Introducción a Apache Spark

Apache Spark es un framework de computación distribuida open source, con el motor de procesamiento con mayor velocidad gracias a su tecnología de procesamiento en memoria de grandes volúmenes de datos. Interesará también al Científico de Datos estar al corriente de las nuevas tendencias de cambio generacional de Hadoop hacia Spark.

Spark tiene varias ventajas sobre Hadoop para el procesamiento de la información y la ejecución de algoritmos. La principal de ellas la velocidad , dado que es hasta cien veces mayor debido a que, a diferencia de Hadoop, Spark utiliza la gestión 'en memoria' y sólo escribe a disco cuando es necesario.

Spark puede ejecutarse de forma independiente o puede convivir como un componente más de Hadoop, de forma que la migración puede planificarse de manera no traumática. Puede por ejemplo utilizar HBase como base de datos, aunque Cassandra se está imponiendo como solución de almacenamiento por su redundancia y escalabilidad. Si atendemos a lo que nos dice la propia web del framework.

Spark ofrece múltiples ventajas respecto a MapReduce-Hadoop

- **Procesamiento en memoria**. Con el fin de mejorar el rendimiento entre operaciones,se permite la persistencia o el almacenamiento en caché de un RDD entre operaciones.
- Las aplicaciones son ejecutadas en Cluster por los nodos y están gestionadas por un maestro
- Escalable y tolerante a fallos

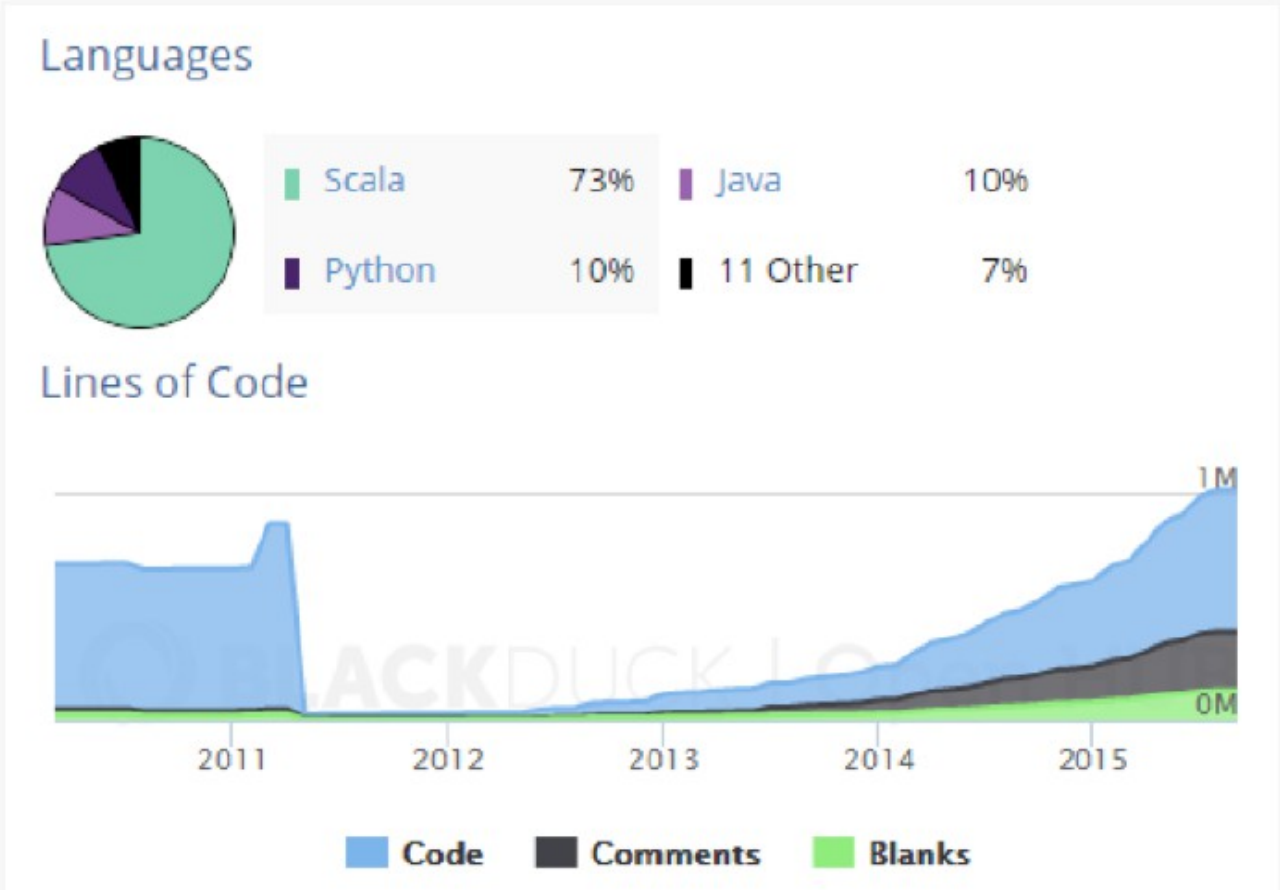
Si el procesamiento se realiza en memoria, el incremento de velocidad con respecto a la alternativa de Hadoop MapReduce es de aproximadamente 100 veces. Ésto se debe principalmente a que los datos en apache Spark se almacenan en RAM, mientras que los datos de su predecesor se almacenaban generalmente en disco en un sistema de ficheros HDFS.

Apache Spark es una **evolución de Hadoop** y se ha vuelto muy popular en los últimos años. Contrariamente a Hadoop y su diseño Java y centrado en lotes, Spark es capaz de producir algoritmos iterativos de una manera rápida y fácil. Además, tiene un conjunto rico de APIs para múltiples lenguajes de programación y soporta nativamente diferentes tipos de procesamiento de datos (aprendizaje de máquina, streaming, análisis de gráficos, SQL).

Apache Spark es un marco de clúster diseñado para aplicaciones rápidas y de uso general para procesamiento de grandes cantidades de datos. Una de las mejoras en la velocidad es el hecho de que los datos, después de cada trabajo, se mantiene en la memoria y no se almacena en el sistema de almacenamiento) como habría sucedido con Hadoop, MapReduce y HDFS. Esta hace que los trabajos iterativos (como el algoritmo de K-means de agrupación) sean más rápidos así como la latencia y el ancho de banda proporcionados por la memoria son más eficientes que el disco físico.

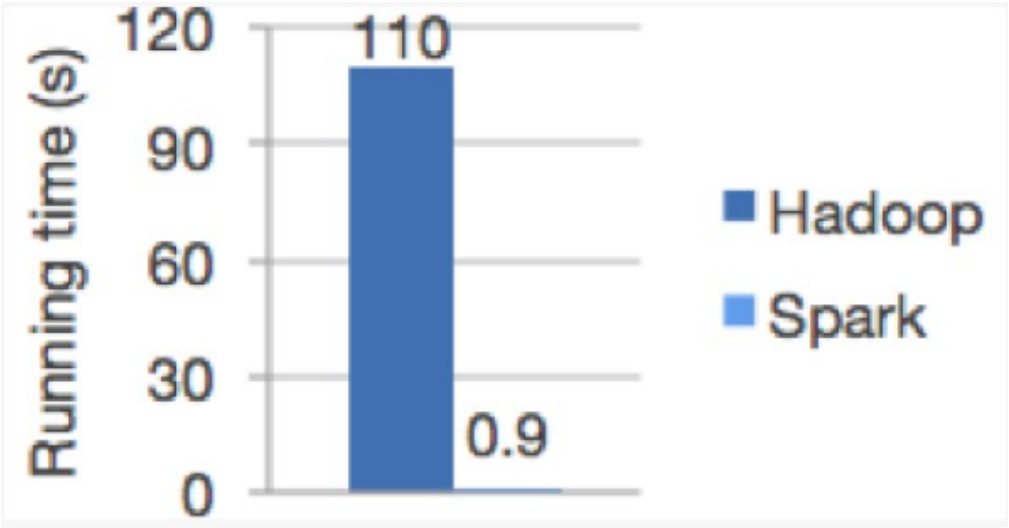
Existen muchas características que hacen de Spark una plataforma especial, pero podríamos englobarlas en cinco aspectos importantes: es una plataforma de código abierto con una comunidad muy activa; es una herramienta rápida; unificada; dispone de una consola interactiva cómoda para los desarrolladores; y también tiene una API para trabajar con grandes cantidades de datos.

1. **Una plataforma de código abierto con una comunidad activa.** Una de las propiedades más interesantes de una solución de código abierto es la actividad de su comunidad. Es la comunidad de desarrolladores la que mejora las características de la plataforma, y ayuda al resto de programadores a implementar soluciones o resolver problemas. La de Apache Spark es una comunidad cada vez más activa : en septiembre de 2013 había más de 113.000 líneas de código; un año después, se superaban las 296.000; y este septiembre de 2015, el volumen de líneas de código ya marca un récord: 620.300.



2. **Una plataforma rápida.** Una de las primeras circunstancias que sorprenden de Spark es que, para ser una plataforma de código abierto, su velocidad es enorme, muy por encima de algunas soluciones propietario. ¿Por qué es tan rápida? Apache Spark permite a los programadores realizar operaciones sobre un gran volumen de datos en clústeres de forma rápida y con tolerancia a fallos. Cuando tenemos que manejar algoritmos, trabajar en memoria y no en disco mejora el rendimiento.

3. **En materia de aprendizaje automático**, Spark ofrece unos tiempos de cálculo en memoria mucho más rápidos que cualquier otra plataforma. El almacenamiento de los datos en la memoria caché hace que la iteración de los algoritmos de machine learning con los datos sea más eficiente. Las transformaciones que se van produciendo de esos datos también se almacenan en memoria, sin tener que acceder dentro del disco. En su página web hay una prueba de competencia que muestra el rendimiento de Spark con respecto a MapReduce : **de 10 a 100 veces más rápida.**



En ese procesamiento de datos en memoria, el equipo de desarrolladores dispone de la flexibilidad suficiente para escoger qué datos quedan en memoria y cuáles pueden volcarse al disco duro porque no son necesarios en ese momento. Eso libera mucho el procesamiento, aumentando su eficacia.

4. **Una gran API para trabajar con los datos.** Apache Spark tiene APIs nativas para los lenguajes de programación Scala, Python y Java. Este conjunto de APIs facilita a los programadores el desarrollo de aplicaciones en estas sintaxis, que se puedan ejecutar en la plataforma de código abierto. Las APIs posibilitan interactuar con los datos de:
   1. El Sistema de Archivos de Hadoop (HDFS).
   2. La base de datos NoSQL de código abierto HBase .
   3. La base de datos NoSQL de código abierto Apache Cassandra

   Las APIs sirven para realizar dos tipos de operaciones sobre los datos:
   - Transformar un grupo de datos.
   - Aplicar operaciones sobre los datos para obtener un resultado.

Spark puede funcionar de dos formas diferentes:

- **Modo autónomo(standalone):** Se ejecuta en su máquina local. En este caso, el máximo de paralelización es el número de núcleos de la máquina local y la cantidad de memoria disponible es exactamente la misma que la local.

- **Modo de clúster:** Se ejecuta en un clúster de múltiples nodos, utilizando un administrador de clúster,como YARN. En este caso, la paralelización máxima es el número de núcleos en todos los nodos que componen el clúster y la cantidad de memoria es la suma de la cantidad de memoria de cada nodo.

Modo cluster

Las aplicaciones para Spark se ejecutan como un grupo independiente de procesos en clústeres, coordinados por el objeto SparkContext. Más específicamente, **SparkContext** puede conectarse a gestores de clúster que son los encargados de asignar recursos en el sistema. Una vez conectados, Spark puede encargar que se creen **ejecutores** o **executors** encargados de la computación en los nodos del clúster. Los trozos de código propio de los que se encargan estos **ejecutores** son denominados **tasks** o **tareas**.