

# Introducción a Big Data

Mtro. José Gustavo Fuentes Cabrera



Facultad de Estudios Superiores

# Acatlán

Apuntes de Temas Selectos de Computación

Licenciatura en Actuaría

# Índice

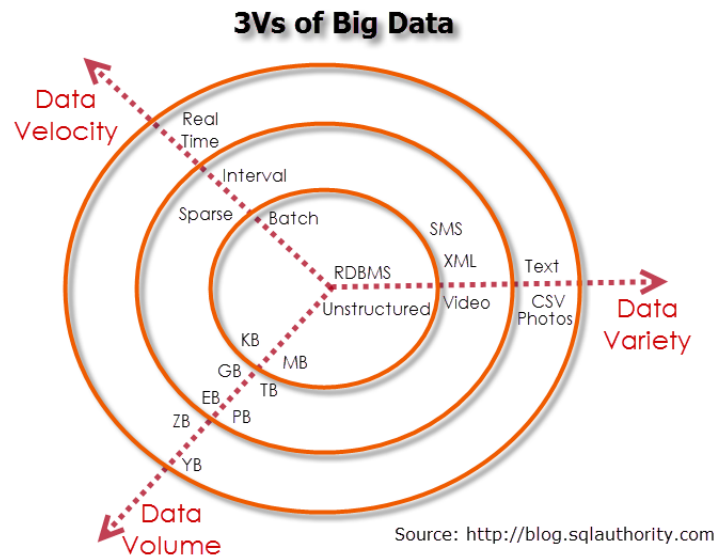
|                                      |    |
|--------------------------------------|----|
| 1. Introducción.                     | 3  |
| 2. El Ecosistema Hadoop.             | 4  |
| 3. Instalación de Hadoop Standalone. | 6  |
| 4. Ingesta de Datos.                 | 9  |
| 5. Explotación de datos con Hive.    | 10 |

## 1. Introducción.

El avance tecnológico vertiginoso ha permitido que mediante la transformación digital de la vida humana se generen enormes cantidades de datos digitales. Si bien en las últimas décadas la importancia del análisis de datos ha gozado de relevancia incremental y se ha generalizado su uso, las necesidades han cambiado conforme los datos han crecido en complejidad con respecto a lo siguiente:

- **Volumen:** El volumen de datos se ha incrementado drásticamente debido a la presencia de máquinas y dispositivos que automáticamente generan grandes cantidades de datos en tiempo real. Esta característica se refiere al tamaño de los datos generados.
- **Variedad** El origen de los datos se caracteriza por su heterogeneidad, provienen en multitud de formatos, plataformas y dispositivos. Datos tradicionales, texto, imágenes, sonidos, videos, información de web, redes sociales, etc.
- **Velocidad** El flujo de datos en la actualidad es masivo y en tiempo real, el reto principal es la recopilación y análisis de los datos en esa escala de tiempo para poder potenciar la toma de decisiones. El tiempo de vida útil de los datos también se ve disminuida por la constante ingestión de nuevos datos.

Lo anterior se conoce como las 3 V's del Big Data, la siguiente figura ilustra los conceptos:



**Figura 1.1:** Las 3 V's del Big Data

En México, la adopción de tecnologías Big Data es limitada todavía, sin embargo, es importante para el futuro profesionista el involucrarse en el uso de estas tecnologías lo más pronto posible pues proporcionan una ventaja competitiva cuya tendencia es prometedora.

## 2. El Ecosistema Hadoop.

De acuerdo con la propia definición del fabricante:

*Hadoop es un entorno de trabajo que permite el procesamiento distribuido de grandes volúmenes de datos a través de clústeres de computadoras usando modelos de programación simples. Está diseñado para escalar desde un único servidor hasta miles de computadoras cada una ofreciendo cómputo y almacenamiento local. En lugar de depender sobre el hardware para alta disponibilidad, está diseñado para detectar y atender fallas en la capa de aplicación, asumiendo proclividad a fallas del cluster de computadoras.*

Hadoop es un proyecto de código abierto desarrollado y mantenido por la fundación Apache, Hadoop es únicamente uno de los proyectos dentro del ecosistema que contiene lo siguiente:

- Hadoop Common: La base del sistema.
- Hadoop Distributed File System: El sistema de archivos distribuido de alto rendimiento.
- Hadoop YARN: Entorno para calendarización de tareas y administración de recursos del cluster.
- Hadoop MapReduce: Un sistema basado en YARN para procesamiento paralelo de grandes conjuntos de datos.

Dentro del ecosistema Hadoop, se han desarrollado proyectos relacionados que complementan diversas tareas necesarias para entornos Big Data, a saber:

- Ambari: Herramienta basada en web para aprovisionamiento, administración y monitoreo de Clusters Apache Hadoop con soporte para HDFS, MapReduce, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig y Sqoop. Provee un tablero de control para visualizar el estado de los clusters así como capacidad para visualización de aplicaciones MapReduce, Pig y Hive.
- Avro: Sistema de serialización de datos.
- Cassandra: Un sistema de bases de datos escalable multi maestro sin puntos únicos de falla.
- Chukwa: Sistema de recolección de datos para administrar grandes sistemas distribuidos.
- HBase: Una base de datos distribuida y escalable que soporta almacenamiento estructurado para tablas grandes.
- Hive: Infraestructura de almacén de datos que provee consultas ad-hoc y resúmenes de datos.
- Mahout: Librería escalable de aprendizaje automático y minería de datos.
- Pig: Lenguaje de flujo de datos de alto nivel para cómputo paralelo.

- **Spark:** Motor rápido de propósito general para datos Hadoop. Provee un modelo de programación simple y expresivo que soporta gran variedad de aplicaciones incluyendo ETL, aprendizaje automático, procesamiento streaming y cómputo para grafos.
- **Tez:**Entorno generalizado de programación para flujo de datos construido sobre Hadoop YARN que provee un poderoso y flexible motor para ejecutar tareas DAG arbitrarias para procesamiento de datos tanto en batch como interactivo. A menudo se perfila como el reemplazo de MapReduce.
- **ZooKeeper:** Servicio de coordinación de alto desempeño para aplicaciones distribuidas.

### 3. Instalación de Hadoop Standalone.

Revisaremos una guía básica para la instalación de un cluster Hadoop de una máquina, se asume el uso de sistemas Linux.

#### 1. Instalar Java.

Hadoop está escrito en Java, por tanto, debemos instalar Java en nuestros equipos primero(ajustar la versión si es que cambia):

```
sudo yum install java-1.8.0-openjdk-devel
ó
sudo apt-get install default-jdk
```

Para sistemas Fedora y Ubuntu respectivamente.

#### 2. Configurar acceso SSH.

Primero generamos una llave pública:

```
ssh-keygen
```

Autorizamos a dicha llave el acceso a nuestra máquina:

```
sudo cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod og-wx ~/.ssh/authorized_keys
```

Posteriormente validamos que la conexión por SSH es exitosa y ce-

rramos la sesión:  
ssh localhost

### 3. Instalación de Hadoop.

Como paso previo, debemos crear la carpeta donde se guardarán los archivos, se recomienda:

```
/usr/local/hadoop/hdfs/data
```

Descargamos ahora los binarios, en este caso estamos usando la versión 2.9.1

```
wget http://www-eu.apache.org/dist/hadoop/common/hadoop-2.
```

```
9.1/hadoop-2.9.1.tar.gz
```

Posteriormente descomprimos el archivo y procederemos a modificar los archivos de configuración.

```
tar -xvf hadoop-2.9.1.tar.gz
```

La lista de archivos a configurar es:

- **hadoop-env.sh**

Solamente necesita cambiarse la línea correspondiente a JAVA HOME

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

- **core-site.xml**

```
1 <configuration>
2   <property>
3     <name>fs.defaultFS</name>
4     <value>hdfs://localhost:9000</value>
5   </property>
6 </configuration>
```

- **mapredsite.xml**

```
1 <configuration>
2   <property>
3     <name>mapreduce.jobtracker.address</name>
4     <value>localhost:54311</value>
5   </property>
6   <property>
7     <name>mapreduce.framework.name</name>
8     <value>yarn</value>
```

```

9   </property>
10  </configuration>

```

#### ■ hdfs-site.xml

```

1  <configuration>
2    <property>
3      <name>dfs.replication</name>
4      <value>1</value>
5    </property>
6    <property>
7      <name>dfs.namenode.name.dir</name>
8      <value>file:///usr/local/hadoop/hdfs/data</value>
9    </property>
10 </configuration>

```

#### ■ yarn-site.xml

```

1  <configuration>
2    <!-- Site specific YARN configuration properties -->
3    <property>
4      <name>yarn.nodemanager.aux-services</name>
5      <value>mapreduce_shuffle</value>
6    </property>
7    <property>
8      <name>yarn.nodemanager.aux-services.mapreduce.
        shuffle.class</name>
9      <value>org.apache.hadoop.mapred.ShuffleHandler</
        value>
10   </property>
11   <property>
12     <name>yarn.resourcemanager.hostname</name>
13     <value>localhost</value>
14   </property>
15 </configuration>

```

#### ■ Master y Slaves

En estos archivos se indican los nodos que fungirán como maestros y esclavos, al ser este ejemplo una instalación Standalone solo agregaremos localhost a cada uno en la primera línea.



Por último, formateamos el cluster y arrancamos los servicios del cluster por medio de los siguientes comandos(dentro de la carpeta sbin):

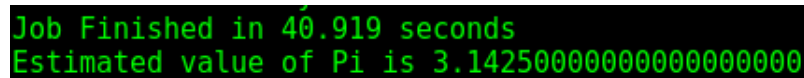
```
./hdfs namenode -format
./start-dfs.sh
./start-yarn.sh
./mr-jobhistory-daemon.sh start historyserver
```

En el navegador, mediante las URL `http://localhost:50070` y `http://localhost:8088` podemos monitorear la salud del cluster.

Hagamos una prueba final para validar que hadoop se encuentra funcionando correctamente consistente en calcular el valor de  $\pi$  mediante MapReduce(dentro de la carpeta bin).

```
./yarn jar ~/server/hadoop-2.9.1/share/hadoop/mapreduce/hadoop-mapreduce-  
pi 16 1000
```

Donde podemos observar el resultado:



```
Job Finished in 40.919 seconds
Estimated value of Pi is 3.14250000000000000000
```

**Figura 3.1:** Resultado de ejecución

## 4. Ingesta de Datos.

Vamos a ejemplificar la ingesta de datos a nuestro cluster mediante linea de comandos usando el comando `hdfs`. El siguiente conjunto de comandos crea la carpeta “rides” en el cluster y carga todos los archivos de la carpeta local `~/Descargas`.

```
./hdfs dfs -mkdir /rides
./hdfs dfs -put ~/Descargas/chicago/* /rides
```

Los archivos pueden encontrarse en la dirección <https://www.kaggle.com/chicago/chicago-taxi-rides-2016/downloads/chicago-taxi-rides-2016.zip>

## 5. Explotación de datos con Hive.

Una vez que nuestro cluster está configurado, funcionando y con datos ingestados, usaremos Hive para realizar consultas (los detalles de la instalación se dejan al lector). Hive es una herramienta de datawarehousing creada por Facebook, puede ser referida como un plugin SQL sobre hadoop. Las consultas se realizan por medio de un lenguaje de consultas similar a SQL conocido como HiveQL. Mediante HiveQL las consultas son convertidas a jobs de MapReduce o de Apache Spark. Veamos un ejemplo simple, cargaremos una de las tablas, por ejemplo, cargaremos la vista correspondiente a los meses de enero, febrero y marzo a una tabla en Hive, para ello creamos primero la estructura como en cualquier manejador de bases de datos:

```
1 CREATE EXTERNAL TABLE IF NOT EXISTS chicago(  
2 taxi_id string,  
3 trip_start_timestamp string,  
4 trip_end_timestamp string,  
5 trip_seconds string,  
6 trip_miles string,  
7 pickup_census_tract string,  
8 dropoff_census_tract string,  
9 pickup_community_area string,  
10 dropoff_community_area string,  
11 fare string,  
12 tips string,  
13 tolls string,  
14 extras string,  
15 trip_total string,  
16 payment_type string,  
17 company string,  
18 pickup_latitude string,  
19 pickup_longitude string,  
20 dropoff_latitude string,  
21 dropoff_longitude string)  
22 ROW FORMAT DELIMITED
```

```
23 FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n'
24 ;
```

Una vez creada la estructura, procedemos a cargar los datos directo del HDFS:

```
LOAD DATA INPATH '/hdfs/rides/chicago_taxi_trips_2016_01.csv'
INTO TABLE chicago;
LOAD DATA INPATH '/hdfs/rides/chicago_taxi_trips_2016_02.csv'
INTO TABLE chicago;
LOAD DATA INPATH '/hdfs/rides/chicago_taxi_trips_2016_03.csv'
INTO TABLE chicago;
```

Corremos un conteo básico para probar nuestra nueva tabla:

```
hive> select count(*) from chicago;
Query ID = jose_20180710162051_0c278a51-bc09-45c3-8b74-c1546459e773
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1531255902309_0001, Tracking URL = http://localhost:8088/proxy/application_1531255902309_0001/
Kill Command = /home/jose/server/hadoop-2.9.1/bin/mapred job -kill job_1531255902309_0001
Hadoop job information for Stage-1: number of mappers: 2; number of reducers: 1
2018-07-10 16:21:02,191 Stage-1 map = 0%, reduce = 0%
2018-07-10 16:21:17,086 Stage-1 map = 50%, reduce = 0%, Cumulative CPU 7.5 sec
2018-07-10 16:21:19,244 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 15.9 sec
2018-07-10 16:21:24,518 Stage-1 map = 100%, reduce = 100%, Cumulative CPU 20.2 sec
MapReduce Total cumulative CPU time: 20 seconds 200 msec
Ended Job = job_1531255902309_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2 Reduce: 1 Cumulative CPU: 20.2 sec HDFS Read: 588258438 HDFS Write: 107 SUCCESS
Total MapReduce CPU Time Spent: 20 seconds 200 msec
OK
5432108
Time taken: 36.959 seconds, Fetched: 1 row(s)
```

Figura 5.1: Resultado de conteo