

Índice

1.0	Introducción.....	3
1.1	Objetivos del proyecto.....	3
1.1.1	Estudio Teórico.....	3
2.0	Big Data.....	3
2.1	las 5 V.....	3
2.2	Tipos de Información.....	4
2.3	Arquitectura	4
2.3.1	Recolección de datos:.....	5
2.3.2	Almacenamiento.....	5
2.3.3	Procesamiento y análisis.....	5
2.3.4	Visualización.....	6
3.0	Hadoop 2.0.6.....	6
3.1	MapReduce.....	6
3.1.2	concepto.....	6
3.1.3	Función Map.....	6
3.1.4	Función Reduce.....	6
3.1.5	Ejemplo.....	7
3.2	HDFS.....	7
3.2.1	NameNode.....	7
3.2.2	DataNode.....	7
3.2.3	Ejemplo.....	8
4.0	Herramientas de hadoop.....	8
4.1	Apache Avro.....	8
4.2	ZooKeeper.....	8
4.3	SOLR.....	9
4.4	Chukwa.....	10
4.5	FLUME.....	10
4.6	Hive.....	11
4.7	MAHOUT.....	11
4.8	OOZIE.....	11
4.9	PIG.....	12
4.10	HUE.....	12
4.11	Sqoop.....	12
4.12	UIMA.....	13
5.0	Distribuciones Hadoop.....	13
5.1	Amazon EMR.....	13
5.2	Cloudera.....	13
5.3	HORTONWORKS.....	14
5.4	IBM InfoSphere BigInsights.....	14
5.6	MapR Technologies.....	14
5.7	Pivotal Software.....	14
6.0	Instalación Hadoop 2.0.6 en Debian.....	14
6.1	Ejecuciones.....	14
6.2	Ejemplos.....	15
6.2.1	Ejemplo en modo Standalone.....	15
6.2.2	Ejemplo en modo servidor-local.....	15
6.3.3	Ejemplo en modo distribuido.....	19
7.0	Hortonworks.....	19
7.1	Ejemplo de Pig (Hortonworks).....	20
7.2	Hadoop-Openstack.....	22

1.0 Introducción

Esta memoria es el resultado del Proyecto de Final de los estudios de Grado Superior de Administración de Sistema Informáticos En Red, impartidos por el I.E.S Gonzalo Nazareno del alumno Sergio Marchena Quirós.

Debido a la envergadura del proyecto He tenido que resaltar tres puntos donde centraremos nuestros estudios:

- Estudio del Big Data
- Estudio MapReduce
- Implementación de Hadoop con MapReduce

1.1 Objetivos del proyecto

Al inicio del proyecto se definieron dos objetivos fundamentales:

1. Estudio teórico Big Data
2. Hadoop.

1.1.1 Estudio Teórico

- Puesta al día de Big Data: definición, motivos de su aparición y evolución.
- Definición de los distintos casos de uso en los que Big Data tiene influencia actualmente y en los que la tendrá en un futuro.
- Estudio y comparación teórica de los distintos paradigmas Big Data así como sus distintas arquitecturas de software y hardware.

2.0 Big Data

En los últimos años la manera de interactuar los usuarios con la tecnología da pie a grandes cantidades de datos, generados por el uso de móviles, redes sociales, blogs... Pero no solo la sociedad, campos como la medicina, ciencia, economía tratan cada vez mas con grandes cantidades de datos.

Big Data es el sector de las tecnologías de la información y la comunicación (TIC) que se preocupa de como almacenar y tratar grandes cantidades de información o conjuntos de datos.

2.1 las 5 V

Es común que cuando se hable de Big Data se haga referencia a grandes cantidades de datos. Pero es más que eso.

- **Volumen:** Un sistema de Big Data es capaz de manejar grandes cantidades de datos. En algunos sistemas de almacenamiento tienen problemas de rendimiento al interactuar con grandes cantidades de datos. Big Data esta pensado para interactuar con grandes cantidades

de datos.

- **Velocidad:** una de las características más importantes en el mundo de la informática es la velocidad de procesamiento. Big Data se centra en procesar grandes cantidades de datos en un tiempo reducido y no solo procesar sino también recibir datos de manera veloz.
- **Variación:** La nueva corriente de almacenamiento de datos no estructurados han cambiado la manera de guardar la información. Big Data es capaz de almacenar y procesar sin tener que estructurar la información.
- **Variabilidad:** Big data debe ser flexible a la hora de adaptarse a nuevos cambios en los formatos de datos, tanto en el almacenamiento, obtención o procesamiento de datos.
- **Valor:** el objetivo final es generar valor de toda la información almacenada a través de distintos procesos de manera eficiente y con el coste más bajo posible.

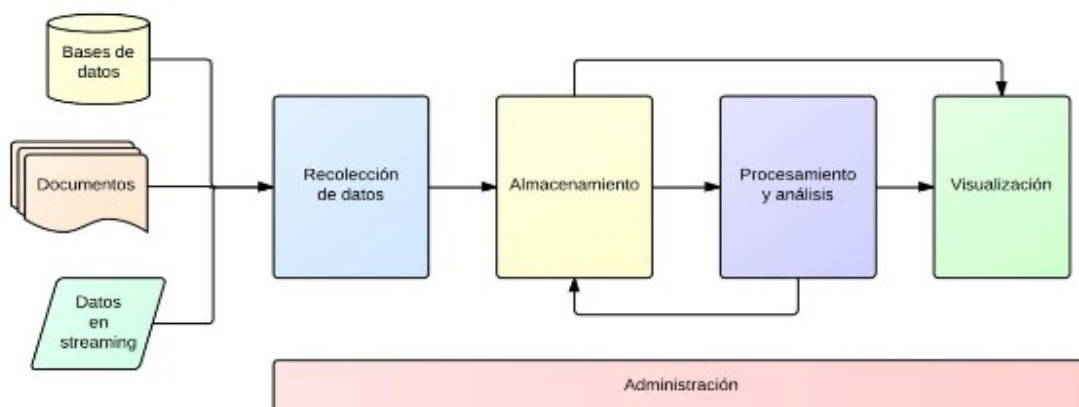
2.2 Tipos de Información

Se puede hablar de tres tipos de datos:

- **Datos Estructurados:** Datos que tienen bien definidos su longitud y su formato, Un ejemplo son las bases de datos relacionales.
- **Datos semi-estructurados:** Datos que no se limitan a campos determinados, poseen sus propios metadatos semi estructurados que describen los objetos y las relaciones entre ellos, ejemplo JSON.
- **Datos Semi-Estructurados:** Datos en el formato tal y como fueron recolectados, carecen de un formato específico. Un ejemplo documentos multimedia.

2.3 Arquitectura

La arquitectura Big Data está compuesta generalmente por cinco capas: recolección de datos, almacenamiento, procesamiento de datos, visualización y administración.



2.3.1 Recolección de datos:

Los fabricamos directa e indirectamente segundo tras segundo.

-Generados por las personas: Enviar correos electrónicos, mensajes por WhatsApp, postear en Facebook...

- **Transacciones de datos:** La facturación, transacciones bancarias..
- **E-marketing y web:** navegación por internet.
- **Machine to Machine (M2M):** medidores, sensores de temperatura, de luz, de altura, de presión, de sonido...
- **Biométrica:** Son el conjunto de datos que provienen de la seguridad, defensa y servicios de inteligencia. Son cantidades de datos generados por lectores biométricos como escáneres de retina, escáneres de huellas digitales.

2.3.2 Almacenamiento

La capa de almacenamiento tiene dos elementos básicos:

- Sistemas de ficheros
- Bases de datos

Hasta hace poco la manera de guardar los datos era en bases de datos relacionales, con el paso del tiempo han surgido nuevas maneras de guardar información como por ejemplo las bases de datos no relacionales. Debido a que Big Data busca la mayor variedad posible los sistemas de fichero han cobrado mayor importancia.

2.3.3 Procesamiento y análisis

Una vez se tienen los datos almacenados, el siguiente paso en un sistema Big Data es explotar la información.

Las herramientas de análisis y procesamiento de información han evolucionado considerablemente, especialmente aquellas que trabajan sobre datos no estructurados. La necesidad de crear nuevas aplicaciones y que éstas ya estén adaptadas a los sistemas de almacenamiento más recientes ha promovido la aparición de nuevos paradigmas. Como por ejemplo MapReduce.

2.3.4 Visualización

Es necesario saber sacar provecho y saber visualizar estos datos, para entenderlos mejor. Para optimizar los beneficios del big data, no podemos limitarnos a llegar a estos datos sin llegar a una correcta comprensión de los mismos. No se trata sólo de contar con la tecnología para obtener y analizar los datos, sino de ser capaces de darle significado.

3.0 Hadoop 2.0.6

Apache Hadoop es un framework de software que soporta aplicaciones distribuidas bajo una licencia libre. Permite a las aplicaciones trabajar con miles de nodos y petabytes de datos.

3.1 MapReduce

MapReduce es un modelo de programación introducido por Google y que en su evolución han participado decenas de colaboradores, apareciendo multitudes de implementaciones. De entre todas esas implementaciones destaca especialmente Hadoop, un proyecto de Apache para proporcionar una base sólida a las arquitecturas y herramientas Big Data.

3.1.2 concepto

No todos los procesos pueden ser abordados desde MapReduce. Concretamente son abordables sólo aquellos que se pueden disgregar en las operaciones de map() y de reduce() y esto es importante a la hora de poder elegir este framework para resolver un problema. Las funciones Map y Reduce están definidas ambas con respecto a datos estructurados en tuplas del tipo (clave, valor).

3.1.3 Función Map

se encarga de dividirlos datos de entrada (uno o varios ficheros de gran tamaño) en varios bloques a ser tratados en paralelo por los nodos de tipo “worker map”. Cada bloque es procesado independientemente del resto por un proceso que ejecuta una función map. Esta función tiene el objetivo de realizar el procesamiento de los datos y dejar los resultados en una lista de pares clave-valor (es decir, se encarga de “mapear” los datos).

$$\text{Map}(k_1, v_1) \rightarrow \text{list}(k_2, v_2)$$

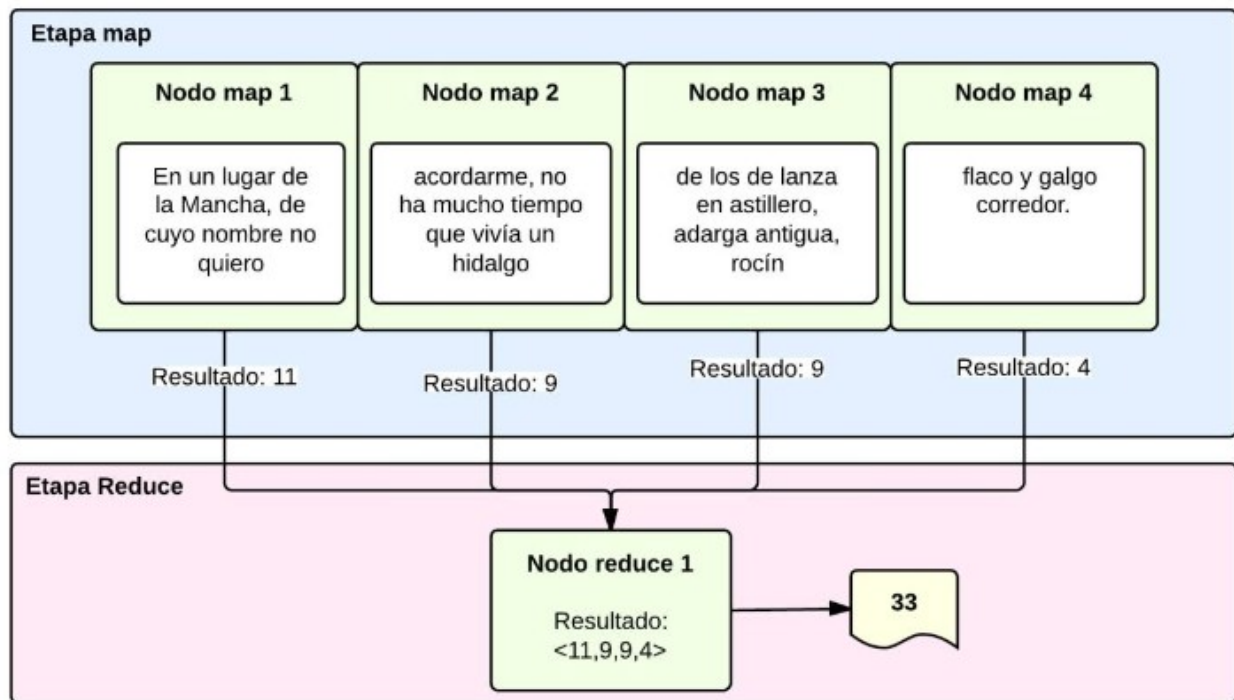
3.1.4 Función Reduce

Los nodos worker de tipo reduce ejecutan una función reduce que recibe como entrada una de las claves generadas en la etapa de map junto con una lista de los valores correspondientes a esa clave. Como salida genera una lista resultante de una función con los valores recibidos. La unión de los resultados puede corresponder a cualquier tipo de función (agregación, suma, máximo, etc.).

$$\text{Reduce}(k_2, \text{list}(v_2)) \rightarrow \text{list}(v_3)$$

3.1.5 Ejemplo

Vamos a ver un pequeño ejemplo con el libro de Don Quijote de la Mancha.



En el proceso map se cuentan las palabras del libro don Quijote, se separan por nodos y se cuentan las palabras, los resultados pasan al proceso reduce que cuenta los resultados y muestra un resultado final.

3.2 HDFS

HDFS es el sistema de almacenamiento, es un sistema de ficheros distribuido. Fue creado a partir del Google File System (GFS). HDFS se encuentra optimizado para grandes flujos y trabajar con ficheros grandes en sus lecturas y escrituras. Su diseño reduce la E/S en la red. La escalabilidad y disponibilidad son otras de sus claves, gracias a la replicación de los datos y tolerancia a los fallos.

3.2.1 NameNode

Sólo hay uno en el cluster. Regula el acceso a los ficheros por parte de los clientes. Mantiene en memoria la metadata del sistema de ficheros y control de los bloques de fichero que tiene cada DataNode.

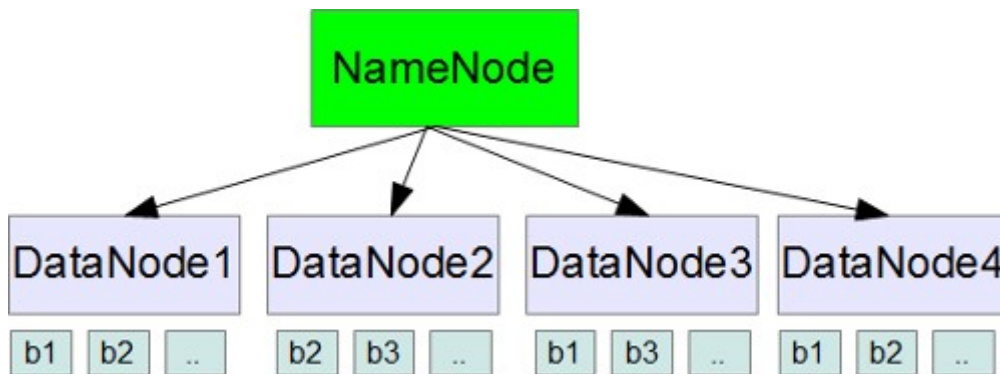
3.2.2 DataNode

Son los responsables de leer y escribir las peticiones de los clientes. Los ficheros están formados por bloques, estos se encuentran replicados en diferentes nodos.

3.2.3 Ejemplo

Vamos a ver una imagen donde se muestra el funcionamiento del sistema de fichero HDFS, podemos observar como es el principal y el que regula los accesos a los datanodes, a su vez los

datanodes leen los fichero por bloques:



4.0 Herramientas de hadoop

En Hadoop tenemos un ecosistema muy diverso, que crece día tras día, por lo que es difícil saber de todos los proyectos que interactúan con Hadoop de alguna forma. A continuación sólo mostraremos los más comunes.

4.1 Apache Avro



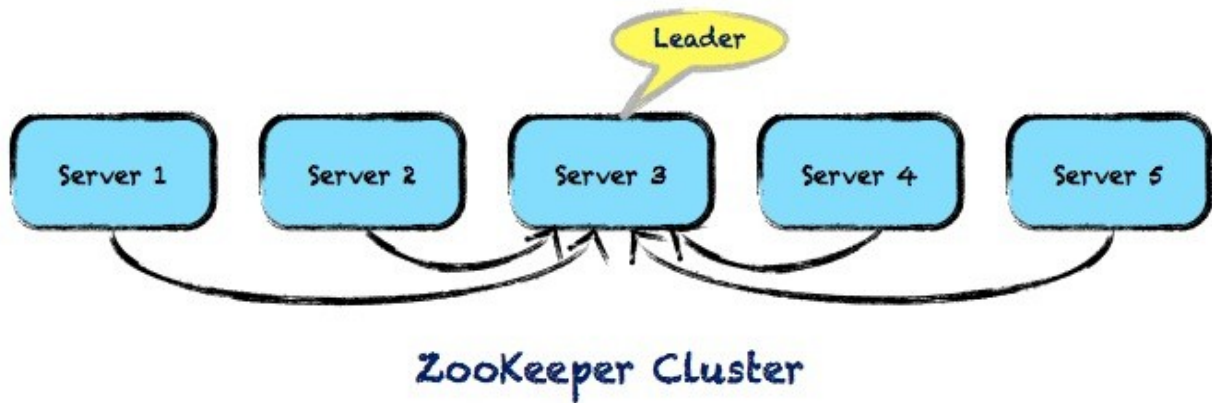
Avro, es un sistema de serialización de datos. En los proyectos en Hadoop, suele haber grandes cantidades de datos, la serialización se usa para procesarlos y almacenar estos datos, de forma que el rendimiento en tiempo sea efectivo. Esta serialización puede ser en texto en plano, JSON, en formato binario. Con Avro podemos almacenar y leer los datos fácilmente desde diferentes lenguajes de programación. Está optimizado para minimizar el espacio en disco necesario para nuestros datos.

4.2 ZooKeeper



Apache ZooKeeper es un proyecto de software de la Apache Software Foundation, que provee un servicio de configuración centralizada y registro de nombres de código abierto para grandes sistemas distribuidos.

La arquitectura de ZooKeeper soporta alta disponibilidad a través de servicios redundantes. Los clientes pueden así preguntar a otro maestro ZooKeeper si el primero falla al responder. Los nodos ZooKeeper guardan sus datos en un espacio de nombres jerárquico, como hace un sistema de archivos. Los clientes pueden leer y escribir desde/a los nodos y de esta forma tienen un servicio de configuración compartido.



www.myjeeva.com



4.3 SOLR

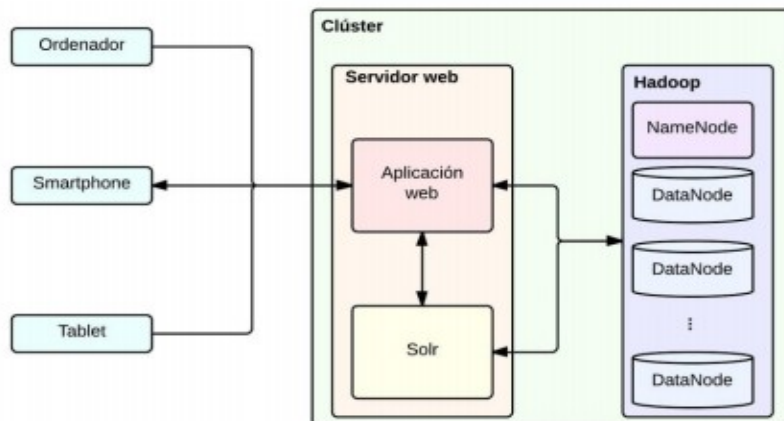


Apache Solr es un motor de búsqueda basado en el Apache Lucene, escrito en Java y que facilita a los programadores el desarrollo de aplicaciones de búsqueda.

Lucene ofrece indexación de información, tecnologías para la búsqueda así como corrección ortográfica, resaltado y análisis de información, entre otras muchas características

Una arquitectura típica de Solr

Cuenta con un servidor web, para que los usuarios puedan interactuar y realizar distintos tipos de búsquedas con conexión directa con Solr y que consulta datos mediante este en Hadoop.



4.4 Chukwa

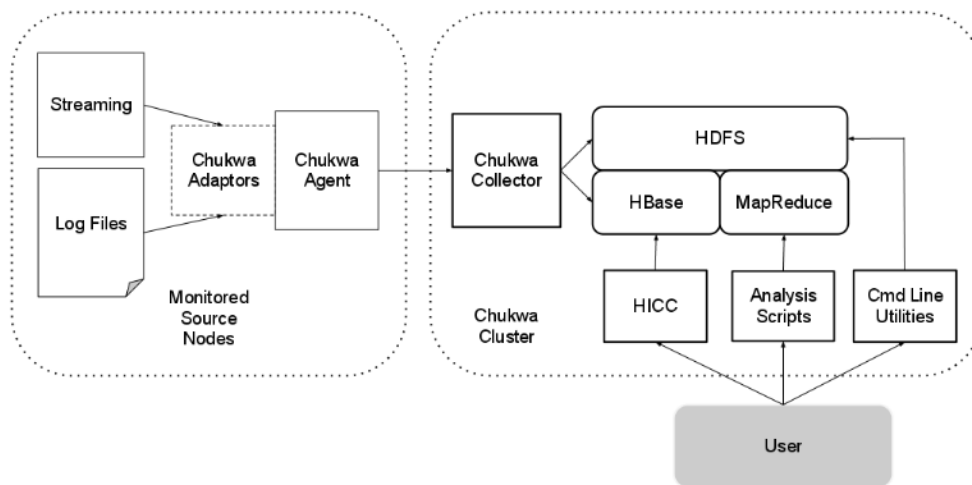


Chukwa es un sistema de captura de datos y framework de análisis que trabaja con Hadoop para procesar y analizar grandes volúmenes de logs. Incluye herramientas para mostrar y monitorizar los datos capturados.

La arquitectura de Chukwa se compone de cuatro componentes:

- Agentes: los procesos que se encargan de capturar datos.

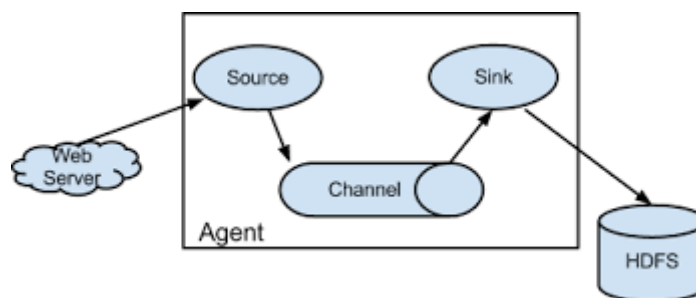
- Colectores: reciben los datos de los agentes y lo escriben en un almacenamiento permanente.
- Trabajos MapReduce para trabajar con los datos.
- HICC: es una interfaz web para visualizar datos.



4.5 FLUME



Apache Flume es un sistema distribuido para capturar de forma eficiente, agregar y mover grandes cantidades de datos log de diferentes orígenes (diferentes servidores) a un repositorio central, simplificando el proceso de recolectar estos datos para almacenarlos en Hadoop y poder analizarlos. Flume y Chukwa son proyectos parecidos, la principal diferencia es que Chukwa está pensado para ser usado en Batch.



4.6 Hive



Es una herramienta para data warehousing que facilita la creación, consulta y administración de grandes volúmenes de datos distribuidos en forma de tablas relacionales. Cuenta con un lenguaje derivado de SQL, llamado Hive QL, que permite realizar las consultar sobre los datos.

A su vez, Hive QL está construido sobre MapReduce, de manera que se aprovecha de las características de éste para trabajar con grandes cantidades de datos almacenados en Hadoop. Esto también provoca que Hive no ofrezca respuestas en tiempo real.

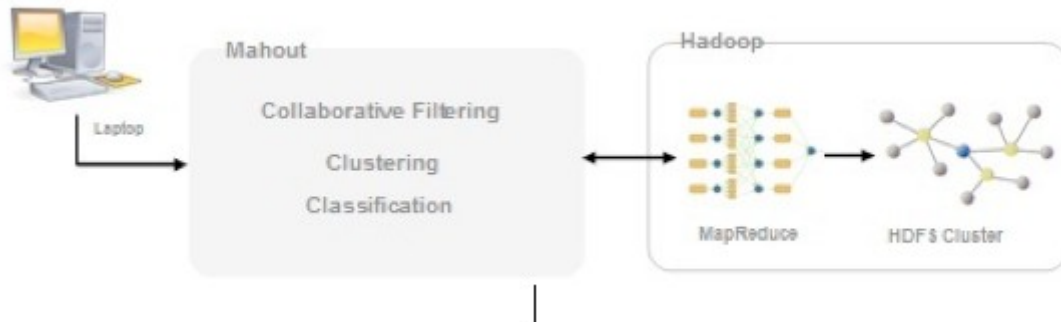
4.7 MAHOUT



Mahout es una librería Java que contiene básicamente funciones de aprendizaje y que está construida sobre MapReduce.

Al usar MapReduce está pensada para trabajar con grandes volúmenes de datos y en sistemas distribuidos.

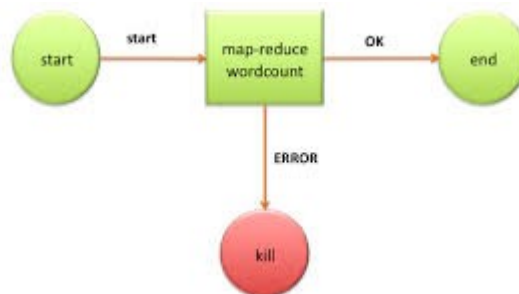
Aquí vemos un pequeño ejemplo:



4.8 OOZIE



Oozie es un planificador de workflows para sistemas que realizan trabajos o procesos Hadoop. Proporciona una interfaz de alto nivel para el usuario no técnico o no experto y que gracias a su abstracción permite a estos usuarios realizar flujos de trabajo complejos.



4.9 PIG



Es una herramienta para analizar grandes volúmenes de datos mediante un lenguaje de alto nivel -PigLatin- que está diseñado para la paralelización del trabajo.

Permite a los usuarios de Hadoop centrarse más en el análisis de los datos y menos en la creación de programas MapReduce.

(Veremos un pequeño ejemplo más adelante).

4.10 HUE



Hue es una herramienta que proporciona a los usuarios y administradores de las distribuciones Hadoop una interfaz web para poder trabajar y administrar las distintas herramientas instaladas.

De esta manera Hue ofrece una serie de editores gráficos, visualización de datos y navegadores para que los usuarios menos técnicos puedan usar Hadoop sin mayores problemas.

The screenshot shows the Hue web interface with a table of query results. The table has columns for 'description', 'salary', and 'salary_c3'. The data is as follows:

	description	salary	salary_c3
0	Dentists, all other specialists	120360	142070 21710
1	Surgeons	191410	206770 15360
2	Oral and maxillofacial surgeons	178440	190420 11980
3	Natural sciences managers	113170	123140 9970
4	Physicians and surgeons, all other	155150	165000 9850
5	Orthodontists	185340	194930 9590
6	Internists, general	167270	176740 9470
7	Political scientists	90050	99320 9270
8	Obstetricians and gynecologists	183600	192780 9180

4.11 Sqoop



Sqoop

Es una herramienta diseñada para transferir de forma eficiente bulk data entre Hadoop y sistemas de almacenamiento con datos estructurados, como bases de datos relacionales.

datos relacionales.

Permite importar tablas individuales o bases de datos enteras a HDFS, genera clases Java que permiten interactuar con los datos importados, además, permite importar de las bases de datos SQL a Hive.



4.12 UIMA



Es un framework para analizar grandes volúmenes de datos no estructurados, como texto, video, datos de audio, etc... y obtener conocimiento que sea relevante para el usuario final.

Por ejemplo a partir de un fichero plano, poder descubrir que entidades son personas, lugares, organizaciones, etc...

5.0 Distribuciones Hadoop

5.1 Amazon EMR

Es un servicio web que facilita el procesamiento rápido y rentable de grandes cantidades de datos, fue uno de los primeros productos comerciales Hadoop en el mercado, y lidera en presencia de mercado global.

Amazon EMR simplifica el procesamiento de big data, al proporcionar un marco de trabajo de Hadoop gestionado que facilita la distribución y el procesamiento de grandes cantidades de datos entre instancias de Amazon EC2 dinámicamente escalables de manera sencilla y rápida.

5.2 Cloudera

Se dedica únicamente a ofrecer soluciones Hadoop para Big Data y es una de las compañías líderes y punteras en este campo. Aparte de las soluciones Big Data, Cloudera también se dedica a ofrecer soporte para sus productos y cuentan con un sistema de entrenamiento y certificaciones profesionales llamado Cloudera University.

5.3 HORTONWORKS

Hortonworks es una compañía de software empresarial. La compañía se centra en el desarrollo y apoyo de Hadoop, un marco que permite el procesamiento distribuido de grandes conjuntos de datos a través de cluster de ordenadores.

5.4 IBM InfoSphere BigInsights

IBM BigInsights extiende los componentes básicos de Hadoop para mejorar la usabilidad. Se añaden características a escala empresarial de IBM para ofrecer un procesamiento y análisis de datos masiva scale-out con una función de la resistencia y tolerancia a fallos. Capacidades de administración y gestión simplificadas, ricas herramientas para desarrolladores y potentes funciones analíticas reducen la complejidad de Hadoop.

5.6 MapR Technologies

MapR es una compañía de software empresarial, que desarrolla y vende software Hadoop-derived. La compañía contribuye a proyectos Apache Hadoop como HBase, Pig, Hive and Zookeeper. Pretende ofrecer una protección completa de datos, sin puntos únicos de fallo, un mejor desempeño,

y facilidad.

5.7 Pivotal Software

Es una compañía de software que está centrada en soluciones Big Data.

Pivotal fue el primer proveedor de EDW en proporcionar un appliance de grado empresarial con todas las características; también fue la primera en lanzar una familia de appliances que integra su Hadoop, EDW y capas de administración de datos en un solo rack.

6.0 Instalación Hadoop 2.0.6 en Debian

Tenemos que descargarnos los paquetes, descomprimos los paquetes, una vez descargados y descomprimidos tenemos que instalar java JDK:

```
aptitude install default-jdk
```

Ahora Tenemos que indicar la variable *JAVA-HOME* que se encuentra dentro de la carpeta hadoop:

```
nano /hadoop-2.6.0/etc/hadoop/hadoop-env.sh
```

```
export JAVA_HOME=/usr/lib/jvm/default-java
```

6.1 Ejecuciones

Existen varios tipos de ejecuciones:

- En modo Standalone: No se necesita configurar nada.
- En modo Servidor - nodo local: Un sistema basado en cliente servidor, pero que se ejecuta en modo local todo.
- En modo distribuido: Infraestructura completa con varios nodos de almacenamiento, ejecución, etc...

6.2 Ejemplos

6.2.1 Ejemplo en modo Standalone.

Vamos a utilizar el modo standalone en el cual no necesitamos configurar nada solamente tener la variable java definida, es un proceso java.

Creamos una carpeta llamada input y copiamos los archivos xml:

```
mkdir input  
cp /hadoop-2.6.0/etc/hadoop/*.xml input/
```

Nos descargamos un ejemplo de internet y ejecutamos la prueba:

```
hadoop-2.6.0/bin/hadoop jar /home/usuario/hadoop-examples-1.1.2.jar wordcount input
```

output

La ejecución crea una carpeta output con los resultados, podemos ver 2 ficheros:

- *part-0000* con la respuesta
- *_SUCCESS*

El resultado muestra las palabras diferentes de los ficheros y las veces que aparecen.

6.2.2 Ejemplo en modo servidor-local.

Para el modo Servidor Necesitamos configurar hadoop de la siguiente manera primero empezamos por instalar jdk:

```
#apt-get install default-jdk
```

Configuramos el acceso SSH:

```
#apt-get install ssh
```

Creamos un grupo:

```
#addgroup hadoop
```

Creamos el usuario y lo metemos dentro del grupo:

```
#adduser --ingroup hadoop hduser
```

Configuramos el acceso ssh al usuario hduser a localhost(conectado como hduser):

```
#ssh-keygen -t rsa -P ""
```

```
#cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

Para probar que se ha configurado correctamente el acceso SSH, nos conectaremos a la máquina local.

```
#ssh localhost
```

salimos del usuario y desactivamos IPv6

```
#nano /etc/sysctl.conf
```

y añadimos esta líneas al final:

```
net.ipv6.conf.all.disable_ipv6 = 1
```

```
net.ipv6.conf.default.disable_ipv6 = 1
```

```
net.ipv6.conf.lo.disable_ipv6 = 1
```

Reiniciar el sistema

```
#reboot
```

Cambiamos los permisos:

```
#chown hduser:hadoop -R /usr/local/hadoop
```

Creamos dos directorios temporales:

```
#mkdir -p /usr/local/hadoop_tmp/hdfs/namenode
```

```
#mkdir -p /usr/local/hadoop_tmp/hdfs/datanode
```

Cambiamos los permisos:

```
#chown hduser:hadoop -R /usr/local/hadoop_tmp/
```

Configuramos Apache Hadoop

Primero configuramos el archivo \$HOME/.bashrc (conectado como hduser):

```
# -- HADOOP ENVIRONMENT VARIABLES START -- #  
  
export JAVA_HOME=/usr/lib/jvm/default-java  
  
export HADOOP_HOME=/usr/local/hadoop  
  
export PATH=$PATH:$HADOOP_HOME/bin  
  
export PATH=$PATH:$HADOOP_HOME/sbin  
  
export HADOOP_MAPRED_HOME=$HADOOP_HOME  
  
export HADOOP_COMMON_HOME=$HADOOP_HOME  
  
export HADOOP_HDFS_HOME=$HADOOP_HOME  
  
export YARN_HOME=$HADOOP_HOME  
  
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native  
  
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
```

Configuramos la variable del fichero: /usr/local/hadoop/etc/hadoop/hadoop-env.sh

```
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
```

Fichero /usr/local/hadoop/etc/hadoop/core-site.xml(dentro de <configuration></configuration>):

```
<property>  
  
<name>fs.default.name</name>
```



```
<value>hdfs://localhost:9000</value>
</property>
```

Fichero /usr/local/hadoop/etc/hadoop/hdfs-site.xml(dentro de <configuration></configuration>):

```
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/usr/local/hadoop_tmp/hdfs/namenode</value>
</property>
<property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/usr/local/hadoop_tmp/hdfs/datanode</value>
</property>
```

Fichero /usr/local/hadoop/etc/hadoop/yarn-site.xml(dentro de <configuration></configuration>):

```
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
```

Hacemos una copia del fichero:

```
#cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template /usr/local/hadoop/etc/hadoop/mapred-site.xml
```

Editamos /usr/local/hadoop/etc/hadoop/mapred-site.xml(dentro de <configuration></configuration>):

```
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
```

La primera vez ejecutamos:

```
#/usr/local/hadoop/bin/hadoop namenode -format
```

Arrancamos el cluster:

```
#/usr/local/hadoop/sbin/start-dfs.sh
#/usr/local/hadoop/sbin/start-yarn.sh
#/usr/local/hadoop/sbin/start-all.sh
```

Ya podemos acceder a la pagina web:

[Http://localhost:8088](http://localhost:8088)

[Http://localhost:50070](http://localhost:50070)

Subimos los ficheros al sistemas de fichero de hadoop, primero creamos una carpeta llamada prueba:

```
# cd /usr/local/hadoop
# bin/hadoop fs -mkdir /prueba
# subimos los ficheros en texto plano:
# bin/hadoop fs -put arthur.txt /prueba/
# bin/hadoop fs -put adavinci.txt /prueba/
```

Ejecutamos el cluster con el siguiente comando:

```
# bin/hadoop jar share/hadoop/tools/lib/hadoop-streaming-2.6.0.jar -file mapper.py -mapper mapper.py -file reducer.py -reducer reducer.py -input /prueba/* -output /prueba/prueba-output
```

Para finalizar se nos crea los mismos dos archivos que en el ejemplo anterior para poder verlo hay que descargarlos del sistema de ficheros con el siguiente comando:

```
# bin/hadoop fs -get /prueba/prueba-output/part-00000
```

6.3.3 Ejemplo en modo distribuido

Lo primero que debemos de hacer es modificar nuestro `/etc/hosts` y establecer el nombre y dirección de los nodos.

Copiamos la clave pública de el nodo principal a los nodos:

```
# ssh-copy-id -i ~/.ssh/id_dsa.pub hadoop@node1
```

Modificamos el archivo `/usr/local/hadoop/etc/hadoop/slaves` y añadimos los esclavos que tengamos.

Copiamos la carpeta `hadoop` con las configuraciones de `hadoop` a todos los nodos esclavos:

```
# scp -r /usr/local/hadoop root@node2:/usr/local
```

Cambiamos configuración(en los esclavos) del fichero `/usr/local/hadoop/etc/hadoop/hdfs-site.xml` y ponemos `NameNode` por `DataNode` así:

```
<name>dfs.datanode.name.dir</name>
```

Ya solo queda arrancar el servidor y los esclavos:

```
# /usr/local/hadoop/bin/start-all.sh → nodo principal
```

```
# /usr/local/hadoop/bin/start-hdfs.sh → todos los esclavos
```

7.0 Hortonworks

Como parte de los ejemplos he utilizado la distribución de Hortonworks que trae dos aplicaciones instaladas, Hive y Pig, Vamos a ver un ejemplo de como utilizar esta distribución y estas dos utilidades.

Empezamos por descargarnos Hortonworks para ello vamos a la pagina oficial y nos da varias posibilidades, utilizaremos la distribución para virtualbox solo tenemos que importarla en virtualbox y acceder a ella mediante la web.

7.1 Ejemplo de Pig (Hortonworks)

Vamos a utilizar el mismo ejemplo que en los anteriores, el ejemplo de contar palabras pero esta vez utilizaremos el lenguaje de alto nivel de pig.

Primero escogemos un archivo `txt` de prueba y lo subimos al sistema de ficheros:

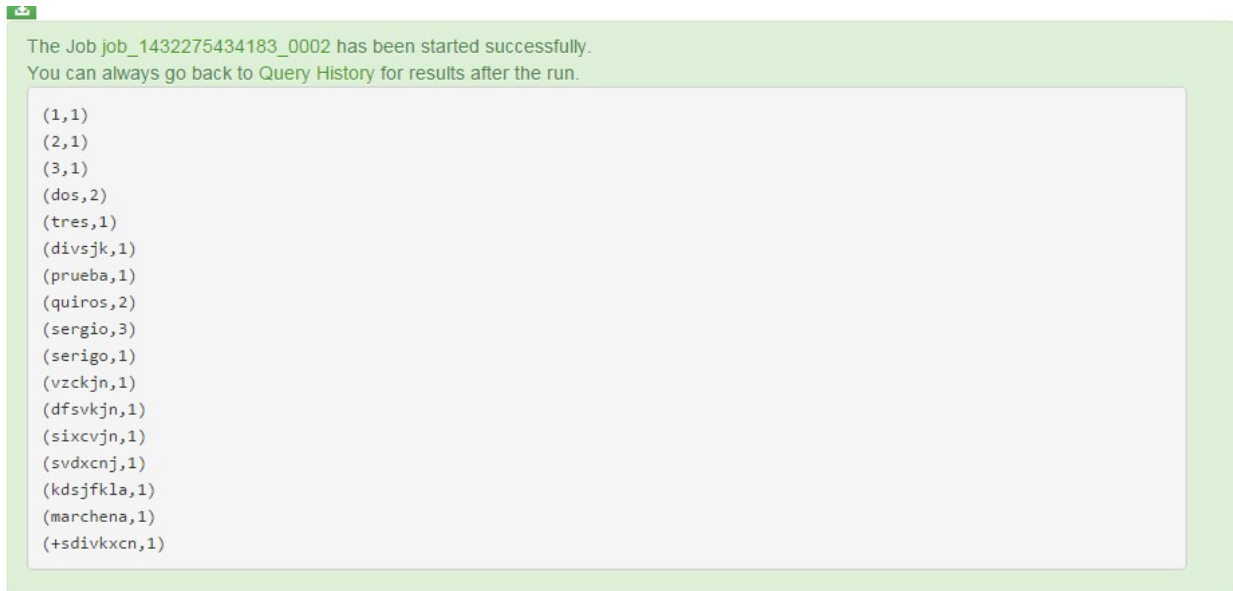
Type	Name	Size	User	Group	Permissions	Date
dir	.		hue	hue	drwxr-xr-x	May 22, 2015 12:22 AM
dir	..		hdfs	hdfs	drwxr-xr-x	April 24, 2015 06:33 AM
dir	.staging		hue	hue	drwx-----	May 22, 2015 12:42 AM
dir	infochimps_dataset_4778_download_16677-csv		hue	hue	drwxr-xr-x	May 21, 2015 11:58 PM
dir	jobsub		hue	hue	drwxrwxrwx	April 24, 2015 06:31 AM
dir	oozie		hue	hue	drwxrwxrwx	April 24, 2015 06:31 AM
file	prueba.txt	153 bytes	hue	hue	-rwxr-xr-x	May 22, 2015 12:22 AM

Ahora vamos a la aplicación Pig (con la carita de un cerdo), pulsamos new script y escribimos lo siguiente:

```
1 lines = LOAD 'prueba.txt' AS (line:chararray);
2 words = FOREACH lines GENERATE FLATTEN(TOKENIZE(line)) as word;
3 grouped = GROUP words BY word;
4 wordcount = FOREACH grouped GENERATE group, COUNT(words);
5 DUMP wordcount;
```

Guardamos (save) y ejecutamos (Execute).

Tenemos que esperar a que el proceso termine o podemos pararlo, una vez terminado te muestra el resultado abajo:



The Job job_1432275434183_0002 has been started successfully.
You can always go back to Query History for results after the run.

```
(1,1)
(2,1)
(3,1)
(dos,2)
(tres,1)
(divsjk,1)
(prueba,1)
(quiros,2)
(sergio,3)
(serigo,1)
(vzckjn,1)
(dfsvkjn,1)
(sixcvjn,1)
(svdxcnj,1)
(kdsjfkla,1)
(marchena,1)
(+sdivkxcn,1)
```

También da la opción de descargarse el fichero.

7.2 Hadoop-Openstack