Hadoop - Tutorial

Hadoop es un marco de código abierto que permite almacenar y procesar grandes datos en un entorno distribuido en clústeres de ordenadores utilizando modelos de programación simple. Está diseñado para aumentar la escala de servidores individuales de miles de máquinas, cada uno de ellos ofrece computación y almacenamiento local.

Este breve tutorial proporciona una introducción rápida a las grandes Datos, algoritmo MapReduce y Hadoop Distributed File System.

Audiencia

Este tutorial se ha elaborado para que los profesionales que aspiran a aprender los conceptos básicos de análisis de datos con Hadoop Marco Hadoop y convertirse en un desarrollador. Los profesionales del Software, análisis profesionales y ETL los desarrolladores son los principales beneficiarios de este curso.

Requisitos previos

Antes de comenzar con este tutorial, vamos a asumir que usted tiene antecedentes de exposición a Core Java, conceptos sobre bases de datos, y cualquiera de los tipos de sistema operativo Linux.

Hadoop- Grandes Datos Generales

"El 90% de los datos de todo el planeta se generó en los últimos años".

Debido a la aparición de nuevas tecnologías, dispositivos y medios de comunicación como los sitios de redes sociales, la cantidad de datos producida por la humanidad está creciendo rápidamente cada año. La cantidad de datos producidos por nosotros desde el principio de los tiempos hasta 2003 fue de 5 millones de gigabytes. Si se acumulan los datos en forma de discos que se pueden llenar todo un campo de fútbol. La misma cantidad se creó en cada dos días en el año 2011, y en cada diez minutos en el año 2013. Esta tasa es aún está creciendo enormemente. A pesar de que toda la información que producen es significativa y puede ser útil cuando se procesan, se descuida.

¿Qué es grande los datos?

Grandes Datos es una colección de conjuntos de datos de gran tamaño que no pueden ser procesados mediante técnicas informáticas tradicionales. No es una técnica o una herramienta, sino que implica muchas áreas de negocios y tecnología.

Lo que viene en los grandes datos?

Grandes los datos implica los datos producidos por los diferentes dispositivos y aplicaciones. A continuación, se presentan algunos de los campos que están bajo el paraguas de Grandes Datos.

- **Los datos de las cajas negras** : es un componente de helicópteros, aviones, y los aviones, etc. que capta las voces de la tripulación de vuelo, las grabaciones de los micrófonos y auriculares, y la información sobre el rendimiento de la aeronave.
- **Los Medios de Comunicación Social** : medios de comunicación social como Facebook y Twitter información y las opiniones de millones de personas en todo el mundo.
- **Bolsa de Valores de datos**: la bolsa de valores de datos contiene información acerca de la 'comprar' y 'vender' las decisiones que se tomen en una proporción de las diferentes empresas hechas por los clientes.
- **Datos de la Red Eléctrica**: La red de suministro de electricidad datos contiene información consumida por un nodo en particular con respecto a una estación base.
- **Datos de transporte** : Transporte incluye un modelo de datos, la capacidad, la distancia y la disponibilidad de un vehículo.
- **Motor de búsqueda de datos**: los motores de búsqueda recuperar gran cantidad de datos de diferentes bases de datos.

Por lo tanto, Grande tiene un enorme volumen de datos, alta velocidad, extensible y variedad de datos. Los datos en el mismo serán de tres tipos.

- **Datos estructurados**: datos relacionales.
- **Semi estructurada de datos** : datos XML.
- **Datos no estructurados**: Word, PDF, Texto, registros medios.

Beneficios de los Grandes Datos

- Con la información que se conserva en la red social como Facebook, las agencias de publicidad están aprendiendo acerca de la respuesta de sus campañas, promociones y otros medios publicitarios.
- Mediante el uso de la información de los medios de comunicación social como las preferencias y percepción producto de sus consumidores, las compañías de productos y organizaciones minoristas están planeando su producción.
- Mediante los datos relativos a la historia médica previa de los pacientes, los hospitales están proporcionando un mejor y rápido servicio.

Las tecnologías de datos grandes

Las tecnologías de datos grandes son importantes a la hora de proporcionar análisis más preciso, lo que puede conducir a más concreta toma de decisiones, consecuencia de una mayor eficiencia operativa, reducir los costes y la reducción de los riesgos para el negocio.

Para aprovechar el poder de los grandes datos, necesitan una infraestructura que puede manejar y procesar grandes volúmenes de datos estructurados y no estructurados en tiempo real y puede proteger la privacidad de los datos y la seguridad.

Existen distintas tecnologías en el mercado de diferentes proveedores, como Amazon, IBM, Microsoft, etc. , para manejar grandes datos. Mientras se mira a las tecnologías que maneje grandes datos, se examinan las siguientes dos clases de tecnología:

Grandes Datos Operacionales

Estos incluyen sistemas como MongoDB que proporcionan las capacidades operacionales en tiempo real y las cargas de trabajo interactivas en las que los datos son principalmente capturan y almacenan.

NoSQL grandes sistemas de datos están diseñados para aprovechar las ventajas de las nuevas arquitecturas cloud computing que han surgido en los últimos diez años para permitir que los cálculos masivos que se ejecute barata y eficiente. Esto hace que grandes cargas de trabajo de datos operacionales mucho más fácil de administrar, más barata y más rápida de aplicar.

NoSQL Algunos sistemas pueden proporcionar información sobre los patrones y las tendencias basadas en datos en tiempo real con el mínimo de código y sin la necesidad de que los datos científicos e infraestructura adicional.

Grandes Datos Analíticos

Estos incluyen sistemas como Massively Parallel Processing (MPP) sistemas de bases de datos y MapReduce que proporcionan capacidades de análisis para análisis retrospectivo y complejo que puede tocar la mayor parte o la totalidad de los datos.

MapReduce proporciona un nuevo método de análisis de los datos que es complementaria a las funciones proporcionadas por SQL, y un sistema basado en MapReduce que pueden ser ampliados en servidores individuales como en miles de alta y baja máquinas.

Estas dos clases de tecnología son complementarias y con frecuencia juntos.

Operativos frente a sistemas analíticos

	Funcionamiento	**Analítica**
Latencia	1 ms - 100 ms	1 min - 100 min
Simultaneidad	1000 - 100,000	1 - 10
Patrón de Acceso	Escribe y Lee	Lee
Consultas	Selectivo	Selectivo
Datos Alcance	Funcionamiento	Retrospectiva
Usuario Final	El Cliente	Los datos científicos
Tecnología	NoSQL	MapReduce, MPP Database

Datos grandes retos

Los principales problemas relacionados con los grandes datos son los siguientes:

- Captura de datos
- Curaduría
- Almacenamiento
- Buscar
- Compartir
- Transferencia
- Análisis
- Presentación

Para cumplir con los retos mencionados, las organizaciones suelen tener la ayuda de los servidores empresariales.

Hadoop- Grandes Soluciones de Datos

Enfoque empresarial tradicional

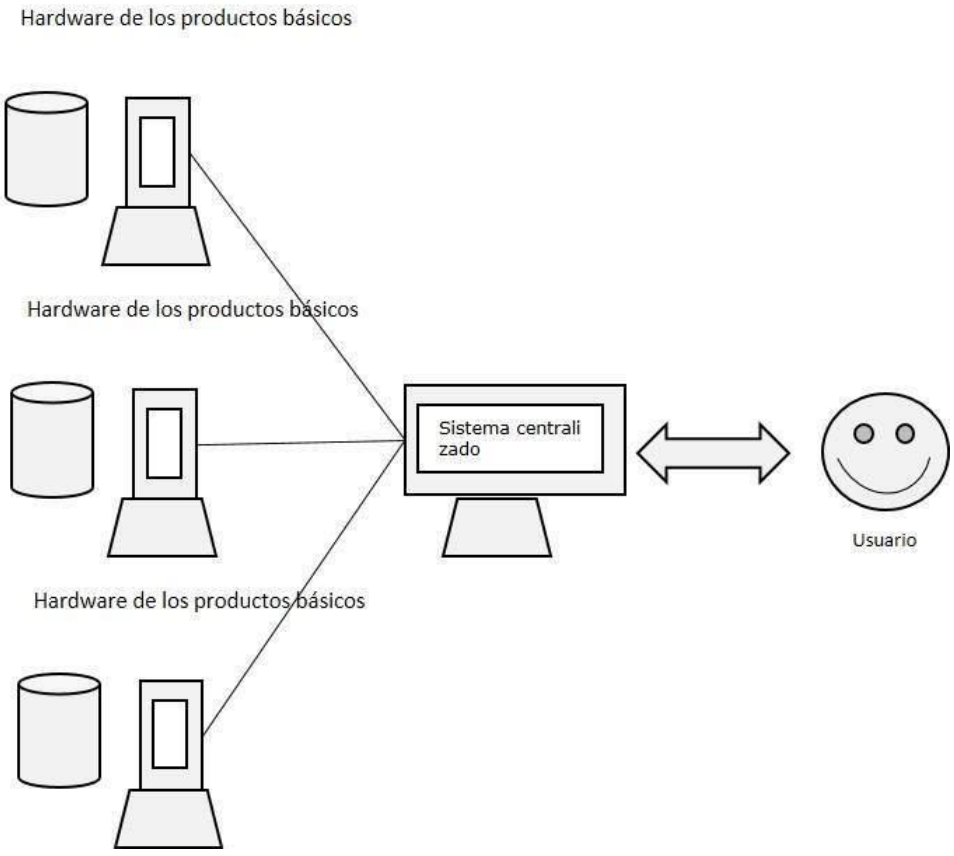
En este enfoque, la empresa tendrá un equipo para almacenar y procesar grandes datos. Para el almacenamiento, los programadores tendrán la ayuda de su elección de proveedores de bases como Oracle, IBM, etc. En este enfoque, el usuario interactúa con la aplicación que, a su vez, controla la parte de almacenamiento de datos y análisis.

Limitación

Este método funciona bien con las aplicaciones de las que menos voluminoso proceso datos que pueden ser atendidas por los servidores de la base estándar, o hasta el límite del procesador que se está procesando los datos. Pero cuando se trata de hacer frente a enormes cantidades de datos escalable, es una frenética tarea de procesar los datos a través de una base de datos única botella.

Solución de Google

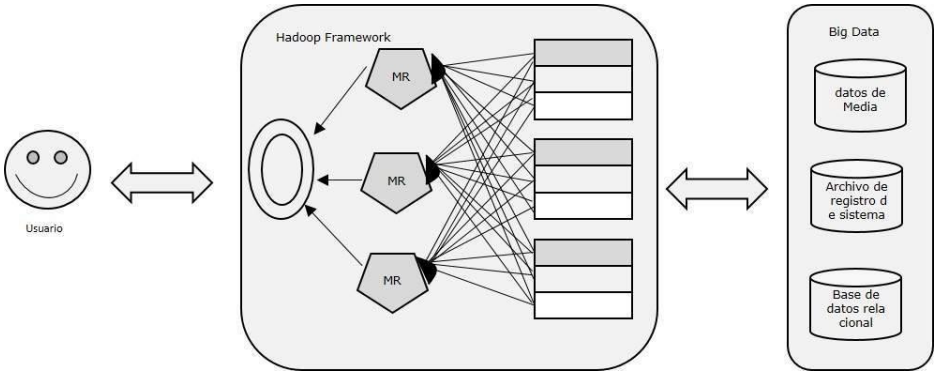
Google soluciona este problema mediante un algoritmo llamado MapReduce. Este algoritmo divide la tarea en partes pequeñas y los asigna a muchos equipos, y recopila los resultados de ellos que si se integra, forma el conjunto de datos de resultado.



Hadoop

Utilizando la solución aportada por Google, Doug y su equipo desarrollaron un proyecto de Código Abierto llamado HADOOP.

Hadoop ejecuta las aplicaciones utilizando el algoritmo MapReduce, donde los datos se procesan en paralelo con los demás. En resumen, Hadoop se utiliza para desarrollar aplicaciones que podrían hacer un análisis estadístico de grandes cantidades de datos.



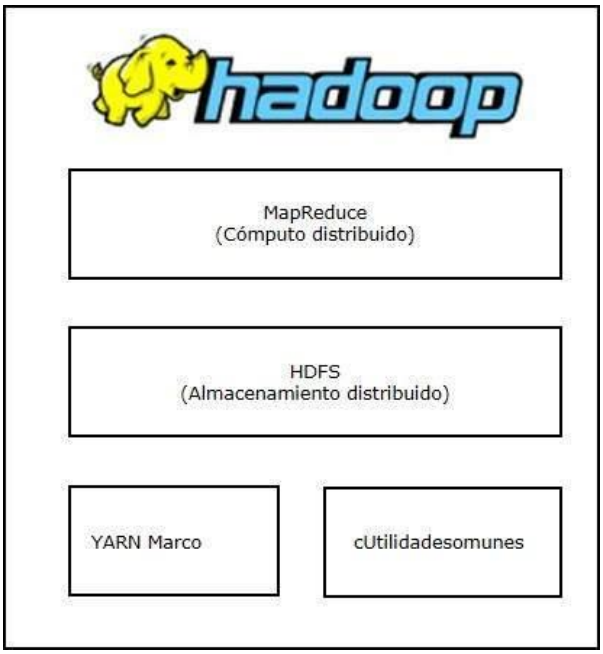
Hadoop- Introducción

Apache Hadoop es un marco de trabajo de código abierto escrito en java que permite procesamiento distribuido de grandes conjuntos de datos a través de clusters de ordenadores mediante sencillos modelos de programación. Hadoop marco la aplicación funciona en un entorno que proporciona almacenamiento distribuido y cálculos en grupos de equipos. Hadoop se ha diseñado para aumentar la escala de un solo servidor a miles de máquinas, cada uno ofreciendo a los cálculos y el almacenamiento.

Hadoop Arquitectura

En su núcleo, Hadoop tiene dos capas principales:

- Informática/Computación capa (MapReduce), y
- Capa de almacenamiento (Hadoop Distributed File System).



MapReduce

MapReduce es un modelo de programación paralela para escribir las aplicaciones distribuidas de Google para garantizar un proceso eficaz de grandes cantidades de datos (datos de varios terabytes de juegos), en grandes grupos (miles de nodos) de hardware básico de manera segura, tolerante a fallos. El programa se ejecuta en MapReduce Hadoop Apache que es un marco de código abierto.

Hadoop Distributed File System

El Hadoop Distributed File System (HDFS) se basa en el Google File System (GFS) y proporciona un sistema de ficheros distribuido que está diseñado para ejecutarse en hardware. Tiene muchas similitudes con los actuales sistemas de archivos distribuidos. Sin embargo, las diferencias con otros sistemas de ficheros distribuidos son importantes. Es muy tolerante a errores y está diseñado para ser instalado en hardware de bajo costo. Proporciona un alto rendimiento en el acceso a datos de aplicaciones y es adecuado para aplicaciones con grandes conjuntos de datos.

Aparte de los mencionados dos componentes básicos, Hadoop marco incluye también los dos módulos siguientes:

- **Hadoop común** : Estas son las bibliotecas de Java y las utilidades requeridas por otros módulos Hadoop.
- **Hadoop HILO** : Este es un marco para la planificación de tareas y administración de recursos de clúster.

¿Cómo Hadoop?

Es bastante caro para construir grandes servidores con configuraciones que manejan grandes transformaciones, pero como una alternativa, usted puede atar muchos ordenadores de consumo con una sola CPU, como un único funcional sistema distribuido y en la práctica, el agrupado las máquinas pueden leer el conjunto de datos en paralelo y proporcionar un rendimiento mucho mayor. Además, es más barato que un servidor de gama alta. Así que este es el primer factor de motivación detrás con Hadoop que agrupados y se extiende a través de bajo costo máquinas.

Hadoop se ejecuta código a través de un cluster de computadoras. Este proceso incluye las siguientes tareas básicas que Hadoop realiza:

- Los datos se dividieron inicialmente en los directorios y archivos. Los archivos se dividen en bloques de tamaño uniforme 128M y 64M (preferiblemente 128M).
- Estos archivos se distribuyen en los distintos nodos del clúster para la transformación posterior.
- HDFS, está en la parte superior del sistema de archivos local, supervisa el proceso.
- Bloques se replican para manejar errores de hardware.
- Comprobar que el código se ejecuta correctamente.
- Realizar la ordenación que se lleva a cabo entre el mapa y reducir las etapas.
- Enviar los datos ordenados de un determinado ordenador.
- Por escrito la depuración de registros para cada trabajo.

Ventajas de Hadoop

- Hadoop marco permite al usuario escribir rápidamente y probar sistemas distribuidos. Es eficiente y automático que distribuye los datos y trabajar a través de las máquinas y a su vez, utiliza el paralelismo de los núcleos de CPU.
- Hadoop no depende de hardware para proporcionar tolerancia a fallos y alta disponibilidad (FTHA), y Hadoop propia biblioteca ha sido diseñado para detectar y controlar errores en el nivel de aplicación.
- Los servidores se pueden añadir o quitar del clúster dinámicamente y Hadoop continúa funcionando sin interrupción.
- Otra gran ventaja de Hadoop es que aparte de ser open source, que es compatible en todas las plataformas ya que está basado en Java.

Hadoop- Configuración Entorno

Hadoop es compatible con GNU/Linux plataforma y sus sabores. Por lo tanto, tenemos que instalar un sistema operativo Linux para configurar Hadoop medio ambiente. En el caso de que usted tenga un SO que no sea Linux, se puede instalar un software en el Virtualbox y Linux dentro del Virtualbox.

De la configuración previa a la instalación

Antes de instalar Hadoop en el entorno de Linux, tenemos que configurar Linux usando ssh (Secure Shell). Siga los pasos que se indican a continuación para configurar el entorno de Linux.

Creación de un usuario

Al principio, se recomienda crear un usuario aparte para Hadoop para aislar Hadoop sistema de archivos del sistema de archivos de Unix. Siga los pasos que se indican a continuación para crear un usuario:

- Abra el usuario root utilizando el comando "su".
- Crear un usuario de la cuenta de root con el comando "useradd usuario".
- Ahora puede abrir una cuenta de usuario existente mediante el comando "su nombre".

Abrir el terminal de Linux y escriba los siguientes comandos para crear un usuario.

```
$ su
   password:
# useradd hadoop
# passwd hadoop
   New passwd:
   Retype new passwd
```

Configuración SSH y la generación de claves

Es necesario realizar una configuración SSH para realizar diferentes operaciones en un clúster como la de iniciar, detener distributed daemon shell las operaciones. Para autenticar usuarios diferentes de Hadoop, es necesaria para proporcionar par de claves pública/privada para un usuario Hadoop y compartirla con los usuarios.

Los siguientes comandos se utilizan para la generación de un par de clave y valor mediante SSH. Copiar las claves públicas forma id_rsa.pub a authorized_keys, y proporcionar el titular, con permisos de lectura y escritura al archivo authorized_keys, respectivamente.

```
$ ssh-keygen -t rsa
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ chmod 0600 ~/.ssh/authorized_keys
```

Instalación de Java

Java es el principal requisito previo para Hadoop. En primer lugar, debe comprobar la existencia de java en el sistema con el comando "java -version". La sintaxis de java versión comando es dada a continuación.

```
$ java -version
```

Si todo está en orden, se le dará el siguiente resultado.

```
java version "1.7.0_71"
Java(TM) SE Runtime Environment (build 1.7.0_71-b13)
Java HotSpot(TM) Client VM (build 25.0-b02, mixed mode)
```

Si java no está instalado en el sistema, a continuación, siga los pasos que se indican a continuación para instalar java.

Paso 1

Descargar Java (JDK <latest version> - X64.tar.gz)visitando el siguiente enlace http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads1880260.html.

A continuación, jdk-7u71-linux-x64.tar.gz se descargará en su sistema.

Paso 2

En general, encontrará el archivo descargado java en carpeta de descargas. Verificar y extraer el jdk-7u71-linux-x64.gz usando los siguientes comandos.

```
$ cd Downloads/
$ ls
jdk-7u71-linux-x64.gz
$ tar zxf jdk-7u71-linux-x64.gz
$ ls
```

```
jdk1.7.0_71   jdk-7u71-linux-x64.gz
```

Paso 3

Para que java disponible para todos los usuarios, tiene que mover a la ubicación "/usr/local/". Abrir root, y escriba los siguientes comandos.

```
$ su
password:
# mv jdk1.7.0_71 /usr/local/
# exit
```

Paso 4

Para la configuración de ruta de acceso y JAVA_HOME variables, agregar los siguientes comandos en el archivo ~/.bashrc

```
export JAVA_HOME=/usr/local/jdk1.7.0_71
export PATH=PATH:$JAVA_HOME/bin
```

Ahora compruebe que el java -version comando desde el terminal, como se explicó anteriormente.

Descargar Hadoop

Descargar y extraer Hadoop 2.4.1 de Apache software foundation usando los siguientes comandos.

```
$ su
password:
# cd /usr/local
# wget http://apache.claz.org/hadoop/common/hadoop-2.4.1/
hadoop-2.4.1.tar.gz
# tar xzf hadoop-2.4.1.tar.gz
# mv hadoop-2.4.1/* to hadoop/
# exit
```

Hadoop Modos de funcionamiento

Una vez que hayas descargado Hadoop, puede utilizar el Hadoop cluster en uno de los tres modos admitidos:

- **Local/Modo autónomo**: Después de descargar Hadoop en su sistema, por defecto, se configura en modo independiente se puede ejecutar como un solo proceso java.
- **Pseudo Modo Distribuido**: es una simulación distribuida en una sola máquina. Cada demonio como Hadoop hdfs, hilos, MapReduce, etc. , se ejecute como un proceso java independiente. Este modo es útil para el desarrollo.
- **Modo Totalmente Distribuida** : Este modo es completamente distribuida con un mínimo de dos o más máquinas como un clúster. Ya hablaremos de este modo en detalle en los próximos capítulos.

Instalar Hadoop en modo autónomo

Vamos a discutir la instalación de **Hadoop 2.4.1** en modo autónomo.

No hay demonios y todo se ejecuta en una sola. Modo autónomo es apto para correr programas MapReduce durante el desarrollo, ya que es fácil de probar y depurar el sistema.

Configuración de Hadoop

Puede establecer las variables de entorno Hadoop anexar los siguientes comandos para**~/.bashrc**archivo.

```
export HADOOP_HOME=/usr/local/hadoop
```

Antes de seguir adelante, usted necesita asegurarse de que Hadoop está trabajando bien. Sólo hay que utilizar el comando siguiente:

```
$ hadoop version
```

Si todo está bien con su configuración, a continuación, usted debe ver el siguiente resultado:

```
Hadoop 2.4.1
Subversion https://svn.apache.org/repos/asf/hadoop/common -r 1529768
Compiled by hortonmu on 2013-10-07T06:28Z
Compiled with protoc 2.5.0
From source with checksum 79e53ce7994d1628b240f09af91e1af4
```

Hadoop significa que tu configuración de modo independiente está trabajando bien. De forma predeterminada, Hadoop está configurado para que se ejecute en un modo distribuido en una sola máquina.

Ejemplo

Veamos un ejemplo sencillo de Hadoop. Hadoop instalación proporciona el siguiente ejemplo MapReduce archivo jar, que proporciona la funcionalidad básica de MapReduce y puede utilizarse para calcular, como valor de PI, recuento de palabras en una determinada lista de archivos, etc.

```
$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-
2.2.0.jar
```

Vamos a tener un directorio de entrada donde vamos a empujar unos cuantos archivos y nuestra obligación es contar el número total de palabras en los archivos. Para calcular el número total de palabras, no es necesario escribir nuestro MapReduce, siempre el archivo .jar contiene la implementación de recuento de palabras. Puede intentar otros ejemplos en los que se usa el mismo archivo .jar, simplemente ejecute los siguientes comandos para comprobar apoya los programas funcionales de MapReduce hadoop mapreduce de ejemplos-2.2.0 .jar.

```
$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-
mapreduceexamples-2.2.0.jar
```

Paso 1

Crear contenido temporal archivos en el directorio de entrada. Puede crear este directorio de entrada en cualquier lugar en el que les gustaría trabajar.

```
$ mkdir input
$ cp $HADOOP_HOME/*.txt input
$ ls -l input
```

Le dará los siguientes archivos en el directorio de entrada:

```
total 24
-rw-r--r-- 1 root root 15164 Feb 21 10:14 LICENSE.txt
-rw-r--r-- 1 root root   101 Feb 21 10:14 NOTICE.txt
-rw-r--r-- 1 root root  1366 Feb 21 10:14 README.txt
```

Estos archivos se han copiado del Hadoop instalación directorio de inicio. Para el experimento, que puede tener diferentes y grandes conjuntos de archivos.

Paso 2

Vamos a comenzar el proceso Hadoop para contar el número total de palabras en todos los archivos disponibles en el directorio de entrada, de la siguiente manera:

```
$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-
mapreduceexamples-2.2.0.jar  wordcount input ouput
```

Paso 3

Paso-2 hará el procesamiento necesario y guardar el resultado de la producción/parte-r00000 archivo, que se puede comprobar mediante:

```
$cat output/*
```

Para ver una lista de todas las palabras junto con su número total disponible en todos los archivos disponibles en el directorio de entrada.

```
"AS      4
"Contribution" 1
"Contributor" 1
"Derivative 1
"Legal 1
"License"      1
"License");     1
"Licensor"      1
"NOTICE”        1
"Not      1
"Object"        1
"Source”        1
"Work”    1
"You"     1
"Your")   1
"[]"      1
```

```
"control"       1
"printed        1
"submitted"     1
(50%)           1
(BIS),          1
(C)             1
(Don't)         1
(ECCN)          1
(INCLUDING      2
(INCLUDING,     2
.............
```

Instalar Hadoop en Pseudo modo distribuido

Siga los pasos que se indican a continuación para instalar Hadoop 2.4.1 en pseudo modo distribuido.

Paso 1: Configuración de Hadoop

Puede establecer las variables de entorno Hadoop anexar los siguientes comandos para**~/.bashrc** archivo.

```
export HADOOP_HOME=/usr/local/hadoop
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=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_INSTALL=$HADOOP_HOME
```

Ahora se aplican todos los cambios en el sistema actual.

```
$ source ~/.bashrc
```

Paso 2: Hadoop Configuración

Usted puede encontrar todos los archivos de configuración Hadoop en la ubicación "$HADOOP_HOME/etc/hadoop". Es necesario realizar cambios en los archivos de configuración según su Hadoop infraestructura.

```
$ Cd $HADOOP_HOME/etc/hadoop
```

Con el fin de desarrollar programas en java Hadoop, tiene que restablecer los java variables de entorno en **hadoop-env.sh** archivo **JAVA_HOME** valor de sustitución con la ubicación de java en su sistema.

```
export JAVA_HOME=/usr/local/jdk1.7.0_71
```

La siguiente es una lista de los archivos que tienes que modificar para configurar Hadoop.

Core-site.xml

El **core-site.xml** contiene información como el número de puerto que se usa para Hadoop ejemplo, memoria asignada para el sistema de archivos, límite de memoria para almacenar los datos, y el tamaño de lectura/escritura.

Abrir el core-site.xml y agregar las siguientes propiedades en entre <configuration>, </configuration> etiquetas.

```
<configuration>

   <property>
      <name>fs.default.name </name>
      <value> hdfs://localhost:9000 </value>
   </property>

</configuration>
```

hdfs-site.xml

La**hdfs-site.xml**contiene información como el valor de los datos de réplica, namenode ruta y datanode las rutas de acceso de los sistemas de archivos locales. Esto significa que el lugar donde se desea almacenar el Hadoop infraestructura.

Supongamos los siguientes datos.

```
dfs.replication (data replication value) = 1
(In the below given path /hadoop/ is the user name.
hadoopinfra/hdfs/namenode is the directory created by hdfs file
system.)
namenode path = //home/hadoop/hadoopinfra/hdfs/namenode
(hadoopinfra/hdfs/datanode is the directory created by hdfs file
system.)
datanode path = //home/hadoop/hadoopinfra/hdfs/datanode
```

Abra este archivo y agregar las siguientes propiedades en entre el <configuration> </configuration> en el archivo.

```
<configuration>
   <property>
      <name>dfs.replication</name>
      <value>1</value>
   </property>

   <property>
      <name>dfs.name.dir</name>
      <value>file:///home/hadoop/hadoopinfra/hdfs/namenode </value>
   </property>

   <property>
      <name>dfs.data.dir</name>
      <value>file:///home/hadoop/hadoopinfra/hdfs/datanode </value>
   </property>

</configuration>
```

Nota: En el archivo anterior, todos los valores de la propiedad son definidos por el usuario y puede realizar cambios en función de su infraestructura Hadoop.

yarn-site.xml

Este archivo se utiliza para configurar yarn en Hadoop. Abra el archivo yarn-site.xml y añadir las siguientes propiedades de entre el <configuration>, </configuration> en el archivo.

```
<configuration>

   <property>
      <name>yarn.nodemanager.aux-services</name>
      <value>mapreduce_shuffle</value>
   </property>

</configuration>
```

mapred-site.xml

Este archivo se utiliza para especificar qué MapReduce framework que estamos usando. De forma predeterminada, Hadoop contiene una plantilla de yarn-site.xml. En primer lugar, es necesario copiar el archivo desde**mapred-site,xml.template** a **mapred-site.xml**con el siguiente comando.

```
$ cp mapred-site.xml.template mapred-site.xml
```

Mapred Abierto de sitio.xml y agregar las siguientes propiedades en entre el <configuración>, < /configuration> etiquetas en este archivo.

```
<configuration>

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

</configuration>
```

Hadoop Instalación Verificación

Los siguientes pasos se utilizan para verificar la instalación Hadoop.

Paso 1: Instalación del nodo Nombre

Configurar el namenode usando el comando "hdfs namenode -format" de la siguiente manera.

```
$ cd ~
$ hdfs namenode -format
```

El resultado esperado es la siguiente.

```
10/24/14 21:30:55 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = localhost/192.168.1.11
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.4.1
...
...
10/24/14 21:30:56 INFO common.Storage: Storage directory
/home/hadoop/hadoopinfra/hdfs/namenode has been successfully
formatted.
10/24/14 21:30:56 INFO namenode.NNStorageRetentionManager: Going to
retain 1 images with txid >= 0
10/24/14 21:30:56 INFO util.ExitUtil: Exiting with status 0
10/24/14 21:30:56 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at localhost/192.168.1.11
************************************************************/
```

Paso 2: Comprobar Hadoop dfs

El siguiente comando se utiliza para iniciar sle. Al ejecutar este comando, se iniciará la Hadoop sistema de archivos.

```
$ start-dfs.sh
```

El resultado esperado es la siguiente:

```
10/24/14 21:37:56
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/hadoop/hadoop
2.4.1/logs/hadoop-hadoop-namenode-localhost.out
localhost: starting datanode, logging to /home/hadoop/hadoop
2.4.1/logs/hadoop-hadoop-datanode-localhost.out
Starting secondary namenodes [0.0.0.0]
```

Paso 3: Comprobación de Script yarn

El siguiente comando se utiliza para iniciar el yarn script. Al ejecutar este comando se inicie el yarn demonios.

```
$ start-yarn.sh
```

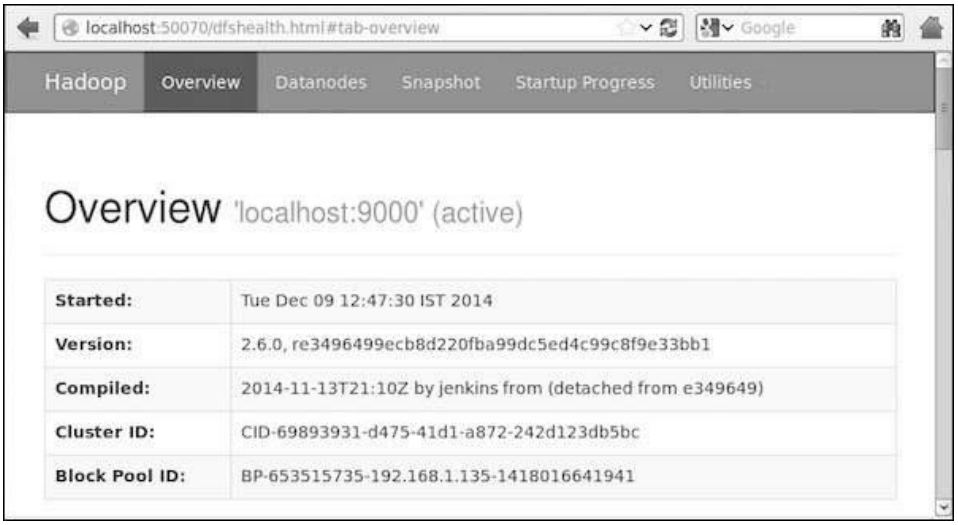
El resultado esperado de la siguiente manera:

```
starting yarn daemons
starting resourcemanager, logging to /home/hadoop/hadoop
2.4.1/logs/yarn-hadoop-resourcemanager-localhost.out
localhost: starting nodemanager, logging to /home/hadoop/hadoop
2.4.1/logs/yarn-hadoop-nodemanager-localhost.out
```

Paso 4: Acceder a Hadoop en el navegador

El número de puerto predeterminado para acceder a Hadoop es 50070. Utilice la siguiente dirección url para obtener Hadoop servicios en el navegador.

http://localhost:50070/



Paso 5: Verifique que todas las solicitudes de clúster

El número de puerto predeterminado para acceder a todas las aplicaciones de clúster es 8088. Utilice la siguiente dirección url para visitar este servicio.

http://localhost:8088/

Hadoop- HDFS Descripción General

Hadoop Sistema de archivos se ha desarrollado utilizando diseño de sistema de archivos distribuidos. Se ejecuta en hardware de productos básicos. A diferencia de otros sistemas distribuidos, HDFS es muy tolerantes y diseñado utilizando hardware de bajo coste.

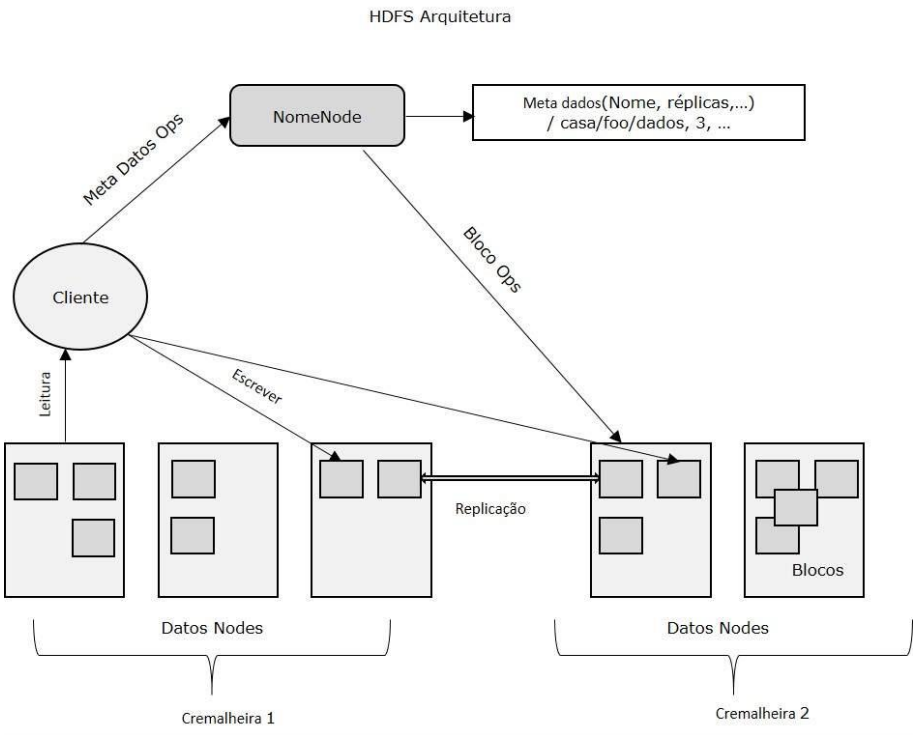
HDFS tiene gran cantidad de datos y proporciona un acceso más fácil. Para almacenar estos datos de gran tamaño, los archivos se almacenan en varias máquinas. Estos archivos se almacenan en forma redundante para rescatar el sistema de posibles pérdidas de datos en caso de fallo. HDFS también permite que las aplicaciones de procesamiento en paralelo.

Características de los HDFS

- Es adecuado para el almacenamiento y procesamiento distribuido.
- Hadoop proporciona una interfaz de comandos para interactuar con HDFS.
- Los servidores de namenode datanode y ayudar a los usuarios a comprobar fácilmente el estado del clúster.
- Streaming el acceso a los datos del sistema de ficheros.
- HDFS proporciona permisos de archivo y la autenticación.

HDFS Arquitectura

A continuación se muestra la arquitectura de un sistema de archivos Hadoop.



HDFS sigue el maestro-esclavo y arquitectura que tiene los siguientes elementos:

Namenode

El namenode es el hardware básico que contiene el sistema operativo GNU/Linux y el software namenode. Es un software que puede ejecutarse en hardware. El sistema que tiene la namenode actúa como el servidor maestro y no las siguientes tareas:

- Administra el espacio de nombres del sistema de archivos.
- Del cliente regula el acceso a los ficheros.
- Además, ejecuta las operaciones del sistema de archivos como el cambio de nombre, cierre y apertura de archivos y directorios.

Datanode

La datanode es un hardware de productos básicos con el sistema operativo GNU/Linux y software datanode. Para cada nodo (Commodity hardware/Sistema) de un clúster, habrá un datanode. Estos nodos gestionar el almacenamiento de datos de su sistema.

- Datanodes realizar operaciones de lectura y escritura de los sistemas de archivos, como por petición del cliente.
- Además, permiten realizar operaciones tales como creación, supresión, con lo que la replicación de acuerdo con las instrucciones del namenode.

Bloque

En general los datos de usuario se almacenan en los archivos de HDFS. El archivo en un sistema de archivos se divide en uno o más segmentos y/o almacenados en los nodos de datos. Estos segmentos se denominan como bloques. En otras palabras, la cantidad

mínima de datos que HDFS puede leer o escribir se llama un bloque. El tamaño de bloque por defecto es de 64 MB, pero puede ser aumentado por la necesidad de cambiar de configuración HDFS.

Objetivos de los HDFS

- **Detección de fallos y recuperación** : Desde los HDFS incluye un gran número de componentes de hardware, fallos de componentes es frecuente. HDFS Por lo tanto debe contar con mecanismos para una rápida y automática detección de fallos y recuperación.
- **Ingentes conjuntos** : HDFS debe tener cientos de nodos por clúster para administrar las aplicaciones de grandes conjuntos de datos.
- **Hardware a una velocidad de transferencia de datos** : una tarea solicitada se puede hacer de una manera eficiente, cuando el cálculo se lleva a cabo cerca de los datos. Especialmente en los casos en que grandes conjuntos de datos se trata, reduce el tráfico de red y aumenta el rendimiento.

Hadoop- HDFS Operaciones

HDFS Inicio

En un principio tienes que formatear el sistema de archivos HDFS namenode, abierto (HDFS server), y ejecute el siguiente comando.

```
$ hadoop namenode -format
```

Después de formatear la HDFS, iniciar el sistema de archivos distribuido. El siguiente comando inicia el namenode, así como los nodos de datos en cluster.

```
$ start-dfs.sh
```

Listado de los archivos en los HDFS

Después de cargar la información en el servidor, podemos encontrar la lista de los archivos de un directorio, el estado de un archivo, utilizando "ls". A continuación se muestra la sintaxis de ls que se puede pasar a un directorio o un nombre de archivo como argumento.

```
$ $HADOOP_HOME/bin/hadoop fs -ls <args>
```

Insertar datos en HDFS

Supongamos que tenemos los datos en el archivo llamado archivo.txt en el sistema local que debe guardarse en el sistema de archivos hdfs. Siga los pasos que se indican a continuación para insertar el archivo requerido en el Hadoop sistema de archivos.

Paso 1

Tiene que crear un directorio de entrada.

```
$ $HADOOP_HOME/bin/hadoop fs -mkdir /user/input
```

Paso 2

Transferir y almacenar un archivo de datos de sistemas locales a la Hadoop sistema de archivos utilizando el comando put.

```
$ $HADOOP_HOME/bin/hadoop fs -put /home/file.txt /user/input
```

Paso 3

Puede comprobar el archivo mediante comando ls.

```
$ $HADOOP_HOME/bin/hadoop fs -ls /user/input
```

Recuperar datos de HDFS

Supongamos que tenemos un archivo llamado into outfile en HDFS. A continuación se ofrece una demostración sencilla para recuperar el archivo necesario de la Hadoop sistema de archivos.

Paso 1

En un primer momento, ver los datos de los HDFS con comando cat.

```
$ $HADOOP_HOME/bin/hadoop fs -cat /user/output/outfile
```

Paso 2

Obtener el archivo de HDFS al sistema de archivos local mediante get.

```
$ $HADOOP_HOME/bin/hadoop fs -get /user/output/ /home/hadoop_tp/
```

Apagar el HDFS

Puede apagar el HDFS, utilizando el siguiente comando.

```
$ stop-dfs.sh
```

Hadoop- Referencia de Comandos

Hay muchos más comandos en **"$HADOOP_HOME/bin/hadoop fs"** que se muestran aquí, aunque estas operaciones básicas serán suficientes para empezar. Ejecuta ./bin/hadoop dfs con argumentos adicionales no se mostrará una lista de todos los comandos que se pueden ejecutar con el sistema FsShell. Por otra parte,

$HADOOP_HOME /bin/hadoop fs -help nombre comando muestra un breve resumen de los usos de la operación de que se trate, si se bloquea.

Una tabla de todas las operaciones se muestra a continuación. Las siguientes convenciones se utilizan para los parámetros:

```
"<path>" means any file or directory name.
"<path>..." means one or more file or directory names.
"<file>" means any filename.
"<src>" and "<dest>" are path names in a directed operation.
"<localSrc>" and "<localDest>" are paths as above, but on the local
file system.
```

Todos los demás archivos y ruta denominaciones se refieren a los objetos en su interior HDFS.

1.	**ls <path>** Muestra el contenido de un directorio especificado en la ruta, con los nombres, los permisos, el propietario, tamaño y fecha de modificación de cada entrada.
2.	**lsr <path>** Se comporta como -ls, pero muestra las entradas de forma recursiva todos los subdirectorios de la ruta.
3.	**du <path>** Muestra uso de disco, en bytes, de todos los archivos que coincidirá con la ruta; los nombres de archivo se comunicaron con el prefijo HDFS protocolo.
4.	**dus <path>** Como -du, pero imprime un resumen del uso del disco de todos los archivos/directorios de la ruta.
5.	**mv <src><dest>** Mueve el archivo o directorio indicado por src a dest, en HDFS.
6.	**cp <src> <dest>** Copia el archivo o directorio identificado por src a dest, en HDFS.
7.	**rm <path>** Elimina el archivo o directorio vacío identificado con la ruta de acceso.
8.	**rmr <path>** Elimina el archivo o directorio identificados con la ruta de acceso. Forma recursiva elimina todas las entradas secundarias (es decir, los archivos o los subdirectorios de la ruta).
9.	**put <localSrc> <dest>** Copia el archivo o directorio del sistema de archivos local identificado por localSrc al dest en el SLE.

10.	**copyFromLocal <localSrc> <dest>** Idéntico a -ponga
11.	**moveFromLocal <localSrc> <dest>** Copia el archivo o directorio del sistema de archivos local identificado por localSrc al dest en HDFS, y, a continuación, elimina la copia local en el éxito.
12.	**get [-crc] <src> <localDest>** Copia el archivo o directorio en HDFS identificados por src a la ruta del sistema de archivos local identificado por localDest.
13.	**getmerge <src> <localDest>** Recupera todos los archivos que coinciden con la ruta src en HDFS, y copia en un único archivo, se funden en el sistema de archivos local identificado por localDest.
14.	**cat <filen-ame>** Muestra el contenido del archivo en stdout.
15.	**copyToLocal <src> <localDest>** Idéntico a -get
16.	**moveToLocal <src> <localDest>** Funciona de forma similar -get, pero elimina la HDFS copia en caso de éxito.
17.	**mkdir <path>** Creates a directory named path in HDFS. Crea los directorios en la ruta principal que faltan (p. ej., mkdir -p en Linux).
18.	**setrep [-R] [-w] rep <path>** Establece el objetivo de replicación de archivos identificados en la ruta de acceso a rep. (La replicación real factor se moverá hacia la meta en el tiempo)
19.	**touchz <path>** Crea un archivo en la ruta que contiene la hora actual como una marca de tiempo. No se si ya existe un archivo en la ruta, a menos que el archivo ya está tamaño 0.
20.	**test -[ezd] <path>** Devuelve 1 si existe una ruta; tiene longitud cero; o es un directorio o de lo contrario, 0.
21.	**stat [format] <path>** Imprime la información acerca de camino. Format es una cadena que acepta tamaño del archivo en bloques (%b), nombre (%n), tamaño de bloque (%o), la replicación (%r), y fecha de modificación (%y, %Y).
22.	**tail [-f] <file2name>**

	Muestra el último archivo de 1KB en stdout.
23.	**chmod [-R] mode,mode,... <path>...** Cambia los permisos de archivo asociado con uno o más objetos identificados en la ruta... Realiza los cambios recursivamente con R. es el modo 3-dígitos modo octal o {augo}+/-{rwxX}. Asume si no se especifica scope y no aplicar un umask.
24.	**chown [-R] [owner][:[group]] <path>...** Establece el usuario propietario y/o un grupo de archivos o directorios identificados por la ruta... Conjuntos recursivamente si propietario -R es especificado.
25.	**chgrp [-R] group <path>...** Establece el grupo propietario de los archivos o directorios identificados por la ruta... Grupo Conjuntos recursivamente si se especifica -R.
26.	**help <cmd-name>** Devuelve información sobre el uso de uno de los comandos listados anteriormente. Debe omitir las principales caracter '-' en cmd.

Hadoop- MapReduce

MapReduce es un marco con la que podemos escribir aplicaciones para procesar grandes cantidades de datos, paralelamente, en grandes grupos de componentes de hardware de manera confiable.

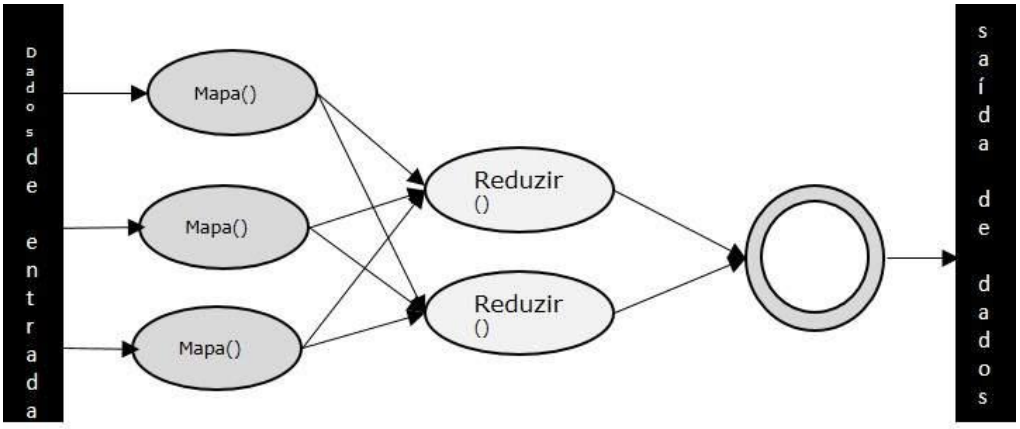
MapReduce ¿Qué es?

MapReduce es una técnica de procesamiento y un programa modelo de computación distribuida basada en java. El algoritmo MapReduce contiene dos tareas importantes, a saber Mapa y reducir. Mapa toma un conjunto de datos y se convierte en otro conjunto de datos, en el que los elementos se dividen en tuplas (pares clave/valor). En segundo lugar, reducir tarea, que toma la salida de un mapa como entrada y combina los datos tuplas en un conjunto más pequeño de tuplas. Como la secuencia de MapReduce el nombre implica, la reducción se realiza siempre después de que el mapa.

La principal ventaja de MapReduce es que es fácil de escalar procesamiento de datos en múltiples nodos. En el modelo MapReduce, el procesamiento de datos primitivos son llamados mapas y reductores. Descomposición de una aplicación de procesamiento de datos en mapas y reductores a veces es no trivial. Pero, una vez que escribir una aplicación en el MapReduce forma, la escala de la aplicación que se ejecuta en cientos, miles o incluso decenas de miles de máquinas en un clúster es simplemente un cambio de configuración. Esta escalabilidad sencilla es lo que ha atraído a muchos programadores a usar el modelo MapReduce.

El algoritmo

- Paradigma MapReduce por lo general se basa en enviar el ordenador a donde residen los datos.
- MapReduce programa se ejecuta en tres etapas, a saber: mapa etapa, shuffle, y reducir.
    - **Mapa etapa** : El mapa o mapa de trabajo es la de procesar los datos de entrada. Por lo general, los datos de entrada se encuentran en la forma de archivo o directorio y se almacena en el sistema de archivos Hadoop (HDFS). El archivo de entrada se pasa a la función mapa línea por línea. El mapper procesa los datos y crea varios pequeños fragmentos de datos.
    - **Reducir etapa**: Esta etapa es la combinación de la **reproducción aleatoria** y la etapa Reducir. La pieza de trabajo es la de procesar los datos que llegan desde el mapa. Después de un proceso de elaboración, se genera un nuevo conjunto de la producción, que se almacena en el HDFS.
- Durante el trabajo, Hadoop MapReduce envía el mapa y reducir las tareas a los servidores correspondientes en el clúster.
- El marco de trabajo administra todos los detalles de los datos de tareas tales como la emisión, verificar la realización de las tareas, y copia de los datos en todo el clúster entre los nodos.
- La mayoría de los informáticos se lleva a cabo en los nodos de datos en los discos locales que reduce el tráfico de la red.
- Tras la finalización de las tareas asignadas, el grupo recopila y disminución de los datos que forman un resultado adecuado y lo envía de vuelta al servidor Hadoop.



Las entradas y salidas (Java)

El MapReduce marco funciona en <key, value> pares, es decir, el marco considera que la entrada en el trabajo como un conjunto de <key, value> pares y produce un conjunto de etiquetas <key, value> de la salida del trabajo, posiblemente de distintos tipos.

La clave y el valor de las clases deben ser serializados por el marco y por lo tanto, necesidad de aplicar la Escritura interfaz. Además, las clases clave tienen que aplicar la

interfaz para facilitar Writable-Comparable ordenar por el marco. Tipos de entrada y salida de MapReduce trabajo: (Entrada)<k1, v1> -> map -> <k2, v2>-> reduce -> <k3, v3>(salida).

	Entrada	Salida
Mapa	<k1, v1>	lista (<k2, v2>)
Reducir	<k2, lista(v2)>	lista (<k3, v3>)

Terminología

- **Carga útil** - Aplicaciones aplicar el mapa y reduzca las funciones y forman el núcleo del trabajo.
- **Mapa** - Mapea la entrada, pares clave/valor, a un conjunto de insumos par clave/valor.
- **NamedNode** - nodo que gestiona el Hadoop Distributed File System (HDFS).
- **DataNode** - Nodo en el que se presentan los datos antes de cualquier transformación.
- **MasterNode** JobTracker - Nodo donde se ejecuta y que acepta las peticiones de trabajo de los clientes.
- **SlaveNode** - Nodo en Mapa y Reducir programa se ejecuta.
- **JobTracker** - Los trabajos y las vías el asignar trabajos a Tarea tracker.
- **Tarea Tracker** - Realiza un seguimiento de la tarea y el estado de los informes de JobTracker.
- **Trabajo**: un programa es una ejecución de un mapa y reductor en un conjunto de datos.
- **Tarea**: La ejecución de un mapa o un reductor en un corte de datos.
- **Tarea intento** - un caso particular de un intento de ejecutar una tarea en un SlaveNode.

Escenario de ejemplo

A continuación se muestra los datos relativos al consumo eléctrico de una organización. Contiene el consumo eléctrico mensual y el promedio anual de diferentes años.

	Jan	Feb	Mar	Apr	Mayo	Jun	Jul	Ago	Sep	Oct	Nov	Dec	Avg
1979	23	23	2	43	24	25	26	26	26	26	25	26	25
1980	26	27	28	28	28	30	31	31	31	30	30	30	29
1981	31	32	32	32	33	34	35	36	36	34	34	34	34
1984	39	38	39	39	39	41	42	43	40	39	38	38	40
1985	38	39	39	39	39	41	41	41	00	40	39	39	45

Si los datos anteriores no se dan como entrada, tenemos que escribir aplicaciones para procesar y producir resultados tales como encontrar el año de uso máximo, año de uso mínimo, y así sucesivamente. Este es un paseo para los programadores con número

finito de registros. Simplemente se escriben la lógica para producir los resultados requeridos, y pasar los datos a la aplicación escrita.

Sin embargo, creo que de los datos que representa el consumo eléctrico de todos los grandes sectores de un estado en particular, desde su formación.

Al escribir aplicaciones para procesar esos datos por lotes,

- Que tendrá un montón de tiempo para que se ejecute.
- Habrá una gran cantidad de tráfico en la red cuando nos movemos los datos desde el origen en el servidor de la red y así sucesivamente.

Para resolver estos problemas, tenemos el MapReduce framework.

Datos de entrada

Los datos anteriores se guardan como **ejemplo.txtand** dada como entrada. El archivo de entrada se ve como se muestra a continuación.

```
1979   23   23   2   43   24   25   26   26   26   26   25   26  25
1980   26   27   28  28   28   30   31   31   31   30   30   30  29
1981   31   32   32  32   33   34   35   36   36   34   34   34  34
1984   39   38   39  39   39   41   42   43   40   39   38   38  40
1985   38   39   39  39   39   41   41   41   00   40   39   39  45
```

Programa de ejemplo

A continuación se muestra el programa de la muestra los datos usando MapReduce framework.

```
package hadoop;

import java.util.*;

import java.io.IOException;
import java.io.IOException;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class ProcessUnits
{
   //Mapper class
   public static class E_EMapper extends MapReduceBase implements
   Mapper<LongWritable ,/*Input key Type */
   Text,                /*Input value Type*/
   Text,                /*Output key Type*/
   IntWritable>        /*Output value Type*/
   {

      //Map function
      public void map(LongWritable key, Text value,
```

```
      OutputCollector<Text, IntWritable> output,
      Reporter reporter) throws IOException
      {
         String line = value.toString();
         String lasttoken = null;
         StringTokenizer s = new StringTokenizer(line,"\t");
         String year = s.nextToken();

         while(s.hasMoreTokens())
            {
               lasttoken=s.nextToken();
            }

         int avgprice = Integer.parseInt(lasttoken);
         output.collect(new Text(year), new IntWritable(avgprice));
      }
   }


   //Reducer class
   public static class E_EReduce extends MapReduceBase implements
   Reducer< Text, IntWritable, Text, IntWritable >
   {

      //Reduce function
      public void reduce( Text key, Iterator <IntWritable> values,
         OutputCollector<Text, IntWritable> output, Reporter reporter)
throws IOException
         {
            int maxavg=30;
            int val=Integer.MIN_VALUE;

            while (values.hasNext())
            {
               if((val=values.next().get())>maxavg)
               {
                  output.collect(key, new IntWritable(val));
               }
            }

         }
   }


   //Main function
   public static void main(String args[])throws Exception
   {
      JobConf conf = new JobConf(Eleunits.class);

      conf.setJobName("max_eletricityunits");
      conf.setOutputKeyClass(Text.class);
      conf.setOutputValueClass(IntWritable.class);
      conf.setMapperClass(E_EMapper.class);
      conf.setCombinerClass(E_EReduce.class);
      conf.setReducerClass(E_EReduce.class);
      conf.setInputFormat(TextInputFormat.class);
      conf.setOutputFormat(TextOutputFormat.class);

      FileInputFormat.setInputPaths(conf, new Path(args[0]));
      FileOutputFormat.setOutputPath(conf, new Path(args[1]));
```

```
      JobClient.runJob(conf);
   }
}
```

Guardar el programa como**ProcessUnits.java.**La compilación y la ejecución del programa se explica a continuación.

Compilación y Ejecución del Programa Unidades de proceso

Supongamos que estamos en el directorio principal de un usuario Hadoop (e.g. /home/hadoop).

Siga los pasos que se indican a continuación para compilar y ejecutar el programa anterior.

Paso 1

El comando siguiente es crear un directorio para almacenar el compilado las clases de java.

```
$ mkdir units
```

Paso 2

Descargar**Hadoop-core-1.2.1.jar,**que se utiliza para compilar y ejecutar el programa MapReduce. Visita el siguiente enlace http://mvnrepository.com/artifact/org.apache.hadoop/hadoop-core/1.2.1 para descargar el jar. Supongamos que el archivo .exe descargado es**/home/hadoop/.**

Paso 3

Los siguientes comandos se utilizan para la elaboración de las**ProcessUnits.java**programa java y crear una jarra para el programa.

```
$ javac -classpath hadoop-core-1.2.1.jar -d units ProcessUnits.java
$ jar -cvf units.jar -C units/ .
```

Paso 4

El siguiente comando se utiliza para crear una entrada de directorio HDFS.

```
$HADOOP_HOME/bin/hadoop fs -mkdir input_dir
```

Paso 5

El siguiente comando se utiliza para copiar el archivo de entrada llamado**sample.txt**el directorio de entrada de HDFS.

```
$HADOOP_HOME/bin/hadoop fs -put /home/hadoop/sample.txt input_dir
```

Paso 6

El siguiente comando se utiliza para verificar los archivos en el directorio de entrada.

```
$HADOOP_HOME/bin/hadoop fs -ls input_dir/
```

Paso 7

El siguiente comando se utiliza para ejecutar la aplicación Eleunit_max teniendo en los archivos de entrada en el directorio de entrada.

```
$HADOOP_HOME/bin/hadoop jar units.jar hadoop.ProcessUnits input_dir
output_dir
```

Espere un momento hasta que el archivo es ejecutado. Después de la ejecución, como se muestra a continuación, la salida contendrá el número de divisiones, el número de tareas mapa, el número de pieza tareas, etc.

```
INFO mapreduce.Job: Job job_1414748220717_0002
completed successfully
14/10/31 06:02:52
INFO mapreduce.Job: Counters: 49
File System Counters

FILE: Number of bytes read=61
FILE: Number of bytes written=279400
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=546
HDFS: Number of bytes written=40
HDFS: Number of read operations=9
HDFS: Number of large read operations=0
HDFS: Number of write operations=2 Job Counters


   Launched map tasks=2
   Launched reduce tasks=1
   Data-local map tasks=2
   Total time spent by all maps in occupied slots (ms)=146137
   Total time spent by all reduces in occupied slots (ms)=441
   Total time spent by all map tasks (ms)=14613
   Total time spent by all reduce tasks (ms)=44120
   Total vcore-seconds taken by all map tasks=146137

   Total vcore-seconds taken by all reduce tasks=44120
   Total megabyte-seconds taken by all map tasks=149644288
   Total megabyte-seconds taken by all reduce tasks=45178880

Map-Reduce Framework

Map input records=5
   Map output records=5
   Map output bytes=45
   Map output materialized bytes=67
   Input split bytes=208
   Combine input records=5
   Combine output records=5
   Reduce input groups=5
```

```
   Reduce shuffle bytes=6
   Reduce input records=5
   Reduce output records=5
   Spilled Records=10
   Shuffled Maps =2
   Failed Shuffles=0
   Merged Map outputs=2
   GC time elapsed (ms)=948
   CPU time spent (ms)=5160
   Physical memory (bytes) snapshot=47749120
   Virtual memory (bytes) snapshot=2899349504
   Total committed heap usage (bytes)=277684224

File Output Format Counters

   Bytes Written=40
```

Paso 8

El siguiente comando se utiliza para verificar los archivos resultantes de la carpeta de salida.

```
$HADOOP_HOME/bin/hadoop fs -ls output_dir/
```

Paso 9

El siguiente comando se utiliza para ver el resultado en la **Part-00000** archivo. Este archivo se genera por HDFS.

```
$HADOOP_HOME/bin/hadoop fs -cat output_dir/part-00000
```

A continuación se muestra la salida generada por el MapReduce programa.

```
1981    34
1984    40
1985    45
```

Paso 10

El siguiente comando se utiliza para copiar la carpeta de salida de los HDFS en el sistema de archivos local para analizar.

```
$HADOOP_HOME/bin/hadoop fs -cat output_dir/part-00000/bin/hadoop dfs
get output_dir /home/hadoop
```

Comandos importantes

Hadoop Todos los comandos son invocados por el **$HADOOP_HOME/bin/hadoop**. Ejecuta el script Hadoop sin argumentos imprime la descripción de todos los comandos.

Usage : hadoop [--config confdir] COMANDO

La siguiente tabla detalla las opciones disponibles y su descripción.

Opciones	Descripción
Namenode -formato	La DFS Formatos archivos.
Secondarynamenode	Ejecuta el DFS namenode secundario.
Namenode	Ejecuta el DFS namenode.
Datanode	Ejecuta un DFS datanode.
Dfsadmin	Ejecuta un DFS cliente admin.
Mradmin	Ejecuta un Map-Reduce cliente admin.
Fsck	Ejecuta una utilidad de comprobación de archivos.
Fs	Archivos genéricos se ejecuta un cliente usuario.
Equilibradora	Ejecuta un balanceo del cluster utilidad.
oiv	Se aplica el visor sin conexión fsimage a un fsimage.
Fetchdt	Obtiene un token de la delegación NameNode.
Jobtracker	Ejecuta el trabajo MapReduce Tracker nodo.
Tubos	Tubos se ejecuta un trabajo.
Tasktracker	Ejecuta una tarea MapReduce Tracker nodo.
Historyserver	Historial de trabajo se ejecuta servidores como un demonio independiente.
Trabajo	Manipula los trabajos MapReduce.
Cola	Obtiene información sobre JobQueues.
Versión	Imprime el número de versión.
jar <jar>	Se ejecuta un archivo jar.
distcp <srcurl> <desturl>	Copia el archivo o directorios recursivamente.
distcp2 <srcurl> <desturl>	DistCp versión 2.
Archive -archiveName NOMBRE -p	Hadoop crea un archivo.
<parent path> <src>* <dest>	
Classpath.	Imprime la ruta de la clase necesaria para obtener el Hadoop jar y las bibliotecas necesarias.
Daemonlog	Obtener/Establecer el nivel de registro para cada demonio

Cómo interactuar con los trabajos MapReduce

Uso: hadoop job [GENERIC_OPTIONS]

Las siguientes son las opciones genéricas disponibles en Hadoop trabajo.

GENERIC_OPTIONS	Descripción
-submit <job-file>	Envía el trabajo.
Estado <job-id>	Imprime el mapa y reducir porcentaje de finalización y contadores de trabajos.
Contador <id de trabajo> <group-name> <countername>	Imprime el valor del contador.
-Matar <id de trabajo>	Mata el trabajo.
-Eventos <id de trabajo> <fromevent- #> < # -de eventos>	Imprime las citas recibidas por los detalles jobtracker para el rango dado.
-Historia [todos] <jobOutputDir> - historia < jobOutputDir>	Imprime los detalles del trabajo, error y mató detalles punta. Más detalles sobre el trabajo, como tareas y éxito los intentos realizados para cada tarea se pueden ver mediante la especificación de la opción [todos].
-List [todos]	Muestra todos los trabajos. -List muestra sólo los trabajos que todavía no se han completado.
-Matar tarea <tarea-id>	Mata a la tarea. Muertas las tareas no se imputarán a intentos fallidos.
-No-tarea <tarea-id>	Falla la tarea. Las tareas no se imputarán a
	Intentos fallidos.
De prioridad <id de trabajo> <prioridad>	Cambia la prioridad del trabajo. Los valores de prioridad son permitidos VERY_HIGH, ALTA, NORMAL, BAJA, VERY_LOW

Para ver el estado de trabajo

```
$ $HADOOP_HOME/bin/hadoop job -status <JOB-ID>
e.g.
$ $HADOOP_HOME/bin/hadoop job -status job_201310191043_0004
```

Para ver la historia de salida del trabajo-dir

```
$ $HADOOP_HOME/bin/hadoop job -history <DIR-NAME>
e.g.
$ $HADOOP_HOME/bin/hadoop job -history /user/expert/output
```

Para matar el trabajo

```
$ $HADOOP_HOME/bin/hadoop job -kill <JOB-ID>
e.g.
$ $HADOOP_HOME/bin/hadoop job -kill job_201310191043_0004
```

Hadoop- Streaming

Hadoop streaming es una utilidad que viene con el Hadoop distribución. Esta utilidad le permite crear y ejecutar Map/Reduce los trabajos con cualquier archivo ejecutable o script como el mapa y/o el reductor.

Ejemplo usando Python

Para Hadoop streaming, vamos a considerar el conteo de palabras. Cualquier trabajo de Hadoop debe tener dos fases: mapa y reductor. Hemos escrito los códigos para el mapa y el reductor de script en python para que se ejecute con Hadoop. También se puede escribir la misma en Perl y Ruby.

Fase Código Mapa

```
!/usr/bin/python
import sys
# Input takes from standard input for myline in sys.stdin:
# Remove whitespace either side myline = myline.strip()
# Break the line into words words = myline.split()
# Iterate the words list for myword in words:
# Write the results to standard output print '%s\t%s' % (myword, 1)
```

Asegúrese de que este archivo tiene permiso de ejecución (chmod +x /home/ expert/hadoop-1.2.1/mapper.py).

Fase Código reductor

```
#!/usr/bin/python
from operator import itemgetter
import sys
current_word = ""
current_count = 0
word = ""
# Input takes from standard input for myline in sys.stdin:
# Remove whitespace either side myline = myline.strip()
# Split the input we got from mapper.py word, count =
myline.split('\t', 1)
# Convert count variable to integer
   try:
      count = int(count)
except ValueError:
   # Count was not a number, so silently ignore this line continue
if current_word == word:
   current_count += count
```

```
else:
   if current_word:
      # Write result to standard output print '%s\t%s' %
(current_word, current_count)
   current_count = count
   current_word = word
# Do not forget to output the last word if needed!
if current_word == word:
   print '%s\t%s' % (current_word, current_count)
```

Guardar el mapa y reductor en los códigos de mapper.py y reductor.py en Hadoop directorio home. Asegúrese de que estos archivos tienen permisos de ejecución (chmod +x mapper.py and chmod +x reducer.py). Sangría Como python es sensible por lo que el mismo código se puede descargar desde el enlace que aparece a continuación.

Ejecución del Programa WordCount

```
$ $HADOOP_HOME/bin/hadoop jar contrib/streaming/hadoop-streaming-1.
2.1.jar \
   -input input_dirs \
   -output output_dir \
   -mapper <path/mapper.py \
   -reducer <path/reducer.py
```

Donde " \" se usa para continuación de línea clara de legibilidad.

Por ejemplo,

```
./bin/hadoop jar contrib/streaming/hadoop-streaming-1.2.1.jar -input
myinput -output myoutput -mapper /home/expert/hadoop-1.2.1/mapper.py -
reducer /home/expert/hadoop-1.2.1/reducer.py
```

Streaming Cómo Funciona

En el ejemplo anterior, tanto el mapa y el reductor son secuencias de comandos python que lee la entrada desde la entrada estándar y emiten la salida a la salida estándar. La utilidad crea un mapa/Reducir trabajo, enviar el trabajo a un clúster, y supervisar el progreso de la tarea hasta que finalice.

Cuando se especifica una secuencia de mapas mapa, cada tarea se iniciará la secuencia de comandos como un proceso independiente cuando el mapa se inicializa. Como el mapa tarea se ejecuta, lo que convierte sus entradas en las líneas de alimentación y las líneas de la entrada estándar (STDIN) del proceso. En el ínterin, el mapa recoge la línea de productos orientada a la salida estándar (STDOUT) del proceso y convierte cada línea en un par de clave/valor, que se recoge en la salida del mapa. De forma predeterminada, el prefijo de la línea hasta el primer carácter de tabulación es la clave y el resto de la línea (excluyendo el carácter de tabulación) será el valor. Si no hay ningún carácter de tabulación en la línea y, a continuación, toda la línea se considera la clave y el valor es null. Sin embargo, esta puede ser personalizada, en función de una necesidad.

Cuando se especifica una secuencia de reductores reductor, cada tarea se iniciará la secuencia de comandos como un proceso separado, a continuación, el reductor se ha inicializado. Como el reductor la tarea se ejecuta, convierte su clave de entrada y los valores en las líneas pares y alimenta las líneas de la entrada estándar (STDIN) del proceso. En el ínterin, el reductor recoge la línea de productos orientada a la salida estándar (STDOUT) del proceso, convierte cada línea en un par de clave/valor, que se recoge en la salida del reductor. De forma predeterminada, el prefijo de la línea hasta el primer carácter de tabulación es la clave y el resto de la línea (excluyendo el carácter de tabulación) es el valor. Sin embargo, esta puede ser personalizada según necesidades específicas.

Comandos importantes

Parámetros	Descripción
-Directorio de entrada/nombre de archivo	Mapa de ubicación de entrada. (Obligatorio)
-Directorio de salida de nombre	Ubicación de salida por reductor. (Obligatorio)
-Mapper ejecutable o script o JavaClassName	Mapa ejecutable. (Obligatorio)
-Reductor ejecutable o script o JavaClassName	Reductor ejecutable. (Obligatorio)
-Archivo nombre de archivo	Hace que el mapa, el reductor, ejecutable o combinadora disponible localmente en los nodos de cálculo.
-Inputformat JavaClassName	Clase de retorno debe pares clave/valor de texto clase. Si no se especifica, TextInputFormat se utiliza como valor predeterminado.
-Outputformat JavaClassName	Clase de debe tener pares clave/valor de texto clase. Si no se especifica, TextOutputformat se utiliza como valor predeterminado.
-Partitioner JavaClassName	Clase que determina qué reducir la clave es enviado a.
-Combinador o JavaClassName streamingCommand	Combinador ejecutable de salida del mapa.
-Cmdenv nombre=valor	Pasa la variable de entorno a los comandos de streaming.
-Inputreader	Para mantener la compatibilidad: especifica un registro clase de lector (en lugar de un formato de entrada clase).
-Verbose	Resultados detallados.
-Lazyoutput	Crea una salida perezosamente. Por ejemplo, si el formato de salida se basa en FileOutputFormat, el

	archivo de salida se crea sólo en la primera llamada a la salida.recoger (o el contexto.write).
-Numreducetasks	Especifica el número de reductores.
-Mapdebug	Script para llamar al mapa tarea falla.
-Reducedebug	Secuencia de comandos para llamar al reducir tarea falla.

Hadoop- Varios nodos de clúster

Como todo el conjunto no puede ser demostrado, estamos explicando la Hadoop cluster medio ambiente utilizando tres sistemas (un maestro y dos esclavos); a continuación, se presentan sus direcciones IP.

- Hadoop Master: 192.168.1.15 (hadoop-master)
- Hadoop Slave: 192.168.1.16 (hadoop-slave-1)
- Hadoop Slave: 192.168.1.17 (hadoop-slave-2)

Siga los pasos que se indican a continuación para que Hadoop clúster de varios nodos.

Instalación de Java

Java es el principal requisito previo para Hadoop. En primer lugar, debe comprobar la existencia de java en el sistema utilizando "java -version". La sintaxis de java versión comando es dada a continuación.

```
$ java -version
```

Si todo funciona bien, le dará el siguiente resultado.

```
java version "1.7.0_71"
Java(TM) SE Runtime Environment (build 1.7.0_71-b13)
Java HotSpot(TM) Client VM (build 25.0-b02, mixed mode)
```

Si java no está instalado en el sistema, a continuación, siga estos pasos para instalar java.

Paso 1

Descargar Java (JDK - X64.tar.gz) visitando el siguiente enlace http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

A continuación, **jdk-7u71-linux-x64.tar.gz** se descargará en su sistema.

Paso 2

En general, encontrará el archivo descargado java en carpeta de descargas. Verificar y extraer el **jdk-7u71-linux-x64.gz** usando los siguientes comandos.

```
$ cd Downloads/
$ ls
jdk-7u71-Linux-x64.gz
$ tar zxf jdk-7u71-Linux-x64.gz
$ ls
jdk1.7.0_71 jdk-7u71-Linux-x64.gz
```

Paso 3

Para que java disponible para todos los usuarios, tiene que mover a la ubicación "/usr/local/". Abra el root, y escriba los siguientes comandos.

```
$ su
password:
# mv jdk1.7.0_71 /usr/local/
# exit
```

Paso 4

Para la configuración de **ruta de acceso** y **JAVA_HOME** variables, agregar los siguientes comandos en el archivo**~/.bashrc**.

```
export JAVA_HOME=/usr/local/jdk1.7.0_71
export PATH=PATH:$JAVA_HOME/bin
```

Ahora compruebe que el **java -version** comando desde el terminal, como se explicó anteriormente. Siga el proceso anterior e instalar java en todos los nodos del clúster.

Creando una cuenta de usuario

Crear una cuenta de usuario del sistema en tanto el maestro como el esclavo sistemas Hadoop para utilizar la instalación.

```
# useradd hadoop
# passwd hadoop
```

La asignación de nodos

Tiene que editar archivo hosts en **/etc/** carpeta en todos los nodos, se especifica la dirección IP de cada sistema seguido por sus nombres de host.

```
# vi /etc/hosts
enter the following lines in the /etc/hosts file.
192.168.1.109 hadoop-master
192.168.1.145 hadoop-slave-1
192.168.56.1 hadoop-slave-2
```

Configuración basada en una clave de acceso

Configuración ssh en cada uno de los nodos, que pueden comunicarse entre sí sin pedir contraseña.

```
# su hadoop
$ ssh-keygen -t rsa
$ ssh-copy-id -i ~/.ssh/id_rsa.pub tutorialspoint@hadoop-master
$ ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop_tp1@hadoop-slave-1
$ ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop_tp2@hadoop-slave-2
$ chmod 0600 ~/.ssh/authorized_keys
$ exit
```

Instalar Hadoop

En el servidor maestro, descargue e instale Hadoop usando los siguientes comandos.

```
# mkdir /opt/hadoop
# cd /opt/hadoop/
# wget http://apache.mesi.com.ar/hadoop/common/hadoop-1.2.1/hadoop-
1.2.0.tar.gz
# tar -xzf hadoop-1.2.0.tar.gz
# mv hadoop-1.2.0 hadoop
# chown -R hadoop /opt/hadoop
# cd /opt/hadoop/hadoop/
```

Configuración Hadoop

Usted tendrá que configurar Hadoop server, haciendo los siguientes cambios como se indica a continuación.

core-site.xml

Abrir el**core-site.xml**archivo y editar, como se muestra a continuación.

```
<configuration>
   <property>
      <name>fs.default.name</name>
      <value>hdfs://hadoop-master:9000/</value>
   </property>
   <property>
      <name>dfs.permissions</name>
      <value>false</value>
   </property>
</configuration>
```

hdfs-site.xml

Abra la**hdfs-site.xml**y editar como se muestra a continuación.

```
<configuration>
   <property>
      <name>dfs.data.dir</name>
      <value>/opt/hadoop/hadoop/dfs/name/data</value>
      <final>true</final>
   </property>
```

```
   <property>
      <name>dfs.name.dir</name>
      <value>/opt/hadoop/hadoop/dfs/name</value>
      <final>true</final>
   </property>

   <property>
      <name>dfs.replication</name>
      <value>1</value>
   </property>
</configuration>
```

mapred-site.xml

mapred-site.xmlarchivo xml y editar tal y como se muestra a continuación.

```
<configuration>
   <property>
      <name>mapred.job.tracker</name>
      <value>hadoop-master:9001</value>
   </property>
</configuration>
```

hadoop-env.sh

Abra el archivo **hadoop-env.sh** y editar JAVA_HOME, HADOOP_CONF_DIR y HADOOP_OPTS como se muestra a continuación.

Nota: Ajuste el JAVA_HOME como por la configuración del sistema.

```
export JAVA_HOME=/opt/jdk1.7.0_17 export HADOOP_OPTS=-
Djava.net.preferIPv4Stack=true export
HADOOP_CONF_DIR=/opt/hadoop/hadoop/conf
```

Instalación de servidores esclavos Hadoop

Instalar Hadoop en todos los servidores esclavos siguiendo el recibir órdenes.

```
# su hadoop
$ cd /opt/hadoop
$ scp -r hadoop hadoop-slave-1:/opt/hadoop
$ scp -r hadoop hadoop-slave-2:/opt/hadoop
```

Hadoop Configuración en el servidor maestro

Abra el servidor maestro y configurar siguiendo el recibir órdenes.

```
# su hadoop
$ cd /opt/hadoop/hadoop
```

Configurar Nodo maestro

```
$ vi etc/hadoop/masters
hadoop-master
```

Configurar Nodo secundario

```
$ vi etc/hadoop/slaves
```

```
hadoop-slave-1
hadoop-slave-2
```

Nombre de formato nodo maestro sobre Hadoop

```
# su hadoop
$ cd /opt/hadoop/hadoop
$ bin/hadoop namenode –format
11/10/14 10:58:07 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG: host = hadoop-master/192.168.1.109
STARTUP_MSG: args = [-format]
STARTUP_MSG: version = 1.2.0
STARTUP_MSG: build =
https://svn.apache.org/repos/asf/hadoop/common/branches/branch-1.2 -r
1479473; compiled by 'hortonfo' on Mon May 6 06:59:37 UTC 2013
STARTUP_MSG: java = 1.7.0_71
************************************************************/ 11/10/14
10:58:08 INFO util.GSet: Computing capacity for map BlocksMap
editlog=/opt/hadoop/hadoop/dfs/name/current/edits
……….
……….
……. 11/10/14 10:58:08 INFO common.Storage: Storage
directory /opt/hadoop/hadoop/dfs/name has been successfully formatted.
11/10/14 10:58:08 INFO namenode.NameNode:
SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at hadoop-master/192.168.1.15
************************************************************/
```

Hadoop Inicio Servicios

El comando siguiente es para iniciar todas las Hadoop servicios en el Hadoop-Master.

```
$ cd $HADOOP_HOME/sbin
$ start-all.sh
```

Agregar un nuevo DataNode en el Hadoop Cluster

A continuación, se presentan los pasos a seguir para agregar nuevos nodos a un clúster Hadoop.

Redes

Agregar nuevos nodos a un clúster Hadoop existente con las correspondientes de la red. Asumir la siguiente configuración de red.

Para la nueva configuración de nodo:

```
IP address : 192.168.1.103
netmask : 255.255.255.0
hostname : slave3.in
```

Agregar el usuario y acceso SSH

Agregar un usuario

En un nodo nuevo, agregue "hadoop" usuario y contraseña de Hadoop usuario "hadoop123" o cualquier cosa que quiera con los siguientes comandos.

```
useradd hadoop
passwd hadoop
```

Contraseña de configuración menor conectividad desde el maestro al nuevo esclavo.

Ejecute el siguiente en el master

```
mkdir -p $HOME/.ssh
chmod 700 $HOME/.ssh
ssh-keygen -t rsa -P '' -f $HOME/.ssh/id_rsa
cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
chmod 644 $HOME/.ssh/authorized_keys
Copy the public key to new slave node in hadoop user $HOME directory
scp $HOME/.ssh/id_rsa.pub hadoop@192.168.1.103:/home/hadoop/
```

Ejecute el siguiente de los esclavos

Iniciar sesión en hadoop. Si no es así, inicie sesión en hadoop usuario.

```
su hadoop ssh -X hadoop@192.168.1.103
```

Copiar el contenido de clave pública en un archivo **"$HOME/.ssh/authorized_keys"** y, a continuación, cambiar el permiso del mismo ejecutando el siguiente comando.

```
cd $HOME
mkdir -p $HOME/.ssh
chmod 700 $HOME/.ssh
cat id_rsa.pub >>$HOME/.ssh/authorized_keys
chmod 644 $HOME/.ssh/authorized_keys
```

Verificar ssh login en la máquina maestra. Ahora compruebe si puede ssh al nuevo nodo sin una contraseña del maestro.

```
ssh hadoop@192.168.1.103 or hadoop@slave3
```

Configurar nombre de host del nodo nuevo

Puede configurar nombre de host en el archivo**/etc/sysconfig/network**

```
On new slave3 machine
NETWORKING=yes
HOSTNAME=slave3.in
```

Para que los cambios surtan efecto, reinicie el equipo o comando hostname a un nuevo equipo con el respectivo nombre de reiniciar es una buena opción).

El esclavo3 equipo de nodo:

hostname slave3.in

Actualizar el archivo **/etc/hosts**en todas las máquinas del clúster con las líneas siguientes:

```
192.168.1.102 slave3.in slave3
```

Ahora, intente hacer ping a la máquina con nombres de host para comprobar si se está resolviendo en IP o no.

En el nuevo equipo de nodo:

```
ping master.in
```

Iniciar el DataNode en Nuevo Nodo

Iniciar el demonio datanode manualmente mediante **$HADOOP_HOME/bin/hadoop-daemon.sh script**. Y automáticamente se contacto con el maestro (NameNode) y unirse al clúster. También debemos añadir el nuevo nodo al archivo conf/esclavos en el servidor maestro. La secuencia de comandos se reconocerá el nuevo nodo.

Iniciar sesión en nuevo nodo

```
su hadoop or ssh -X hadoop@192.168.1.103
```

Inicio HDFS en un nodo secundario recién agregado mediante el siguiente comando

```
./bin/hadoop-daemon.sh start datanode
```

La salida de jps comando en un nodo nuevo. Se ve de la siguiente manera.

```
$ jps
7141 DataNode
10312 Jps
```

Extracción de un DataNode del Hadoop Cluster

Podemos eliminar un nodo de un clúster sobre la marcha, mientras se está ejecutando, sin ninguna pérdida de datos. HDFS característica proporciona una clausura, el cual asegura que eliminar un nodo se lleva a cabo en forma segura. Para utilizar esta función, siga los pasos que aparecen a continuación:

Paso 1: Iniciar sesión en master

Iniciar sesión en master usuario de la máquina donde Hadoop está instalado.

```
$ su hadoop
```

Paso 2: Cambiar configuración de clúster

El excluir archivo debe estar configurado antes de iniciar el clúster. Agregar una clave con nombre dfs.hosts.excluir a nuestro **$HADOOP_HOME/etc/hadoop/hdfs-site.xml**. El valor asociado a la clave ofrece la ruta de acceso completa a un archivo en el NameNode local del sistema de archivos que contiene una lista de las máquinas que no están autorizados a conectarse a HDFS.

Por ejemplo, agregar las siguientes líneas al **etc/hadoop/hdfs sitio.xml**.

```
<property>
   <name>dfs.hosts.exclude</name>
   <value>/home/hadoop/hadoop-1.2.1/hdfs_exclude.txt</value>
   <description>DFS exclude</description>
</property>
```

Paso 3: Determinar los hosts para retirar

Cada una de las máquinas que se van a desmantelar se debe agregar en el archivo identificado por el hdfs_exclude.txt, uno de los nombres de dominio por línea. De esta manera se impedirá que conecta a la NameNode. El contenido del**"/home/hadoop/hadoop-1.2.1/hdfs_exclude.txt"**archivo se muestra a continuación, si desea eliminar DataNode2.

```
slave2.in
```

Paso 4: Fuerza a cargar configuración

Ejecute el comando**"$HADOOP_HOME/bin/hadoop dfsadmin -refreshNodes"**sin las comillas.

```
$ $HADOOP_HOME/bin/hadoop dfsadmin -refreshNodes
```

Esto hará que el NameNode volver a leer la configuración, incluyendo el recientemente actualizado "excluye" archivo. Se retira los nodos en un período de tiempo, lo que permite tiempo para cada uno de los nodos de bloques que se replican en las máquinas que están programadas para permanecer activo.

En**slave2.in**, verificar la jps salida de comando. Después de algún tiempo, se verá el proceso DataNode se desconecta automáticamente.

Paso 5: Apagar los nodos

Después de que el proceso de retirada se ha completado, el hardware puede ser retirada con seguridad cerrado por mantenimiento. Ejecute el comando de informe a dfsadmin para comprobar el estado de retirada. El siguiente comando se describen el estado de los nodos y retirar los nodos conectados al panel de instrumentos.

```
$ $HADOOP_HOME/bin/hadoop dfsadmin -report
```

Paso 6: Editar excluye archivo de nuevo

Una vez que las máquinas han sido clausuradas, se pueden retirar de la "excluye" archivo. Ejecución de**"$HADOOP_HOME/bin/hadoop dfsadmin -refreshNodes"** nuevo a leer la excluye de nuevo en el NameNode; lo que permite que el DataNodes volver a unirse a un grupo después de la conservación se ha terminado, o se necesita más capacidad en el clúster nuevo, etc.

Nota especial: Si el proceso descrito anteriormente es seguido y el tasktracker proceso todavía está en funcionamiento en el nodo, es necesario que se cierre. Una manera de hacerlo es para desconectar la máquina como lo hicimos en los pasos anteriores. El Maestro se reconocerá el proceso automáticamente y se declaran como muertos. No hay necesidad de seguir el mismo proceso para la extracción de la tasktracker porque no es tan crucial como en comparación con el DataNode. DataNode contiene los datos que desea quitar de forma segura sin ninguna pérdida de datos.

La tasktracker se puede ejecutar/shutdown de la mosca con el siguiente comando en cualquier momento.

```
$ $HADOOP_HOME/bin/hadoop-daemon.sh stop tasktracker
$HADOOP_HOME/bin/hadoop-daemon.sh start tasktracker
```