Conceptos y aplicaciones en Big Data

Práctica 2 - Hadoop MapReduce

1) El dataset Jacobi tiene los coeficientes de un sistema de 15 ecuaciones de 15 incógnitas. Las ecuaciones en el archivo ya están "despejadas" como lo requiere el método de Jacobi. Cada línea del archivo posee:

   <var_N, term_ind, coef_var1, coef_var2, … , coef_ var15>

   Por simplicidad, para cada variable *N* su correspondiente coeficiente es cero. Implemente en MapReduce el cálculo del método de Jacobi para la solución del sistema de ecuaciones dado.

2) Resuelva el problema del método de Jacobi con el dataset jacobi2.

   Este dataset tiene los datos almacenados de la siguiente forma:

   <incognita_i, coef_i, valor>

   Donde para cada incógnita, los términos de la ecuación correspondiente aparecen en distintas tuplas, inclusive el término independiente. El valor de coef_i es, o bien el nombre de la incógnita afectada por el coeficiente valor, o bien el string “TI”, haciendo referencia a que valor es el término independiente de dicha ecuación. Ejemplo:

incognita_i	coef_i	valor
X	TI	1
X	Y	3
X	Z	0.5
Y	TI	-4
Y	X	1/10
Y	Z	1/10
Z	TI	1
Z	X	1/2
Z	Y	1/2

   Nota: Continúe enviando los valores de las incógnitas por parámetros a los *mappers* y *reducers*, según corresponda.

3) Cómo plantearía una solución MapReduce al siguiente algoritmo secuencial:

   i. entrada

   ```
   datos: array [1..N] of <int1, int2, ..., intM>
   ```

   ii. algoritmo

   ```
   error = 0.001; dif = 1; K=5; M=5

   prom = [ 0, 0, 0, 0, 0 ]; N=len(lines)

   for t in lines:
       v = t.split("\t")
   ```

```
    for m in range(M):
        prom[m] = prom[m] + float(v[m])

C=[]

for m in range(K):
    e=[]
    for m in range(M):
        e.append( prom[m] / N + random.random())
    C.append(e)
while dif > error:
    S=[]
    for m in range(K):
        S.append([ [0,0,0,0,0], 0 ])

    for t in lines:
        v = t.split("\t")
        min=9999999
        for q in range(K):
            a=0
            for m in range(M):
                a=a + (C[q][m] - float(v[m]))**2
            if(a < min):
                min = a
                Q = q
        for m in range(M):
            S[Q][0][m]= S[Q][0][m] + float(v[m])
        S[Q][1] = S[Q][1] + 1

    for q in range(K):
        if S[q][1] > 0:
            for m in range(M):
                S[q][0][m]= S[q][0][m] / S[q][1]

    dif=0
    for q in range(K):
        for m in range(M):
            dif=dif + (S[q][0][m] - C[q][m])**2
            if(S[q][1] > 0):
                C[q][m] = S[q][0][m]
```