Informática como servicio

Práctica de Map–Reduce, 2014/15

En este guión se pretende dar algunas indicaciones sobre la realización de la práctica de Map–Reduce. Todo ha sido ya comentado en clase.

Aspectos generales

- Planteamos tres ejercicios:

  A) Contar las veces que aparece cada palabra en un conjunto de textos

  B) Producto de una matriz por un vector

  C) Integración numérica

  Es necesario realizar **los tres ejercicios**.

- Se utilizará el sistema Elastic Map Reduce de la plataforma Amazon; no obstante, antes de subir las funciones conviene probar su comportamiento en local, en el propio ordenador, con ejemplos sencillos.

- El plazo de entrega de la práctica finaliza el sábado **15 de noviembre**.

- El profesor está disponible para atender las dudas que vayan surgiendo, bien en el despacho 2.09 de la Facultad de Informática o bien por correo electrónico (`arregui@udc.es`).

- Deben implementarse las funciones `map` y `reduce` para cada uno de los problemas, utilizando un lenguaje de programación adecuado (Python, Java, ...).

- Cada estudiante deberá enviar, por correo electrónico, los códigos con las funciones `map` y `reduce` de cada ejercicio. En caso de funcionamiento incorrecto o de dudas por parte del profesor, éste le contactará para su aclaración.

Elastic Map Reduce (EMR)

- Para utilizar EMR debemos colocar nuestras funciones y archivos de datos en el sistema de almacenamiento S3 de Amazon.

- Los procesos se crean desde la ventana de EMR (botón "Create New Job Flow"); hay que proporcionar lo siguiente:

  - "Cluster configuration":
    - "Cluster name": un nombre del proceso
    - "Termination protection": NO
    - "Log folder S3 location": carpeta donde se situarán los archivos *log*

- "Tags"
- "Software configuration"
- "File system configuration"
- "Hardware configuration"
- "Security and acces"
- "IAM roles"
- "Bootstrap actions"
- "Steps":
  - "Add step": seleccionaremos la opción "Streaming program"
  - En el botón "Configure and add":
    - "Mapper": archivo que contiene la función `map`
    - "Reducer": archivo que contiene la función `reduce`
    - "Input S3 location": directorio que contiene los archivos de datos
    - "Output S3 location": ruta a un directorio nuevo

    Las rutas que indican un directorio deben terminar con una barra (`/`)
  - "Auto terminate": YES

- Las funciones deben estar programadas para leer (escribir) datos desde (en) pantalla, mediante entrada (salida) estándar; por ejemplo, en PYTHON:

```
import sys
for line in sys.stdin:
     key = ...
     value = ...
.................................

print key, value
```

- Si el proceso ha terminado correctamente, en la ventana de EMR aparece con *status* "Completed" y podremos ver en S3 que se ha creado el directorio indicado en "Output location", con los ficheros de salida.
- Si el proceso no ha finalizado correctamente, el *status* que aparece es "Failed"; si pinchamos en el proceso vemos, en la parte inferior de la ventana, alguna información sobre el mismo.
- Si hemos habilitado el *"Debugging"* (en el cuarto paso, "Advanced options"), en la ruta indicada aparece una nueva carpeta con una gran cantidad de contenido que también puede ayudarnos a detectar posibles errores.

Ejercicio A) Recuento de palabras

Se trata de contar cuántas veces aparece una palabra en un grupo de textos. Cada texto está almacenado en un fichero, pudiendo ser el número de ficheros muy elevado.

De forma muy esquemática,

- el *mapper* localiza las palabras y les asigna un 'uno' cada vez que aparecen
- el *reducer* suma todos los 'unos' de cada palabra.

En la carpeta `s3n://ics-mei-udc/mapreduce/practica1/` hay siete archivos de tipo texto.

Ejercicio B) Producto matriz vector

El objetivo es multiplicar una matriz por un vector. La matriz puede ser muy grande por lo que estará *troceada* en bloques, cada uno de los cuales se almacenará en un fichero distinto.

Para simplificar la implementación, definiremos el vector a multiplicar dentro de la función `map`; por ejemplo, un vector con todas sus componentes iguales a *uno*.

Consideraremos dos formas de almacenamiento: en filas y en columnas.

Almacenamiento en filas

En estos casos, cada archivo contiene una fila de la matriz. Las carpetas `s3n://ics-mei-udc/mapreduce/practica2/datarow01/` y `s3n://ics-mei-udc/mapreduce/practica2/datarow02/` contienen la descomposición por filas de dos matrices distintas. En ambos conjuntos de ficheros, cada línea contiene tres cifras:

- el índice de fila i
- el índice de columna j
- el coeficiente a_{ij}.

Almacenamiento en columnas

En estos casos, cada bloque es solamente una columna o una fila de la matriz. Las carpetas `s3n://ics-mei-udc/mapreduce/practica2/datacol01/` y `s3n://ics-mei-udc/mapreduce/practica2/datacol02/` contienen la descomposición por filas de dos matrices distintas. En ambos conjuntos de ficheros, cada línea contiene tres cifras:

- el índice de fila i
- el índice de columna j
- el coeficiente a_{ij}.

Ejercicio C) Integración numérica

Dada una función de una variable, sabemos que:

$$\int_a^b f(x)\,dx = \int_{x_0}^{x_1} f(x)\,dx + \int_{x_1}^{x_2} f(x)\,dx + \ldots + \int_{x_i}^{x_{i+1}} f(x)\,dx + \ldots + \int_{x_{N-1}}^{x_N} f(x)\,dx\,,$$

donde $a = x_0 < x_1 < x_2 < \ldots < x_i < x_{i+1} < \ldots < x_{N-1} < x_N = b$. Cada una de las integrales puede aproximarse por la fórmula del trapecio:

$$\int_{x_i}^{x_{i+1}} f(x)\,dx \approx \frac{x_{i+1} - x_i}{2}\left[f(x_i) + f(x_{i+1})\right]\,.$$

De manera similar, si g es una función de dos variables,

$$\int_{\Omega} g(x,y)\,dx\,dy = \sum_{i=1}^{N} \int_{\Omega_i} g(x,y)\,dx\,dy\,,$$

donde $\Omega = \cup\Omega_i$ $(i = 1, 2, \ldots, N)$. En el caso particular de que el dominio de integración sea un triángulo de vértices a, b y c, la integral sobre dicho triángulo puede aproximarse por:

$$\int_{T} g(x,y)\,dx\,dy \approx \frac{\text{Area }(T)}{3}\left(g(a_1,a_2) + g(b_1,b_2) + g(c_1,c_2)\right)\,.$$

En la carpeta `s3n://ics-mei-udc/mapreduce/practica3/` hay dos carpetas con sendos conjuntos de datos:

- `data01` contiene los datos de una subdivisión del intervalo $(0, 3)$; cada archivo contiene las abscisas de los dos extremos.
- `data02` contiene los datos de una subdivisión del cuadrado $(0, 1)\times(0, 1)$; cada archivo proporciona las coordenadas de los vértices (a_1, a_2, b_1, b_2, c_1, c_2) de cada triángulo.