**PONTIFICIA UNIVERSIDAD CATÓLICA MADRE Y MAESTRA  
FACULTAD DE CIENCIAS E INGENIERÍA**

A logo of a university

Description automatically generated with low confidence

**Tecnologías Emergentes.**

**ISC-573**

Proyecto Final-Hadoop

|  |  |  |
| --- | --- | --- |
| **Presentado por:** |  | Junior Hernández 2018-0999. Nicol Ureña 2018-1669 |
| **Entregado a:** |  | Alfredo Vásquez. |
| **Fecha de entrega:** |  | 1 de agosto del 2023. |
| **Santiago de los caballeros; República Dominicana.** | | |

**Introducción**

La arquitectura de un clúster Hadoop es esencial para el procesamiento y almacenamiento distribuido de grandes volúmenes de datos. Antes de configurar los nodos maestros y trabajadores, es fundamental comprender los diferentes componentes que conforman este sistema. El nodo maestro, compuesto por el NameNode y el ResourceManager, mantiene el conocimiento del sistema de archivos distribuidos y administra la asignación de recursos en el clúster. Por otro lado, los nodos trabajadores, que incluyen los DataNodes y los NodeManagers, almacenan los datos reales y proporcionan la potencia de procesamiento para ejecutar los trabajos.

En este reporte, se explicará paso a paso cómo configurar un clúster Hadoop en modo distribuido en entornos Linux. Se proporciona información sobre la instalación de Java, la descarga e instalación de Hadoop, así como la configuración de las variables de entorno y los archivos de configuración necesarios para el correcto funcionamiento del clúster de 2 nodos.

**Arquitectura de un Clúster Hadoop**

Antes de configurar los nodos maestros y trabajadores, es importante comprender los diferentes componentes de un clúster Hadoop.

Nodo Maestro:

El nodo maestro mantiene el conocimiento sobre el sistema de archivos distribuidos, similar a la tabla de inodos en un sistema de archivos ext3, y gestiona la asignación de recursos. En esta guía, el nodo maestro se denominará "node-master" y alojará dos demonios:

NameNode: Administra el sistema de archivos distribuidos y conoce la ubicación de los bloques de datos almacenados en el clúster.

ResourceManager: Gestiona los trabajos YARN y se encarga de programar y ejecutar procesos en los nodos trabajadores.

Nodos Trabajadores:

Los nodos trabajadores almacenan los datos reales y proporcionan la potencia de procesamiento para ejecutar los trabajos. En esta guía, habrá dos nodos trabajadores, llamados "node1" y "node2", cada uno alojando dos demonios:

DataNode: Gestiona los datos físicos almacenados en el nodo; está vinculado al NameNode.

NodeManager: Gestiona la ejecución de tareas en el nodo.

Creación del archivo hosts en cada nodo

Para que cada nodo se comunique entre sí por nombre, edita el archivo /etc/hosts para agregar las direcciones IP privadas de los tres servidores. No olvides reemplazar las direcciones IP de muestra con las de tus nodos, por ejemplo:

192.0.2.1 node-master

192.0.2.2 node1

Instalación de Hadoop

Antes de instalar Hadoop, es importante conocer los diferentes tipos de implementaciones que pueden realizarse:

Modo local: Se ejecuta en un solo nodo como un solo proceso Java.

Modo pseudo-distribuido local: Se ejecuta en un solo nodo, pero cada servicio se ejecuta en un proceso Java diferente.

Modo distribuido: Se ejecuta utilizando múltiples nodos completamente distribuidos.

Para montar Hadoop en producción, se utiliza el modo distribuido:

Hadoop funciona en entornos Linux. Se recomienda comenzar a explorar Hadoop utilizando una máquina virtual.

Instalación de Java:

Antes de instalar Hadoop, asegúrate de tener Java 1.8 o superior instalado. Si no tienes Java, puedes instalarlo con el siguiente comando:

sudo apt-get update

sudo apt install default-jdk default-jre -y

Verifica la versión de Java instalada:

java -version

Descargar e instalar Hadoop:

Para instalar Hadoop, primero descarga la versión adecuada desde el sitio web oficial de Apache Hadoop:

wget <https://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz>

Descomprime el archivo descargado y muévelo a la carpeta donde se instalará Hadoop:

tar -xzvf hadoop-3.3.6.tar.gz

sudo mv hadoop-3.3.6 /usr/local/hadoop

Crear una carpeta para almacenar los registros (logs) y cambiar el propietario:

sudo mkdir /usr/local/hadoop/logs

sudo chown -R hadoop:hadoop /usr/local/Hadoop

Configurar SSH:

Los nodos en Hadoop se comunican mediante SSH, por lo que debes configurar este protocolo:

sudo apt install openssh-server openssh-client -y

Crear una clave SSH para permitir la comunicación entre los nodos del clúster sin necesidad de ingresar contraseña cada vez:

ssh-keygen -t rsa -P ''

Cuando generes esta clave, deja el campo de contraseña en blanco para que el usuario de Hadoop pueda comunicarse sin ser solicitado.

Visualiza la clave pública del nodo maestro y cópiala al portapapeles para usarla en cada uno de los nodos trabajadores:

less /home/hadoop/.ssh/id\_rsa.pub

Agrega la nueva clave generada a la lista de claves autorizadas, para que Hadoop pueda usar SSH sin solicitar contraseña:

cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys

La configuración se realizará en el nodo maestro y se replicará en los demás nodos.

Establecer JAVA\_HOME:

Encuentra la ubicación de tu instalación de Java. Esto se conoce como JAVA\_HOME. Si instalaste OpenJDK desde el administrador de paquetes, puedes encontrar la ruta con el siguiente comando:

update-alternatives --display java

Toma el valor del enlace actual y elimina el /bin/java final. Por ejemplo, en Debian, el enlace es /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java, por lo que JAVA\_HOME debería ser /usr/lib/jvm/java-8-openjdk-amd64/jre.

Si instalaste Java desde Oracle, JAVA\_HOME es la ruta donde descomprimiste el archivo de Java.

Edita el archivo ~/hadoop/etc/hadoop/hadoop-env.sh y reemplaza esta línea:

export JAVA\_HOME=${JAVA\_HOME}

con la ruta de instalación real de Java. Por ejemplo, en un Debian 9 Linode con open-jdk-8, esto se vería así:

export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre

Configurar las variables de entorno de Hadoop:

Modifica el archivo ~/.bashrc para configurar las variables de entorno del usuario que utilizará Hadoop:

sudo nano ~/.bashrc

Agrega las siguientes líneas al final del archivo:

#HADOOP VARIABLES START

export HADOOP\_HOME=/usr/local/hadoop

export HADOOP\_INSTALL=$HADOOP\_HOME

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\_OPTS="-Djava.library.path=$HADOOP\_HOME/lib/native"

#HADOOP VARIABLES END

Guarda el archivo y ciérralo.

Ejecuta el siguiente comando para que el sistema reconozca las nuevas variables de entorno creadas, de modo que las variables estarán visibles cada vez que se reinicie la máquina virtual:

source ~/.bashrc

Luego configuramos el archivo hadoop-env.sh, el cual contiene algunas de las variables de entorno que utiliza Hadoop. Para esto ejecutamos el siguiente comando:

sudo nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh

Al final del archivo incorporar las siguientes variables:

export JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64 export HADOOP\_CLASSPATH+=" $HADOOP\_HOME/lib/\*.jar"

Luego cambiar el directorio de trabajo a /usr/local/hadoop/lib y bajar un archivo \*.jar, el cual es necesario para el manejo correcto de Hadoop. Para esto ejecutar los siguientes comandos:

cd /usr/local/hadoop/lib

sudo wget https://jcenter.bintray.com/javax/activation/javax.activation- api/1.2.0/javax.activation-api-1.2.0.jar

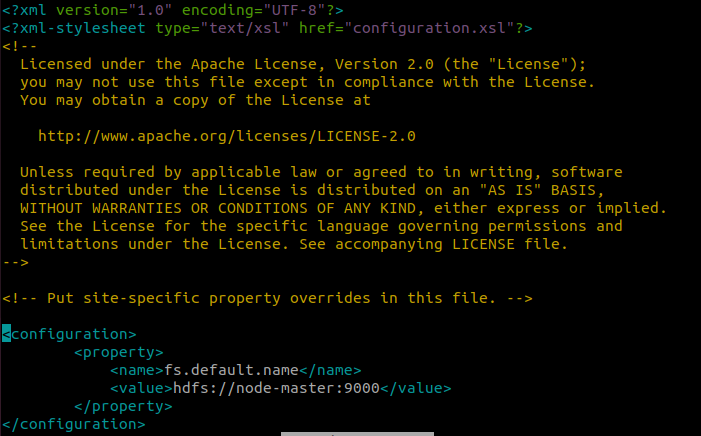
Después de bajada la librería javax.activation-api-1.2.0.jar, ejecutar el comando hadoop versión, para obtener la versión de hadoop y confirmar que todo funciona bien hasta este punto:

hadoop versión

Ahora procederemos a modificar todos los archivos necesarios para la configuración del clúster Pseudo- Distribuido de Hadoop. Comenzaremos con el archivo core-site.xml, el cual contiene las propiedades que usa Hadoop en su arranque. Para esto ejecutamos el siguiente comando:

sudo nano /usr/local/hadoop/etc/hadoop/core-site.xml

Después editar el archivo para que se vea como el siguiente bloque:

El archivo core-site.xml es una configuración importante en el ecosistema de Hadoop. Este archivo contiene propiedades clave-valor que definen la configuración básica del sistema de archivos distribuido de Hadoop (HDFS) y otros componentes centrales del clúster Hadoop. Veamos el significado e importancia de cada elemento en esta configuración:

<?xml version="1.0" encoding="UTF-8"?>: Esta declaración indica que el archivo sigue la sintaxis XML versión 1.0 y utiliza el conjunto de caracteres UTF-8 para codificar el contenido del archivo.

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>: Esta línea especifica que se puede aplicar una hoja de estilo XML (XSL) al archivo para transformar su apariencia. En este caso, el archivo de estilo se denomina "configuration.xsl" y se aplica al archivo XML para mejorar su legibilidad visual.

<configuration>: Esta etiqueta envuelve todas las propiedades y define el inicio del archivo de configuración de Hadoop.

<property>: Cada configuración en core-site.xml se define dentro de la etiqueta <property>, que representa una propiedad específica que se va a configurar.

<name>fs.default.name</name>: Es el nombre de la propiedad que se está configurando. En este caso, se establece la propiedad fs.default.name, que indica la dirección del sistema de archivos por defecto que se utilizará en Hadoop.

<value>hdfs://node-master:9000</value>: Es el valor asignado a la propiedad fs.default.name. En este caso, el valor es hdfs://node-master:9000, lo que indica que el sistema de archivos por defecto es HDFS y que el nombre del nodo maestro (NameNode) es node-master, y la comunicación se realiza a través del puerto 9000.

La importancia de esta configuración radica en que fs.default.name es una de las configuraciones más fundamentales para el correcto funcionamiento de Hadoop. Define la dirección del sistema de archivos HDFS, que es el componente central para almacenar y administrar grandes volúmenes de datos en un clúster de Hadoop. Al configurar esta propiedad correctamente, se garantiza que otros servicios y aplicaciones de Hadoop se comuniquen con el sistema de archivos adecuado.

Si la configuración no es precisa, los servicios de Hadoop pueden tener dificultades para encontrar el sistema de archivos correcto o incluso no funcionar en absoluto. Por lo tanto, es esencial asegurarse de que core-site.xml esté correctamente configurado antes de implementar y utilizar un clúster Hadoop de manera efectiva.

Ya sabiendo esto es momento de configurar el archivo hdfs-site.xml, el cual debe ser editado en cada máquina del cluster. Se usa para especificar los directorios que serán usados como NameNode y DataNode. Pero antes de crear este fichero, se deben crear los directorios que contengan el NameNone y DataNode, ejecutando los siguientes comandos:

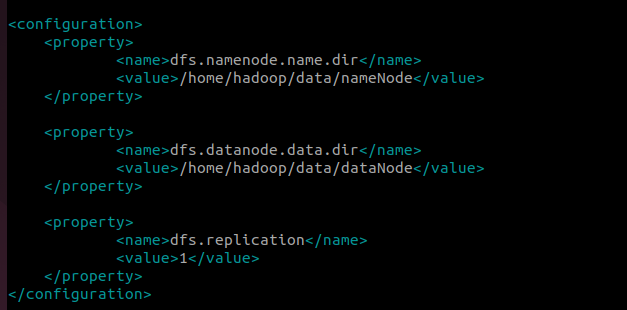
sudo mkdir -p /home/hadoop/hdfs/{nameNode,dataNode}

sudo chown -R hadoop:hadoop /home/hadoop/hdfs

Ejecutar este comando para activar la edición del archivo:

sudo nano /usr/local/hadoop/ etc/hadoop/hdfs-site.xml

Después editar el archivo para que se vea como el siguiente bloque:



La última propiedad, dfs.replication, indica cuántas veces se replican los datos en el clúster. Puede configurar 2 para duplicar todos los datos en los dos nodos. No ingrese un valor mayor que el número real de nodos trabajadores.

El archivo hdfs-site.xml es uno de los archivos de configuración en Hadoop, específicamente relacionado con el sistema de archivos distribuido de Hadoop (HDFS). A continuación, explicaré la importancia y el significado de cada elemento en la configuración proporcionada:

<property>: Este es un elemento que envuelve cada propiedad o configuración específica dentro del archivo. Cada propiedad tiene un nombre (<name>) y un valor (<value>).

<name>: El nombre de la propiedad que se está configurando. Específicamente en este contexto, cada nombre se refiere a una configuración particular del sistema de archivos distribuido de Hadoop.

<value>: El valor asociado con la propiedad específica. Este valor representa la configuración deseada para la propiedad en cuestión.

Ahora, expliquemos cada propiedad específica en la configuración:

<property>

<name>dfs.namenode.name.dir</name>

<value>/home/hadoop/data/nameNode</value>

Significado: Esta propiedad indica la ubicación en el sistema de archivos local donde se almacenarán los datos del NameNode en el clúster de Hadoop. El NameNode es el nodo maestro en HDFS que realiza un seguimiento de la ubicación y el estado de todos los bloques de datos en el clúster. Es esencial mantener los metadatos del NameNode de forma persistente, por lo que se configura una ubicación local para almacenar esta información.

<property>

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

<value>/home/hadoop/data/dataNode</value>

Significado: Esta propiedad indica la ubicación en el sistema de archivos local donde se almacenarán los bloques de datos de los DataNodes. Los DataNodes son los nodos esclavos en HDFS, que almacenan los datos en el clúster. Esta configuración especifica dónde se almacenarán físicamente los bloques de datos en cada DataNode.

<property>

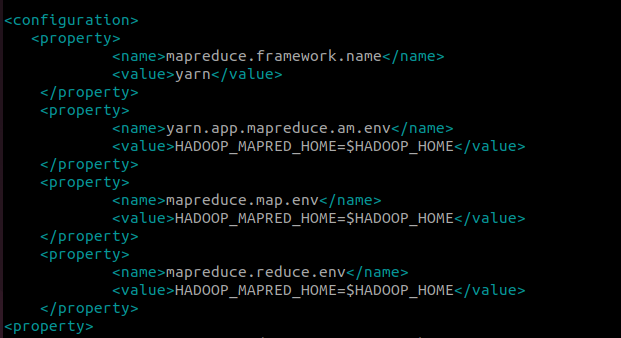
<name>dfs.replication</name>

<value>1</value>

Significado: Esta propiedad establece el factor de replicación predeterminado para los bloques de datos en HDFS. El valor 1 significa que cada bloque de datos se replicará una vez, lo que implica que habrá una copia adicional del bloque. La replicación en HDFS es importante porque proporciona tolerancia a fallos y alta disponibilidad. Si un DataNode falla, el NameNode puede acceder a copias de los bloques de datos en otros DataNodes y, por lo tanto, garantizar la continuidad del servicio y la integridad de los datos.

En resumen, la configuración proporcionada en hdfs-site.xml especifica la ubicación de los directorios de almacenamiento del NameNode y los DataNodes, así como el nivel de replicación de los bloques de datos en el sistema de archivos distribuido de Hadoop (HDFS). Estas configuraciones son fundamentales para el correcto funcionamiento y rendimiento del clúster Hadoop.

Ahora, edite el archivo mapred-site.xml, configurando YARN como el marco predeterminado para las operaciones de MapReduce:



El archivo mapred-site.xml es uno de los archivos de configuración en Hadoop, específicamente relacionado con el framework MapReduce, que es una parte fundamental de Hadoop utilizado para procesar y analizar grandes volúmenes de datos distribuidos en un clúster. A continuación, explicaré la importancia y el significado de cada elemento en la configuración proporcionada:

<property>: Al igual que en el archivo hdfs-site.xml, este es un elemento que envuelve cada propiedad o configuración específica dentro del archivo. Cada propiedad tiene un nombre (<name>) y un valor (<value>).

<name>: El nombre de la propiedad que se está configurando. Cada nombre se refiere a una configuración particular relacionada con el framework MapReduce.

<value>: El valor asociado con la propiedad específica. Este valor representa la configuración deseada para la propiedad en cuestión.

Ahora, expliquemos cada propiedad específica en la configuración:

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

Significado: Esta propiedad indica el framework que se utilizará para ejecutar los trabajos MapReduce. En este caso, el valor yarn especifica que los trabajos MapReduce se ejecutarán dentro del administrador de recursos YARN (Yet Another Resource Negotiator) de Hadoop. YARN es responsable de gestionar los recursos del clúster y programar las tareas para su ejecución.

<property>

<name>yarn.app.mapreduce.am.env</name>

<value>HADOOP\_MAPRED\_HOME=$HADOOP\_HOME</value>

Significado: Esta propiedad define el entorno para el ApplicationMaster (AM) de MapReduce cuando se ejecuta en YARN. El ApplicationMaster es responsable de negociar recursos con el ResourceManager y coordinar la ejecución de trabajos MapReduce en el clúster. En esta configuración, se está estableciendo la variable de entorno HADOOP\_MAPRED\_HOME con el valor de la variable HADOOP\_HOME, que generalmente apunta al directorio de instalación de Hadoop. Esta configuración permite que el AM y los trabajos MapReduce tengan acceso a las bibliotecas y configuraciones de Hadoop necesarias para su ejecución.

<property>

<name>mapreduce.map.env</name>

<value>HADOOP\_MAPRED\_HOME=$HADOOP\_HOME</value>

<property>

<name>mapreduce.reduce.env</name>

<value>HADOOP\_MAPRED\_HOME=$HADOOP\_HOME</value>

Significado: Estas propiedades definen el entorno para los procesos de mapeo (map) y reducción (reduce) cuando se ejecutan como parte de un trabajo MapReduce. Al igual que la propiedad anterior, se establece la variable de entorno HADOOP\_MAPRED\_HOME con el valor de HADOOP\_HOME para que los procesos de map y reduce tengan acceso a las bibliotecas y configuraciones necesarias de Hadoop.

En resumen, la configuración proporcionada en mapred-site.xml define el framework MapReduce que se utilizará (YARN en este caso) y establece variables de entorno importantes para el ApplicationMaster y los procesos de mapeo y reducción en un clúster de Hadoop. Estas configuraciones son fundamentales para el correcto funcionamiento de los trabajos MapReduce y su integración con el sistema de administración de recursos de YARN.

Luego de esto, edite el yarn-site.xml, que contiene las opciones de configuración para YARN. En el campo de valor de yarn.resourcemanager.hostname.

La configuración del archivo yarn-site.xml en Hadoop es esencial para el funcionamiento del administrador de recursos YARN (Yet Another Resource Negotiator). Cada propiedad dentro de esta configuración tiene una importancia específica para el comportamiento del clúster Hadoop. A continuación, se explica el significado y la importancia de cada elemento en la configuración proporcionada:

<property>: Este elemento engloba cada propiedad de configuración individual en el archivo yarn-site.xml. Dentro de cada <property>, se encuentran el nombre de la propiedad (<name>) y su valor (<value>).

<name>: Es el nombre de la propiedad de configuración de YARN. Define qué aspecto del sistema está configurando. En el ejemplo, los nombres de las propiedades son yarn.acl.enable, yarn.resourcemanager.hostname, y yarn.nodemanager.aux-services.

<value>: Es el valor asignado a la propiedad de configuración definida en <name>. Es lo que determina el comportamiento o la configuración del sistema para esa propiedad en particular. En el ejemplo, los valores son 0, 203.0.113.0, y mapreduce\_shuffle.

Ahora, veamos la importancia y el significado de cada elemento en la configuración:

yarn.acl.enable: Esta propiedad determina si se habilitan los controles de acceso basados en listas de control de acceso (ACL) en YARN. Cuando está establecido en 1, se habilitan los controles de acceso y se deben configurar las listas de control de acceso para los diferentes componentes de YARN (administrador de recursos y nodos). Al establecerlo en 0, como en el ejemplo, se deshabilitan los controles de acceso y se permitirá el acceso a todos los usuarios. Esta configuración puede ser útil en entornos donde no es necesario un control de acceso detallado.

yarn.resourcemanager.hostname: Esta propiedad especifica la dirección IP o el nombre de host del nodo donde se encuentra el administrador de recursos de YARN (ResourceManager). El ResourceManager es responsable de asignar los recursos a las aplicaciones que se ejecutan en el clúster. Al configurar esta propiedad, los nodos del clúster sabrán dónde buscar el ResourceManager para registrar su disponibilidad y solicitar recursos para ejecutar aplicaciones.

yarn.nodemanager.aux-services: Esta propiedad establece el nombre del servicio auxiliar (aux-service) que se ejecutará en el NodeManager. Los servicios auxiliares son procesos adicionales que se pueden ejecutar en los nodos para proporcionar características específicas a las aplicaciones. En este caso, el valor mapreduce\_shuffle indica que el servicio auxiliar "mapreduce\_shuffle" estará activo en los nodos. Este servicio es necesario para admitir la operación de mezcla (shuffle) de datos durante el procesamiento de trabajos MapReduce.

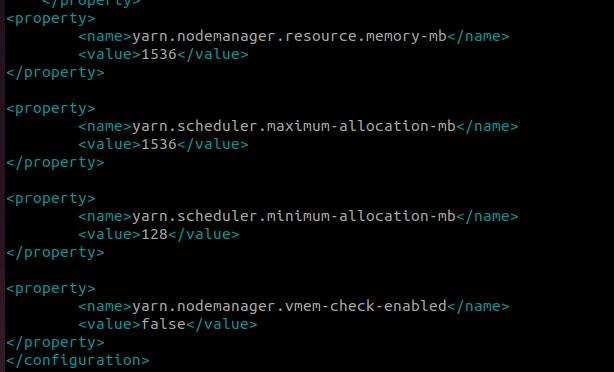
En resumen, la configuración del archivo yarn-site.xml es esencial para el funcionamiento adecuado del clúster Hadoop con YARN como administrador de recursos. Cada propiedad tiene una función específica, como habilitar o deshabilitar los controles de acceso, establecer la ubicación del ResourceManager y activar servicios auxiliares necesarios para características específicas de Hadoop, como el procesamiento de trabajos MapReduce.

Los scripts de inicio utilizan File Workers para iniciar los demonios necesarios en todos los nodos. Edite ~/hadoop/etc/hadoop/workers para incluir ambos nodos:

node-master

node1

Edite /home/hadoop/hadoop/etc/hadoop/yarn-site.xml y agregue las siguientes líneas:

<property>: Al igual que en el ejemplo anterior, este elemento engloba cada propiedad de configuración individual en el archivo yarn-site.xml. Cada propiedad tiene un nombre y un valor asociado.

yarn.nodemanager.resource.memory-mb: Esta propiedad define la cantidad máxima de memoria RAM que puede utilizar un nodo de trabajo (NodeManager) para ejecutar tareas de aplicación. El valor proporcionado, 1536 en este caso, establece el límite de memoria en megabytes (MB) que cada nodo puede asignar a las aplicaciones. Este valor es importante para evitar que una sola aplicación agote todos los recursos del nodo, lo que podría afectar negativamente a otras aplicaciones que se ejecuten en el mismo nodo.

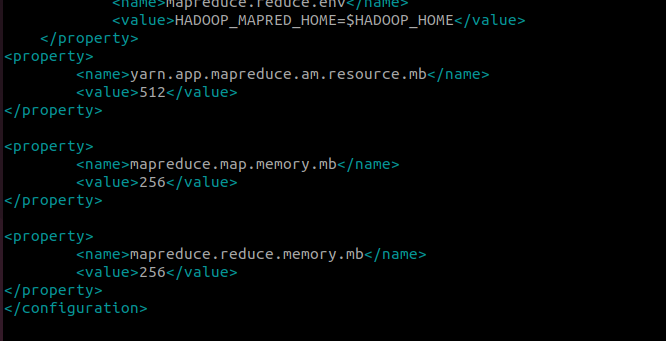
yarn.scheduler.maximum-allocation-mb: Esta propiedad establece la cantidad máxima de memoria RAM que puede solicitar una aplicación en el clúster. En este caso, el valor es 1536, lo que significa que ninguna aplicación puede solicitar más de 1536 MB de memoria. Esta configuración es otra medida de control para garantizar que ninguna aplicación pueda monopolizar los recursos del clúster solicitando una cantidad excesiva de memoria.

yarn.scheduler.minimum-allocation-mb: Esta propiedad define la cantidad mínima de memoria RAM que puede solicitar una aplicación en el clúster. Con un valor de 128, significa que las aplicaciones no pueden solicitar menos de 128 MB de memoria. Esta configuración es útil para evitar que aplicaciones muy pequeñas desperdicien recursos solicitando cantidades insignificantes de memoria.

yarn.nodemanager.vmem-check-enabled: Esta propiedad determina si se habilita la comprobación de memoria virtual (virtual memory) en el nodo de trabajo (NodeManager). Si se establece en false, como en el ejemplo, se desactiva la comprobación de la memoria virtual. Esto significa que YARN no verificará si la memoria virtual del nodo supera los límites establecidos, lo que podría permitir que las aplicaciones utilicen más memoria virtual de la que está disponible físicamente en el nodo. Sin embargo, esta configuración también podría aumentar el riesgo de que una aplicación agote la memoria del sistema.

La última propiedad deshabilita la verificación de la memoria virtual, lo que puede evitar que los contenedores se asignen correctamente con JDK8 si está habilitado.

Edite /home/hadoop/hadoop/etc/hadoop/mapred-site.xml y agregue las siguientes líneas:



Significado: Esta propiedad establece la cantidad de memoria en megabytes (MB) que se asigna al ApplicationMaster (AM) de MapReduce cuando se ejecuta en el clúster Hadoop con YARN. El ApplicationMaster es responsable de negociar recursos con el ResourceManager y coordinar la ejecución de trabajos MapReduce en el clúster. Establecer un valor adecuado para esta propiedad es importante para garantizar que el AM tenga suficiente memoria para administrar de manera eficiente los trabajos MapReduce y las tareas asociadas.

<property>

<name>mapreduce.map.memory.mb</name>

<value>256</value>

Significado: Esta propiedad establece la cantidad de memoria en megabytes (MB) que se asigna a cada tarea de mapeo (map) en un trabajo MapReduce. El valor configurado aquí define el límite máximo de memoria que un proceso de mapeo puede utilizar durante su ejecución. Establecer una cantidad adecuada de memoria para los mappers es crucial para evitar problemas de falta de memoria durante la ejecución de trabajos que implican la transformación de grandes volúmenes de datos.

<property>

<name>mapreduce.reduce.memory.mb</name>

<value>256</value>

Significado: Esta propiedad establece la cantidad de memoria en megabytes (MB) que se asigna a cada tarea de reducción (reduce) en un trabajo MapReduce. Al igual que la propiedad anterior, el valor configurado aquí define el límite máximo de memoria que un proceso de reducción puede utilizar durante su ejecución. Asegurarse de que haya suficiente memoria disponible para las tareas de reducción es importante para evitar cuellos de botella y asegurar un rendimiento óptimo en la fase de reducción del trabajo.

Copie los archivos binarios de Hadoop en los nodos trabajadores:

cd /home/hadoop/

scp hadoop-\*.tar.gz node1:/home/Hadoop

HDFS debe formatearse como cualquier sistema de archivos clásico. En node-master, ejecute el siguiente comando:

hdfs namenode -format

Inicie HDFS ejecutando el siguiente script desde node-master:

start-dfs.sh

Los servicios al final quedaron así:  
A black screen with white text

Description automatically generated

A black screen with white text

Description automatically generated

Pasos Preparar y Ejecutar un mapper y reducer de Python en lo configurado anteriormente

Crear un Directorio Llamado ventas:

mkdir prueba

Descargar el archivo de DataSetVentas.txt

Crear la siguiente carpeta en el HDFS

hdfs dfs -mkdir -p /user/bigdata/

Una vez tengamos todos los ficheros correctamente creados los subiremos a HDFS con el siguiente comando:

hdfs dfs -put ventas/ /user/bigdata/

Crear un Directorio Llamado codigo:

mkdir código

Copiar los archivos mapper1.py y reducer1.py al directorio "codigo" y luego convertir los mismos a Ejecutable

sudo chmod +x mapper1.py

sudo chmod +x reducer1.py

Subir los archivos el directorio "codigo" al directorio /user/bigdata/ del HDFS:

hdfs dfs -put codigo/ /user/bigdata/

Para Ejecutar MapReduce, utilizando Hadoop Streaming:

mapred streaming -input /user/hadoop/ventas \

-output /user/hadoop/resultado/stream02 \

-mapper /home/hadoop/codigo/mapper1.py \

-reducer /home/hadoop/codigo/reducer1.py

Ver resultado:

hdfs dfs -cat /user/hadoop/resultado/stream02/part-00000

A computer screen shot of a number

Description automatically generated

A computer screen shot of a black screen

Description automatically generated

Utilizando el DataSet presentado en el punto anterior, desarrollar Mappers and Reducers que se encarguen de determinar las siguientes informaciones:

a. Mostrar el total de compras enviado a cada estado, utilizando First Class como Modo de Envío. Los datos a mostrar son:

Estado CantidadOrdenes MontoTotalOrdenes

Mapper total de compras:

#!/usr/bin/env python3  
  
import sys  
  
for line in sys.stdin:  
 line = line.strip()  
 cols = line.split("\t")  
 ship\_mode = cols[4]  
 state = cols[10]  
 sales = cols[17]  
   
 if cols[0] != 'Row ID' and ship\_mode == 'First Class':  
 print(f"{state}\t1\t{sales}")

Reducer total de compras

#!/usr/bin/env python3  
  
import sys  
  
state\_total\_orders = {}  
state\_total\_sales = {}  
  
for line in sys.stdin:  
 line = line.strip()  
 state, order\_count, total\_sales = line.split("\t")  
 order\_count = int(order\_count)  
 total\_sales = float(total\_sales)  
   
 if state in state\_total\_orders:  
 state\_total\_orders[state] += order\_count  
 state\_total\_sales[state] += total\_sales  
 else:  
 state\_total\_orders[state] = order\_count  
 state\_total\_sales[state] = total\_sales  
  
for state in state\_total\_orders:  
 print(f"{state}\t\t{state\_total\_orders[state]}\t{state\_total\_sales[state]}")

Explicacion

El código del Mapper (Mapper total de compras) recibe líneas de entrada desde la entrada estándar (stdin). Cada línea representa una compra con diferentes campos separados por tabulaciones. El código realiza lo siguiente:

Importa el módulo sys para leer las líneas de entrada y realizar operaciones de entrada y salida.

Itera sobre cada línea en la entrada estándar usando un bucle for.

Elimina los espacios en blanco al inicio y al final de la línea usando line.strip().

Divide la línea en columnas utilizando el carácter de tabulación como separador y almacena las columnas en la lista cols.

Extrae los valores de interés de las columnas, como el modo de envío, el estado y las ventas.

Luego, comprueba si la primera columna (cols[0]) no es igual a 'Row ID' para evitar procesar la fila de encabezado.

Si el modo de envío es 'First Class', imprime una línea de salida que contiene el estado, el número 1 (indicando una compra), y el monto de ventas asociado con esa compra.

El código del Reducer (Reducer total de compras) recibe la salida del Mapper, que está ordenada por estado. El Reducer realiza lo siguiente:

Importa el módulo sys para leer las líneas de entrada y realizar operaciones de entrada y salida.

Crea dos diccionarios vacíos: state\_total\_orders y state\_total\_sales. Estos diccionarios se utilizarán para almacenar el total de órdenes y ventas para cada estado.

Itera sobre cada línea en la entrada estándar usando un bucle for.

Divide la línea en tres valores: el estado, el número de órdenes y el monto total de ventas, utilizando el carácter de tabulación como separador.

Convierte el número de órdenes en un entero y el monto total de ventas en un número de punto flotante.

Luego, comprueba si el estado ya está presente en los diccionarios state\_total\_orders y state\_total\_sales. Si es así, suma los nuevos valores al estado existente. Si no, crea una nueva entrada en los diccionarios para el estado con los valores actuales.

Finalmente, el Reducer imprime, para cada estado, una línea de salida que contiene el estado, el número total de órdenes y el monto total de ventas.

Resultado:

A computer screen shot of a number

Description automatically generated

b. Mostrar la Categoría con mayor y menor cantidad de órdenes. Los datos a mostrar son:

Categoría CantidadOrdenes MontoTotalOrdenes

Mapper categorías

#!/usr/bin/env python3  
  
import sys  
  
for line in sys.stdin:  
 try:  
 line = line.strip()  
 cols = line.split("\t")  
 categoria = cols[14]  
 monto = float(cols[len(cols) - 1])  
 if cols[0] != 'Row ID':  
 print(f"{categoria}\t1\t{monto}")  
 except ValueError:  
 continue

Reducer categorias

#!/usr/bin/env python3  
  
import sys  
  
categoria\_actual = None  
cantidad\_ordenes\_categoria = 0  
monto\_total\_categoria = 0  
  
categoria\_mayor\_ordenes = None  
cantidad\_mayor\_ordenes = 0  
monto\_mayor\_ordenes = 0  
  
categoria\_menor\_ordenes = None  
cantidad\_menor\_ordenes = float('inf')  
monto\_menor\_ordenes = 0  
  
for line in sys.stdin:  
 line = line.strip()  
 categoria, cantidad\_ordenes, monto = line.split("\t")  
 cantidad\_ordenes = int(cantidad\_ordenes)  
 monto = float(monto)  
  
 if categoria\_actual == categoria:  
 cantidad\_ordenes\_categoria += cantidad\_ordenes  
 monto\_total\_categoria += monto  
 else:  
 if categoria\_actual:  
 if cantidad\_ordenes\_categoria > cantidad\_mayor\_ordenes:  
 categoria\_mayor\_ordenes = categoria\_actual  
 cantidad\_mayor\_ordenes = cantidad\_ordenes\_categoria  
 monto\_mayor\_ordenes = monto\_total\_categoria  
  
 if cantidad\_ordenes\_categoria < cantidad\_menor\_ordenes:  
 categoria\_menor\_ordenes = categoria\_actual  
 cantidad\_menor\_ordenes = cantidad\_ordenes\_categoria  
 monto\_menor\_ordenes = monto\_total\_categoria  
  
 categoria\_actual = categoria  
 cantidad\_ordenes\_categoria = cantidad\_ordenes  
 monto\_total\_categoria = monto  
  
# Para el último registro  
if categoria\_actual:  
 if cantidad\_ordenes\_categoria > cantidad\_mayor\_ordenes:  
 categoria\_mayor\_ordenes = categoria\_actual  
 cantidad\_mayor\_ordenes = cantidad\_ordenes\_categoria  
 monto\_mayor\_ordenes = monto\_total\_categoria  
  
 if cantidad\_ordenes\_categoria < cantidad\_menor\_ordenes:  
 categoria\_menor\_ordenes = categoria\_actual  
 cantidad\_menor\_ordenes = cantidad\_ordenes\_categoria  
 monto\_menor\_ordenes = monto\_total\_categoria  
  
print(f"{categoria\_mayor\_ordenes}\t{cantidad\_mayor\_ordenes}\t{monto\_mayor\_ordenes}")  
print(f"{categoria\_menor\_ordenes}\t{cantidad\_menor\_ordenes}\t{monto\_menor\_ordenes}")

Explicación

El código del Mapper (Mapper categorías) procesa líneas de entrada desde la entrada estándar (stdin) que contienen información sobre compras con diferentes campos separados por tabulaciones. El código realiza lo siguiente:

Importa el módulo sys para leer las líneas de entrada y realizar operaciones de entrada y salida.

Itera sobre cada línea en la entrada estándar usando un bucle for.

Elimina los espacios en blanco al inicio y al final de la línea usando line.strip().

Divide la línea en columnas utilizando el carácter de tabulación como separador y almacena las columnas en la lista cols.

Extrae la categoría del producto y el monto de la compra de las columnas.

Comprueba si la primera columna (cols[0]) no es igual a 'Row ID' para evitar procesar la fila de encabezado.

Si no es la fila de encabezado, imprime una línea de salida que contiene la categoría, un 1 (indicando una compra) y el monto de la compra asociado con esa categoría.

El código del Reducer (Reducer categorias) recibe la salida del Mapper y realiza un análisis de las categorías para encontrar la categoría con la mayor y menor cantidad de órdenes, junto con sus montos totales. El Reducer realiza lo siguiente:

Importa el módulo sys para leer las líneas de entrada y realizar operaciones de entrada y salida.

Inicializa algunas variables para mantener un seguimiento de la categoría actual, la cantidad total de órdenes y el monto total para esa categoría. También se inicializan variables para mantener un seguimiento de las categorías con la mayor y menor cantidad de órdenes y sus respectivos montos.

Itera sobre cada línea en la entrada estándar usando un bucle for.

Divide la línea en tres valores: la categoría, la cantidad de órdenes y el monto, utilizando el carácter de tabulación como separador.

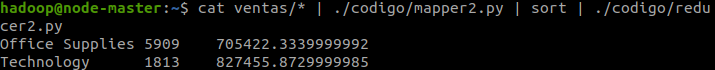
Convierte la cantidad de órdenes en un entero y el monto en un número de punto flotante.

Compara la categoría actual con la categoría anterior. Si es la misma categoría, suma la cantidad de órdenes y el monto al total actual. Si es una nueva categoría, compara las cantidades totales de órdenes para determinar si es mayor o menor que las categorías anteriores y actualiza las variables correspondientes.

Continúa iterando hasta que todas las líneas hayan sido procesadas.

Finalmente, imprime la categoría con la mayor cantidad de órdenes y su monto total, y luego imprime la categoría con la menor cantidad de órdenes y su monto total.

Resultado:



c. Mostrar la cantidad de órdenes enviadas por cada Modo de Envío y Estado. Los datos a mostrar son:

ModoEnvio Estado CantidadOrdenes

Mapper cantidad ordenes

#!/usr/bin/env python3  
  
import sys  
  
for line in sys.stdin:  
 line = line.strip()  
 cols = line.split("\t")  
 ship\_mode = cols[4]  
 state = cols[10]  
 if cols[0] != 'Row ID':  
 print(f"{ship\_mode}\t{state}\t{1}")

Reducer cantidad ordenes

#!/usr/bin/env python3  
  
import sys  
  
current\_key = None  
order\_count = 0  
  
for line in sys.stdin:   
 line = line.strip()  
 ship\_mode, state, count = line.split("\t")  
   
 try:  
 count = int(count)  
 except ValueError:  
 continue  
   
 if current\_key == (ship\_mode, state):  
 order\_count += count  
 else:  
 if current\_key is not None:  
 print(f"{current\_key[0]}\t{current\_key[1]}\t{order\_count}")  
 current\_key = (ship\_mode, state)  
 order\_count = count  
  
# Don't forget to print the last record  
if current\_key is not None:  
 print(f"{current\_key[0]}\t{current\_key[1]}\t{order\_count}")

Explicacion:

El código del Mapper (Mapper cantidad ordenes) procesa líneas de entrada desde la entrada estándar (stdin) que contienen información sobre órdenes con diferentes campos separados por tabulaciones. El código realiza lo siguiente:

Importa el módulo sys para leer las líneas de entrada y realizar operaciones de entrada y salida.

Itera sobre cada línea en la entrada estándar usando un bucle for.

Elimina los espacios en blanco al inicio y al final de la línea usando line.strip().

Divide la línea en columnas utilizando el carácter de tabulación como separador y almacena las columnas en la lista cols.

Extrae el modo de envío y el estado de las columnas.

Comprueba si la primera columna (cols[0]) no es igual a 'Row ID' para evitar procesar la fila de encabezado.

Si no es la fila de encabezado, imprime una línea de salida que contiene el modo de envío, el estado y el número 1 (indicando una orden).

El código del Reducer (Reducer cantidad ordenes) recibe la salida del Mapper y suma las órdenes para cada combinación de modo de envío y estado. El Reducer realiza lo siguiente:

Importa el módulo sys para leer las líneas de entrada y realizar operaciones de entrada y salida.

Inicializa algunas variables para mantener un seguimiento de la clave actual (que es una tupla que contiene el modo de envío y el estado) y el contador de órdenes para esa combinación.

Itera sobre cada línea en la entrada estándar usando un bucle for.

Divide la línea en tres valores: el modo de envío, el estado y el número de órdenes, utilizando el carácter de tabulación como separador.

Convierte el número de órdenes en un entero.

Compara la clave actual con la clave anterior. Si es la misma clave, suma el número de órdenes al contador actual. Si es una nueva clave, imprime la clave anterior junto con el número total de órdenes para esa clave.

Continúa iterando hasta que todas las líneas hayan sido procesadas.

Finalmente, imprime la última clave junto con el número total de órdenes para esa clave.

Resultado:

A computer screen shot of a computer

Description automatically generated

d. Mostrar las 2 SubCategorías que más se han vendido en Diciembre 2017 y Diciembre 2018. Los datos a mostrar son:

SubCategoría MontoTotalOrdenes

Mapper de subcategorías

#!/usr/bin/env python3  
  
import sys  
from datetime import datetime  
  
format = '%d/%m/%Y'  
  
for line in sys.stdin:  
 line = line.strip()  
 cols = line.split("\t")  
 categoria = cols[14]  
 subcategoria = cols[15]  
 monto = cols[-1]  
 if cols[0] != 'Row ID':  
 order\_date = datetime.strptime(cols[2], format)  
  
 if order\_date.year in [2017, 2018] and order\_date.month == 12:  
 print(f"{subcategoria}\t{monto}")

Reducer de subcategorías

#!/usr/bin/env python3  
  
import sys  
  
subcategoria\_actual = None  
monto\_total\_actual = 0  
top\_subcategorias = []  
  
def agregar\_subcategoria(subcategoria, monto):  
 global top\_subcategorias  
 top\_subcategorias.append((subcategoria, monto))  
 top\_subcategorias.sort(key=lambda x: x[1], reverse=True)  
 top\_subcategorias = top\_subcategorias[:2]  
  
for line in sys.stdin:  
 line = line.strip()  
 subcategoria, monto = line.split("\t")  
 monto = float(monto)  
  
 if subcategoria\_actual == subcategoria:  
 monto\_total\_actual += monto  
 else:  
 if subcategoria\_actual is not None:  
 agregar\_subcategoria(subcategoria\_actual, monto\_total\_actual)  
 subcategoria\_actual = subcategoria  
 monto\_total\_actual = monto  
  
if subcategoria\_actual is not None:  
 agregar\_subcategoria(subcategoria\_actual, monto\_total\_actual)  
  
for subcategoria, monto in top\_subcategorias:  
 print(f"{subcategoria}\t{monto}")

Explicacion:

El código del Mapper (Mapper de subcategorías) procesa líneas de entrada desde la entrada estándar (stdin) que contienen información sobre órdenes con diferentes campos separados por tabulaciones. El código realiza lo siguiente:

Importa el módulo sys para leer las líneas de entrada y realizar operaciones de entrada y salida.

Importa el módulo datetime para trabajar con fechas.

Define el formato de fecha como '%d/%m/%Y'.

Itera sobre cada línea en la entrada estándar usando un bucle for.

Elimina los espacios en blanco al inicio y al final de la línea usando line.strip().

Divide la línea en columnas utilizando el carácter de tabulación como separador y almacena las columnas en la lista cols.

Extrae la categoría, subcategoría y monto de las columnas.

Comprueba si la primera columna (cols[0]) no es igual a 'Row ID' para evitar procesar la fila de encabezado.

Si no es la fila de encabezado, convierte la fecha de la columna 2 a un objeto de fecha y hora utilizando el formato definido.

Comprueba si el año de la fecha está en la lista [2017, 2018] y si el mes es diciembre (mes número 12).

Si se cumple esta condición, imprime una línea de salida que contiene la subcategoría y el monto de la venta asociado con esa subcategoría.

El código del Reducer (Reducer de subcategorías) recibe la salida del Mapper y encuentra las dos subcategorías con los montos de ventas más altos durante el mes de diciembre de 2017 y 2018. El Reducer realiza lo siguiente:

Importa el módulo sys para leer las líneas de entrada y realizar operaciones de entrada y salida.

Inicializa algunas variables para mantener un seguimiento de la subcategoría actual y el monto total actual para esa subcategoría.

También se inicializa una lista llamada top\_subcategorias para almacenar las dos subcategorías con los montos más altos.

Define una función llamada agregar\_subcategoria que agrega una subcategoría a la lista top\_subcategorias y luego la ordena de mayor a menor según el monto de ventas y mantiene solo las dos subcategorías con los montos más altos.

Itera sobre cada línea en la entrada estándar usando un bucle for.

Divide la línea en dos valores: la subcategoría y el monto, utilizando el carácter de tabulación como separador.

Convierte el monto en un número de punto flotante.

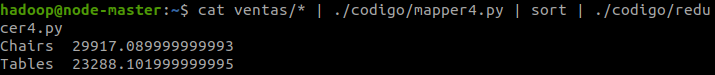
Compara la subcategoría actual con la subcategoría anterior. Si es la misma subcategoría, suma el monto al monto total actual. Si es una nueva subcategoría, llama a la función agregar\_subcategoria para agregar la subcategoría anterior a la lista top\_subcategorias.

Continúa iterando hasta que todas las líneas hayan sido procesadas.

Finalmente, llama a la función agregar\_subcategoria para agregar la última subcategoría a la lista top\_subcategorias.

Imprime las dos subcategorías con los montos de ventas más altos en el formato: "subcategoría monto".

Resultado:



e. Mostrar el Total Ordenado para los Meses Enero y Marzo en la Ciudades de Wilmington, Houston y New York City. Los datos a mostrar son:

Mes Ciudad MontoTotalOrdenes

Mapper total ordenado

#!/usr/bin/env python3  
  
import sys  
from datetime import datetime  
  
format = '%d/%m/%Y'  
meses = ["", "Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre"]  
  
for line in sys.stdin:  
 line = line.strip()  
 cols = line.split("\t")  
 ciudad = cols[9]  
 mes = None  
 monto = None  
  
 if cols[0] != 'Row ID':  
 dt = datetime.strptime(cols[2], format)  
 mes = meses[dt.month]  
 monto = float(cols[len(cols) - 1])  
  
 if mes in ["Enero", "Marzo"] and ciudad in ["Wilmington", "Houston", "New York City"]:  
 print(f"{mes}\t{ciudad}\t{monto}")

Reducer total ordenado

#!/usr/bin/env python3  
  
import sys  
  
mes\_actual = None  
ciudad\_actual = None  
monto\_total\_actual = 0  
  
for line in sys.stdin:  
 line = line.strip()  
 mes, ciudad, monto = line.split("\t")  
 monto = float(monto)  
  
 if mes\_actual == mes and ciudad\_actual == ciudad:  
 monto\_total\_actual += monto  
 else:  
 if mes\_actual and ciudad\_actual:  
 print(f"{mes\_actual}\t{ciudad\_actual}\t{monto\_total\_actual}")  
 mes\_actual = mes  
 ciudad\_actual = ciudad  
 monto\_total\_actual = monto  
  
# Imprimir el último resultado fuera del loop  
if mes\_actual and ciudad\_actual:  
 print(f"{mes\_actual}\t{ciudad\_actual}\t{monto\_total\_actual}")

Explicación:

El código del Mapper (Mapper total ordenado) procesa líneas de entrada desde la entrada estándar (stdin) que contienen información sobre órdenes con diferentes campos separados por tabulaciones. El código realiza lo siguiente:

Importa el módulo sys para leer las líneas de entrada y realizar operaciones de entrada y salida.

Importa el módulo datetime para trabajar con fechas.

Define el formato de fecha como '%d/%m/%Y'.

Define una lista de nombres de meses.

Itera sobre cada línea en la entrada estándar usando un bucle for.

Elimina los espacios en blanco al inicio y al final de la línea usando line.strip().

Divide la línea en columnas utilizando el carácter de tabulación como separador y almacena las columnas en la lista cols.

Extrae la ciudad de la columna 9 (índice 9).

Inicializa las variables mes y monto como None.

Comprueba si la primera columna (cols[0]) no es igual a 'Row ID' para evitar procesar la fila de encabezado.

Si no es la fila de encabezado, convierte la fecha de la columna 2 (índice 2) a un objeto de fecha y hora utilizando el formato definido.

Se asigna el nombre del mes correspondiente a la variable mes y convierte el monto de la última columna (cols[len(cols) - 1]) a un número de punto flotante y lo asigna a la variable monto.

Comprueba si mes tiene el valor de Enero o Marzo y si la ciudad se encuentra en la lista ["Wilmington", "Houston", "New York City"].

Si ambas condiciones son verdaderas, imprime una línea de salida que contiene el nombre del mes, la ciudad y el monto de la orden asociado con esa combinación.

El código del Reducer (Reducer total ordenado) recibe la salida del Mapper y suma los montos de órdenes para cada combinación de mes y ciudad. El Reducer realiza lo siguiente:

Importa el módulo sys para leer las líneas de entrada y realizar operaciones de entrada y salida.

Inicializa algunas variables para mantener un seguimiento del mes actual, la ciudad actual y el monto total actual para esa combinación.

Itera sobre cada línea en la entrada estándar usando un bucle for.

Divide la línea en tres valores: el mes, la ciudad y el monto, utilizando el carácter de tabulación como separador.

Convierte el monto en un número de punto flotante.

Compara el mes actual y la ciudad actual con los valores del mes y la ciudad de la línea actual. Si ambos son iguales, suma el monto al monto total actual.

Si alguno de los valores es diferente, imprime la combinación anterior junto con el monto total para esa combinación y actualiza las variables mes\_actual, ciudad\_actual y monto\_total\_actual con los valores de la nueva combinación.

Continúa iterando hasta que todas las líneas hayan sido procesadas.

Finalmente, imprime la última combinación junto con el monto total para esa combinación.

Resultado:

A computer screen with numbers and letters

Description automatically generated

**Conclusión**

La arquitectura de un clúster Hadoop es una solución poderosa para el procesamiento y almacenamiento de grandes volúmenes de datos distribuidos. Comprender los diferentes componentes que conforman este sistema es fundamental para su configuración y despliegue exitoso. La combinación de nodos maestros y trabajadores garantiza una gestión eficiente del sistema de archivos distribuidos y una óptima ejecución de los trabajos YARN.

En este proyecto, se ha aprendido cómo instalar Java y Hadoop en modo distribuido, y a configurar las variables de entorno y los archivos de configuración necesarios. Además, se presentaron ejemplos prácticos de ejecución de un job MapReduce utilizando Hadoop Streaming para obtener información relevante sobre distintos atributos del data set de ventas.

La arquitectura de un clúster Hadoop ofrece un enfoque escalable y eficiente para el procesamiento y análisis de grandes volúmenes de datos. Con esta configuración y conocimiento, los profesionales de la informática y los científicos de datos pueden aprovechar el poder de Hadoop para abordar problemas complejos y realizar análisis de datos a gran escala.