Conceptos y Aplicaciones de Big Data

MapReduce (emulador MRE)

Desarrollo en Python

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

Usando más de un map como entrada a un job

```
job = Job(inputDir1, outputDir, fMap1, fRed)

job.addInputPath(inputDir2, fmap2)
job.addInputPath(inputDir3, fmap3)

success = job.waitForCompletion()
```

Al directorio de entrada y función map pasados al momento de crear un *Job*, se le pueden agregar más directorios de entrada, cada uno con su función map correspondiente

Comparador - Shuffle

```
def fShuffleCmp(aKey, anotherKey):
    if(aKey[0] == anotherKey[0]):
        return 0
    elif(aKey[0] < anotherKey[0]):
        return -1
    else:
        return 1
```

Se puede implementar una función que compare dos claves durante la tarea shuffle. Lo que se recibe en esta función son las claves intermedias (k2) escritas por los *mappers* y *combiners*.

Comparador - Shuffle

```
def fShuffleCmp(aKey, anotherKey):
    if(aKey[0] == anotherKey[0]):
        return 0
    elif(aKey[0] < anotherKey[0]):
        return -1
    else:
        return 1
```

Esta función debe devolver 0 si *aKey* es igual a *anotherKey* (por el criterio que se necesite), devolver -1 si *aKey* es menor que *anotherKey* y devolver 1 si *aKey* es mayor que *anotherKey*

Comparador - Shuffle

```
def fShuffleCmp(aKey, anotherKey):
    if(aKey[0] == anotherKey[0]):
        return 0
    elif(aKey[0] < anotherKey[0]):
        return -1
    else:
        return 1
```

En este ejemplo y sabiendo que las claves son strings, se están comparando por su primer letra. Es decir todas las palabras que comiencen con el mismo carácter serán recibidas por el mismo reducer.

Comparador - Sort

```
def fSortCmp(aKey, anotherKey):
    if(len(aKey) == len(anotherKey)):
        return 0
    elif(len(aKey) < len(anotherKey)):
        return -1
    else:
        return 1
```

Se puede implementar una función que compare dos claves durante la tarea sort. Lo que se recibe en esta función son las claves intermedias (k2) escritas por los *mappers* y *combiners*.

Comparador - Sort

```
def fSortCmp(aKey, anotherKey):
    if(len(aKey) == len(anotherKey)):
        return 0
    elif(len(aKey) < len(anotherKey)):
        return -1
    else:
        return 1
```

Esta función debe devolver 0 si *aKey* es igual a *anotherKey* (por el criterio que se necesite), devolver -1 si *aKey* es menor que *anotherKey* y devolver 1 si *aKey* es mayor que *anotherKey*

Comparador - Sort

```
def fSortCmp(aKey, anotherKey):
    if(len(aKey) == len(anotherKey)):
        return 0
    elif(len(aKey) < len(anotherKey)):
        return -1
    else:
        return 1
```

En este ejemplo y sabiendo que las claves son strings, se están comparando por su longitud. Es decir todas las palabras que reciba un mismo *reducer*, vendran en orden ascendente por cantidad de caracteres

Comparador

```
job = Job(inputDir, outputDir, fmap, fred)
job.setShuffleCmp(fShuffleCmp)
job.setSortCmp(fSortCmp)
success = job.waitForCompletion()
```

Los comparadores personalizados se setean mediante los métodos *setShuffleCmp* y *setSortCmp* del job.