Institución de Educación Superior
UCEVA
Unidad Central del Valle del Cauca

MAP REDUCE Y HIVE

Tema 5

Institución de Educación Superior
UCEVA
Unidad Central del Valle del Cauca

Motivación

Imaginen que tenemos que contar todas las palabras de una biblioteca con 1 millón de libros. ¿Lo haría una sola persona o sería mejor dividir el trabajo? ¿Cómo hacerlo?

Paradigma MapReduce

- Modelo de programación paralela/distribuida usada para escribir aplicaciones que procesan grandes cantidades de datos de manera confiable.

- Cuando ***los enfoques tradicionales*** de procesamiento de grandes volúmenes de datos ***comenzaron a alcanzar sus límites*** surgió un nuevo paradigma de análisis de datos basado en el uso de sistemas de computación distribuida.

- Puede trabajar con datos estructurados y no estructurados.

Paradigma MapReduce

- Limitaciones de los entornos tradicionales:
  - El programador es quien resuelve problemas de comunicación y sincronización.
  - Cuellos de botella en el acceso a datos
  - Procesamiento centralizado poco escalable.
  - Fallos parciales.

Proceso MapReduce

- ***Mapear y reducir*** son dos conceptos basados en la ***programación funcional*** donde la salida de la función se basa únicamente en la entrada.

- *Se deben implementar dos funciones:*
  - ***Map:*** procesa entrada y genera pares intermedios clave/valor (key/value).
  - ***Reduce:*** agrupa (merge) todos los pares clave/valor asociados con una misma clave y realiza una operación de agregación .

Institución de Educación Superior
UCEVA
Unidad Central del Valle del Cauca

Proceso MapReduce

- Al igual que en una función matemática, ***ƒ (x) = y***, y depende de x. Proporciona una función u operación para un mapa y la reduce.

- ***MapReduce*** simplifica la ejecución de código en paralelo pues solo ***necesita crear, asignar y reducir tareas***, y no tiene que preocuparse por múltiples subprocesos, sincronización o problemas de concurrencia.

Componentes

- ***Función map() – Mapper:***
  - El sistema toma los datos de entrada y los divide en pequeños fragmentos.
  - Crea fragmentos de datos (k,v) – parejas clave-valor
  - Agrupa las parejas ordenadas por cada clave formando nuevos grupos.

- ***Función reduce() – Reducer:***
  - Recibe cada grupo formado por el mapper
  - Combina los grupos formados por el mapper:
    Reduce (k, List(v)) → list(v')
  - Produce una lista clave-valor definitiva

Componentes

- Elementos para modelar los datos: conjuntos (datasets), registros y pares clave-valor (key-value).

- Dataset: colección de registros, usualmente procesada en paralelo en una plataforma distribuida (ejemplos: bloques de HDFS, RDD de Spark, etc.)

- Registro: Se representa en la forma de par clave-valor (key-value).

- Clave-valor: representan atributos de los datos.
  - ***Clave***: identificador del atributo (por ejemplo, el nombre de un atributo)
  - ***Valor***: dato correspondiente a la clave, puede ser escalar(número, string, etc.) o complejo(lista de objetos, etc.).

Usos

- Google:
  - Construcción de índices para el buscador
  - Clustering de artículos en Google News
  - Búsqueda de rutas en Google Maps
  - Traducción estadística

- Facebook:
  - Minería de Datos
  - Optimización de ADS
  - Detección de SPAM
  - Gestión de logs

- Investigación: Bioinformática, física de partículas, simulación climática.

Arquitectura

Proceso:

- Dividir las entradas en M particiones de tamaño entre 64 y 128 MB. El programa MapReduce se comienza a instanciar en las diversas máquinas del clúster.

- Una de las copias del programa toma el papel de "maestro“ y las otras copias funcionan como "workers“, recibiendo la asignación de sus tareas desde el master.

- Un worker que tenga asignada una tarea específica de map() toma como entrada la partición que le corresponda y produce los pares (clave, val para crear una salida.

Arquitectura

Proceso:

- Periódicamente, los ***pares clave-valor*** almacenados en el buffer se ***escriben en el disco local*** y son pasados al master

- Los ***pares recibidos*** por el master son redirigidos a los workers que tienen ***las tareas de reduce()***

- Cuando un worker de tipo reduce es notificado por el master con la localización de una partición, emplea llamadas remotas para hacer lecturas de la información almacenada en los discos duros de los diversos workers de tipo map().

Arquitectura

Proceso:

- El worker de tipo reduce() ordena la entrada recibida e itera sobre el conjunto de valores ordenados intermedios, y lo hace por cada una de las claves únicas encontradas.

- Cuando todas las tareas map() y reduce() se han completado, el master retorna el control al código de un usuario.

- La tarea Map-Reduce queda definida por dos funciones:
- map:(f,{(k1;v1)})→ {(k2;v2)}
- reduce:(k2;{v2}) → {(k3;v3)}

Ejemplo WordCount

- Conteo de Palabras en un Gran Conjunto de Textos:

- Map: Cada nodo toma una porción del texto y emite pares clave-valor donde la clave es una palabra y el valor es 1 (por ejemplo, "perro": 1).

- Reduce: Los nodos agrupan todos los pares con la misma palabra y suman los valores, resultando en un conteo total de cada palabra en el texto.

Ejemplo Análisis de Logs

- Análisis de Logs de Servidor:

- Map: Cada nodo procesa un fragmento de los archivos de log y emite pares clave-valor donde la clave puede ser una URL y el valor el número de veces que ha sido accedida.

- Reduce: Se suman las cantidades para cada URL, obteniendo un resumen de las URLs más visitadas.

Ejemplo Redes Sociales

- Procesamiento de Redes Sociales:

- Map: Cuando se analiza la actividad de los usuarios en una red social, cada nodo puede procesar datos de actividad individual y emitir pares clave-valor como {usuario: número de interacciones}.

- Reduce: Se agrupan las interacciones de cada usuario para obtener métricas agregadas, como la actividad total de un usuario.

Clase WordCount

- Con Map-reduce puede asignar y reducir datos en función de una variedad de criterios.

- La clase JavaWordCount, como su nombre indica, asigna (extrae) palabras de una fuente de entrada y reduce (resume) los resultados, devolviendo un recuento de cada palabra.

- La salida será un archivo de texto con una lista de palabras y sus frecuencias de ocurrencia en los datos de entrada.

Clase WordCount

Entrada:

$K_1 = \Phi$, $V_1 = línea$

Salida:

pares (palabra, No. ocurrencias)

Seudocódigo del *MAP*:

```
map (key , value):
// key: vacío, value: línea de texto
        for each row w in value emit (w, 1)
```

Seudocódigo del *REDUCE*:

```
reduce (key, values):
//key: palabra;  values: un iterador sobre los 1
        emit (key, sum(values))
```

Clase WordCount

La función ***map():***

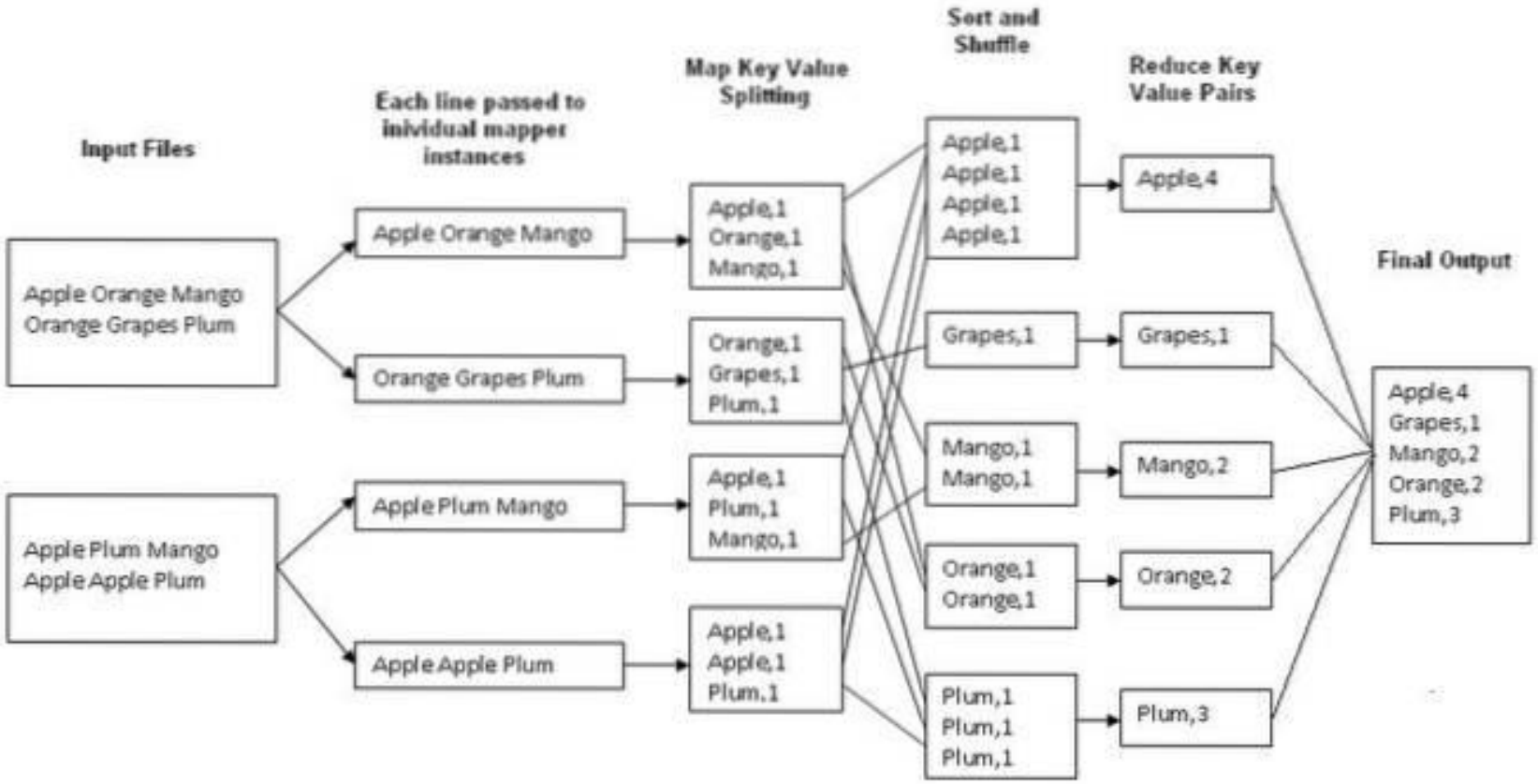
Divide un documento en palabras (lo tokeniza) mediante el empleo de un analizador léxico

Emite una serie de tuplas de la forma (clave, valor) donde la clave es la palabra y el valor es "1".

Por ejemplo, del documento "La casa de la pradera" la función map retornaría: ("la", "1"), ("casa", "1"), ("de", "1"), ("la", "1"), ("pradera", "1").

La función ***reduce()*** recibe la salida de la función anterior y sólo necesita la ***suma de todos los valores de su entrada*** para encontrar el total de las apariciones de esa palabra.

Clase WordCount



Ejemplo: Análisis de ventas por región

Objetivo: Analizar un conjunto de datos de ventas para encontrar los ingresos totales por región, dado un archivo en formato CSV.

```
id,region,producto,precio,cantidad
1,Centro,Televisor,500,2
2,Norte,Laptop,700,1
3,Sur,Teléfono,300,3
```

Mapper: Extrae (region, precio * cantidad).

Reducer: Suma los ingresos por región y genera (region, ingresos totales)

Ejemplo: Análisis de ventas por región

- Salida del Mapper:

```
Centro  1000.0
Norte   700.0
Sur 900.0
Centro  1000.0
Norte   600.0
Sur 1600.0
```

Salida del Reducer:

```
Centro  2000.0
Norte   1300.0
Sur 2500.0
```

Ejecución práctica

En la práctica ejecutaremos el proceso WorCount de Map-Reduce para identificar las diferentes cadenas de caracteres presentes en un conjunto de archivos y la cantidad de ocurrencias de éstas.

Para conseguirlo debe ejecutar los siguientes pasos:

1. Crear ubicaciones de entrada en HDFS:

   wordcount/input (Contendrá en HDFS el conjunto de archivos a procesar)

2. Al ejecutarse, el sistema crea automáticamente:

   wordcount/output (Contendrá la salida correspondiente a la lista de parejas *clave/valor*)

3. En la máquina virtual ubique la aplicación "***WordCount***". Los archivos de texto suministrados por el profesor deben ser copiados en el directorio *input* creado en HDFS.

Ejecución práctica

4. Ejecute la aplicación WordCount desde el archivo JAR, pasando las rutas a los directorios de entrada y salida en HDFS:

```
hadoop jar wordcount.jar org.myorg.WordCount /user/cloudera/wordcount/input /user/cloudera/wordcount/output
```

5. Verifique que en directorio output creado en HDFS se hayan generado dos archivos, que corresponden al resultado del procesamiento Map-Reduce.

6. Mueva al entorno local el archivo Part-0000 generado en la carpeta *output* de HDFS

7. Verifique el contenido del archivo e interprete los resultados.

APACHE HIVE

Es una infraestructura de almacenamiento de datos implementada en ***Hadoop***.

Resume grandes volúmenes de datos y facilita las consultas y análisis de los mismos.

Inicialmente desarrollado por Facebook, después la *Apache Software Foundation* continuó su desarrollo.

Utilizada por empresas como Netflix y Amazon (Amazon Web Services)

Conceptos Iniciales

Hive no es

- Una base de datos relacional
- Un diseño para *OnLine Transaction Processing* (*OLTP*)
- Un lenguaje en tiempo real para ejecución consultas y actualizaciones a nivel de fila.

Conceptos Iniciales

Características de Hive

- Esquema que almacena información en una base de datos y se procesan los datos en *HDFS*.
- Proporciona un lenguaje de consulta tipo *SQL*: ***HiveQL*** o ***HQL***.
- Es familiar, rápido, escalable y extensible.
- Hive convierte la consulta ***HQL a Jobs distribuidos***.

Hive Data Warehouse

Antes: se usaban las mismas bases de datos para procesar las transacciones y para hacer consultas analíticas.

Ahora: Se separan las consultas analíticas en una base de datos distinta → ***Data Warehouse***.

Data Warehouse contiene copias de *solo lectura* de todos los datos en los sistemas transaccionales y operacionales (**OLTP**)

Proceso ETL: Extracción – Transformación – Limpieza

Estructura de datos HIVE

- Provee una estructura de tablas basada en HDFS

- Soporta tres tipos de estructura:
  - Tablas externas
    - Similares a las tablas SQL, corresponden a directorios HDFS
    - Al eliminar la tabla, los datos continúan en HDFS

  - Particiciones
    - División de las tablas en subdirectorios
    - Mejora el rendimiento de las consultas en el filtrado de la cláusula *Where.*

  - Buckets
    - Particiones Hash
    - Optimizan las operaciones tipo Join

Tipos de Dato

Todos los tipos de datos en Hive se clasifican en cuatro características, de la siguiente manera:

- Tipos de columna:
  - TINYINT (Ej. 10Y), SMALLINT (Ej. 10S), INT (Ej. 10), BIGINT (Ej. 10L), FLOAT, DOUBLE
  - VARCHAR, CHAR, STRING
  - TIMESTAMP
  - DECIMAL (M,N) – Ej. DECIMAL (10,2)
- Valores Null
- Tipos complejos:
  - ARRAY <DATA_TYPE>
  - STRUCT <COL_NAME, DATA_TYPE>
  - MAPS <DATA_TYPE, DATA_TYPE>

Ejemplo Datos Complejos

```
CREATE TABLE Ejemplo_Datos_Complejos
(
  Ciudades  ARRAY<string>,
  Asignaturas   STRUCT<Codigo:string, Creditos:int>
  lista_Clase  MAP<string,string>,
  );
```

Instrucciones

Instrucción Create Database:

Usada crear una base de datos en *Hive*.

Una base de datos en *Hive* es un espacio de nombres o una colección de tablas.

Sintaxis: CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database_name>

Conectarse a un esquema de base de datos:

Sintaxis: use <database_name>

Instrucciones

- Consultar esquemas de base de datos:

  La siguiente consulta se utiliza para verificar una lista de bases de datos:

  show databases;

- Instrucción Drop Database:

  Es una declaración que afecta todas las tablas y elimina la base de datos.

  Sintaxis: DROP DATABASE IF EXISTS <database_name>;

- Consultar tablas del esquema:

  Sintaxis: show tables;

Instrucciones

Instrucción Create Table

Es una declaración utilizada para crear una tabla en *Hive*.

La sintaxis es la siguiente:

CREATE TABLE IF NOT EXISTS <Table_Name> (col1 typeCol1, col2 typeCol2, …)
[COMMENT 'table_comment']
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '<char>'
LINES TERMINATED BY '<char>'
STORED AS TEXTFILE;

Instrucciones

- Instrucción Load Data
  - Después de crear una tabla en SQL, podemos introducir los datos utilizando la instrucción *Insert*. Pero en Hive, podemos introducir datos mediante la sentencia LOAD DATA.
  - Al insertar datos en Hive, es mejor utilizar la carga a granel.
  - Sintaxis:
  - LOAD DATA [LOCAL] INPATH 'filepath' [OVERWRITE] INTO TABLE tablename

Institución de Educación Superior
UCEVA
Unidad Central del Valle del Cauca

Instrucciones

Ejemplo: Crear la tabla Empleado:

```
CREATE TABLE EMPLEADO (
Nombre String,
Edad int,
Nivel int)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
LINES TERMINATED BY '\n'
STORED AS TEXTFILE;
```

Instrucciones

- Ejemplo cargar, en la tabla creada, los datos de un archivo de texto:
  - Usar archivo emp.txt en su entorno de usuario (En el entorno Linux)
  - Cargar datos del archivo emp.txt en la tabla Empleado:
    - LOAD DATA LOCAL INPATH '/home/cloudera/practica/hive/emp.txt' INTO TABLE EMPLEADO (En el entorno *Hive*)

Instrucciones

- Instrucción Alter Table
  - Se utiliza para modificar una tabla en Hive.
  - Sintaxis:
    - ALTER TABLE name RENAME TO new_name
    - ALTER TABLE name ADD COLUMNS (col_spec[, col_spec ...])
    - ALTER TABLE name DROP [COLUMN] column_name
    - ALTER TABLE name CHANGE column_name new_name new_type
    - ALTER TABLE name REPLACE COLUMNS (col_spec[, col_spec ...])

Instrucciones

- Instrucción Drop Table
  - Se utiliza para eliminar una tabla en Hive.
  -
  - Cuando se elimina una tabla de Hive Metastore, quita la tabla y los datos de la columna y sus metadatos.

  - Sintaxis:
  - DROP TABLE [IF EXISTS] table_name;

Consultas

- La Hive Query Language (HiveQL) es un lenguaje de consulta de Hive para procesar y analizar datos estructurados en un Metastore.

- Sintaxis:

  SELECT [ALL | DISTINCT] select_expr, select_expr, ...
  FROM table_reference
  [WHERE where_condition]
  [GROUP BY col_list]
  [HAVING having_condition]
  [LIMIT number];

Institución de Educación Superior
UCEVA
Unidad Central del Valle del Cauca

Funciones

- Funciones de operadores aritméticos: +, -, *, /, %
- Funciones de operadores lógicos: AND, OR, NOT
- Funciones resúmen: SUM(), AV(), MAX(), MIN(), COUNT()
- Toda la lista de funciones puede visualizarse con ***show functions***
- La explicación de una función se puede visualizar con ***describe function nombreFuncion;***

Preguntas

Institución de Educación Superior
UCEVA®
Unidad Central del Valle del Cauca

www.uceva.edu.co