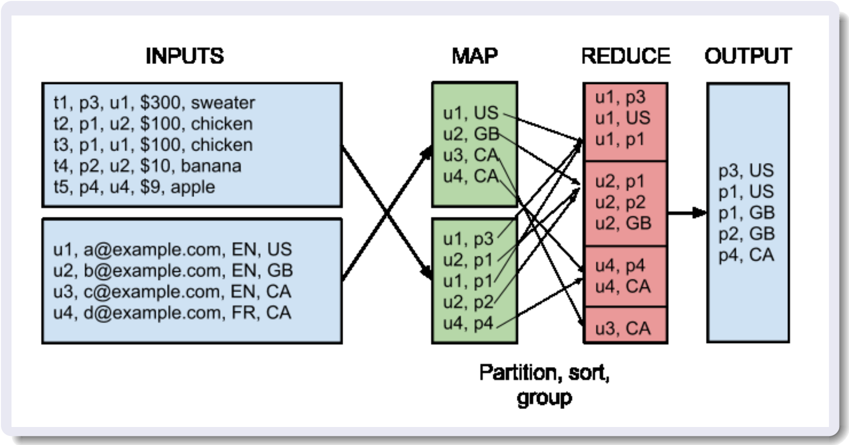
UNIVERSIDAD COMPLUTENSE MADRID

4) MapReduce and Hadoop MapReduce.

The five common steps of parallel computing

1. Preparing the Map() input: This will take the input data row wise and emit key value pairs per rows, or we can explicitly change as per the requirement.
   - Map input: list (k1, v1).
2. Run the user-provided Map() code.
   - Map output: list (k2, v2).
3. Shuffle the Map output to the Reduce processors. Also, shuffle the similar keys (grouping them) and input them to the same reducer.
4. Run the user-provided Reduce() code: This phase will run the custom reducer code designed by developer to run on shuffled data and emit key and value.
   - Reduce input: (k2, list(v2)).
   - Reduce output: (k3, v3).
5. Produce the final output: Finally, the master node collects all reducer output and combines and writes them in a text file.

4) MapReduce and Hadoop MapReduce.



4) MapReduce and Hadoop MapReduce.

Hadoop MapReduce

Is a popular Java framework for easily written applications. It processes vast amounts of data (multiterabyte datasets) in parallel on large clusters (thousands of nodes) of commodity hardware in a reliable and fault-tolerant manner.

Listing Hadoop MapReduce entities

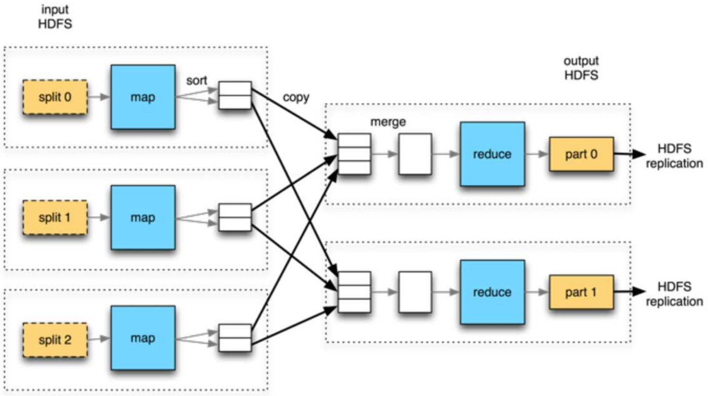
- **Client**: This initializes the job.
- **JobTracker**: This monitors the job.
- **TaskTracker**: This executes the job.
- **HDFS**: This stores the input and output data.

UNIVERSIDAD COMPLUTENSE MADRID

4) MapReduce and Hadoop MapReduce.

Hadoop MapReduce scenario

- The loading of data into HDFS.
- The execution of the Map phase.
- Shuffling and sorting.
- The execution of the Reduce phase.



4) MapReduce and Hadoop MapReduce.

Understanding the Hadoop MapReduce fundamentals

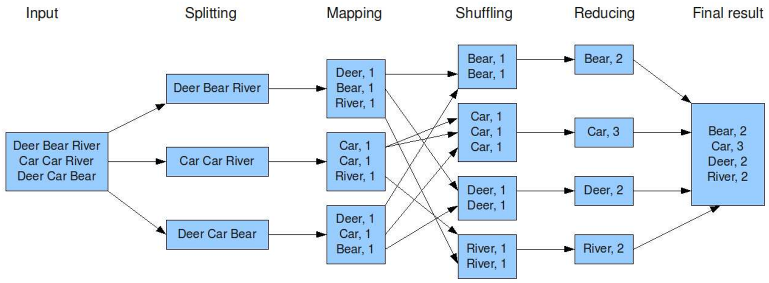
- **MapReduce objects**.
  i) Mapper. ii) Reducer. iii) Driver.

- **MapReduce dataflow**.
  1. Preloading data in HDFS.
  2. Running MapReduce by calling Driver.
  3. Reading of input data by the Mappers, which results in the splitting of the data execution of the Mapper custom logic and the generation of intermediate key-value pairs.
  4. Executing Combiner and the shuffle phase to optimize the overall Hadoop MapReduce process.
  5. Sorting and providing of intermediate key-value pairs to the Reduce phase. The Reduce phase is then executed. Reducers take these partitioned key-value pairs and aggregate them based on Reducer logic.
  6. The final output data is stored at HDFS.

UNIVERSIDAD COMPLUTENSE MADRID

5) Hadoop MapReduce.

Example Word Count



4) MapReduce and Hadoop MapReduce.

Example Word Count (1).

- **Map**:

```
public static class MapClass extends MapReduceBase
implements Mapper {
  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();

  public void map(WritableComparable key, Writable value,
  OutputCollector output, Reporter reporter)
  throws IOException {
  String line = ((Text)value).toString();
  StringTokenizer itr = new StringTokenizer(line);
  while (itr.hasMoreTokens()) {
  word.set(itr.nextToken());
  output.collect(word, one);
  }
  }
}
```

4) MapReduce and Hadoop MapReduce.

Example Word Count (2)

- **Reduce**:

```
public static class Reduce extends MapReduceBase implements Reducer {
public void reduce(WritableComparable key, Iterator
values, OutputCollector output, Reporter reporter)
  throws IOException {
  int sum = 0;
  while (values.hasNext()) {
  sum += ((IntWritable) values.next()).get();
  }
  output.collect(key, new IntWritable(sum));
  }
}
```

UNIVERSIDAD COMPLUTENSE MADRID

4) MapReduce and Hadoop MapReduce.

Example Word Count (3)

- **WordCount**:

```
public static void main(String[ ]args) throws IOException {
  //checking goes here
  JobConf conf = new JobConf();

  conf.setOutputKeyClass(Text.class);
  conf.setOutputValueClass(IntWritable.class);

  conf.setMapperClass(MapClass.class);
  conf.setCombinerClass(Reduce.class);
  conf.setReducerClass(Reduce.class);

  conf.setInputPath(new Path(args[0]));
  conf.setOutputPath(new Path(args[1]));

  JobClient.runJob(conf);
}
```

UNIVERSIDAD COMPLUTENSE MADRID

4) MapReduce and Hadoop MapReduce.

What is MapReduce used for?

1. **At Google**
   - Index construction for Google Search (replaced in 2010 by Caffeine).
   - Article clustering for Google News.
   - Statistical machine translation.
2. **At Yahoo!**
   - "Web map" powering Yahoo! Search.
   - Spam detection for Yahoo! Mail.
3. **At Facebook**
   - Data mining, Web log processing.
   - SearchBox (with Cassandra).
   - Facebook Messages (with HBase).
   - Spam detection and Ad optimization.
4. **In research**
   - Bioinformatics (Maryland).
   - Particle physics (Nebraska).

UNIVERSIDAD COMPLUTENSE MADRID

5) How to link R and Hadoop?

Learning RHadoop

RHadoop is a great open source software framework of **R** for performing data analytics with the Hadoop platform via R functions.

RHadoop with R packages

These three different **R packages** have been designed on Hadoop's two main features HDFS and MapReduce:

rhdfs: This is an R package for providing all Hadoop HDFS access to R. All distributed files can be managed with R functions.

rmr: This is an R package for providing Hadoop MapReduce interfaces to R. With the help of this package, the Mapper and Reducer can easily be developed.

rhbase: This is an R package for handling data at HBase distributed database through R.

5) How to link R and Hadoop?

Learning RHIPE

- **R and Hadoop Integrated Programming Environment (RHIPE)** is a free and open source project. RHIPE is widely used for performing Big Data analysis with **D & R analysis**.
- **D & R analysis** is used to divide huge data, process it in parallel on a distributed network to produce intermediate output, and finally recombine all this intermediate output into a set.
- **RHIPE** is designed to carry out D & R analysis on complex Big Data in R on the Hadoop platform.

RHadoop

R
Hadoop
RHadoop

UNIVERSIDAD COMPLUTENSE MADRID

5) How to link R and Hadoop?

Learning Hadoop streaming

- **Hadoop streaming** is a utility that comes with the Hadoop distribution. This utility allows you to create and run MapReduce jobs with any executable or script as the Mapper and/or Reducer. This is supported by R, Python, Ruby, Bash, Perl, and so on. We will use the R language with a bash script.
- There is one R package named **HadoopStreaming** that has been developed for performing data analysis on Hadoop clusters with the help of R scripts, which is an interface to Hadoop streaming with R.

Finally

Three ways to link R and Hadoop are as follows:

- **RHIPE**.
- **RHadoop**.
- **Hadoop streaming**.

Thanks!!

References

- Glen Mules, Warren Pettit, *Introduction to MapReduce Programming*, September 2013.
- John Maindonald, W. John Braun, *Introduction and Hadoop Overview*, Lab Course: Databases & Cloud Computing, University of Freiburg, 2012.
- Vignesh Prajapati, *Big Data Analytics with R and Hadoop*, November 2013.
- Tom White, *Hadoop: The Definitive Guide*, Second Edition, October 2010.