Conceptos y Aplicaciones de Big Data

MAPREDUCE (EMULADOR MRE)

DESARROLLO EN PYTHON

Prof. Waldo Hasperué
whasperue@lidi.info.unlp.edu.ar

Temario

Uso de un emulador

Desarrollo de soluciones en MapReduce

- Python

Emulador MapReduce

Este semestre haremos los ejercicios usando un emulador de MapReduce.

Este emulador si bien permite la ejecución de jobs MapReduce, no se ejecuta en un ambiente distribuido ni hace acceso a un DFS.

El emulador es provisto por la cátedra en el archivo MRE.py y solo debe ser importado desde cualquier script en python para hacer uso de la clase Job:

```
from  MRE import Job
```

Emulador – Lectura de archivos

El emulador solo lee archivos de texto plano.

Cada linea del archivo (finalizada con un "Enter") es convertida a una tupla para la función map.

Emulador – Lectura de archivos

Si las líneas leídas tienen al menos un carácter tabulador, la primer "columna" es usada como *K1* y el resto es interpretado como *V1*.

K1	V1				
32893864	Ywjwg	10	3	1977	4174941
11949606	Qfmqiw	23	8	1981	7126895
11215371	Liytxx	11	6	1984	9462881
38440764	Tjjcxqx	25	12	1967	1147482
13230837	Xgmuadg	4	6	1962	5334710

Cinco tuplas a procesar

Emulador – Lectura de archivos

Si las líneas leídas NO tienen al menos un carácter tabulador, el offset de la línea dentro del archivo es usado como *K1* y la línea completa como *V1*.

K1	V1
0	FRAY BALTASAR
13	Fray Baltasar estaba perplejo ante su pupitre, en el …
45	--¡Seis horas sin lograr nada, pensó. Dios me ayude …
72	Se encaminó al coro lentamente, pensando sin …
93	Entonaron los frailes los suaves cánticos rituales; …

Cinco tuplas a procesar

MapReduce

Desarrollo de una aplicación en Python.

Ejemplo - WordCount

WordCount es un programa que contabiliza la ocurrencia de cada palabra que aparece en un texto.

Ejemplo:

- Entrada:

  "Si tú crees que puedes, puedes. Si tú crees que no puedes, no puedes"

- Salida:

Puedes	4	Crees	2
Si	2	Que	2
Tú	2	No	2

WordCount – Función map

```
def fmap(key, value, context):
    words = value.split()
    for w in words:
        context.write(w, 1)
```

Para la ejecución de un job se debe implementar una función map, la cual va a recibir una *key*,su *value* asociado y un *context*.

WordCount – Función map

```
def fmap(key, value, context):
    words = value.split()
    for w in words:
        context.write(w, 1)
```

key y *value* siempre serán strings, por lo que habrá que hacer todas las conversiones pertinentes, para poder trabajar con valores numéricos.

WordCount – Función map

```
def fmap(key, value, context):
    words = value.split()
    for w in words:
        context.write(w, 1)
```

La ejecución de una invocación a esta función puede generar cero, una o más tuplas de salida. Las tuplas <k2, v2> deben ser escritas usando el método *write* de *context.* Éste método recibe dos parametros: *k2* y *v2.*

WordCount – Función reduce

```
def fred(key, values, context):
    c=0
    for v in values:
        c=c+1
    context.write(key, c)
```

Para la ejecución de un job se debe implementar una función reduce, la cual va a recibir una *key*, su lista de *values* asociado y un *context*.

WordCount - Driver

```
inputDir = root_path + "WordCount/input/"
outputDir = root_path + "WordCount/output/"

job = Job(inputDir, outputDir, fmap, fred)
success = job.waitForCompletion()
```

Se deben establecer los directorios de lectura y escritura.
ATENCIÓN: el directorio de escritura será vaciado por el propio emulador, al momento de ejecutar un job.

WordCount - Driver

```
inputDir = root_path + "WordCount/input/"
outputDir = root_path + "WordCount/output/"

job = Job(inputDir, outputDir, fmap, fred)
success = job.waitForCompletion()
```

Se crea un objeto *Job* pasándole los directorios de entrada y salida, la función *map* y la función *reduce*. Luego se invoca al método *waitForCompletion* para la ejecución del job.