UNIVERSIDAD DE SANTIAGO DE COMPOSTELA

Técnologías de computación para datos masivos

Autor:
Luis Ardévol Mesa

Profesor:
Tomás Fernández Pena

Escola Técnica Superior de Enxeñaría
Master en Tecnoloxías de Análise de Datos Masivos: Big Data

November 14, 2024

Contents

1 Big Data y Map Reduce

las claves que van a un reducer se giardan en un sitio y las claves que van a otro reducer se guardan en otro sitio.. los splits tienen un numero entero de lineas

VER LOS PROCESOS DEL MASTER Y DE LOS DEMONIOS

2 Introducción a Hadoop

i. Nada

ii. Libreria que permite programar las tareas: MapReduce.

iii. Esta seria la idea de ocmo se hacen las aplicaciones e nhadoop. Map Reduce se usa por encima del almacenamiento deistribuidos. Pig y Hive (version de SQL) son lenguajes de alto nivel que se traducen a tareas MapReduce. Hay otros tipos de aplicaciones como SParK que no usan MapReduce, sino se ejecutan directamente sobre YARN. La base de datos no relacional HBase se ejecuta directamente sobre HDFS y hay otros proyectos como Mahout para machine learning, Flume para ingestar datos, etc.

iv. La parte de instalacion la haremos en practica. Se puede probar en tu ordenador local con un unico nodo o de un modo pseudodistribuido.

v. La unica dificultad de la instalacion es configurarlo. Los 4 mas importantes son esos que vemos

vi. Este es de ejemplo

vii. Veamos las dos partes que tiene. La primera HDFS. No trabaja tan bien con ficheros pequeños. Latencia - cuanto tardamos en empezar a leerlo, ancho de banda - lo que tardamos en leerlo. Las modificaciones siempre van al final. Cada reducer debe tener su propio fuchero ya que no pueden escribir dos a la vez en uno.

viii. en el master se ejecuta un demonio que se llama namenode, que es el que manteiene la info. En los esclavos se ejecuta un demonio que se llama datanode. El namenode es el que sabe donde estan los bloques de los ficheros. El datanode es el que tiene los bloques de los ficheros, pero no tienen idea sobre los ficheros.

ix. el backup cada cierto tiempo realiza un checkpoint por si en namenode falla.

x. el RM en el master y el NM en los esclavos. El AM se introduce mas adelante ya que cuando aumentaba mucho el numero de tareas la carga del RM era demasiado alta. USa figura de 17 y salta a esa.

xi. (vuelve aqui desde la 17) y sigue. El RM se divide en dos, el scheduler y el applications manager AsM. Le da un poco igual

xii. no mucho

xiii. no mucho, ya lo dijo todo en la 17. YARN es un poco más complicado

xiv. El RM inicializa una tarea y justo asigna un AM. La aplicacion app mstr lanza los contenedores pidiendo al RM donde puede lanzar X tareas map y esas tareas, mientras se ejecutan, van hablandole al app master, que vuelve a hablar al RM para pedir mas espacio. Por lo que el RM se encarga de inicializar tareas y conceder recursos.

xv. blabla

xvi. blabla

xvii. comenta con el blabla

xviii. el app master pide contenedores con x recursos.

xix. Hadoop streaming supone una perdida de rendimeinto muy muy grande

3 Cluster Hadoop

```
ssh -i "hadoop.pem" ubuntu@DNS_publico_NNRM
ssh -i "hadoop.pem" -fNT -L 9870:localhost:9870 -L 8088:
   localhost:8088 ubuntu@DNS_publico_NNRM
ssh -i "hadoop.pem" -fNT -L 8188:localhost:8188
   ubuntu@DNS_publico_BKTL
yarn --daemon start timelineserver #en BKTL
```

echo "ip-172-31-13-193.ec2.internal" » /opt/bd/hadoop/etc/hadoop/dfs.include echo "ip-172-31-10-249.ec2.internal" » /opt/bd/hadoop/etc/hadoop/dfs.include echo "ip-172-31-6-138.ec2.internal" » /opt/bd/hadoop/etc/hadoop/dfs.include echo "ip-172-31-1-58.ec2.internal" » /opt/bd/hadoop/etc/hadoop/dfs.include ESTE fuera

echo "ip-172-31-1-58.ec2.internal" » /opt/bd/hadoop/etc/hadoop/dfs.exclude echo "ip-172-31-1-58.ec2.internal" » /opt/bd/hadoop/etc/hadoop/yarn.exclude

Nuevos: echo "ip-172-31-4-16.ec2.internal" » /opt/bd/hadoop/etc/hadoop/dfs.include echo "ip-172-31-4-16.ec2.internal" » /opt/bd/hadoop/etc/hadoop/yarn.include echo "ip-172-31-0-239.ec2.internal" » /opt/bd/hadoop/etc/hadoop/dfs.include echo "ip-172-31-0-239.ec2.internal" » /opt/bd/hadoop/etc/hadoop/yarn.include

172.31.13.193 /rack1 172.31.10.249 /rack1 172.31.6.138 /rack2 172.31.0.239 /rack3 172.31.4.16 /rack3

TAREA 2: Cuestion 1: Caben 3: estamos indicando que solo se pueden crear hasta 4 entradas de tipo archivo o directorio en esa ubicación (INCLUYENDO EL PROPIO DIRECTORIO). Al alcanzar la cuota de 4 archivos, el NameNode interrumpe cualquier operación que genere nuevas entradas en ese directorio. Equilibra el uso de recursos del sistema.

Cuestion 2:

193 - 32 bloques (rack1); 16 - 8 bloques (rack3).

Aparecen 35 bloques under-replicated, 2 perdidos y 1 archivo corrupto. Bloques perdidos 14. BP-1359368873-172.31.14.95-1729179225409:$\mathrm{blk}_1073741854_1030 len = 67108864 MISSING!15. BP - 1359368873 - 172.31.14.95 - 1729179225409 : blk_1073741855_1031 len = 41943040 MISSING!Sihaybloquesperdidosnosepuederecuperarel f$

Nuevo datanode: los tres tienen 35 bloques

CUestion 3: espacio utilizado menor por eficiencia de EC.

Los datos se dividen en bloques de datos y distribuye entre los datanodes. Da tolerancia a fallos y usa menos espacio.