Conceptos y Aplicaciones de Big Data

MapReduce (emulador MRE)

Desarrollo en Python

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

Combiner - Implementación

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

La función *combiner* se implementa con la misma filosofía que una función *reduce*.
Recibe una clave y todos los valores asociados a esa clave (sólo los generados hasta el momento)

Combiner - Seteo

```
job = Job(inputDir, outputDir, fMap, fRed)
job.setCombiner(fComb)
success = job.waitForCompletion()
```

La función *combiner* se setea mediante el método *setCombiner* del job creado

Combiner - Seteo

```
job = Job(inputDir, outputDir, fMap, fRed)
job.setCombiner(fRed)
success = job.waitForCompletion()
```

En la mayoría de los problemas como función *combiner*, podremos usar la misma función *reduce*.

Ejecutando varios jobs

```
job1 = Job(inputDir, tmpDir, fmap1, fred1)
success = job1.waitForCompletion()

job2 = Job(tmpDir, outputDir, fmap2, fred2)
success = job2.waitForCompletion()
```

El segundo job se ejecutará una vez que finalice el primero

Cada job puede configurarse con sus propias funciones *map* y *reduce*. Eventualmente dos o más jobs podrían usar las mismas funciones.

Ejecutando varios jobs

```
job1 = Job(inputDir, tmpDir, fmap1, fred1)
success = job1.waitForCompletion()

job2 = Job(tmpDir, outputDir, fmap2, fred2)
success = job2.waitForCompletion()
```

El segundo job debería leer el directorio usado como salida por el primer job.

Ejecutando varios jobs – Proceso iterativo

```
continuar = True
while (continuar):
    job = Job(inputDir, outputDir,
                        fmap, fred)
    success = job.waitForCompletion()
    continuar = evaluarFin()
```

Esta evaluación (función *evaluarFin*) involucra leer el resultado del job (que está en *outputDir*) para saber si hay que continuar o no con el proceso

Parametrizando jobs

```
job = Job(inputDir, outputDir, fmap, fred)
diccionario = {"param1": 3, " param2": 5}
job.setParams(diccionario)
success = job.waitForCompletion()
```

La parametrización del job se hace mediante el método *setParams* del job. Como parámetro recibe un diccionario. Este diccionario estará disponible para usarlo en la función *map*.

Parametrizando jobs

```
def fmap(key, value, context):
    words = value.split()
    for w in words:
        if(len(w) > context["param1"]):
            context.write(w, 1)
```

En la función *map* podemos obtener los valores del diccionario que necesitemos mediante su acceso a través de *context*.

WordCount extendido
Cuenta palabras de más de cinco caracteres y las devuelve si ocurren más de 10 veces

```
def fmap(key, value, context):
    words = value.split()
    for w in words:
        if(len(w) > context["min_len"]):
            context.write(w, 1)

def fred(key, values, context):
    c=0
    for v in values:
        c=c+1
    if (c > context["min_ocur"]):
         context.write(key, c)

job = Job(inputDir, outputDir, fmap, fred)
job.setCombiner(fred)
job.setParams({"min_len": 5, "min_ocur": 10})
success = job.waitForCompletion()
```