https://www.linode.com/docs/guides/install-configure-run-spark-on-top-of-hadoop-yarn-cluster/

Spark es un sistema de computación cluster de propósito general. Puede desplegar y ejecutar aplicaciones paralelas en clusters que van desde un solo nodo hasta miles de nodos distribuidos. Spark fue diseñado originalmente para ejecutar aplicaciones de Scala, pero también soporta Java, Python y R.

Spark puede funcionar como un gestor de clústeres independiente o aprovechando marcos de gestión de clústeres dedicados como Apache Hadoop YARN o Apache Mesos.

Antes de comenzar

1. El nodo maestro en este caso (HDFS NameNode y YARN ResourceManager) se llama **node-master** y los nodos esclavos (HDFS DataNode y YARN NodeManager) se llaman **node1** y **node2**.

Descargar e instalar binarios de spark

Los binarios de Spark están disponibles en la página de descargas de Apache Spark. Ajuste cada comando a continuación para que coincida con el número de versión correcto.

1. Obtenga la URL de descarga de la página de descarga de Spark, descárguela y descomprímala.

Para Spark 2.2.0 con Hadoop 2.7 o posterior, inicie sesión node-master como el hadoop usuario, y correr:

```
cd /home/hadoop
wget https://d3kbcqa49mib13.cloudfront.net/spark-2.2.0-bin-hadoop2.7.tgz
tar -xvf spark-2.2.0-bin-hadoop2.7.tgz
mv spark-2.2.0-bin-hadoop2.7 spark
```

2. Añade el directorio de binarios de Spark a tu PATH. Edita /home/hadoop/.profile y añadir la siguiente línea:

Para sistemas Debian/Ubuntu:

File: /home/hadoop/.profile

```
PATH=/home/hadoop/spark/bin:$PATH
```

Integrar la spark con el hilo

Para comunicarse con el YARN Resource Manager, Spark necesita conocer su configuración de Hadoop. Esto se hace a través de la HADOOP_CONF_DIR variable de entorno. El SPARK_HOME no es obligatoria, pero es útil cuando se envían trabajos de Spark desde la línea de comandos.

1. Editar el *hadoop* perfil del usuario /home/hadoop/.profile y añadir las siguientes líneas:

File: /home/hadoop/.profile

```
export HADOOP_CONF_DIR=/home/hadoop/hadoop/etc/hadoop
```

```
export SPARK_HOME=/home/hadoop/spark
export
LD_LIBRARY_PATH=/home/hadoop/hadoop/lib/native:$LD_LIBRARY_PATH
```

2. Reinicie su sesión cerrando la sesión y volviéndola a abrir.

3. Cambie el nombre del archivo de configuración de la plantilla por defecto de la spark:

```
mv $SPARK_HOME/conf/spark-defaults.conf.template
$SPARK_HOME/conf/spark-defaults.conf
```

4. Edita $SPARK_HOME/conf/spark-defaults.conf y el conjunto spark.master para yarn:

File: $SPARK_HOME/conf/spark-defaults.conf

```
spark.master    yarn
```

Spark está ahora listo para interactuar con tu grupo de YARN.

Entender el modo de cliente y de clúster

Las tareas de spark pueden ejecutarse en YARN en dos modos: el modo de *clúster* y el modo de *cliente*. Es importante comprender la diferencia entre los dos modos para elegir una configuración apropiada de asignación de memoria y para enviar los trabajos como se espera.

Un trabajo de spark consiste en dos partes: Ejecutores de spark que ejecutan las tareas reales y un conductor de spark que programa los ejecutores.

- Modo **cluster**: todo funciona dentro del cluster. Puede iniciar un trabajo desde su portátil y el trabajo seguirá ejecutándose, aunque cierre el ordenador. En este modo, el Spark Driver está encapsulado dentro del YARN Application Master.
- Modo de **cliente** el driver de la spark se ejecuta en un cliente, como su portátil. Si se cierra el cliente, el trabajo falla. Los Spark Executors todavía se ejecutan en el cluster, y para programar todo, se crea un pequeño YARN Application Master.

El modo cliente es muy adecuado para trabajos interactivos, pero las aplicaciones fallarán si el cliente se detiene. Para trabajos de larga duración, el modo de clúster es más apropiado.

Configurar la asignación de memoria

La asignación de los contenedores de spark que se ejecutan en los contenedores YARN puede fallar si la asignación de memoria no está configurada correctamente. Para los nodos con menos de 4G de RAM, la configuración por defecto no es adecuada y puede desencadenar el swapping y un rendimiento pobre, o incluso el fallo de la inicialización de la aplicación debido a la falta de memoria.

Asegúrese de entender cómo Hadoop YARN maneja la asignación de memoria antes de editar los ajustes de la memoria de spark para que sus cambios sean compatibles con los límites de su clúster YARN.

Dé a sus contenedores de YARN la máxima memoria permitida

Si la memoria solicitada está por encima del máximo permitido, YARN rechazará la creación del contenedor, y su aplicación Spark no se iniciará.

1. Obtener el valor de yarn.scheduler.maximum-allocation-mb en $HADOOP_CONF_DIR/yarn-site.xml. Este es el valor máximo permitido, en MB, para un solo contenedor.

2. Asegúrese de que los valores para la asignación de la memoria de spark, configurados en la siguiente sección, estén por debajo del máximo.

Esta guía utilizará un valor de muestra de 1536 para yarn.scheduler.maximum-allocation-mb. Si su configuración es más baja, ajuste las muestras con su configuración.

Configurar la asignación de memoria del Spark Driver en el modo de clúster

En el modo de clúster, el Spark Driver funciona dentro del YARN Application Master. La cantidad de memoria solicitada por la Spark en la inicialización se configura en spark-defaults.confo a través de la línea de comandos.

Desde spark-defaults.conf

- Configure la cantidad de memoria por defecto asignada al Spark Driver en el modo de clúster mediante spark.driver.memory (este valor es por defecto 1G). Para ponerlo en 512MBEditar el archivo:

File: $SPARK_HOME/conf/spark-defaults.conf

```
spark.driver.memory    512m
```

Desde la Línea de Comando

- Usa la opción --driver-memory para especificar la cantidad de memoria solicitada por spark-submit. Vea la siguiente sección sobre la presentación de solicitudes para ver ejemplos.

Nota: Los valores dados desde la línea de comandos anularán lo que se haya establecido en spark-defaults.conf.

Configurar la asignación de memoria maestra de la aplicación de la spark en el modo de cliente

En el modo de cliente, el controlador de la spark no se ejecutará en el clúster, por lo que la configuración anterior no tendrá ningún efecto. Aún es necesario crear un maestro de aplicación de YARN para programar el ejecutor de la spark, y puedes establecer sus requerimientos de memoria.

Configure la cantidad de memoria asignada al Maestro de aplicaciones en el modo de cliente con spark.yarn.am.memory (por defecto a 512M)
File: $SPARK_HOME/conf/spark-defaults.conf

```
spark.yarn.am.memory    512m
```

Este valor no se puede establecer desde la línea de comandos.

Configurar la asignación de memoria de los ejecutores Spark

La asignación de memoria de los ejecutores Spark se calcula en función de dos parámetros internos $SPARK_HOME/conf/spark-defaults.conf:

- spark.executor.memory: establece la memoria de base utilizada en el cálculo
- spark.yarn.executor.memoryOverheadse añade a la memoria base. Por defecto es el 7% de la memoria base, con un mínimo de 384MB

Nota
Asegúrese de que la memoria solicitada por el Ejecutor, **incluyendo** la memoria de sobrecarga, esté por debajo del tamaño máximo del contenedor YARN, de lo contrario la aplicación Spark no se inicializará.

Ejemplo:
para spark.executor.memory de 1Gb, la memoria requerida es de 1024+384=1408MB. Para 512MB, la memoria requerida será de 512+384=896MB
Para establecer la memoria de ejecución en 512MB, editar $SPARK_HOME/conf/spark-defaults.conf y añadir la siguiente línea:
File: $SPARK_HOME/conf/spark-defaults.conf

```
spark.executor.memory        512m
```

Cómo presentar una solicitud de spark al Cluster de YARN

Las solicitudes se presentan con el spark-submit comando. El paquete de instalación de la spark contiene aplicaciones de ejemplo, como el cálculo paralelo de *Pi*...que puedes correr para practicar el inicio de los trabajos de la spark.

Para ejecutar el cálculo de *Pi* de muestra, utilice el siguiente comando:

```
spark-submit --deploy-mode client \
        --class org.apache.spark.examples.SparkPi \
        $SPARK_HOME/examples/jars/spark-examples_2.11-2.2.0.jar 10
```

El primer parámetro, --deploy-modeLa opción "Modo de transporte", especifica el modo que se debe utilizar, client o cluster.
Para ejecutar la misma aplicación en modo de clúster, sustituya --deploy-mode clientcon --deploy-mode cluster.

Monitoree sus aplicaciones de spark

Cuando envías un trabajo, Spark Driver inicia automáticamente una interfaz de usuario web en el puerto 4040 que muestra información sobre la aplicación. Sin embargo, cuando la ejecución termina, la interfaz de usuario Web se descarta con el controlador de la aplicación y ya no se puede acceder a ella.

Spark proporciona un Servidor de Historial que recopila los registros de aplicación del HDFS y los muestra en una interfaz de usuario web persistente. Los siguientes pasos permitirán la persistencia del registro en el HDFS:

1. Edita $SPARK_HOME/conf/spark-defaults.conf y agregar las siguientes líneas para permitir que las tareas de spark se registren en el HDFS:

   File: $SPARK_HOME/conf/spark-defaults.conf

   ```
   spark.eventLog.enabled  true
   spark.eventLog.dir hdfs://node-master:9000/spark-logs
   ```

2. Crear el directorio de registro en HDFS:

   ```
   hdfs dfs -mkdir /spark-logs
   ```

3. Configurar las propiedades relacionadas con el Servidor de historiales en $SPARK_HOME/conf/spark-defaults.conf:

   File: $SPARK_HOME/conf/spark-defaults.conf

   ```
   spark.history.provider            org.apache.spark.deploy.history.FsHistoryProvider
   spark.history.fs.logDirectory     hdfs://node-master:9000/spark-logs
   spark.history.fs.update.interval  10s
   spark.history.ui.port             18080
   ```

   Es posible que desee utilizar un intervalo de actualización diferente al predeterminado 10s. Si especifica un intervalo mayor, tendrá un cierto retraso entre lo que ve en el Servidor de Historial y el estado en tiempo real de su aplicación. Si utiliza un intervalo más corto, aumentará las E/S en el HDFS.

4. Ejecute el Servidor del Historial:

   ```
   $SPARK_HOME/sbin/start-history-server.sh
   ```

5. Repita los pasos de la sección anterior para iniciar un trabajo con spark-submit que generará algunos registros en el HDFS:
6. Acceda al Servidor de Historial navegando a http://node-master:18080 en un navegador web:

Ejecutar el Spark Shell

El Spark shell ofrece una forma interactiva de examinar y trabajar con sus datos.

1. Ponga algunos datos en el HDFS para su análisis. Este ejemplo utiliza el texto de *Alicia en* el País de las Maravillas del proyecto Gutenberg:

```
cd /home/hadoop
wget -O alice.txt https://www.gutenberg.org/files/11/11-0.txt
hdfs dfs -mkdir inputs
hdfs dfs -put alice.txt inputs
```

2. Inicie el proyecto de Spark:

```
spark-shell

var input = spark.read.textFile("inputs/alice.txt")
// Count the number of non blank lines
input.filter(line => line.length()>0).count()
```

Puedes encontrar la documentación oficial en Official Apache Spark documentation.