Pasos instalación YARN con Hadoop

El nodo maestro mantiene el conocimiento sobre el sistema de archivos distribuido, como una tabla inode en un ext4 programa la asignación de recursos. El nodo maestro se encargará de esta función en esta guía, y albergará dos demonios:

- El **NameNode** gestiona el sistema de archivos distribuido y sabe dónde están los bloques de datos almacenados dentro del cluster (hdfs --daemon start namenode)
- El **ResourceManager** gestiona los trabajos de YARN y se encarga de programar y ejecutar procesos en los nodos de trabajadores.

Los nodos de trabajo almacenan los datos reales y proporcionan potencia de procesamiento para ejecutar los trabajos. Albergarán dos demonios:

- El **DataNode** gestiona los datos físicos almacenados en el nodo NameNode.
- El **NodeManager** gestiona la ejecución de tareas en el nodo.

Configuración de yarn y mapreduce Hadoop:

En todos los nodos configuramos yarn-site.xml

- yarn.resourcemanager.hostname

El hostname del resource manager.

- yarn.nodemanager.hostname

El hostname de los nodemanager

- yarn.nodemanager.aux-services

Una lista de servicios separados por comas en la que el nombre del servicio solo debe contener a-zA-Z0-9_ y no puede comenzar con números:

map-reduce_shuffle: Realiza todas las funciones entre la parte de map y la parte reduce.

En monitor de recursos de ubuntu, podemos ver tamaño máximo y minimo de cada contenedor y el número de subprocesos utilizados para manejar las solicitudes del administrador de aplicaciones.

Configuramos mapred-site.xml

- mapreduce.framework.name

El marco de tiempo de ejecución para ejecutar trabajos de MapReduce. Puede ser local, classic o yarn.

- mapreduce.application.classpath

CLASSPATH para aplicaciones mapreduce. Una lista separada por comas de entradas CLASSPATH.

Si se establece mapreduce.application.framework, debe especificar la classpath adecuada para ese archivo, y el nombre del archivo debe estar presente en la classpath.

Si mapreduce.app-submission.cross-platform es false, se usaría la sintaxis de expansión variable del entorno específico de la plataforma para construir las entradas CLASSPATH predeterminadas.

Para Linux:

$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*, $HADOOP_MAPRED_HOME/share/hadoop/mapreduce/lib/*.

Para Windows:

%HADOOP_MAPRED_HOME%/share/hadoop/mapreduce/*, %HADOOP_MAPRED_HOME%/share/hadoop/mapreduce/lib/*.

Si mapreduce.app-submission.cross-platform es true, se usaría CLASSPATH predeterminado independiente de la plataforma para aplicaciones mapreduce:

{{HADOOP_MAPRED_HOME}}/share/hadoop/mapreduce/*, {{HADOOP_MAPRED_HOME}}/share/hadoop/mapreduce/lib/*

El marcador de expansión de parámetros será reemplazado por NodeManager en el inicio del contenedor en función del sistema operativo subyacente en consecuencia.

Nosotros lo configuraremos como con las rutas de estas 2 carpetas:

…/share/hadoop/mapreduce/*, …/share/hadoop/yarn/lib/*

Ejecutamos el demonio: yarn –daemon start resourcemanager (mejor usar el script start-yarn.sh si tenemos workers configurado)

Ejercicio

Creamos fichero datos.txt y lo subimos a hdfs:

Uno

Uno dos

Uno dos tres

Uno dos tres cuatro

Localizamos ejemplo jar de hadoop: hadoop-mapreduce-example.jar:

yarn jar /(…ruta)/ hadoop-mapreduce-example.jar

vemos todas las clases que incluye

yarn jar /(…ruta)/ hadoop-mapreduce-example.jar wordcount datos.txt /(ruta)/salida

Nota: si ruta de salida ya existe da error

Podemos ver como distribuye la información, entrada, salida, y temporal donde realiza las funciones mapper, shuflle y reduce. En la salida creara un archivo SUCCESS que indica que se realizó con éxito y tantos archivos cmo reduce hay realizado, si es pequeño, solo uno.

Desde terminal con hdfs dfs -cat ../salida/*

Podemos ver como trabaja abriendo el monitor de recursos de Ubuntu.

Como práctica, probar otras aplicaciones del archivo ejemplos, por ejemplo pi vale para indicar cuantos mappers y reduces se quieren usar.

Por ejemplo descargar quijote.txt

Como funciona la aplicación contador, video 2 a partir minuto 6

Las propiedades de la asignación de memoria

Un trabajo de YARN se ejecuta con dos tipos de recursos:

- Un maestro de *aplicaciones* (AM) se encarga de supervisar la aplicación y de coordinar a los ejecutores distribuidos en el cluster.
- Algunos ejecutores que son creados por la AM en realidad ejecutan el trabajo. Para los trabajos de MapReduce, realizarán la operación de mapeo o reducción, en paralelo.

Ambos se ejecutan en *contenedores* en los nodos de los trabajadores. Cada nodo trabajador ejecuta un demonio *NodeManager* que es responsable de la creación de contenedores en el nodo. Todo el cluster es gestionado por un *ResourceManager* que programa la asignación de contenedores en todos los nodos de trabajadores, dependiendo de las necesidades de capacidad y de la carga actual.

Es necesario configurar adecuadamente cuatro tipos de asignaciones de recursos para que el grupo funcione. Estos son:

1. Cuánta memoria se puede asignar a los contenedores YARN en un solo nodo. Este límite debe ser más alto que todos los demás; de lo contrario, la asignación de

contenedores será rechazada y las solicitudes fracasarán. Sin embargo, no debería ser la cantidad total de RAM del nodo.

Este valor se configura en yarn-site.xml en yarn.nodemanager.resource.memory-mb.

2. Cuánta memoria puede consumir un solo contenedor y la asignación de memoria mínima permitida. Un contenedor nunca será más grande que el máximo, o bien la asignación fallará y siempre será asignada como un múltiplo de la cantidad mínima de RAM.

Estos valores se configuran en yarn-site.xml en yarn.scheduler.maximum-allocation-mb y yarn.scheduler.minimum-allocation-mb.

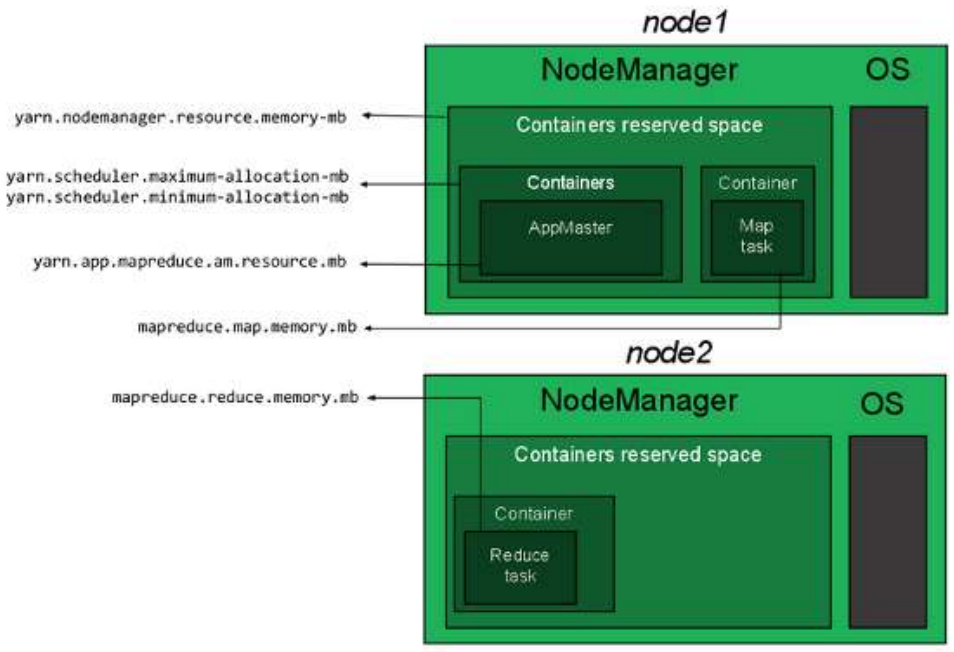
3. Cuánta memoria se asignará al ApplicationMaster. Este es un valor constante que debe caber en el tamaño máximo del contenedor.

Esto se configura en mapred-site.xml en yarn.app.mapreduce.am.resource.mb.

4. Cuánta memoria se asignará a cada mapa o reducir la operación. Este debe ser menor que el tamaño máximo.

Esto se configura en mapred-site.xml en mapreduce.map.memory.mb y mapreduce.reduce.memory.mb.

La relación entre todas estas propiedades se puede ver en la siguiente figura:



Ejemplo de configuración para nodos de 2GB

Para los nodos de 2GB, puede ser una configuración de trabajo:

yarn.nodemanager.resource.memory-mb	1536
yarn.scheduler.maximum-allocation-mb	1536
yarn.scheduler.minimum-allocation-mb	128
yarn.app.mapreduce.am.resource.mb	512
mapreduce.map.memory.mb	256
mapreduce.reduce.memory.mb	256

1. Edita yarn-site.xml y añadir las siguientes líneas:

```
<property>
    <name>yarn.nodemanager.resource.memory-mb</name>
    <value>1536</value>
</property>

<property>
    <name>yarn.scheduler.maximum-allocation-mb</name>
    <value>1536</value>
</property>

<property>
    <name>yarn.scheduler.minimum-allocation-mb</name>
    <value>128</value>
```

```
</property>

<property>
      <name>yarn.nodemanager.vmem-check-enabled</name>
      <value>false</value>
</property>
```

La última propiedad desactiva la comprobación de la memoria virtual, lo que puede impedir que los contenedores se asignen correctamente con JDK8 si está activada.

2. Edita mapred-site.xml y añadir las siguientes líneas:

```
<property>
      <name>yarn.app.mapreduce.am.resource.mb</name>
      <value>512</value>
</property>

<property>
      <name>mapreduce.map.memory.mb</name>
      <value>256</value>
</property>

<property>
      <name>mapreduce.reduce.memory.mb</name>
      <value>256</value>
</property>
```

Monitor YARN

Los símbolos yarn proporciona utilidades para gestionar su clúster YARN. También puede imprimir un informe de los nodos en ejecución con el comando:

```
1. yarn node -list
```

Del mismo modo, puede obtener una lista de aplicaciones en ejecución con el comando:

```
yarn application -list
```

Para obtener todos los parámetros disponibles del yarn comando, ver