# EJERCICIOS

### EJERCICIO 1

Utilizando una de las tareas de ejemplo de MapReduce (sudoku) resuelve el siguiente sudoku:

puzzle1.dta

8 5 ? 3 9 ? ? ? ?

? ? 2 ? ? ? ? ? ?

? ? 6 ? 1 ? ? ? 2

? ? 4 ? ? 3 ? 5 9

? ? 8 9 ? 1 4 ? ?

3 2 ? 4 ? ? 8 ? ?

9 ? ? ? 8 ? 5 ? ?

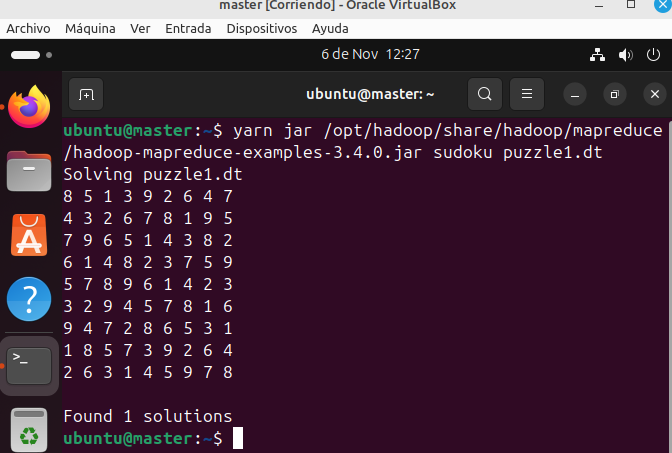
? ? ? ? ? ? 2 ? ?

? ? ? ? 4 5 ? 7 8

Primero muéstralo por pantalla y después vuélvelo a ejecutar, pero esta vez guardando la solución directamente en el fichero local de salida puzzle1resuelto.dta (recuerda las redirecciones en Linux).

Para acabar sube el fichero inicial y la solución a hadoop.

yarn jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.4.0.jar sudoku puzzle1.dt



yarn jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.4.0.jar sudoku puzzle1.dt >> puzzle1resulto.dt



hdfs dfs -put puzzle1.dt

hdfs dfs -put puzzle1resulto.dt



<https://learn.microsoft.com/es-es/azure/hdinsight/hadoop/apache-hadoop-run-samples-linux>

### EJERCICIO 2 (opcional- sólo si te sobra el tiempo)

Monta un clúster Hadoop sobre centOS sin entorno gráfico (instalación mínima con herramientas de desarrollo y herramientas de administración del sistema) usando 3 máquinas: 1 master y 2 trabajadores.

### EJERCICIO 3 (opcional sólo si has hecho el ejercicio 2)

Accede a las aplicaciones gráficas de hdfs y yarn del clúster anterior desde otro equipo que tenga entorno gráfico

### EJERCICIO 4

En el clúster anterior vamos a lanzar el programa de wordcount pero en Python.

* Crea el fichero pymap.py:

*#!/usr/bin/python3*

*import sys*

*for line in sys.stdin:*

*line = line.strip()*

*words = line.split()*

*for word in words:*

*print("{}\t1".format(word))*

* Crea el fichero pyreduce.py

*#!/usr/bin/python3*

*from operator import itemgetter*

*import sys*

*lastword = None*

*lastcount = 0*

*curword = None*

*i=1*

*for line in sys.stdin:*

*line = line.strip()*

*curword, count = line.split('\t', 1)*

*count = int(count)*

*#print("\*\*",curword,str(count),"\*\*")*

*if lastword == curword:*

*lastcount += count*

*else:*

*if lastword:*

*print("{}\t{}".format(lastword, lastcount))*

*lastcount = count*

*lastword = curword*

*if lastword == curword:*

*print("{}\t{}".format(lastword, lastcount))*

* Descarga un libro del [proyecto Gutenberg](https://www.gutenberg.org/) y súbelo a hadoop al directorio /practicas

*wget -O alice.txt https://www.gutenberg.org/files/11/11-0.txt*

*hdfs dfs -mkdir practicas*

*hdfs dfs -put frankenstein.txt practicas*

* Lanza el proceso y guarda la salida en el directorio /ejercicio4

*hadoop jar /opt/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar -files pymap.py,pyreduce.py -mapper pymap.py -reducer pyreduce.py -input ruta\_libroXX.txt -output ruta\_salida*

*hadoop jar /opt/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.4.0.jar -files pymap.py,pyreduce.py -mapper pymap.py -reducer pyreduce.py -input practicas/frankenstein.txt -output ejercicios/ejercicio4/*

* Visualiza la salida

*hdfs dfs -cat ejercicios/ejercicio4/part-00000*

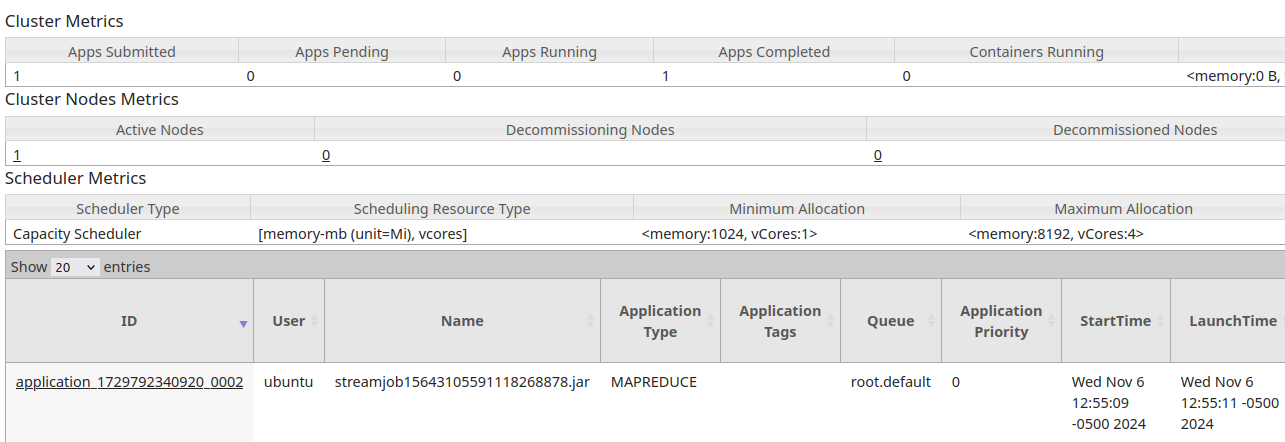
* Descarga a local el fichero de salida

*hdfs dfs -get ejercicios/ejercicio4/part-00000 conteodepalabrasfrank.txt*

* Muestra la salida ordenada de mayor a menor y de forma paginada según las palabras más frecuentes

*sort -k2,2nr -k1,1 conteodepalabrasfrank.txt | less*

* Desde la aplicación web mira cómo ha ido la tarea



### EJERCICIO 5

Repite el ejercicio anterior pero ahora crea tus propias funciones map y reduce para contar el número de veces que aparece cada letra, pero ahora guarda la salida en /ejercicio5.

*pymapword.py*

*#!/usr/bin/python3*

*import sys*

*for line in sys.stdin:*

*line = line.strip()*

*for char in line:*

*print("{}\t1".format(char))*

*pyreduceword.py*

*!/usr/bin/python3*

*from operator import itemgetter*

*import sys*

*lastchar= None*

*lastcount = 0*

*curchar = None*

*i=1*

*for line in sys.stdin:*

*curchar, count = line.split('\t', 1)*

*count = int(count)*

*#print("\*\*",curword,str(count),"\*\*")*

*if lastchar == curchar:*

*lastcount += count*

*else:*

*if lastchar:*

*print("{}\t{}".format(lastchar, lastcount))*

*lastcount = count*

*lastchar = curchar*

*if lastchar == curchar:*

*print("{}\t{}".format(lastchar, lastcount))*

*chmod a+x pymapword.py*

*chmod a+x pyreduceword.py*

*hadoop jar /opt/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.4.0.jar -files pymapword.py,pyreduceword.py -mapper pymapword.py -reducer pyreduceword.py -input practicas/frankenstein.txt -output ejercicios/ejercicio5*

### EJERCICIO 6

Descarga los datos en formato csv correspondientes al año 2018 de la siguiente página [retrasos en los vuelos](https://www.kaggle.com/datasets/yuanyuwendymu/airline-delay-and-cancellation-data-2009-2018) y desde Python:

* Carga la información del fichero en un dataframe de pandas

*import pandas as pd*

*df\_vuelos = pd.read\_csv('/home/ubuntu/Documentos/2018.csv', nrows=10000)*

* Muestra las primeras líneas del dataframe

*df\_vuelos.head(5)*

* Guarda la información en un fichero parquet

*df\_vuelos.to\_parquet('/home/ubuntu/Documentos/vuelos.parquet')*

* Carga la información del fichero parquet en un dataframe de pandas

*df\_vuelosparquet=pd.read\_parquet('/home/ubuntu/Documentos/vuelos.parquet')*

* Muestra las primeras líneas del dataframe del punto anterior

*df\_vuelosparquet.head()*

* Muestra el tamaño del archivo parquet y el tiempo que se ha tardado en crear el archivo

*import pandas as pd*

*import os*

*import time*

*df\_vuelos = pd.read\_csv('/home/ubuntu/Documentos/2018.csv', nrows=1000)*

*df\_vuelos.head(5)*

*# Definir la ruta donde guardar el archivo Parquet*

*parquet\_file\_path = '/home/ubuntu/Documentos/vuelos.parquet'*

*# Medir el tiempo de creación del archivo*

*start\_time = time.time()*

*# Guardar el DataFrame como archivo Parquet*

*df\_vuelos.to\_parquet(parquet\_file\_path)*

*# Medir el tiempo de ejecución*

*end\_time = time.time()*

*elapsed\_time = end\_time - start\_time*

*# Obtener el tamaño del archivo Parquet*

*file\_size = os.path.getsize(parquet\_file\_path) # Tamaño en bytes*

*# Mostrar los resultados*

*print(f"El archivo Parquet se ha creado en {elapsed\_time:.2f} segundos.")*

*print(f"El tamaño del archivo Parquet es: {file\_size / (1024 \* 1024):.2f} MB") #Convertido a MB*