



**BIG DATA Y MINERÍA DE DATOS GEOESPACIALES**

**3 PROCESAMIENTO DISTRIBUIDO DE DATOS**

MapReduce es tan solo un modelo de programación. Es necesario un *framework* que proporcione la infraestructura de soporte para la ejecución de programas MapReduce. Estos son **Apache Hadoop** y **Apache Spark** (aunque no son los únicos son los más usados y el estándar de uso *de facto*)

**3.2 MRJob**

A. Martín

1



**BIG DATA Y MINERÍA DE DATOS GEOESPACIALES**

**3 PROCESAMIENTO DISTRIBUIDO DE DATOS**

**3.3 El Framework Apache Hadoop**

2

## BIG DATA Y MINERÍA DE DATOS GEOESPACIALES

### 3 PROCESAMIENTO DISTRIBUIDO DE DATOS

- Apache Hadoop es un proyecto de Apache Software Foundation de código abierto que proporciona un entorno a MapReduce para el almacenamiento y procesamiento distribuido de datos.

3.3 Apache Hadoop
NODE 04
A. Martín

3

## BIG DATA Y MINERÍA DE DATOS GEOESPACIALES

### 3 PROCESAMIENTO DISTRIBUIDO DE DATOS

- Creado por Doug Cutting y Mike Cafarella en 2004 a partir de dos importantes *papers* de Google:
 

*J. Dean and, S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", Communications of the ACM, Jan 2008, VI 51 No. 1.*

*S. Ghemawat, H. Gobioff and S.T. Leung, "The Google File System", SOSP'03, October 19–22, 2003, Bolton Landing, New York, USA*

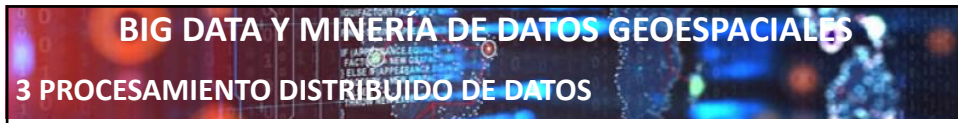
3.3 Apache Hadoop
NODE 04
A. Martín

4

## 5

## 6

- Usado por múltiples organizaciones como Facebook, X, Last.fm, eBay, LinkedIn, Rackspace, Yahoo!, Amazon, etc.
- Modos de funcionamiento:
  1. Standalone: todo en un nodo, para pruebas
  2. Pseudodistribuido: funciona como una instalación completa, pero en un solo nodo, con tantos hilos (*threads*) como núcleos tenga la máquina.
  3. Totalmente distribuido, en un clúster




## BIG DATA Y MINERÍA DE DATOS GEOESPACIALES

### 3 PROCESAMIENTO DISTRIBUIDO DE DATOS

- Hadoop se compone de tres partes básicas:
  - a) El sistema de almacenamiento distribuido: Hadoop Distributed File System (HDFS)
  - a) El motor de ejecución de trabajos MapReduce
  - a) El Ecosistema de Hadoop: Una colección de herramientas en continua expansión y mejora que usan HDFS y MapReduce como núcleo

3.3 Apache Hadoop
NODE 04
A. Martín

7



## BIG DATA Y MINERÍA DE DATOS GEOESPACIALES

### 3 PROCESAMIENTO DISTRIBUIDO DE DATOS

#### 3.3.1 HDFS

- El diseño de HDFS está basado en el diseño del sistema de ficheros de Google (GFS, Google File System)
- El lenguaje de programación usado en su diseño en Java
- Los datos se particionan y son distribuidos por los diferentes nodos. El tamaño por defecto es de 64 Mb. Los datos se distribuyen con no menos de dos réplicas.
- Los datos se pueden tratar en paralelo.
- Existe tolerancia a fallos puesto que existen múltiples copias de los datos en diferentes nodos.
- Está especialmente optimizado para lecturas largas secuenciales en post-proceso, no es bueno para hacer múltiples lecturas cortas.

3.3 Apache Hadoop
NODE 04
A. Martín

8



## BIG DATA Y MINERIA DE DATOS GEOESPACIALES

### 3 PROCESAMIENTO DISTRIBUIDO DE DATOS

#### 3.3.1 HDFS

datafile.csv

64 MB

64 MB

9 MB

137 MB

3.3 Apache Hadoop


A. Martín

9

## BIG DATA Y MINERIA DE DATOS GEOESPACIALES

### 3 PROCESAMIENTO DISTRIBUIDO DE DATOS

#### 3.3.1 HDFS

- Se interactúa con HDFS principalmente a través de la terminal, después de iniciar todos los procesos de java (*daemons*), usando el comando *hdfs* con el siguiente formato:

`$ hdfs dfs -option <arg>` en la versión 2 sigue siendo válido el comando usado en la versión 1: `$ hadoop fs -option <arg>`

- Comandos básicos:

Comando para HDFS	Descripcion	Comando en Linux
hdfs dfs -ls	Lista los ficheros del directorio de trabajo en HDFS del usuario	ls

3.3 Apache Hadoop


A. Martín

10

**BIG DATA Y MINERÍA DE DATOS GEOESPACIALES**

**3 PROCESAMIENTO DISTRIBUIDO DE DATOS**

**3.3.1 HDFS**

Comando para HDFS	Descripción	Comando en Linux
<code>hdfs dfs -mkdir</code>	Crea un directorio en HDFS. El usuario debe tener permisos de escritura en el directorio padre	<code>mkdir</code>
<code>hdfs dfs -put</code>	Copia un fichero del sistema de archivos local a HDFS	<code>cp</code>
<code>hdfs dfs -get</code>	Copia un fichero almacenado en HDFS al sistema de archivos local	<code>cp</code>
<code>hdfs dfs -text</code>	Muestra el contenido de un fichero de texto situado en HDFS	<code>cat</code>
<code>hdfs dfs -rmr</code>	Elimina de forma recursiva un directorio en HDFS	<code>rm -rf</code>
<code>hdfs dfs -appendToFile</code>	Añade datos a un fichero existente en HDFS	

<http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/FileSystemShell.html>

**3.3 Apache Hadoop** A. Martín

11

**BIG DATA Y MINERÍA DE DATOS GEOESPACIALES**

**3 PROCESAMIENTO DISTRIBUIDO DE DATOS**

**3.3.1 HDFS**

La principal diferencia entre Hadoop V.2 y Hadoop V.3 es que la solución de Hadoop V.2 para tolerancia a fallos es proporcionada por la técnica de replicación donde cada bloque de información se copia para crear 2 réplicas. Esto significa que en lugar de almacenar una pieza de información, Hadoop V.2 almacena tres veces más. Esto plantea el problema de espacio en disco.

En Hadoop 3, la tolerancia a fallos la proporciona la codificación de borrado (*erasure coding*). Este método permite recuperar un bloque de información utilizando un bloque de réplica y el llamado “bloque de paridad”. Hadoop 3 crea un bloque de paridad de cada dos bloques de datos (utiliza un método parecido a la compresión de archivos). Esto requiere solo 1,5 veces más espacio en disco. El nivel de tolerancia a fallos en Hadoop 3 sigue siendo el mismo, pero se requiere menos espacio en disco para sus operaciones.

**3.3 Apache Hadoop** A. Martín

12

## BIG DATA Y MINERIA DE DATOS GEOESPACIALES

### 3 PROCESAMIENTO DISTRIBUIDO DE DATOS

#### 3.3.2 MapReduce

- Por encima del sistema de ficheros, Hadoop incorpora un motor de ejecución de trabajos MapReduce. MapReduce permite el procesado a gran escala de conjuntos de datos distribuidos.
- La idea básica es mover la computación cerca de los datos.
- Estructura Master/Slave.
- La conexión entre los nodos mapper y los reducer se hace escribiendo en HDFS.

3.3 Apache Hadoop

NODE 04
A. Martín

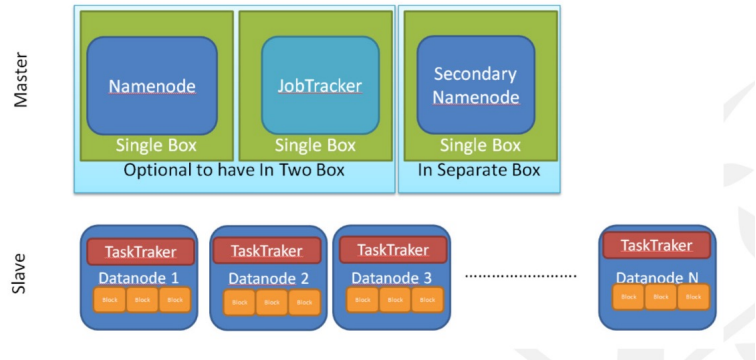
13

## BIG DATA Y MINERIA DE DATOS GEOESPACIALES

### 3 PROCESAMIENTO DISTRIBUIDO DE DATOS

#### 3.3.2 MapReduce

- Primera versión (Hadoop V.1): un único *Job Tracker* que gestiona y asigna tareas y múltiples *Task Trackers* que ejecutan tareas



The diagram illustrates the Hadoop V.1 architecture, divided into Master and Slave nodes.

**Master Node:** Contains three components:
 

- Namenode:** Labeled "Single Box".
- JobTracker:** Labeled "Single Box".
- Secondary Namenode:** Labeled "Single Box" and "In Separate Box".

 A note below the first two boxes states: "Optional to have In Two Box".

**Slave Node:** Contains multiple **TaskTracker** instances, each associated with a **Datanode** (Datanode 1, Datanode 2, Datanode 3, ..., Datanode N). Each TaskTracker box contains three small orange boxes labeled "Block".

3.3 Apache Hadoop

NODE 04
A. Martín

14

# BIG DATA Y MINERÍA DE DATOS GEOESPACIALES

## 3 PROCESAMIENTO DISTRIBUIDO DE DATOS

### 3.3.2 MapReduce

- Primera versión (Hadoop V.1)

**Hadoop Cluster**

```
graph LR; Client[Client] --> JobTracker[Job Tracker]; JobTracker --> TT1[Task Tracker]; JobTracker --> TT2[Task Tracker]; JobTracker --> TT3[Task Tracker]; JobTracker --> TT4[Task Tracker]; NameNode[NameNode] --> TT1; NameNode --> TT2; NameNode --> TT3; NameNode --> TT4; TT1 --> DN1[DataNode]; TT1 --> DN2[DataNode]; TT2 --> DN3[DataNode]; TT2 --> DN4[DataNode]; TT3 --> DN1; TT3 --> DN2; TT4 --> DN3; TT4 --> DN4;
```

The diagram illustrates the Hadoop V1 cluster architecture. A Client (red box) connects to the Job Tracker (blue box). The Job Tracker is connected to four Task Trackers (purple boxes). The NameNode (orange box) is connected to all four Task Trackers. Each Task Tracker is connected to two DataNodes (green boxes). The DataNodes are arranged in two columns, with each Task Tracker connected to one DataNode in each column.

### 3.3 Apache Hadoop

A. Martín

15

# BIG DATA Y MINERÍA DE DATOS GEOESPACIALES

## 3 PROCESAMIENTO DISTRIBUIDO DE DATOS

### 3.3.2 MapReduce

- segunda versión, 2012 (Hadoop V.2): YARN (Yet Another Resource Negotiator):

The diagram illustrates the YARN architecture components and their interactions:

- Application client (client node):** Submits a YARN application to the Resource Manager (1: submit YARN application).
- Resource Manager (resource manager node):** Manages resources and interacts with the NameNode and Node Manager. It allocates resources (3: allocate resources (heartbeat)) to the Node Manager.
- NameNode:** Manages the file system metadata.
- Node Manager (node manager node):** Manages resources on the node. It launches containers (2b: launch) and starts containers (4a: start container) for the Application process.
- DataNode:** Stores data blocks. It also has a Node Manager and Container for the Application process (4b: launch).

Key interactions shown:

- 1: submit YARN application (Application client to Resource Manager)
- 2a: start container (Resource Manager to Node Manager)
- 2b: launch (Node Manager to Container)
- 3: allocate resources (heartbeat) (Resource Manager to Node Manager)
- 4a: start container (Node Manager to Container)
- 4b: launch (Node Manager to Container)

### 3.3 Apache Hadoop

A. Martín

16



## BIG DATA Y MINERÍA DE DATOS GEOESPACIALES

### 3 PROCESAMIENTO DISTRIBUIDO DE DATOS

#### 3.3.2 MapReduce

- segunda versión, 2012 (Hadoop V.2): YARN (Yet Another Resource Negotiator):

The diagram illustrates the Hadoop YARN architecture. On the left, two Client nodes (one red, one purple) send Job Submission (dashed arrows) to a central Resource Manager (yellow box). The Resource Manager manages three Node Managers (yellow boxes). Each Node Manager contains an Application Master (App Mstr, purple oval) and several Containers (pink ovals). The Resource Manager sends Resource Requests (dash-dot arrows) to the App Masters. The App Masters send Node Status (dotted arrows) back to the Resource Manager. Within each Node Manager, the App Master and Containers communicate via MapReduce Status (solid arrows). A legend at the bottom left defines the arrow types: solid for MapReduce Status, dashed for Job Submission, dotted for Node Status, and dash-dot for Resource Request.

**3.3 Apache Hadoop** A. Martín

17

## BIG DATA Y MINERÍA DE DATOS GEOESPACIALES

### 3 PROCESAMIENTO DISTRIBUIDO DE DATOS

#### 3.3.2 MapReduce

- tercera versión, 2017 (Hadoop V.3): YARN V.2

YARN se actualizó a la versión 2 en Hadoop 3. Hay varios cambios significativos que mejoran la usabilidad y la escalabilidad:

- YARN 2 soporta los flujos - grupos lógicos de la aplicación YARN
- La separación entre los procesos de recopilación (escritura de datos) y los procesos de servicio (lectura de datos) mejora la escalabilidad

**3.3 Apache Hadoop** A. Martín

18



## BIG DATA Y MINERIA DE DATOS GEOESPACIALES

### 3 PROCESAMIENTO DISTRIBUIDO DE DATOS

#### 3.3.2 MapReduce. Hadoop Streaming

- Hadoop está basado en Java, si queremos escribir nuestro código en Python, por ejemplo, Hadoop Streaming es la herramienta que traduce el código a ejecutables.
- Dentro de la herramienta Hadoop Streaming es posible usar la clase MRJob, tanto para trabajar con ficheros grabados en directorios locales (únicamente se usa el motor MapReduce) como ficheros en HDFS (se usa el sistema de almacenamiento HDFS y el motor MapReduce).
- Hadoop Streaming se puede usar siempre y cuando todos los procesos Hadoop estén arrancados y ejecutándose correctamente (todos los daemons)

3.3 Apache Hadoop


NODE 04
A. Martín

19



## BIG DATA Y MINERIA DE DATOS GEOESPACIALES

### 3 PROCESAMIENTO DISTRIBUIDO DE DATOS

#### 3.3.3 Ecosistema. En la Web de Hadoop

- **Ambari™**: A web-based tool for provisioning, managing, and monitoring Apache Hadoop clusters which includes support for Hadoop HDFS, Hadoop MapReduce, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig and Sqoop. Ambari also provides a dashboard for viewing cluster health such as heatmaps and ability to view MapReduce, Pig and Hive applications visually along with features to diagnose their performance characteristics in a user-friendly manner.
- **Avro™**: A data serialization system.
- **Cassandra™**: A scalable multi-master database with no single points of failure.
- **Chukwa™**: A data collection system for managing large distributed systems.
- **HBase™**: A scalable, distributed database that supports structured data storage for large tables.
- **Hive™**: A data warehouse infrastructure that provides data summarization and ad hoc querying.
- **Mahout™**: A Scalable machine learning and data mining library.
- **Pig™**: A high-level data-flow language and execution framework for parallel computation.
- **Spark™**: A fast and general compute engine for Hadoop data. Spark provides a simple and expressive programming model that supports a wide range of applications, including ETL, machine learning, stream processing, and graph computation.
- **Tez™**: A generalized data-flow programming framework, built on Hadoop YARN, which provides a powerful and flexible engine to execute an arbitrary DAG of tasks to process data for both batch and interactive use-cases. Tez is being adopted by Hive™, Pig™ and other frameworks in the Hadoop ecosystem, and also by other commercial software (e.g. ETL tools), to replace Hadoop™ MapReduce as the underlying execution engine.
- **ZooKeeper™**: A high-performance coordination service for distributed applications.

3.3 Apache Hadoop


NODE 04
A. Martín

20

**BIG DATA Y MINERIA DE DATOS GEOESPACIALES**

**3 PROCESAMIENTO DISTRIBUIDO DE DATOS**

**3.3.3 Ecosistema. Otros proyectos**

- **Hue:** Interfaz web para simplificar el uso de Hadoop
- **Oozie:** Planificadores de workflows para gestionar trabajos Hadoop
- **Sqoop:** Transferencia eficiente de datos entre Hadoop y bases de datos relacionales.
- **Storm:** Procesamiento de flujo de datos (stream processing).
- **Flume:** Obtención, agregación y movimiento de grandes ficheros de logs a HDFS.

**3.3 Apache Hadoop** **A. Martín**

21

**BIG DATA Y MINERIA DE DATOS GEOESPACIALES**

**3 PROCESAMIENTO DISTRIBUIDO DE DATOS**

**3.3.4 Acceso a HDFS desde python**

- Existen librerías específicas (pydoop, WebHDFS, snakebyte) que permiten conectarse a HDFS desde Python (siempre que todos los procesos Hadoop estén arrancados y ejecutándose correctamente). Estas librerías presentan limitaciones y, debido a las versiones, pueden no funcionar correctamente.
- La interacción se basa en escribir comandos directamente en la ventana de la terminal. Para ello, la manera más sencilla es usar la librería de Python *Subprocess* que viene instalada con la distribución de Python.

**3.3 Apache Hadoop** **A. Martín**

22

## BIG DATA Y MINERIA DE DATOS GEOESPACIALES

### 3 PROCESAMIENTO DISTRIBUIDO DE DATOS

#### 3.3.4 Acceso a HDFS desde python



```

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Wed Sep 6 09:40:02 2023
5
6 @author: angel
7 """
8 import subprocess
9
10 def run_cmd(args_list):
11     """
12     run linux commands
13     """
14     proc = subprocess.Popen(args_list, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
15     s_output, s_err = proc.communicate()
16     return s_output, s_err
17
18 ##Run Hadoop -ls / command in Python
19 (out, err) = run_cmd(['hdfs', 'dfs', '-ls', '/'])
20 print(out)
21
22 ##Run Hadoop -mkdir command in Python
23 (out, err) = run_cmd(['hdfs', 'dfs', '-mkdir', '/input2'])
24
25 ##Run Hadoop -ls / command in Python
26 (out, err) = run_cmd(['hdfs', 'dfs', '-ls', '/'])
27 print(out)
28
29 ##Run Hadoop -put command in Python
30 (out, err) = run_cmd(['hdfs', 'dfs', '-put',
31                      '/home/angel/Documents/BigData/mapreduce/WordFrequency/Book.txt', '/input2'])
32
33 ##Run Hadoop -get command in Python
34 (out, err) = run_cmd(['hdfs', 'dfs', '-get',
35                      '/input2/Book.txt', '/home/angel/Documents/BigData/hadoop/'])
36

```

3.3 Apache Hadoop

A. Martín