

# Conceptos y Aplicaciones de Big Data

---

MAPREDUCE (EMULADOR MRE)

DESARROLLO EN PYTHON

Prof. Waldo Hasperué  
[whasperue@lidi.info.unlp.edu.ar](mailto:whasperue@lidi.info.unlp.edu.ar)

# Combiner - Implementación

---

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

---

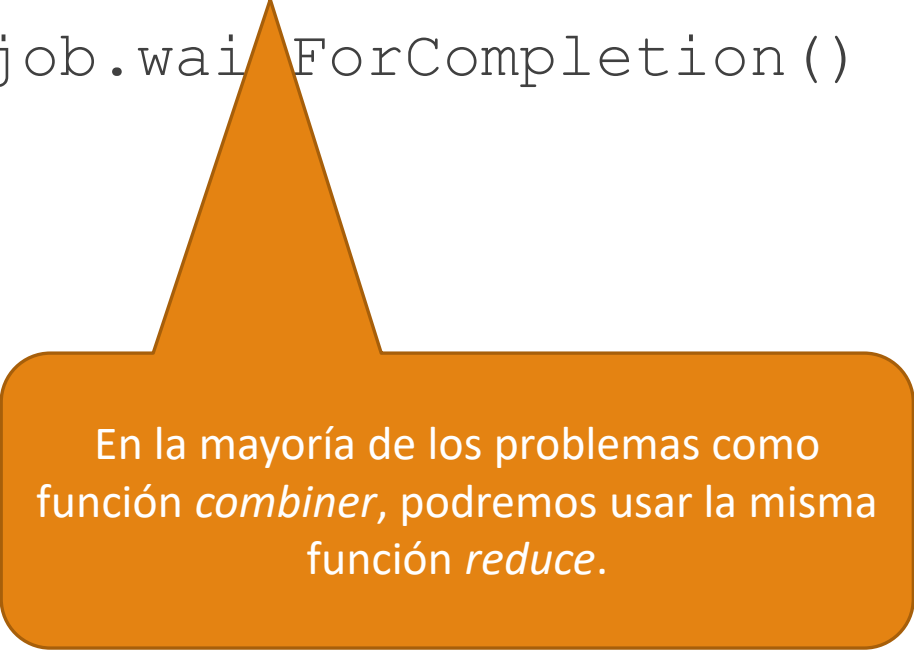
```
job = Job(inputDir, outputDir, fMap, fRed)  
job.setCombiner(fComb)  
success = job.waitForCompletion()
```

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.