Lección 5.2 Entorno Hadoop/MapReduce:

Tras motivar en la lección anterior la necesidad de arquitecturas específicas para dar soporte al procesamiento y análisis de Big Data y enumerar las principales arquitecturas existentes, en esta y siguientes lecciones nos centraremos en el estudio del entorno Hadoop/MapReduce y las herramientas desarrolladas sobre él, siendo esta arquitectura y herramientas de código abierto unas de las más usadas por las empresas y desarrolladores que implementan Big Data

1. Hadoop

- Proyecto de código abierto de alto nivel dirigido por Apache Software Foundation (ASF)
    - Al ser de código abierto, podemos descargarlo de forma gratuita, ya sea listo para su utilización o el código fuente, de forma que podemos usarlo tal cual, contribuir a su desarrollo, o modificarlo para usarlo en productos comerciales (bajo los términos de la licencia de Apache).
- Entorno que describe un método de procesamiento de datos distribuido y que permite escalar usando hardware comercial configurado para funcionar como un clúster de computadores
    - Esta arquitectura permite a Hadoop gestionar y analizar enormes volúmenes de datos
    - A un coste significativamente menor que los métodos tradicionalmente usados en la tecnología de almacenes de datos (Data Warehousing) (ej. con BD's Relacionales)
- En su núcleo, Hadoop tiene dos funciones principales
    - Almacenar Datos (HDFS)
    - Procesar datos (MapReduce)

2. Sistema de archivos distribuidos HDFS

- Hadoop Distributed File System (HDFS): Sistema de archivos diseñado para permitir su distribución a través de un clúster de servidores (comerciales, compuestos por Hardware comercial)
    - Basado en tecnología Java
    - Permite escalar respecto al volumen de datos mediante la adición de nuevos equipos al cluster → (Lo cual implica) Soporte para conjuntos de datos masivos al mismo tiempo que se mantiene un coste bajo y de crecimiento lineal

    - Tolerante a fallos
- Permite almacenar cualquier tipo de archivo pero no es un sistema de archivos “real”
    - No podemos acceder directamente a él desde el SO
- Diseñado para la lectura de secuencial de grandes volúmenes de datos
    - No adecuado para la lecturas y escrituras aleatorias de datos (En principio, pues han surgido algunas alternativas→ BD’s NoSQL sobre Hadoop como Cassandra y HBase)

3. Sistema de archivos distribuidos HDFS

- Los archivos creados en HDFS se componen uno o más bloques (de datos) HDFS
    - El tamaño de bloque por defecto es de 64 MB
    - Cada bloque es replicado (tolerancia a fallos) y distribuido a través de las máquinas que forman el cluster HDFS
- Está formado por dos componentes principales
    - NameNode
        - Un único NameNode en el clúster
        - Es el responsable de dirigir el sistema de archivos y, además, es el responsable para asignación de directorios y archivos
        - Gestionan los bloques de datos HDFS que están almacenados en los DataNode
    - DataNode
        - Existe uno por cada computador del clúster
        - Se encargan de gestionar el almacenamiento local en cada nodo (lecturas y escrituras)

También existe la figura del “Secondary NameNode”. En caso de que falle el NameNode este pasa a ser el nuevo NameNode.

4. Map Reduce. Módelo de programación

- **MapReduce:** (Simple pero potente) Modelo de programación para el procesamiento de datos en paralelo
    - Nace de los principios conocidos desde los años 80 para la computación distribuida
    - Desarrollado por Google antes de existir Hadoop
- Hadoop es una implementación de MapReduce

- La combinación MapReduce con HDFS permite procesar cientos de Gigabytes de datos, e incluso Terabytes, en menos de un minuto (es muy eficiente)

5. Map Reduce. Motor de ejecución

Además de un modelo de programación...

- (Pero) En el contexto de Hadoop, MapReduce también es un motor de ejecución
    - Los usuarios desarrollan programas MapReduce que se envían al motor MapReduce para su procesamiento
- Los programas creados por los desarrolladores reciben el nombre de trabajos o "jobs"
    - En lenguaje de programación Java
    - (Aunque también) Hadoop proporciona una API que permite escribir trabajos MapReduce en otros lenguajes distintos de Java
- El motor de ejecución de MapReduce es usado para distribuir la carga de trabajo a lo largo de clúster HDFS y es el responsable de la ejecución de los trabajos (jobs) MapReduce
    - Los trabajos MapReduce son gestionados por el **JobTracker** (Existe un único JobTracker por clúster Hadoop)
    - El JobTracker se comunicada con el HDFS NameNode para determinar la localización de los datos

6. Map Reduce. Procesos Map y Reduce

Cada trabajo Map Reduce se descompone en dos procesos clave para su ejecución

- **Map**: Divide la entrada en muchas piezas pequeñas de forma que cada pieza pueda ser procesada de forma independiente y en paralelo
    - Se cargan los registros en la forma clave:valor, para lo que hemos de determinar qué parte es la clave y cuál es el valor, pudiendo operar sobre el valor de cada registro de forma individual.
    - Múltiples procesos Map se lanzan en paralelo a través del clúster de nodos, aprovechando la localidad de los datos
- **Reduce:** (Una vez terminado el proceso Map) Los resultados del procesamiento de cada pieza son recopilados, agregados y procesados
    - Los datos con la misma clave se agregan según la función definida y cada proceso Reduce puede almacenar su propia salida

- Además, existe un tercer procesos denominado Shuffle, generado de forma automática, donde los valores se ordenan por clave y se distribuyen a múltiples procesos Reduce, enviando todos los registros que tengan las misma clave al mismo proceso Reduce

Ejemplo

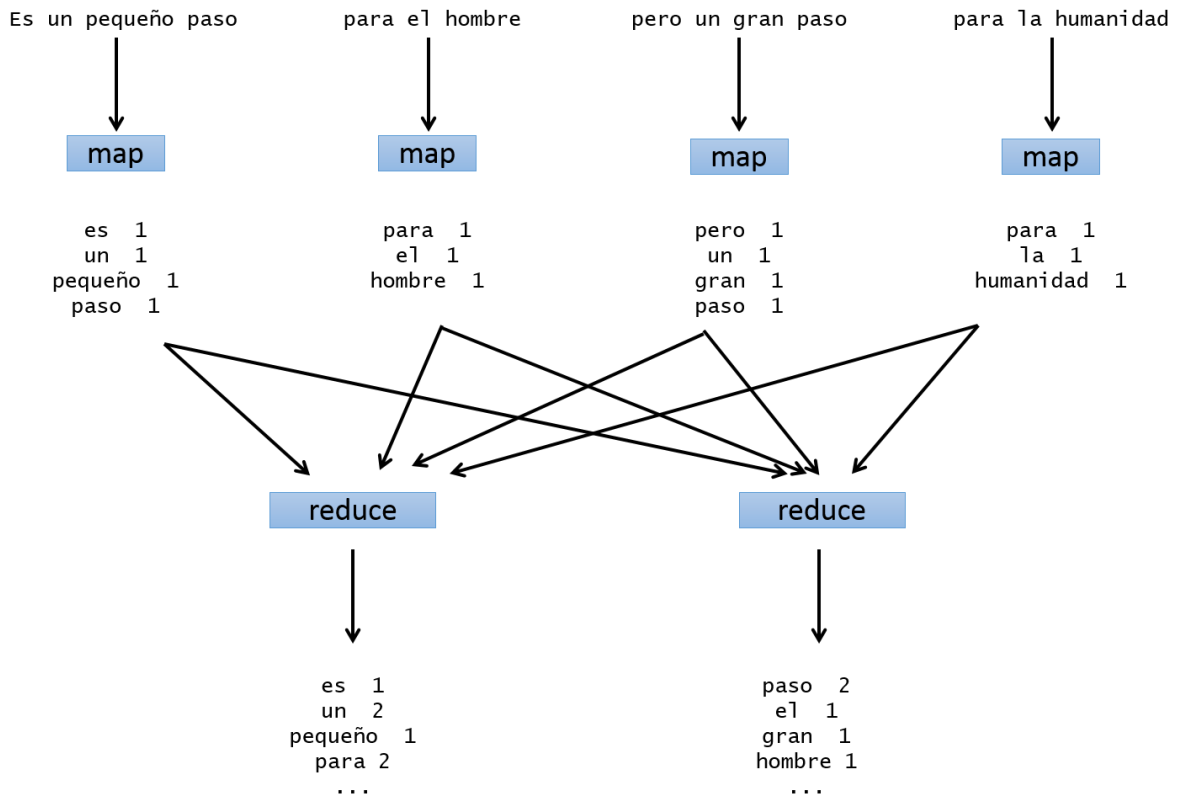
Con el fin de mostrar el funcionamiento de MapReduce, aplicaremos este modelo de programación para el análisis de la frecuencia de aparición de cada palabra en el siguiente texto:

"Es un pequeño paso
para el hombre,
pero un gran paso
para la humanidad"

En la fase map, dividimos la entrada (el texto) en cuatro líneas, enviando cada una a uno de los procesos map distribuidos a lo largo del clúster. En estos procesos, cada frase se divide palabras (las claves) y a cada palabra se le asigna un 1 como valor para representar que la palabra ha sido vista 1 en el texto. Ya tenemos como clave:valor, palabra y valor 1.

Tras esto, en la fase shuffle (generada de forma automática) los valores se ordenan por clave y se distribuyen a múltiples procesos Reduce, enviando todos los registros que tengan las misma clave al mismo proceso Reduce. Si hay dos Reducers podemos decidir enviar las palabras que empiezan por A-L al primero y el resto, M-Z al segundo.

Finalmente, en el proceso Reduce se agregan los resultados con la misma clave aplicando la suma sobre los valores. Es decir, obtenemos el número de veces que la palabra aparece en el texto. En ese momento cada proceso Reduce puede encargarse de almacenar su salida (o servir como entrada de otro trabajo MapReduce)



6. Map Reduce. Conclusiones

- Los procesos MapReduce permiten el procesamiento y análisis de los grandes volúmenes de datos almacenados en el clúster HDFS pero...
- MapReduce está pensado para trabajar con datos almacenados en la forma de clave:valor (considerado esto como un tipo de almacenamiento NoSQL)
    - (Aunque) puede procesar cualquier tipo de archivo almacenado en el clúster HDFS si diseñamos el proceso map correspondiente donde se determine la clave y el valor.
- La dificultad para el desarrollo de estos procesos es muy elevada
    - Por ejemplo, si la comparamos con el uso de lenguajes de consulta y manipulación de datos clásicos como SQL
    - Por ello han aparecido diversas herramientas del entorno Hadoop construidas sobre la base MapReduce / HDFS que proporcionan un nivel de abstracción

mayor al programador, simplificando el diseño de los procesos para el tratamiento y análisis de los datos. (Las veremos en la siguiente lección)