Lección 5.5 Apache Hive:

Aquellos que estéis acostumbrados a trabajar con bases de datos relacionales (ej. MySQL) o, incluso con Almacenes de Datos, y en general, los que habéis seguido con atención las lecciones anteriores de este curso, seguramente hace tiempo que os estáis planteando algunas cuestiones tan relevantes como ...

- ¿Cómo implementar un Almacén de Datos (DW) en Hadoop?
- ¿Cómo portar una infraestructura de DW previamente existente sobre una BD relacional?
- ¿ Qué ocurre con los diseñadores y administradores de bases de datos los cuales son expertos en el uso de SQL (Structured Query Language)?
- ¿Y con los usuarios finales con conocimientos de SQL?
    - Ej. Ejecutivos de nivel medio y alto, analistas, emprendedores,...
- ¿Tienen que aprender nuevos lenguajes como Pig Latin o formarse en la programación de complicados procesos Map Reduce?

1. Apache Hive

- La solución a estas cuestiones es **Apache Hive**, una de las herramientas más importantes desarrollada para el entorno Hadoop
- Hive proporciona un dialecto de SQL, denominado Hive Query Language (HQL)[1], que permite realizar consultas sobre los datos almacenados en el clúster Hadoop HDFS
    - Ventaja: SQL es una forma efectiva e intuitiva para la organización y el uso de los datos
    - La implementación de las operaciones con las que están familiarizados los usuarios de SQL en MapReduce puede llegar a ser muy compleja (supone un gran número de líneas de código, incluso para sencillas operaciones como el típico ejemplo del recuento de palabras)
- De esta forma, Hive al igual que Pig proporciona una capa de abstracción sobre el núcleo de Hadoop (HDFS y Map Reduce)
    - Las sentencias SQL se traducen de forma invisible al programador en uno o más procesos MapReduce.

[1] Se parece bastante al dialecto de MySQL

Comparación visual del número de líneas de código entre consulta SQL para hacer el ejemplo del recuento de palabras en un texto con el proceso Map Reduce correspondiente

```
package org.myorg;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class WordCount {

 public static class Map extends Mapper<LongWritable, Text, Text,
IntWritable> {
   private final static IntWritable one = new IntWritable(1);
   private Text word = new Text();

   public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
      String line = value.toString();
      StringTokenizer tokenizer = new StringTokenizer(line);
      while (tokenizer.hasMoreTokens()) {
         word.set(tokenizer.nextToken());
         context.write(word, one);
      }
   }
 }
```

```
public static class Reduce extends Reducer<Text,  IntWritable, Text,
IntWritable> {
   public void reduce(Text key, Iterable<IntWritable> values, Context context)
    throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
         sum += val.get();
      }
      context.write(key, new IntWritable(sum));
   }
 }

 public static void main(String[] args) throws Exception {
   Configuration conf = new Configuration();

      Job job = new Job(conf, "wordcount");

   job.setOutputKeyClass(Text.class);
   job.setOutputValueClass(IntWritable.class);

   job.setMapperClass(Map.class);
   job.setReducerClass(Reduce.class);
   job.setInputFormatClass(TextInputFormat.class);
   job.setOutputFormatClass(TextOutputFormat.class);

   FileInputFormat.addInputPath(job, new Path(args[0]));
   FileOutputFormat.setOutputPath(job, new Path(args[1]));

   job.waitForCompletion(true);
 }
 }
```

Recordemos el ejemplo, cuando explicamos los procesos Map Reduce en la lección 4.2, del programa para computar la frecuencia de aparición de cada palabra en un texto dado como entrada. En esta imagen podemos ver la implementación del mismo usando la API Java de Hadoop Map Reduce. Con ella podemos hacernos una idea del número de líneas de código que supone desarrollar un sencillo proceso de análisis como este.

Si implementamos el mismo ejemplo usando SQL, la sentencia o programa resultante es el siguiente:

A simple vista nos sorprende como el número de líneas de código es notablemente menor. De esta forma Hive, al permitir el uso de SQL, no solo proporciona soporte al gran número de profesionales que conocen este lenguaje de consulta de datos, si no que reduce en gran medida la complejidad del desarrollo de procesos de análisis sobre Big Data usando Hadoop.

2. Data Warehousing con Hive

Pero ¿para qué es adecuado Hive?

- Adecuado para Almacenes de Datos (Data Warehousing) donde:
  - Se analizan grandes volúmenes de **datos estáticos** o que apenas cambian en el tiempo
  - No es requerido un tiempo de respuesta de la consulta rápido → **Alta latencia de consulta**
- (Es importante entender que) Hive no es un base de datos completa ...
  - Soporte limitado a las transacciones: Si necesitamos OLTP mejor usar una BD no SQL (ej. Apache HBase)
  - No permite la actualización, inserciones y borrados a nivel de fila

        - Sólo se permite la inserción de conjuntos de filas de forma secuencial en una nueva tabla o al final de una existente
        - Borrados solo a nivel de tabla
    - Hadoop es un sistema orientado al procesamiento de grandes **lotes** de datos
        - Alta latencia de consulta: Consultas que una base de datos relacional llevarían segundos en Hive puede llevar minutos
        - Sin embargo, para el análisis grandes de volúmenes de datos Big Data (5v's) Hive tiene un rendimiento muy alto

3. Hive como herramienta OLAP

- Hive es "**casi**" una herramienta **OLAP** (On-Line Analytical Processing)(Ya vimos que es el análisis OLAP en la lección 2.2 del módulo 2 cuando hablamos de Inteligencia de Negocio o BI)
    - Adecuado para el análisis de grandes volúmenes de datos históricos Big Data...
    - (Pero) **No satisface** la característica **On-Line**: Alta latencia de consulta
- De esta forma, Hive está indicado para….
    - Aplicaciones de BI que usan Almacenes de Datos (Data Warehousing, DW) para la generación de informes, minería de datos para extracción de conocimiento implícito en los datos…
    - Cuyos requerimientos de tiempo no son elevados (minutos, horas, días,...)
- Para la conexión con herramientas de BI o cualquier aplicación que desarrollemos, dispone de conectores JDBC o ODBC
    - Funcionan de las misma forma que los de las bases de datos relacionales

4. Hive como herramienta OLAP

- El análisis OLAP "**En-Línea**" y cuadros de mando requieren un baja latencia de consulta
    - Tiempo de consulta (<segundo)
- Para dar soporte a este tipo de aplicaciones existen diversas alternativas
    - 1. Usar herramientas de BI (como las que vimos en la lección 2.5 del módulo 2) que soporten la **carga de datos en memoria (In-Memory Analytics)** desde Hive.
        - Ej. Pentaho Enterprise Edition, Tableau, Qlikview, Excel con Power Pivot …
        - Se trata de bases de datos NoSQL en memoria (veremos este tipo de almacenamiento en la siguiente lección)

    - 2. Portar los datos ya estructurados a un Almacén de Datos externo a Hadoop
        - Sobre un SGBDR o, mejor, a un SGBDR con características extendidas y procesamiento MPP
    - 3. Otras opciones
        - Uso de discos SSD para el clúster HDFS, OLAP sobre BD's NoSQL ,…

Por último comentar que...

- En la actualidad se está intentando reducir la latencia de consulta Hive para soportar OLAP
    - Hortornworks con Apache Tez y la iniciativa Stinger[2]
    - Cloudera Impala
    - Actualmente llegan a multiplicar la velocidad de consulta x40 y pretenden alcanzar tiempos de consulta inferiores al segundo (en 2015 se pretende que Stinger esté totalmente implementado en Hive en la distribución de Hortornworks)

Por todo ello, llegamos a la conclusión de que Hive es una las herramientas fundamentales a tener en cuenta en la configuración de la arquitectura de nuestra aplicación de Inteligencia de Negocio (BI) para Big Data.

[2] http://hortonworks.com/hadoop/tez/
http://hortonworks.com/labs/stinger/