Objetivos del proyecto y resultados

Esta semana hemos conocido dos implementaciones diferentes de la abstracción de programación MapReduce: Hadoop y Spark.

Hadoop fue uno de los primeros sistemas de programación MapReduce distribuidos populares, y realmente allanó el camino para el análisis de Big data a escalas masivas en hardware básico. Como se describe en las conferencias de esta semana, sus principales puntos fuertes son tres: 1) escalabilidad, 2) programabilidad, y 3) tolerancia a las fallas. Hadoop permite a los programadores escribir aplicaciones Java sencillas que se ejecutan en miles de máquinas conectadas en red y que, gracias a sus abstracciones MapReduce, pueden recuperarse de un fallo completo del hardware de muchas de esas máquinas.

Spark, por su parte, es un sucesor de Hadoop que hace hincapié en el almacenamiento de datos en memoria caché y en una API más flexible. Mientras que Hadoop persiste regularmente en replicar los datos en disco para garantizar la tolerancia a fallos, Spark, en cambio, almacena la información en caché en memoria y se asegura de recordar las transformaciones que se aplicaron para generar un dato concreto. Spark también ofrece más que simples transformaciones map y reduce.

También has aprendido sobre el algoritmo PageRank, y cómo se puede utilizar para clasificar sitios web interconectados en base a los rangos existentes y el recuento de enlaces.

En este mini-proyecto, ganará experiencia con Spark implementando el algoritmo PageRank utilizando transformaciones de Spark. En concreto, esta tarea le pedirá que implemente las transformaciones necesarias para completar una única iteración del algoritmo iterativo PageRank dado un conjunto de datos distribuidos Spark (RDD) de sitios web.

Recordemos de las clases de esta semana que un concepto importante en los marcos MapReduce es el de "pares clave-valor". La **clave** de un par clave-valor es un identificador único de un objeto lógico, por ejemplo, un sitio web. El **valor** son los metadatos asociados al objeto lógico que identifica la clave, por ejemplo, información sobre los enlaces salientes de un sitio web. En Spark, una Pareja clave-valor se representa utilizando la clase Tuple2. Por ejemplo, para crear una nueva Pareja clave-valor a partir de alguna clave k y algún valor v se puede utilizar el siguiente código:

```
1
Tuple2 kvPair = new Tuple2(k, v);
```

Spark utiliza objetos RDD de propósito especial para almacenar conjuntos de datos que contienen objetos Tuple2 (es decir, pares clave-valor). En lugar de utilizar la clase JavaRDD estándar, los RDDs de objetos Tuple2 utilizan la clase JavaPairRDD.

Configuración del proyecto

Por favor, consulte el Mini-Proyecto 0 para una descripción del proceso de construcción y pruebas utilizado en este curso.

Una vez que haya descargado y descomprimido los archivos del proyecto utilizando el botón gris etiquetado miniproject_1.zip en la parte superior de esta descripción, debería ver el archivo de código fuente del proyecto en

miniproject_1/src/main/java/edu/coursera/distributed/PageRank.javaminiproject_1/src/main/java/edu/coursera/distributed/PageRank.javastart color red, start verbatim, miniproject_1/src/main/java/edu/coursera/distributed/PageRank.java, end verbatim, end color red

y las pruebas del proyecto en

miniproject_1/src/test/java/edu/coursera/distributed/SparkTest.javaminiproject_1/src/test/java/edu/c

oursera/distributed/SparkTest.javastart color red, start verbatim, miniproject_1/src/test/java/edu/coursera/distributed/SparkTest.java, end verbatim, end color red

Se recomienda que revise el vídeo de demostración de esta semana antes de comenzar esta tarea, en el que el profesor Sarkar trabaja con un ejemplo similar.

Instrucciones del proyecto

Las modificaciones deben realizarse íntegramente dentro de PageRank.java. No debe cambiar las firmas de ningún método público o protegido dentro de PageRank.java, pero puede editar los cuerpos de los métodos y agregar los métodos nuevos que desee. Utilizaremos nuestra copia de SparkTest.java en el proceso de calificación final, así que no modifique ese archivo ni ningún otro excepto PageRank.java.

Sus objetivos principales para esta tarea son los siguientes:

1. Dentro de PageRank.java, implementar el método sparkPageRank para realizar una única iteración del algoritmo PageRank utilizando las API Java de Spark.

En PageRank.java hay TODOs útiles que te ayudarán a guiar tu implementación.

NOTA: Hemos recibido quejas de algunos alumnos que intentaron ejecutar el Proyecto 1 localmente en sus máquinas Apple Silicon. Aunque es posible configurar el proyecto para que se ejecute en Apple Silicon (véase el POST de uno de los alumnos aquí: https://www.coursera.org/learn/distributed-programming-in-java/discussions/forums/F75SP7QkEeaElQ6tBsFbjg/threads/tPZWkqvVEe2TGRJeZ1mA3Q), también es posible ejecutar el Proyecto 1 (y los proyectos posteriores de este curso) sin modificaciones en Apple Silicon simplemente instalando y utilizando una versión x86_64 de Java en su Mac M1 o M2 a través de Rosetta 2.

Evaluación del proyecto

Su tarea debe consistir únicamente en el archivo PageRank.java que modificó para implementar este miniproyecto. Tenga en cuenta que el rendimiento observado para las pruebas en su máquina local puede diferir. Puntuación basada en lo siguiente:

- 10% - rendimiento de testUniformThirtyThousand en 2 núcleos
- 10% - rendimiento de testUniformOneHundredThousand en 2 núcleos
- 5% - rendimiento de testIncreasingThirtyThousand en 2 núcleos
- 5% - rendimiento de testIncreasingOneHundredThousand en 2 núcleos
- 10% - rendimiento de testRandomThirtyThousand en 2 núcleos
- 10% - rendimiento de testRandomOneHundredThousand en 2 núcleos
- 10% - rendimiento de testUniformThirtyThousand en 4 núcleos
- 10% - rendimiento de testUniformOneHundredThousand en 4 núcleos
- 5% - rendimiento de testIncreasingThirtyThousand en 4 núcleos
- 5% - rendimiento de testIncreasingOneHundredThousand en 4 núcleos
- 10% - rendimiento de testRandomThirtyThousand en 4 núcleos
- 10% - rendimiento de testRandomOneHundredThousand en 4 núcleos