Spark

Spark fue diseñado para soportar en memoria algoritmos iterativos que se pudiesen desarrollar sin escribir un conjunto de resultados cada vez que se procesaba un dato. Esta habilidad para mantener todo en memoria es una técnica de computación de alto rendimiento aplicado al análisis avanzado, la cual permite que Spark tenga unas velocidades de procesamiento que sean 100 veces más rápidas que las conseguidas utilizando MapReduce.

Desde hace bastante tiempo, se han buscado mecanismos que aporten a Apache Hadoop la capacidad de procesamiento en tiempo real, ya que el framework inicialmente fue orientado a procesos batch. Se han propuesto varias soluciones como Apache HBase, Impala, Apache Flume, etc. pero estas soluciones solamente permiten cubrir en parte esta necesidad de tiempo real y procesamiento en modo stream. En el caso del proyecto de Apache Spark, éste ya cuenta de forma nativa con una librería que permite el procesamiento de datos en tiempo real, Apache Spark Streaming.

DESVENTAJAS DE HODOOP

- Latencia para el acceso a datos: HDFS está orientado a procesos batch y operaciones en streaming. Por lo tanto, la latencia de cualquier operación IO no ha sido optimizada y sistemas de archivos tradicionales (como ext4, XFS...) suelen ser más rápidos en estos aspectos.

- Cantidades grandes de ficheros pequeños: El límite del número de ficheros en este sistema está limitado por la memoria del NameNode, que es en su RAM donde se encuentran los metadata. Cada fichero, directorio y bloque ocupa un tamaño de entre 150 y 200 bytes en los metadata, lo que quiere decir, que si hay millones de ficheros pequeños va a ocupar mucho más espacio en la RAM que si tenemos menos cantidad de ficheros de gran tamaño (recomendable 100 MB o más).

- Escribe una vez, lee varias: En HDFS los ficheros solo se pueden escribir una vez (aunque HDFS se ha mejorado con el modo "append", Cloudera no suele recomendarlo porque no lo considera estable).

- No se puede acceder con los comandos tradicionales de Linux (ls, cat, vim...). Esto complica mucho la integración con otras herramientas comerciales (como sistemas de backup, por ejemplo). Y aunque exista "HDFS fuse" para montar HDFS como cualquier otro sistema de archivo Linux, esta solución no ofrece un buen rendimiento.

DESVENTAJAS DE MAPREDUCE

Es muy difícil de depurar: Al procesarse el programa en los nodos donde se encuentran los bloques de datos, no es fácil encontrar los fallos de código. Tampoco es conveniente utilizar funciones de escritura de logs en el código ya que eso podría suponer un gran aumento en la ejecución de procesos MapReduce.

- No todos los algoritmos se pueden escribir con el paradigma MapReduce. Por ejemplo, el famoso algoritmo de grafos Dijkstra, al necesitar un procesamiento secuencial, no se puede escribir en MapReduce.

Latencia: cualquier job MapReduce suele tardar por lo menos 10 segundos. Por lo tanto, si el volumen de información a tratar es pequeño, es posible que Hadoop no osea la solución más rápida

DOCKER

VENTAJAS

- Consumen menos recursos hardware que las máquinas virtuales y éstos van exclusivamente a la aplicación.
- Las instancias se arrancan en pocos segundos.
- Es fácil de automatizar e implantar en entornos de integración continua.
- Existen muchas imágenes que pueden descargarse y modificarse libremente.
- Ambientes Replicables.
- Exactitud de las Pruebas.
- Escalabilidad.
- Alta Disponibilidad.

DESVENTAJAS

- Sólo puede usarse de forma nativa en entornos Unix aunque se puede virtualizar gracias a boot2docker tanto en OSX como en Windows.
- Las imágenes sólo pueden estar basadas en versiones de Linux modernas (kernel 3.8 mínimo).
- Como es relativamente nuevo, puede haber errores de código entre versiones.