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.

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)
```

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 son strings, por lo que habrá que hacer todas las conversiones pertinentes

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

WordCount – Función reduce

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

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()
```

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.

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 (clase del emulador de MapReduce).