Conceptos y Aplicaciones de Big Data

MAPREDUCE

PROBLEMAS ITERATIVOS
PARAMETRIZACIÓN DE TASKS

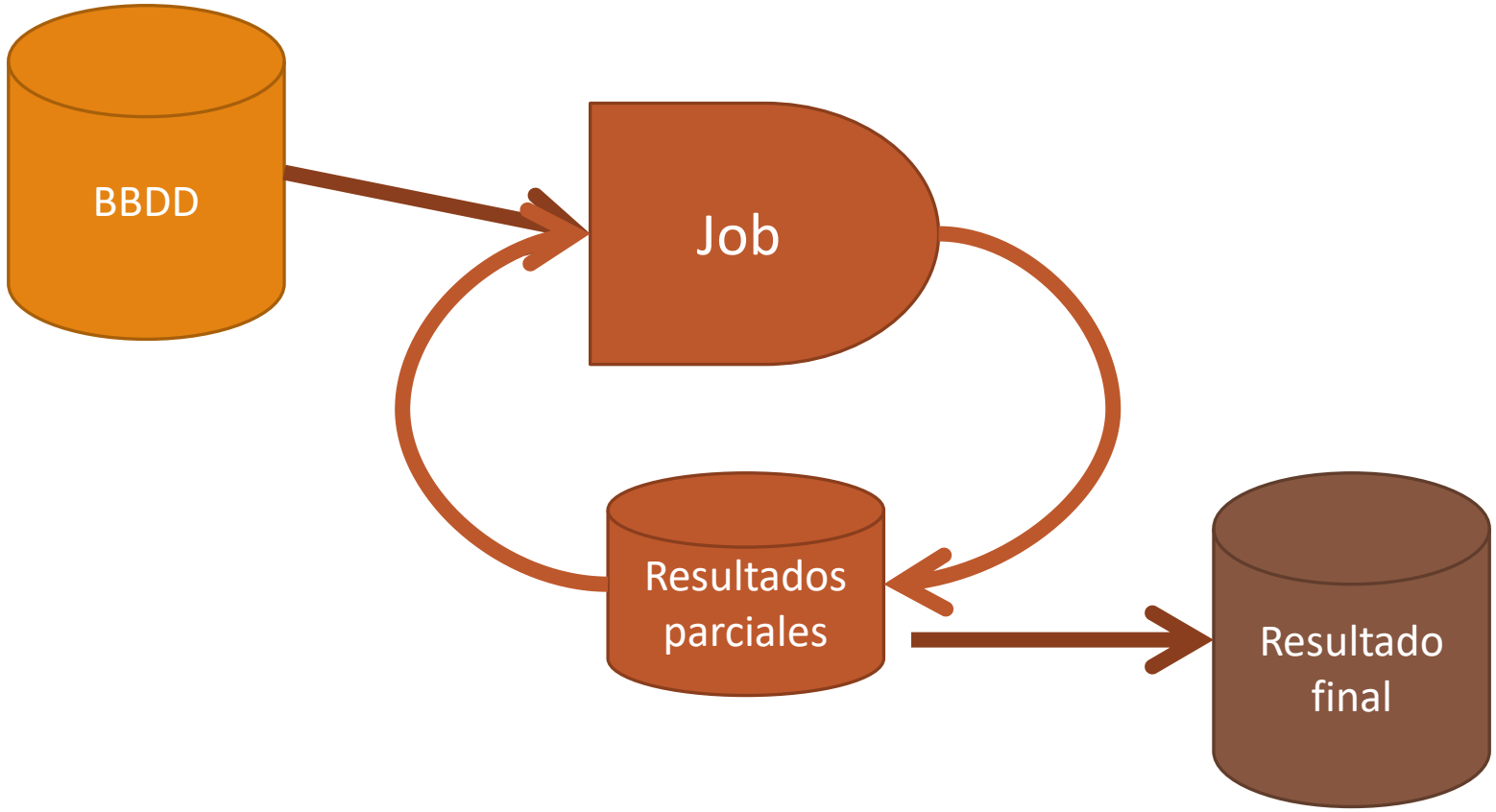
Prof. Waldo Hasperué
whasperue@lidi.info.unlp.edu.ar

Ejecutando varios jobs

Existen muchos problemas donde se necesita "iterar" sobre el mismo conjunto de datos varias veces.

- cálculos de índices
- clustering,
- árboles de clasificación,
- redes neuronales,
- algoritmos evolutivos,
- meteheurísticas de optimización,
- simulación
- etc.

DAG ejemplo de un proceso iterativo



Ejemplo - Método de Jacobi

El método de Jacobi es un método iterativo, usado para resolver sistemas de ecuaciones lineales cuadrados (igual cantidad de ecuaciones que de incógnitas):

$$
\begin{aligned}
X - 3Y - \tfrac{1}{2}Z &= 1 \\
-\tfrac{1}{10}X + Y - \tfrac{1}{10}Z &= -4 \\
-\tfrac{1}{2}X - \tfrac{1}{2}Y + Z &= 1
\end{aligned}
$$

Ejemplo - Método de Jacobi

Se inicia con un valor arbitrario para cada incógnita.
Ejemplo: ($X_0 = 1$; $Y_0 = 2$; $Z_0 = 3$)

Se despeja una incógnita diferente en cada ecuación:

$$X_1 = 1 + 3\,Y_0 + 1/2\,Z_0$$

$$Y_1 = -4 + 1/10\,X_0 + 1/10\,Z_0$$

$$Z_1 = 1 + 1/2\,X_0 + 1/2\,Y_0$$

Ejemplo - Método de Jacobi

Con los valores de la iteración *i* se calculan las variables de la iteración *i*+1:

$$(X_i = 1; Y_i = 2; Z_i = 3)$$

$$X_{i+1} = 1 + 3\,Y_i + 1/2\,Z_i = 8.5$$
$$Y_{i+1} = -4 + 1/10\,X_i + 1/10\,Z_i = -3.6$$
$$Z_{i+1} = 1 + 1/2\,X_i + 1/2\,Y_i = 2.5$$

$$(X_{i+1} = 8{,}5; Y_{i+1} = -3{,}6; Z_{i+1} = 2{,}5)$$
$$(X_{i+2} = -8{,}55; Y_{i+2} = -2{,}9; Z_{i+2} = 3{,}45)$$

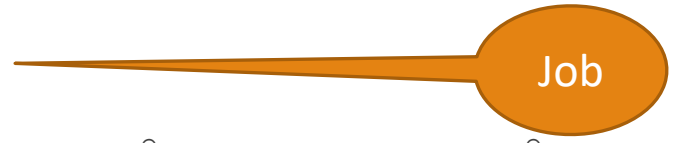
Ejemplo - Método de Jacobi

Iteración	X	Y	Z	Dif_ant
0	1	2	3	
1	8.5	-3.6	2.5	87.86
2	-8.55	-2.9	3.45	292.095
3	-5.975	-4.51	-4.725	76.05335
4	-14.8925	-5.07	-4.2425	80.06821
5	-16.3313	-5.9135	-8.98125	25.23725
…	…	…	…	…
19	-43.3555	-10.7602	-25.5819	0.950582
…	…	…	…	…
38	-49.2383	-11.8579	-29.4935	0.012495
39	-49.3203	-11.8732	-29.5481	0.009948

Ejemplo - Método de Jacobi

El método finaliza cuando la diferencia entre los valores de las incógnitas de dos iteraciones consecutivas son menores a un cierto error preestablecido:

```
error = 0.01
dif = 1
incog = {"X": 1, "Y": 2, "Z": 3}
while (dif >= error):
      calcular (incog_i, incog_i+1)
      dif = (X_i+1 - X_i)^2 + (Y_i+1  - Y_i)^2 + (Z_i+1 - Z_i)^2
```



Método de Jacobi en MapReduce

Supongamos tener un Dataset con N ecuaciones de N incógnitas en el formato que espera el método de Jacobi

<var_1, t_i, c_1, c_2, . . ., c_N>

<var_2, t_i, c_1, c_2, . . ., c_N>

<var_3, t_i, c_1, c_2, . . ., c_N>

. . .

<var_N, t_i, c_1, c_2, . . ., c_N>

¿Cómo se implementan el map y el reduce?

¿Cuál es el problema al cambiar de iteración?

Método de Jacobi – Fase map

< X	1	0	3	1/2 >
< Y	-4	1/10	0	1/10 >
< Z	1	1/2	1/2	0 >

```
def map(key, value, context):
      vars = (1, 1, 2, 3)
      coefs = value.split("\t")
      res = 0
      for v in range(4):
            res = res + vars[i] * coefs[i]
      context.write(key, res)
```

Método de Jacobi – Fase reduce

```
<X        8.5 >
<Y       -3.6 >
<Z        2.5 >
```

```
def reduce(key, values, context):
    res = 0
    for v in values:
        res = v
    context.write(key, res)
```

Método de Jacobi – ¿Problema?

```
def map(key, value, context):
      vars = (1, 1, 2, 3)

      coefs = value.split("\t")
      res = 0
      for v in range(4):
            res = res + vars[i] * coefs[i]
      context.write(key, res)
```

Estos valores deberían ser reemplazados por (1, 8.5, -3.6, 2,5)

Parametrizando jobs

A veces, en algunos problemas, resulta útil pasar parámetros a los diferentes jobs para que estos realicen su tarea.

Los valores se setean el driver y estos son pasados a los TaskTracker que ejecutan los mappers y los reducers.

Método de Jacobi en MapReduce

Inicialmente el driver envía un valor arbitrario para cada incógnita.

Esos valores son enviados a todos los TaskTracker

Finalizado el job, lee los datos que resultaron del job y los utiliza para:

- Calcular el error para chequear la condición de fin
- Enviarlos nuevamente a los TaskTracker en una nueva iteración, si se debe continuar

Método de Jacobi – Fase map

```
def map(key, value, context):
    vars = context["incognitas"]
    coefs = value.split("\t")
    res = 0
    for v in range(4):
        res = res + vars[i] * coefs[i]
    context.write(key, res)
```

Método de Jacobi – Driver

```
job = Job(inputDir, outputDir, fmap, fred)


coefs = {"incognitas": [1, 1, 2 , 3]}
job.setParams(coefs)


success = job.waitForCompletion()
```