

Desarrollo de programa MapReduce en Python

1. Desarrollo de programa tempMax:

Problema: calcular la temperatura máxima de cada año

Subproblema: por cada año calculamos la temperatura máxima

Clave: año

Valor: temperatura que ocurrió en ese año

Reduce: recibe un año junto con todas sus temperaturas. Entonces calcula la máxima

Map: recibe una línea de texto que contiene el año\tmes\ttemperatura

(ej 1999 Enero 3). Emite <año, temperatura> (<1999, 3>)

- a. Creamos una carpeta: mkdir /home/moranjesus/Desktop/tempMax

```
[moranjesus@localhost ~]$ mkdir /home/moranjesus/Desktop/tempMax  
[moranjesus@localhost ~]$
```

- b. Nos ubicamos en esa carpeta desde la terminal:

```
cd /home/moranjesus/Desktop/tempMax  
[moranjesus@localhost ~]$ cd /home/moranjesus/Desktop/tempMax  
[moranjesus@localhost tempMax]$
```

- c. Creamos un archivo llamado mapperMaxTemp.py, para ello:

```
touch mapperMaxTemp.py  
[moranjesus@localhost tempMax]$ touch mapperMaxTemp.py  
[moranjesus@localhost tempMax]$
```

- d. Damos doble click en ese archivo y escribimos:

```
#!/usr/bin/python3  
import sys
```

```
'''
```

```
Mapper de maxTemp
```

```
Creado por Jesus Moran
```

```
'''
```

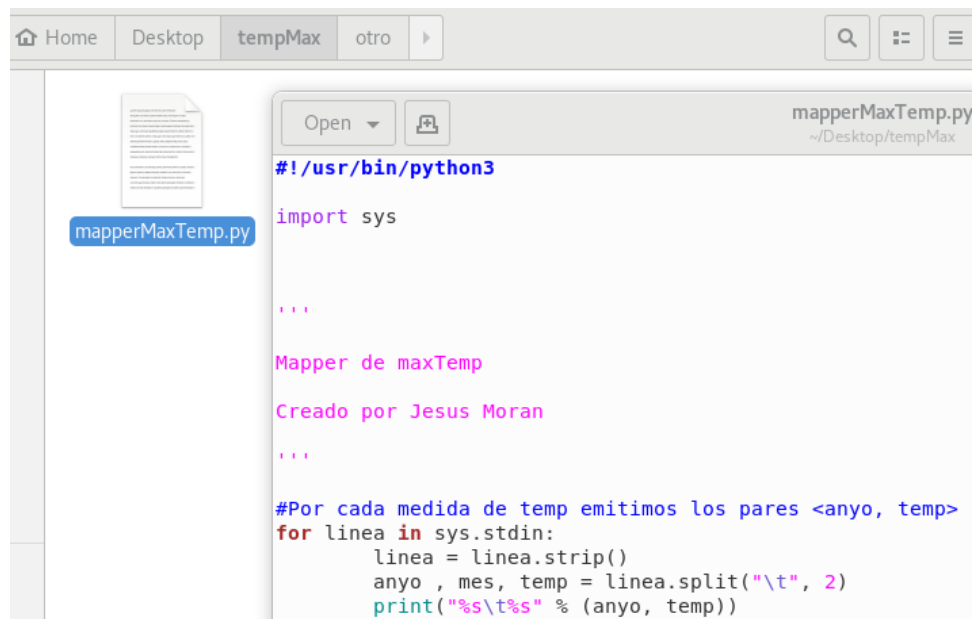
```
#Por cada medida de temp emitimos los pares <anyo, temp>
```

```
for linea in sys.stdin:
```

```
    linea = linea.strip()
```

```
    anyo , mes, temp = linea.split("\t", 2)
```

```
    print("%s\t%s" % (anyo, temp))
```



- e. Le damos permisos de ejecución: `chmod u+x mapperMaxTemp.py`
- ```

[moranjesus@localhost tempMax]$ chmod u+x mapperMaxTemp.py
[moranjesus@localhost tempMax]$

```

- f. Creamos un archivo llamado `reducerMaxTemp.py`, para ello:
- ```

touch reducerMaxTemp.py
[moranjesus@localhost tempMax]$ touch reducerMaxTemp.py
[moranjesus@localhost tempMax]$

```

- g. Damos doble click en ese archivo y escribimos:

```

#!/usr/bin/python3
import sys

'''
Reducer de MaxTemp
Creado por Jesus Moran
'''

```

```

subproblema = None
tempMaxima = None

```

```

for claveValor in sys.stdin:
    anyo, temp = claveValor.split("\t", 1)

```

```

    #convertimos la temp a float
    temp = float(temp)

```

```

    #El primer subproblema es el primer anyo de reducir
    (y la temp maxima de momento tambien)
    if subproblema == None:
        subproblema = anyo
        tempMaxima = temp

```

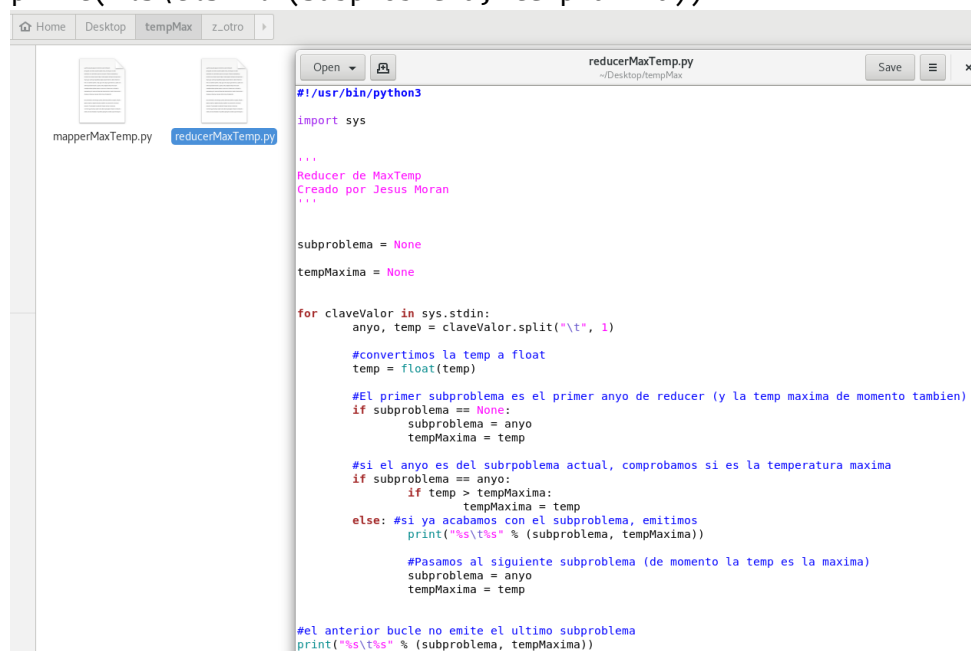
```

        #si el anyo es del subproblema actual, comprobamos
        si es la temperatura maxima
        if subproblema == anyo:
            if temp > tempMaxima:
                tempMaxima = temp
        else: #si ya acabamos con el subproblema, emitimos
            print("%s\t%s" % (subproblema, tempMaxima))

        #Pasamos al siguiente subproblema (de momento
        la temp es la maxima)
        subproblema = anyo
        tempMaxima = temp

#el anterior bucle no emite el ultimo subproblema
print("%s\t%s" % (subproblema, tempMaxima))

```



```

#!/usr/bin/python3
import sys

'''
Reducer de MaxTemp
Creado por Jesus Moran
'''

subproblema = None
tempMaxima = None

for claveValor in sys.stdin:
    anyo, temp = claveValor.split("\t", 1)

    #convertimos la temp a float
    temp = float(temp)

    #El primer subproblema es el primer anyo de reducir (y la temp maxima de momento tambien)
    if subproblema == None:
        subproblema = anyo
        tempMaxima = temp

    #si el anyo es del subproblema actual, comprobamos si es la temperatura maxima
    if subproblema == anyo:
        if temp > tempMaxima:
            tempMaxima = temp
    else: #si ya acabamos con el subproblema, emitimos
        print("%s\t%s" % (subproblema, tempMaxima))

    #Pasamos al siguiente subproblema (de momento la temp es la maxima)
    subproblema = anyo
    tempMaxima = temp

#el anterior bucle no emite el ultimo subproblema
print("%s\t%s" % (subproblema, tempMaxima))

```

h. Le damos permisos de ejecución: `chmod u+x reducerMaxTemp.py`

```

[moranjesus@localhost tempMax]$ chmod u+x reducerMaxTemp.py
[moranjesus@localhost tempMax]$

```

2. Creamos un archivo llamado medidas.txt con varias temperaturas

a. Creamos el archivo: `touch medidas.txt`

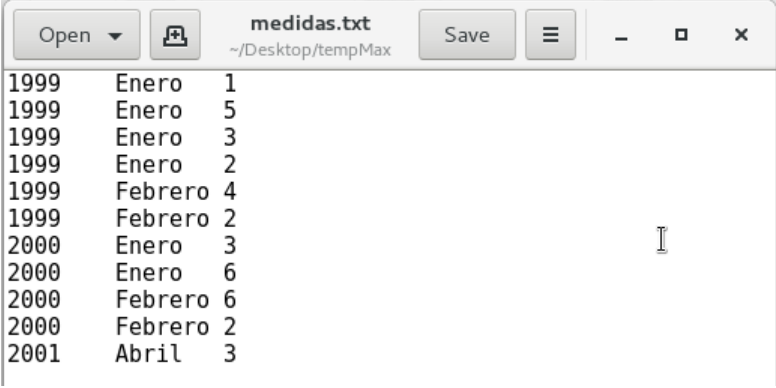
b. Damos doble click al archivo y añadimos varias temperaturas

```

1999 Enero 1
1999 Enero 5
1999 Enero 3
1999 Enero 2
1999 Febrero 4
1999 Febrero 2
2000 Enero 3
2000 Enero 6
2000 Febrero 6
2000 Febrero 2
2001 Abril 3

```

(no dejar una línea vacía)



1999	Enero	1
1999	Enero	5
1999	Enero	3
1999	Enero	2
1999	Febrero	4
1999	Febrero	2
2000	Enero	3
2000	Enero	6
2000	Febrero	6
2000	Febrero	2
2001	Abril	3

3. Ejecutamos en Hadoop el programa MapReduce:

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -files ./mapperMaxTemp.py,./reducerMaxTemp.py -mapper ./mapperMaxTemp.py -reducer ./reducerMaxTemp.py -input medidas.txt -output ./salidaMaxTemp1
```

IMPORTANTE: al copiar-pegar del pdf hay veces que no deja copiar todas las líneas a la vez. En tal caso, fijarse siempre que lo copiado coincide con la siguiente imagen

```
[moranjesus@localhost tempMax]$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -files ./mapperMaxTemp.py,./reducerMaxTemp.py -mapper ./mapperMaxTemp.py -reducer ./reducerMaxTemp.py -input medidas.txt -output ./salidaMaxTemp1
```

Lo anterior también se puede ejecutar con el siguiente comando que utiliza file en lugar de files. Se recomienda utilizar files.

```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -file ./mapperMaxTemp.py -mapper ./mapperMaxTemp.py -file ./reducerMaxTemp.py -reducer ./reducerMaxTemp.py -input medidas.txt -output ./salidaMaxTemp1
```

```
[moranjesus@localhost tempMax]$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -file ./mapperMaxTemp.py -mapper ./mapperMaxTemp.py -file ./reducerMaxTemp.py -reducer ./reducerMaxTemp.py -input medidas.txt -output ./salidaMaxTemp1
```

Una vez ejecutado, se tiene:

```

File System Counters
  FILE: Number of bytes read=143337
  FILE: Number of bytes written=768839
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
Map-Reduce Framework
  Combine input records=0
  Combine output records=0
  Reduce input groups=3
  Reduce shuffle bytes=105
  Reduce input records=11
  Reduce output records=3
  Spilled Records=11
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=0
  Total committed heap usage (bytes)=148410368
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Output Format Counters
  Bytes Written=39

```

IMPORTANTE: Si se ejecuta dos veces el programa con el mismo comando, se tiene un mensaje de error como el siguiente:

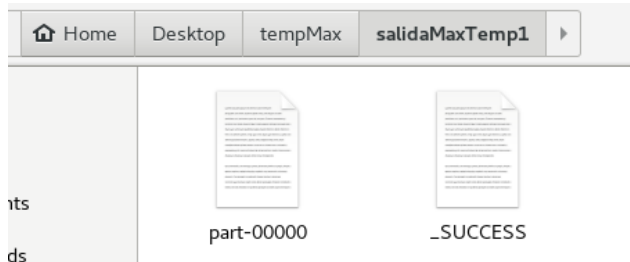
```

2021-11-18 14:53:02,649 ERROR streaming.StreamJob: Error Launching job : Output
directory file:/home/moranjesus/Desktop/tempMax/salidaMaxTemp1 already exists
Streaming Command Failed!

```

Esto quiere decir que o bien borramos la carpeta salidaMaxTemp1 o bien guardamos el resultado en otra carpeta

4. Si queremos ver la salida, la tenemos en ./salidaMaxTemp1



5. Si le damos doble click, podemos verla:



6. Ejecutamos en Hadoop el programa MapReduce con Combiner:

```

hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar
-files ./mapperMaxTemp.py./reducerMaxTemp.py -mapper ./mapperMaxTemp.py
-reducer ./reducerMaxTemp.py -combiner ./reducerMaxTemp.py -input
medidas.txt -output ./salidaMaxTemp2

```

```
[moranjesus@localhost tempMax]$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -files ./mapperMaxTemp.py,./reducerMaxTemp.py -mapper ./mapperMaxTemp.py -reducer ./reducerMaxTemp.py -combiner ./reducerMaxTemp.py -input medidas.txt -output ./salidaMaxTemp2
```

File System Counters

```
FILE: Number of bytes read=286300  
FILE: Number of bytes written=1539068  
FILE: Number of read operations=0  
FILE: Number of large read operations=0  
FILE: Number of write operations=0
```

Map-Reduce Framework

```
Map input records=11  
Map output records=11  
Map output bytes=77  
Map output materialized bytes=39  
Input split bytes=101  
Combine input records=11  
Combine output records=3  
Reduce input groups=3  
Reduce shuffle bytes=39  
Reduce input records=3  
Reduce output records=3  
Spilled Records=6  
Shuffled Maps =1  
Failed Shuffles=0  
Merged Map outputs=1  
GC time elapsed (ms)=25  
Total committed heap usage (bytes)=296968192
```

Shuffle Errors

```
BAD_ID=0  
CONNECTION=0  
IO_ERROR=0  
WRONG_LENGTH=0  
WRONG_MAP=0  
WRONG_REDUCE=0
```

File Input Format Counters

```
Bytes Read=150
```

File Output Format Counters

```
Bytes Written=39
```

Otras opciones: Depurar Reducer

Supongamos que queremos ver cómo se procesó la parte Reducer, para ello:

7. Creamos un archivo llamado reducerMaxTempDebug.py, para ello:

touch reducerMaxTempDebug.py

8. Damos doble click en ese archivo y escribimos:

```
#!/usr/bin/python3
import sys
import os

'''
Reducer de MaxTemp
Creado por Jesus Moran
'''

subproblema = None
tempMaxima = None

tarea = os.environ.get('mapreduce_task_partition')
print("-----")
print("-----")
print("Tarea " + str(tarea))

numReducer = 1
print("\t-----")
print("\tReducer " + str(numReducer))
numReducer = numReducer + 1

for claveValor in sys.stdin:
    anyo, temp = claveValor.split("\t", 1)

    #convertimos la temp a float
    temp = float(temp)

    #El primer subproblema es el primer anyo de reducer (y la
    temp maxima de momento tambien)
    if subproblema == None:
        subproblema = anyo
        tempMaxima = temp

    #si el anyo es del subrpoblema actual, comprobamos si es
    la temperatura maxima
    if subproblema == anyo:
        print("\t\tEntrada: " + claveValor)
        if temp > tempMaxima:
            tempMaxima = temp
    else: #si ya acabamos con el subproblema, emitimos
print("\t\t\tSalida: %s\t%s" % (subproblema, tempMaxima))
```

```

        #Pasamos al siguiente subproblema (de momento la
temp es la maxima)
        print("\t-----")
        print("\tReducer " + str(numReducer))
        numReducer = numReducer + 1
        print("\t\tEntrada: " + claveValor)
        subproblema = anyo
        tempMaxima = temp

#el anterior bucle no emite el ultimo subproblema
print("\t\t\tSalida: %s\t%s" % (subproblema, tempMaxima))

```

```

subproblema = None
tempMaxima = None

tarea = os.environ.get('mapreduce_task_partition')
print("-----")
print("-----")
print("Tarea " + str(tarea))

numReducer = 1
print("\t-----")
print("\tReducer " + str(numReducer))
numReducer = numReducer + 1

for claveValor in sys.stdin:
    anyo, temp = claveValor.split("\t", 1)

    #convertimos la temp a float
    temp = float(temp)

    #El primer subproblema es el primer anyo de reducer (y la temp maxima de momento tambien)
    if subproblema == None:
        subproblema = anyo
        tempMaxima = temp

    #si el anyo es del subproblema actual, comprobamos si es la temperatura maxima
    if subproblema == anyo:
        print("\t\tEntrada: " + claveValor)

        if temp > tempMaxima:
            tempMaxima = temp

    else: #si ya acabamos con el subproblema, emitimos
        print("\t\t\tSalida: %s\t%s" % (subproblema, tempMaxima))

        #Pasamos al siguiente subproblema (de momento la temp es la maxima)
        print("\t-----")
        print("\tReducer " + str(numReducer))
        numReducer = numReducer + 1

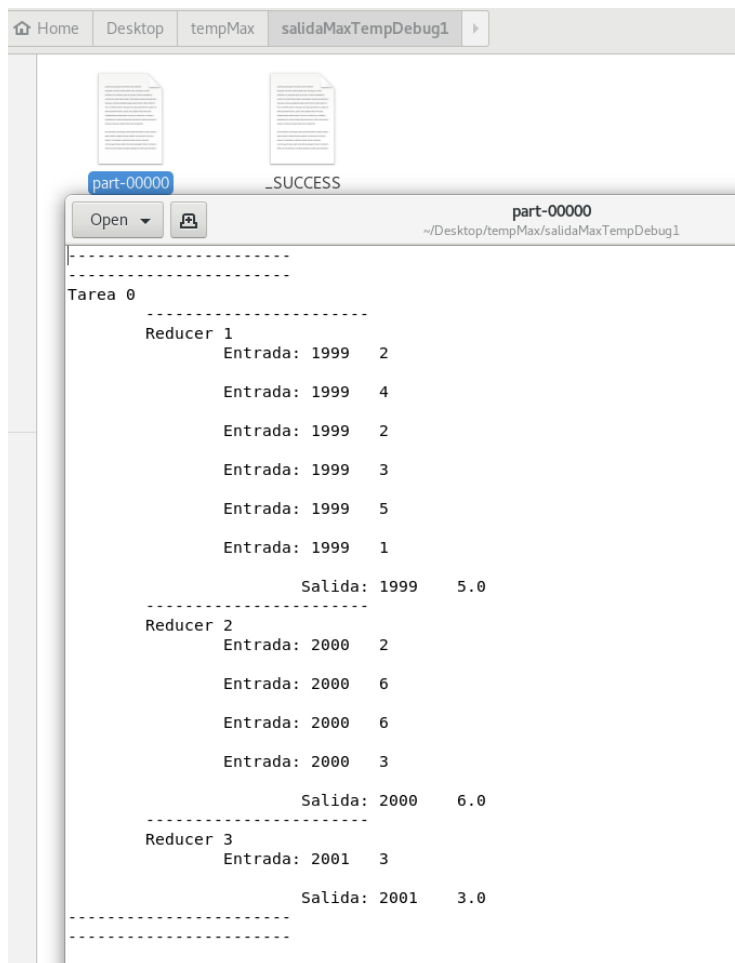
        print("\t\tEntrada: " + claveValor)
        subproblema = anyo
        tempMaxima = temp

#el anterior bucle no emite el ultimo subproblema
print("\t\t\tSalida: %s\t%s" % (subproblema, tempMaxima))

print("-----")
print("-----")

```

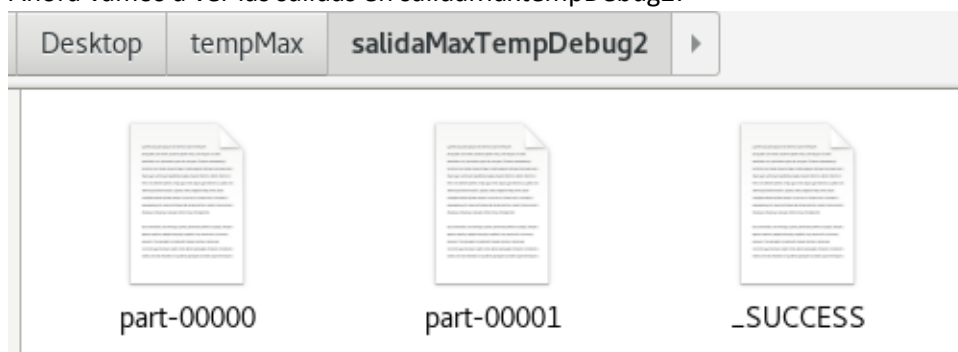
9. Le damos permisos de ejecución: `chmod u+x reducerMaxTempDebug.py`
10. Lo ejecutamos: `hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -files ./mapperMaxTemp.py,./reducerMaxTempDebug.py -mapper ./mapperMaxTemp.py -reducer ./reducerMaxTempDebug.py -input medidas.txt -output ./salidaMaxTempDebug1`
`[moranjesus@localhost tempMax]$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -files ./mapperMaxTemp.py,./reducerMaxTempDebug.py -mapper ./mapperMaxTemp.py -reducer ./reducerMaxTempDebug.py -input medidas.txt -output ./salidaMaxTempDebug1`
11. Ahora vamos a ver la salida. Para ello abrimos la carpeta `salidaMaxTempDebug1` y el archivo `part-r-00000`



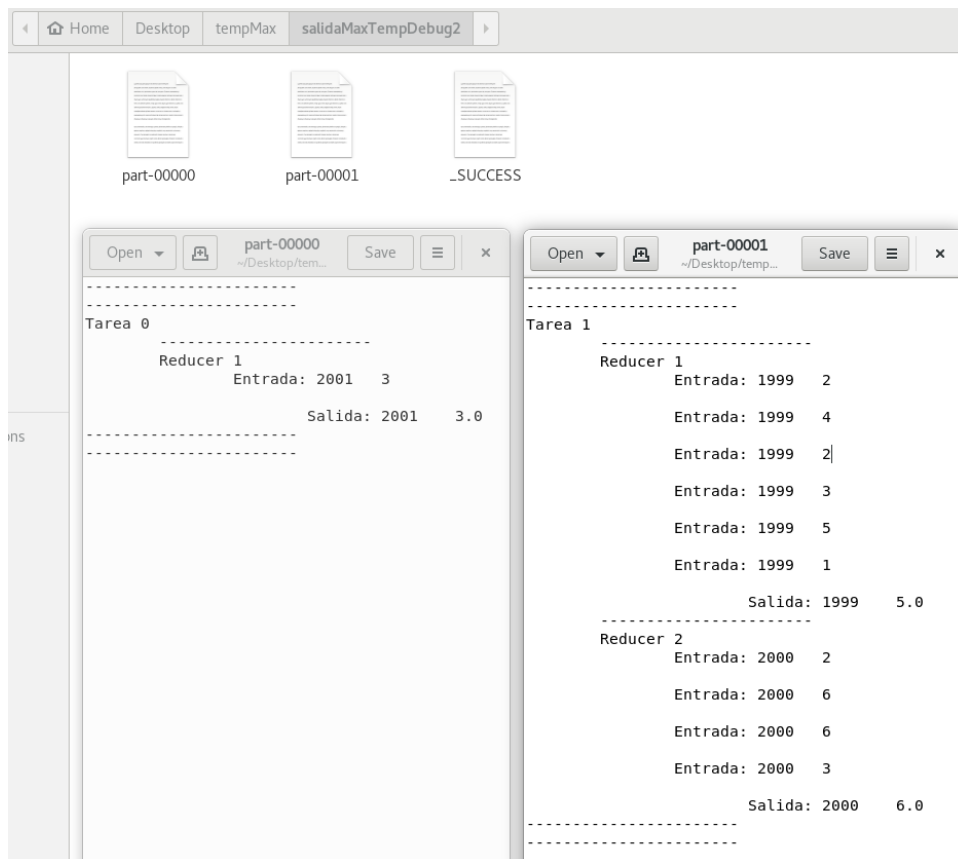
Vemos que tuvimos 3 subproblemas (unos con más entradas que otros). Y luego cada uno de ellos emite una salida (reducer puede emitir varias salidas por subproblema).

Si tenemos algún defecto en el programa, entonces podemos imprimir todo lo que se está haciendo. Y una vez que se solucione el defecto, entonces se borran los mensajes y se deja sólo la salida

12. Ahora ejecutamos el programa "simulando 2 ordenadores en la parte reducer" (2 tareas Reducer), para ello: `hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -D mapred.reduce.tasks=2 -files ./mapperMaxTemp.py,./reducerMaxTempDebug.py -mapper ./mapperMaxTemp.py -reducer ./reducerMaxTempDebug.py -input medidas.txt -output ./salidaMaxTempDebug2`
13. Ahora vamos a ver las salidas en salidaMaxtempDebug2:



14. Está el resultado de la tarea Reducer 0 y de la tarea Reducer 1, las abrimos:



15. Para ver cómo MapReduce hace el reparto entre claves y tareas Reducer, podemos ejecutar el siguiente programa Java:

```
import java.util.*;
import java.lang.Integer;

public class Main {
    public static void main(String[] args) throws Exception {
        int numberOfTasks = 2;

        printPartitionInformation("1999", numberOfTasks);
        printPartitionInformation("2000", numberOfTasks);
        printPartitionInformation("2001", numberOfTasks);
    }
}
```

```
public static void printPartitionInformation(String key, int numberOfTasks){
    System.out.println("-----");
    System.out.println("Key: " + key);
}
```

```

        System.out.println("Key hash: " + key.hashCode());

        System.out.println("Key hash & Integer.MaxInt = " + (key.hashCode() &
Integer.MAX_VALUE));

        System.out.println("Partition: " + (key.hashCode() & Integer.MAX_VALUE) %
numberOfTasks);

    }
}

```

```

import java.util.*;
import java.lang.Integer;

public class Main {
    public static void main(String[] args) throws Exception {
        int numberOfTasks = 2;

        printPartitionInformation("1999", numberOfTasks);
        printPartitionInformation("2000", numberOfTasks);
        printPartitionInformation("2001", numberOfTasks);
    }

    public static void printPartitionInformation(String key, int numberOfTasks){
        System.out.println("-----");
        System.out.println("Key: " + key);
        System.out.println("Key hash: " + key.hashCode());
        System.out.println("Key hash & Integer.MaxInt = " + (key.hashCode() & Integer.MAX_VALUE));
        System.out.println("Partition: " + (key.hashCode() & Integer.MAX_VALUE) % numberOfTasks);
    }
}

```

Nota: Hadoop no hace el hashCode de un String, sino que lo hará de un XWritable (ej IntWritable, Text, etc). Por lo que los resultados pueden ser ligeramente diferentes. Para más control, se puede convertir "1999", "2000", etc a Text y llamar a la función de Particionado.

Tendremos:

```

-----
Key: 1999
Key hash: 1516360
Key hash & Integer.MaxInt = 1516360
Partition: 0
-----
Key: 2000
Key hash: 1537214
Key hash & Integer.MaxInt = 1537214
Partition: 0
-----
Key: 2001
Key hash: 1537215
Key hash & Integer.MaxInt = 1537215
Partition: 1

```

Otras opciones: Depurar Mapper

Supongamos que queremos ver cómo se procesó la parte Reducer, para ello:

16. Creamos un archivo llamado mapperMaxTempDebug.py, para ello:
touch mapperMaxTempDebug.py

17. Damos doble click en ese archivo y escribimos:

```
#!/usr/bin/python3
```

```
import sys
```

```
import os
```

```
'''
```

```
Mapper de maxTemp
```

```
Creado por Jesus Moran
```

```
'''
```

```
tarea = os.environ.get('mapreduce_task_partition')
```

```
#Por cada medida de temp emitimos los pares <anyo, temp>
```

```
numLinea = 0
```

```
entrada = ""
```

```
salida = ""
```

```
for linea in sys.stdin:
```

```
    numLinea = numLinea + 1
```

```
    entrada = linea
```

```
    linea = linea.strip()
```

```
    anyo , mes, temp = linea.split("\t", 2)
```

```
    salida = str(anyo) + "\t" + str(temp)
```

```
print("Tarea " + tarea + " Linea " + str(numLinea) + " Entrada: " + entrada)
```

```
print("Tarea " + tarea + " Linea " + str(numLinea) + " Salida: " + salida)
```

```
mapperMaxTempDebug.py x reducerMaxTempDebug.py x
#!/usr/bin/python3

import sys
import os

'''
Mapper de maxTemp
Creado por Jesus Moran
'''

tarea = os.environ.get('mapreduce_task_partition')

#Por cada medida de temp emitimos los pares <anyo, temp>
numLinea = 0
entrada = ""
salida = ""
for linea in sys.stdin:
    numLinea = numLinea + 1
    entrada = linea
    linea = linea.strip()
    anyo , mes, temp = linea.split("\t", 2)
    salida = str(anyo) + "\t" + str(temp)

    print("Tarea " + tarea + " Linea " + str(numLinea) + " Entrada: " + entrada)
    print("Tarea " + tarea + " Linea " + str(numLinea) + " Salida: " + salida)
```

18. Le damos permisos de ejecución: `chmod u+x mapperMaxTempDebug.py`
19. Lo ejecutamos: `hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -file ./mapperMaxTempDebug.py -mapper ./mapperMaxTempDebug.py -input medidas.txt -output ./salidaMaxTempDebug3`

```
[moranjesus@localhost tempMax]$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -file ./mapperMaxTempDebug.py -mapper ./mapperMaxTempDebug.py -input medidas.txt -output ./salidaMaxTempDebug3
```

20. Ahora abrimos el archivo `part-00000` de la carpeta `salidaMaxTempDebug3`, tendremos:

Tarea 0	Linea 1	Entrada: 1999	Enero	1
Tarea 0	Linea 1	Salida: 1999		1
Tarea 0	Linea 10	Entrada: 2000	Febrero	2
Tarea 0	Linea 10	Salida: 2000		2
Tarea 0	Linea 11	Entrada: 2001	Abril	3
Tarea 0	Linea 11	Salida: 2001		3
Tarea 0	Linea 2	Entrada: 1999	Enero	5
Tarea 0	Linea 2	Salida: 1999		5
Tarea 0	Linea 3	Entrada: 1999	Enero	3
Tarea 0	Linea 3	Salida: 1999		3
Tarea 0	Linea 4	Entrada: 1999	Enero	2
Tarea 0	Linea 4	Salida: 1999		2
Tarea 0	Linea 5	Entrada: 1999	Febrero	4
Tarea 0	Linea 5	Salida: 1999		4
Tarea 0	Linea 6	Entrada: 1999	Febrero	2
Tarea 0	Linea 6	Salida: 1999		2
Tarea 0	Linea 7	Entrada: 2000	Enero	3
Tarea 0	Linea 7	Salida: 2000		3
Tarea 0	Linea 8	Entrada: 2000	Enero	6
Tarea 0	Linea 8	Salida: 2000		6
Tarea 0	Linea 9	Entrada: 2000	Febrero	6
Tarea 0	Linea 9	Salida: 2000		6

21. Ahora vamos a ejecutar el programa simulando que tenemos varios “servidores ejecutando Mappers”, es decir, con varias tareas mapper: `hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -Dmapreduce.input.lineinputformat.linespermap=3 -inputformat org.apache.hadoop.mapred.lib.NLineInputFormat -file ./mapperMaxTempDebug.py -mapper ./mapperMaxTempDebug.py -input medidas.txt -output ./salidaMaxTempDebug4`

```
[moranjésus@localhost tempMax]$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -Dmapreduce.input.lineinputformat.linespermap=3 -inputformat org.apache.hadoop.mapred.lib.NLineInputFormat -file ./mapperMaxTempDebug.py -mapper ./mapperMaxTempDebug.py -input medidas.txt -output ./salidaMaxTempDebug4
```

Con esto estamos indicando que cada Mapper debe ejecutar 3 líneas de datos. Por lo tanto, como tenemos 11 líneas, se ejecutan $\text{redondeo_hacia_arriba}(11/3)$ tareas Mapper. Es decir, se ejecutan 4 tareas Mapper (tarea 0, 1, 2 y 3). Todas las tareas Mapper ejecutarán 3 datos, salvo la última que ejecuta 2.

Importante: como veremos más adelante, esto no está del todo bien porque Hadoop Streaming por defecto hace que las Mapper reciban registros del tipo `<linea>`, pero `NLineInputFormat` lo que hace es que Mapper reciba `<offset, linea>`. En este ejemplo, el Mapper interpretará que `anyo` es el `offset`, `mes` que es el `anyo`, y `temp` que es `mes\temperatura`. Entonces o bien se adapta el programa para depurarlo (no recomendable), o se indica que no se utilice `offset`. Nota hay

escenarios en los que nos interesará tener el offset y otros en los que no. Más adelante se explica como quitar el offset

22. Ahora abrimos el archivo part-00000 de la carpeta salidaMaxTempDebug4, tendremos:

Tarea 0	Linea 1	Entrada: 39	1999	Enero	2
Tarea 0	Linea 1	Salida: 39	Enero	2	
Tarea 0	Linea 2	Entrada: 52	1999	Febrero	4
Tarea 0	Linea 2	Salida: 52	Febrero	4	
Tarea 0	Linea 3	Entrada: 67	1999	Febrero	2
Tarea 0	Linea 3	Salida: 67	Febrero	2	
Tarea 1	Linea 1	Entrada: 82	2000	Enero	3
Tarea 1	Linea 1	Salida: 82	Enero	3	
Tarea 1	Linea 2	Entrada: 95	2000	Enero	6
Tarea 1	Linea 2	Salida: 95	Enero	6	
Tarea 1	Linea 3	Entrada: 108	2000	Febrero	6
Tarea 1	Linea 3	Salida: 108	Febrero	6	
Tarea 2	Linea 1	Entrada: 0	1999	Enero	1
Tarea 2	Linea 1	Salida: 0	Enero	1	
Tarea 2	Linea 2	Entrada: 13	1999	Enero	5
Tarea 2	Linea 2	Salida: 13	Enero	5	
Tarea 2	Linea 3	Entrada: 26	1999	Enero	3
Tarea 2	Linea 3	Salida: 26	Enero	3	
Tarea 3	Linea 1	Entrada: 123	2000	Febrero	2
Tarea 3	Linea 1	Salida: 123	Febrero	2	
Tarea 3	Linea 2	Entrada: 138	2001	Abril	3
Tarea 3	Linea 2	Salida: 138	Abril	3	

Notar que al utilizar el NLineInputFormat nos cambia la salida del Mapper ya que aparece un número (ej. Tarea 0 Linea 1 Entrada: **39**1999 Enero 2). Ese número es el offset. Si utilizamos un NLineInputFormat se tiene que (1) adecuar el programa para que reciba <offset, línea>, o (2) poner la opción -

Dstream.map.input.ignoreKey=true de forma que el programa recibirá como entrada <línea> tal y como pasaba antes de utilizar el NLineInputFormat en Hadoop Streaming

23. Ahora vamos a ejecutar el programa simulando correctamente que tenemos varios "servidores ejecutando Mappers", es decir, con varias tareas mapper: `hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -Dstream.map.input.ignoreKey=true -Dmapreduce.input.lineinputformat.linespermap=3 -inputformat org.apache.hadoop.mapred.lib.NLineInputFormat -file ./mapperMaxTempDebug.py -mapper ./mapperMaxTempDebug.py -input medidas.txt -output ./salidaMaxTempDebug5`

```
[moranjesus@localhost tempMax]$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -Dstream.map.input.ignoreKey=true -Dmapreduce.input.lineinputformat.linespermap=3 -inputformat org.apache.hadoop.mapred.lib.NLineInputFormat -file ./mapperMaxTempDebug.py -mapper ./mapperMaxTempDebug.py -input medidas.txt -output ./salidaMaxTempDebug5
```

24. Ahora abrimos el archivo part-00000 de la carpeta salidaMaxTempDebug, tendremos:

Tarea 0	Línea 1	Entrada: 1999	Enero	2
Tarea 0	Línea 1	Salida: 1999		2
Tarea 0	Línea 2	Entrada: 1999	Febrero	4
Tarea 0	Línea 2	Salida: 1999		4
Tarea 0	Línea 3	Entrada: 1999	Febrero	2
Tarea 0	Línea 3	Salida: 1999		2
Tarea 1	Línea 1	Entrada: 2000	Enero	3
Tarea 1	Línea 1	Salida: 2000		3
Tarea 1	Línea 2	Entrada: 2000	Enero	6
Tarea 1	Línea 2	Salida: 2000		6
Tarea 1	Línea 3	Entrada: 2000	Febrero	6
Tarea 1	Línea 3	Salida: 2000		6
Tarea 2	Línea 1	Entrada: 1999	Enero	1
Tarea 2	Línea 1	Salida: 1999		1
Tarea 2	Línea 2	Entrada: 1999	Enero	5
Tarea 2	Línea 2	Salida: 1999		5
Tarea 2	Línea 3	Entrada: 1999	Enero	3
Tarea 2	Línea 3	Salida: 1999		3
Tarea 3	Línea 1	Entrada: 2000	Febrero	2
Tarea 3	Línea 1	Salida: 2000		2
Tarea 3	Línea 2	Entrada: 2001	Abril	3
Tarea 3	Línea 2	Salida: 2001		3

Alternativa: Crear programa con un particionado específico

Supongamos que ahora queremos calcular la temperatura máxima de cada mes de cada año. Entonces tendremos tantos subproblemas como meses-años: 1999-Enero, 1999-Febrero...2000-Enero, 2000-Febrero....

Por tanto, la clave ya no es el año, sino el compuesto de año-Mes

En primer lugar, vamos a crear un archivo medidas2.txt con la siguiente información:

```
1999 Enero 1
1999 Enero 5
1999 Enero 3
1999 Enero 2
1999 Febrero 4
1999 Febrero 2
2000 Enero 3
2000 Enero 6
2000 Febrero 6
2000 Febrero 2
2001 Abril 3
1999 Enero 20
(no dejar línea en blanco)
```

25. Creamos un archivo llamado mapperAnyoMes.py, para ello:

```
touch mapperAnyoMes.py
```

26. Damos doble click en ese archivo y escribimos:

```
#!/usr/bin/python3
import sys

'''
Mapper de maxTemp
Creado por Jesus Moran
'''

#Por cada medida de temp emitimos los pares <{anyo, mes}, temp>
(emitirmos los tres datos)
for linea in sys.stdin:
    linea = linea.strip()
    anyo , mes, temp = linea.split("\t", 2)
    print("%s\t%s\t%s" % (anyo, mes, temp))
```

```
#!/usr/bin/python3

import sys

'''
Mapper de maxTemp
Creado por Jesus Moran
'''

#Por cada medida de temp emitimos los pares <{anyo, mes}, temp> (emitirmos los tres datos)
for linea in sys.stdin:
    linea = linea.strip()
    anyo , mes, temp = linea.split("\t", 2)
    print("%s\t%s\t%s" % (anyo, mes, temp))
```

27. Le damos permisos de ejecución: `chmod u+x mapperAnyoMes.py`

28. Creamos un archivo llamado reducerMaxTempYearMonthDebug.py, para ello:

touch reducerMaxTempYearMonthDebug.py

29. Damos doble click en ese archivo y escribimos:

```
#!/usr/bin/python3
```

```
import sys
```

```
'''
```

```
Reducer de MaxTemp
```

```
Creado por Jesus Moran
```

```
'''
```

```
subproblema = None
```

```
tempMaxima = None
```

```
numReducer = 1
```

```
print("-----")
```

```
print("Reducer " + str(numReducer))
```

```
numReducer = numReducer + 1
```

```
for claveValor in sys.stdin:
```

```
    anyo, month, temp = claveValor.split("\t", 2)
```

```
    #convertimos la temp a float
```

```
    temp = float(temp)
```

```
    #El primer subproblema es el primer anyo-mes de reducer (y la temp maxima de momento tambien)
```

```
    if subproblema == None:
```

```
        subproblema = [anyo, month]
```

```
        tempMaxima = temp
```

#si el anyo-mes es del subproblema actual, comprobamos si es la temperatura maxima

if subproblema == [anyo, month]:

 print(" Input: " + claveValor)

 if temp > tempMaxima:

 tempMaxima = temp

else: #si ya acabamos con el subproblema, emitimos

 print(" Output: %s\t%s" % (subproblema, tempMaxima))

#Pasamos al siguiente subproblema (de momento la temp es la maxima)

 print("-----")

 print("Reducer " + str(numReducer))

 numReducer = numReducer + 1

 print(" Input: " + claveValor)

 subproblema = [anyo, month]

 tempMaxima = temp

#el anterior bucle no emite el ultimo subproblema

print(" Output: %s\t%s" % (subproblema, tempMaxima))

```
#!/usr/bin/python3

import sys

'''
Reducer de MaxTemp
Creado por Jesus Moran
'''

subproblema = None
tempMaxima = None

numReducer = 1
print("-----")
print("Reducer " + str(numReducer))
numReducer = numReducer + 1

for claveValor in sys.stdin:
    anyo, month, temp = claveValor.split("\t", 2)

    #convertimos la temp a float
    temp = float(temp)

    #El primer subproblema es el primer anyo-mes de reducir (y la temp maxima de
    momento tambien)
    if subproblema == None:
        subproblema = [anyo, month]
        tempMaxima = temp

    #si el anyo-mes es del subproblema actual, comprobamos si es la temperatura maxima
    if subproblema == [anyo, month]:
        print("    Input: " + claveValor)

        if temp > tempMaxima:
            tempMaxima = temp

    else: #si ya acabamos con el subproblema, emitimos
        print("    Output: %s\t%s" % (subproblema, tempMaxima))

        #Pasamos al siguiente subproblema (de momento la temp es la maxima)
        print("-----")
        print("Reducer " + str(numReducer))
        numReducer = numReducer + 1

        print("    Input: " + claveValor)
        subproblema = [anyo, month]
        tempMaxima = temp

#el anterior bucle no emite el ultimo subproblema
print("    Output: %s\t%s" % (subproblema, tempMaxima))
```

30. Ejecutamos el programa MapReduce (sin Combiner de momento porque queremos depurar si lo está haciendo bien): `hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -files ./mapperAnyoMes.py,./reducerMaxTempYearMonthDebug.py -mapper ./mapperAnyoMes.py -reducer ./reducerMaxTempYearMonthDebug.py -input medidas2.txt -output ./salidaMaxTempAnyoMes1`

```
[moranjesus@localhost tempMax]$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -files ./mapperAnyoMes.py,./reducerMaxTempYearMonthDebug.py -mapper ./mapperAnyoMes.py -reducer ./reducerMaxTempYearMonthDebug.py -input medidas2.txt -output ./salidaMaxTempAnyoMes1
```

(NOTA: esta ejecución no funcionará porque no estamos indicando que hay una clave compuesta por dos campos. Ver a continuación)

31. Abrimos el archivo de salida:

```

part-00000
-----
Reducer 1
  Input: 1999 Enero    20

      Output: ['1999', 'Enero']    20.0
-----
Reducer 2
  Input: 1999 Febrero  2

      Input: 1999 Febrero  4

      Output: ['1999', 'Febrero']  4.0
-----
Reducer 3
  Input: 1999 Enero    2

      Input: 1999 Enero    3

      Input: 1999 Enero    5

      Input: 1999 Enero    1

      Output: ['1999', 'Enero']    5.0
-----
Reducer 4
  Input: 2000 Febrero  2

      Input: 2000 Febrero  6

      Output: ['2000', 'Febrero']  6.0
-----
Reducer 5
  Input: 2000 Enero    6

      Input: 2000 Enero    3

      Output: ['2000', 'Enero']    6.0
-----
Reducer 6
  Input: 2001 Abril    3

      Output: ['2001', 'Abril']    3.0

```

Podemos ver que la salida no es correcta porque en 1999 Enero hay dos salidas. Esto nos da pistas de que los datos no están ordenados-agrupados como queremos. El problema es que queremos que agrupe por Año-Mes y no se lo indicamos en ningún sitio. Hadoop interpreta que la clave es año (pares <año, {mes, temperatura}>), cuando la clave debería ser el mes y el año (pares <{año, mes}, temperatura>). Entonces para indicar que tenemos una clave compuesta utilizaremos los parámetros `-Dmap.output.key.field.separator="\t"` y `-Dstream.num.map.output.key.fields=2`. Estamos indicando que la clave está compuesta por 2 elementos y que están separados por una tabulación.

32. Ejecutamos el programa MapReduce indicando clave compuesta (sin Combiner de momento porque queremos depurar si lo está haciendo bien): `hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -Dmap.output.key.field.separator="\t" -Dstream.num.map.output.key.fields=2 -files ./mapperAnyoMes.py,./reducerMaxTempYearMonthDebug.py -mapper ./mapperAnyoMes.py -reducer ./reducerMaxTempYearMonthDebug.py -input medidas2.txt -output ./salidaMaxTempAnyoMes2`

```

[moranjesus@localhost tempMax]$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/h
adoop-streaming-3.3.0.jar -Dmap.output.key.field.separator="\t" -Dstream.num.map
.output.key.fields=2 -files ./mapperAnyoMes.py,./reducerMaxTempYearMonthDebug.py
-mapper ./mapperAnyoMes.py -reducer ./reducerMaxTempYearMonthDebug.py -input me
didas2.txt -output ./salidaMaxTempAnyoMes2

```

33. Ahora abrimos el archivo de salida:

```

-----
Reducer 1
  Input: 1999 Enero   20
    Input: 1999 Enero   2
    Input: 1999 Enero   3
    Input: 1999 Enero   5
    Input: 1999 Enero   1
      Output: ['1999', 'Enero']    20.0
-----
Reducer 2
  Input: 1999 Febrero  2
    Input: 1999 Febrero  4
      Output: ['1999', 'Febrero']  4.0
-----
Reducer 3
  Input: 2000 Enero    6
    Input: 2000 Enero    3
      Output: ['2000', 'Enero']    6.0
-----
Reducer 4
  Input: 2000 Febrero  2
    Input: 2000 Febrero  6
      Output: ['2000', 'Febrero']  6.0
-----
Reducer 5
  Input: 2001 Abril    3
      Output: ['2001', 'Abril']    3.0

```

Ahora ya hace lo que esperamos porque 1999-Enero lo trata como un subproblema, 1999-Febrero, como otro, etc.

34. Ahora que ya creemos que funciona bien, lo que podemos hacer es crear una copia del programa y quitarle la parte de depuración (muy importante cambiar la parte de emitir resultado para que emita tres valores: año, mes y temperatura. Lo llamamos reducerMaxTempYearMonth.py:

```
#!/usr/bin/python3
```

```
import sys
```

```
'''
```

```
Reducer de MaxTemp
```

```
Creado por Jesus Moran
```

```
'''
```

```
subproblema = None
```

```
tempMaxima = None
```

```
for claveValor in sys.stdin:
```

```
    anyo, month, temp = claveValor.split("\t", 2)
```

```
    #convertimos la temp a float
```

```
    temp = float(temp)
```

```
    #El primer subproblema es el primer anyo-mes de reducir (y la temp maxima de momento tambien)
```

```
    if subproblema == None:
```

```
        subproblema = [anyo, month]
```

```
        tempMaxima = temp
```

```
    #si el anyo-mes es del subproblema actual, comprobamos si es la temperatura maxima
```

```
    if subproblema == [anyo, month]:
```

```
        if temp > tempMaxima:
```

```
            tempMaxima = temp
```

```
    else: #si ya acabamos con el subproblema, emitimos
```

```
        print("%s\t%s\t%s" % (subproblema[0], subproblema[1], tempMaxima))
```

```
    #Pasamos al siguiente subproblema (de momento la temp es la maxima)
```

```
    subproblema = [anyo, month]
```

```
    tempMaxima = temp
```

```
#el anterior bucle no emite el ultimo subproblema
```

```
print("%s\t%s\t%s" % (subproblema[0], subproblema[1], tempMaxima))
```

```
#!/usr/bin/python3

import sys

'''
Reducer de MaxTemp
Creado por Jesus Moran
'''

subproblema = None
tempMaxima = None

for claveValor in sys.stdin:
    anyo, month, temp = claveValor.split("\t", 2)

    #convertimos la temp a float
    temp = float(temp)

    #El primer subproblema es el primer anyo-mes de reducir (y la temp maxima de
    momento tambien)
    if subproblema == None:
        subproblema = [anyo, month]
        tempMaxima = temp

    #si el anyo-mes es del subproblema actual, comprobamos si es la temperatura maxima
    if subproblema == [anyo, month]:
        if temp > tempMaxima:
            tempMaxima = temp

    else: #si ya acabamos con el subproblema, emitimos
        print("%s\t%s\t%s" % (subproblema[0], subproblema[1], tempMaxima))

        #Pasamos al siguiente subproblema (de momento la temp es la maxima)
        subproblema = [anyo, month]
        tempMaxima = temp

#el anterior bucle no emite el ultimo subproblema
print("%s\t%s\t%s" % (subproblema[0], subproblema[1], tempMaxima))
```

35. Lo ejecutamos: `hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -Dmap.output.key.field.separator="\t" -Dstream.num.map.output.key.fields=2 -files ./mapperAnyoMes.py,./reducerMaxTempYearMonth.py -mapper ./mapperAnyoMes.py -reducer ./reducerMaxTempYearMonth.py -input medidas2.txt -output ./salidaMaxTempAnyoMes3`

```
[moranjesus@localhost tempMax]$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -Dmap.output.key.field.separator="\t" -Dstream.num.map.output.key.fields=2 -files ./mapperAnyoMes.py,./reducerMaxTempYearMonth.py -mapper ./mapperAnyoMes.py -reducer ./reducerMaxTempYearMonth.py -input medidas2.txt -output ./salidaMaxTempAnyoMes3
```

Map-Reduce Framework

```
Map input records=12
Map output records=12
Map output bytes=165
Map output materialized bytes=195
Input split bytes=102
Combine input records=0
Combine output records=0
Reduce input groups=5
Reduce shuffle bytes=195
Reduce input records=12
Reduce output records=5
Spilled Records=24
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=25
Total committed heap usage (bytes)=331489280
```

36. Observamos la salida:

part-00000		
1999	Enero	20.0
1999	Febrero	4.0
2000	Enero	6.0
2000	Febrero	6.0
2001	Abril	3.0

37. Ahora lo ejecutamos con Combiner: `hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -Dmap.output.key.field.separator="\t" -Dstream.num.map.output.key.fields=2 -files ./mapperAnyoMes.py,./reducerMaxTempYearMonth.py -mapper ./mapperAnyoMes.py -combiner ./reducerMaxTempYearMonth.py -reducer ./reducerMaxTempYearMonth.py -input medidas2.txt -output ./salidaMaxTempAnyoMes4`

```
[moranjesus@localhost tempMax]$ hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.0.jar -Dmap.output.key.field.separator="\t" -Dstream.num.map.output.key.fields=2 -files ./mapperAnyoMes.py,./reducerMaxTempYearMonth.py -mapper ./mapperAnyoMes.py -combiner ./reducerMaxTempYearMonth.py -reducer ./reducerMaxTempYearMonth.py -input medidas2.txt -output ./salidaMaxTempAnyoMes4
```

Map-Reduce Framework

```
Map input records=12
Map output records=12
Map output bytes=165
Map output materialized bytes=96
Input split bytes=102
Combine input records=12
Combine output records=5
Reduce input groups=3
Reduce shuffle bytes=96
Reduce input records=5
Reduce output records=5
Spilled Records=10
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=48
Total committed heap usage (bytes)=331489280
```

38. Observamos la salida:

part-00000		
1999	Enero	20.0
1999	Febrero	4.0
2000	Enero	6.0
2000	Febrero	6.0
2001	Abril	3.0