CURSO DE SPARK – DISTRIBUIÇÃO E PROCESSAMENTO DE DADOS

PROF. CARLOS M. D. VIEGAS

Roteiro de implantação do Apache Hadoop 3.x (standalone)

Requisitos

- Distribuição GNU/Linux;
- Máquina Java (versão 11);
- Aplicação SSH (cliente e servidor).

O Hadoop permite que os processos sejam executados em três modos:

- Instância única (*standalone*), no qual apenas um nó executa as tarefas, sem que seja necessário fazer qualquer configuração adicional (padrão). Este modo é útil para o propósito de *debugging*.
- Pseudo-distribuído (*pseudo-distributed*), no qual um único nó pode executar múltiplos processos Java de maneira a simular o processamento em paralelo. Este modo é útil para quando não se dispõe de um cluster com múltiplos nós, mas deseja-se realizar o processamento de dados com paralelismo;
- Totalmente distribuído (*fully-distributed*), no qual é criado um cluster com vários nós que irão armazenar e processar as informações de maneira paralela, com conexão em rede.

Este roteiro apresentará os modos de execução em instância única e pseudo-distribuído.

Instalação e Configuração

1. No sistema GNU/Linux, baixar o Apache Hadoop na versão 3.3.4 a partir do seguinte endereço: https://hadoop.apache.org/releases.html

2. Extrair o arquivo por meio do comando abaixo em um emulador de terminal:

```
tar -xzvf hadoop-3.3.4.tar.gz -C ~
```

3. Configurar o ambiente Hadoop para que reconheça o local de instalação da máquina Java:

   Arquivo: **~/hadoop-3.3.4/etc/hadoop/hadoop-env.sh**

   (linha 54)

```
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64/
```

   (o local da instalação da máquina Java pode variar conforme o sistema operacional e a versão Java)

4. Para verificar se o Hadoop pode ser executado, digite em um emulador de terminal:

```
~/hadoop-3.3.4/bin/hadoop
```

Como resposta ao comando acima, deverão ser apresentadas informações adicionais sobre o comando (help):

```
Usage: hadoop [OPTIONS] SUBCOMMAND [SUBCOMMAND OPTIONS]
 or    hadoop [OPTIONS] CLASSNAME [CLASSNAME OPTIONS]
  where CLASSNAME is a user-provided Java class

  OPTIONS is none or any of:

buildpaths                       attempt to add class files from build tree
--config dir                     Hadoop config directory
--debug                          turn on shell script debug mode
--help                           usage information
hostnames list[,of,host,names]   hosts to use in slave mode
hosts filename                   list of hosts to use in slave mode
loglevel level                   set the log4j level for this command
workers                          turn on worker mode

  SUBCOMMAND is one of:


    Admin Commands:

daemonlog     get/set the log level for each daemon

    Client Commands:

archive       create a Hadoop archive
checknative   check native Hadoop and compression libraries availability
classpath     prints the class path needed to get the Hadoop jar and the required libraries
conftest      validate configuration XML files
credential    interact with credential providers
distch        distributed metadata changer
distcp        copy file or directories recursively
dtutil        operations related to delegation tokens
envvars       display computed Hadoop environment variables
fs            run a generic filesystem user client
gridmix       submit a mix of synthetic job, modeling a profiled from production load
jar <jar>     run a jar file. NOTE: please use "yarn jar" to launch YARN applications, not
this command.
jnipath       prints the java.library.path
kdiag         Diagnose Kerberos Problems
kerbname      show auth_to_local principal conversion
key           manage keys via the KeyProvider
rumenfolder   scale a rumen input trace
rumentrace    convert logs into a rumen trace
s3guard       manage metadata on S3
trace         view and modify Hadoop tracing settings
version       print the version

    Daemon Commands:

kms           run KMS, the Key Management Server
registrydns   run the registry DNS server

SUBCOMMAND may print help when invoked w/o parameters or with -h.
```

Se chegamos até este ponto sem problemas/erros, o Hadoop já está configurado para execução de aplicações MapReduce no modo standalone.

5. Para facilitar a execução dos comandos do Hadoop no sistema, sem ter que digitar sempre o local dos executáveis, podemos proceder com a configuração das variáveis de ambiente, conforme abaixo:

   Arquivo: **~/.bashrc**

   (ao final do arquivo acrescentar)

```
export HADOOP_HOME="/home/spark/hadoop-3.3.4"
export HADOOP_COMMON_HOME="${HADOOP_HOME}"
export HADOOP_HDFS_HOME="${HADOOP_HOME}"
export HADOOP_MAPRED_HOME="${HADOOP_HOME}"
export HADOOP_YARN_HOME="${HADOOP_HOME}"
export HADOOP_CONF_DIR="${HADOOP_HOME}/etc/hadoop"
export HADOOP_COMMON_LIB_NATIVE_DIR="${HADOOP_HOME}/lib/native"
export HADOOP_OPTS="$HADOOP_OPTS -XX:-PrintWarnings -
Djava.library.path=${HADOOP_COMMON_LIB_NATIVE_DIR}"
export LD_LIBRARY_PATH="${HADOOP_COMMON_LIB_NATIVE_DIR}"
export JAVA_HOME="/usr/lib/jvm/java-11-openjdk-amd64/"
export HDFS_NAMENODE_USER="spark"
export HDFS_DATANODE_USER="${HDFS_NAMENODE_USER}"
export HDFS_SECONDARYNAMENODE_USER="${HDFS_NAMENODE_USER}"
export YARN_RESOURCEMANAGER_USER="${HDFS_NAMENODE_USER}"
export YARN_NODEMANAGER_USER="${HDFS_NAMENODE_USER}"
export PATH="$PATH:${HADOOP_HOME}/sbin:${HADOOP_HOME}/bin:${JAVA_HOME}/bin"
```

   (o local da variável HADOOP_HOME pode variar conforme o nome do usuário do sistema)

6. É recomendado especificar algumas propriedades adicionais para que o HADOOP não apresente *warnings* sobre bibliotecas não nativas do JAVA:

   Arquivo: **~/hadoop-3.3.4/etc/hadoop/hadoop-env.sh**

   (linha 90)

```
export              HADOOP_OPTS="$HADOOP_OPTS              -XX:-PrintWarnings              -
Djava.library.path=${HADOOP_COMMON_LIB_NATIVE_DIR}"
```

Primeira execução

- Modo: Instância única

1. Vamos agora fazer um primeiro processamento utilizando o MapReduce em instância única. Como exemplo, vamos utilizar um arquivo de entrada (dataset) em formato `.json` e realizar a contagem de ocorrências de uma determinada palavra por meio de uma expressão regular.

   a. Primeiramente vamos criar uma pasta para guardar o dataset e em seguida baixar o arquivo `dataset-1.json` disponibilizado na plataforma Moodle do CEAJUD:

```
mkdir ~/hadoop-3.3.4/datasets
cp dataset-1.json ${HADOOP_HOME}/datasets
```

b. Em seguida, executamos a tarefa do MapReduce sobre o arquivo de entrada para realizar a contagem de ocorrências de uma determinada palavra. Neste exemplo, uma expressão regular foi utilizada para determinar um padrão desejado. Este comando faz a busca desse padrão, conta as ocorrências do mesmo e grava em um arquivo de saída na pasta `output`:

```
hadoop jar ${HADOOP_HOME}/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.4.jar grep datasets output 'doc[a-z.]+'
```

(É possível experimentar outros arquivos como entrada e também alterar a expressão regular de busca)

c. Para visualizar o resultado:

```
cat output/*
```

(Importante: Sempre que um novo processo for executado, a pasta output deve ser renomeada no comando ou removida do sistema)

- Modo: <u>Pseudo-distribuído</u>

1. Para realizar o processamento no modo pseudo-distribuído (localmente) com MapReduce são necessárias algumas configurações adicionais no Apache Hadoop.

a. Editar o arquivo `core-site.xml` e acrescentar:

`Arquivo:` **`~/hadoop-3.3.4/etc/hadoop/core-site.xml`**

```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value>
    </property>
</configuration>
```

b. Editar o arquivo `hdfs-site.xml` e acrescentar:

`Arquivo:` **`~/hadoop-3.3.4/etc/hadoop/hdfs-site.xml`**

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
```

```
</configuration>
```

c. Criar uma chave criptográfica para o servidor ssh permitir o login sem solicitar usuário e senha:

```
ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 0600 ~/.ssh/authorized_keys
```

d. Formatar o sistema de arquivos HDFS no NameNode:

```
hdfs namenode -format
```

e. Iniciar o daemon (processo) do HDFS

```
start-dfs.sh
```

f. Utilizar um navegador para analisar as estatísticas do NameNode:

http://localhost:9870/

g. Criar os diretórios de usuário no HDFS para executar as tarefas do MapReduce:

```
hdfs dfs -mkdir -p /user/<username>
```

(Recomenda-se utilizar o mesmo nome de usuário do sistema local)

Exemplo de alguns comandos úteis do HDFS:

Comando	**Descrição**
`hdfs dfs -ls`	Lista um diretório
`hdfs dfs -rm`	Remover um arquivo ou pasta do HDFS
`hdfs dfs -mkdir`	Cria um diretório
`hdfs dfs -copyFromLocal`	Copia arquivos do sistema local para o HDFS
`hdfs dfs -put`	Copia arquivos do sistema local para o HDFS
`hdfs dfs -copyToLocal`	Copia o arquivo do HDFS para o sistema local
`hdfs dfs -get`	Copia o arquivo do HDFS para o sistema local
`hdfs dfs -cat`	Lê de um arquivo no HDFS

Lista com a referência de todos os comandos `hdfs`:

https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HDFSCommands.html

h. Copiar arquivos de entrada para o HDFS. Considere utilizar o mesmo dataset anteriormente baixado.

```
hdfs dfs -mkdir input
hdfs dfs -put datasets/*.json input
```

i. Executar a contagem de ocorrências:

```
hadoop jar ${HADOOP_HOME}/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.4.jar grep input output 'doc[a-z.]+'
```

j. Examine os dados diretamente do HDFS:

```
hdfs dfs -cat output/*
```

k. [OPCIONAL] Para encerrar o daemon do HDFS:

```
stop-dfs.sh
```

Alguns exemplos de programas pré-existentes no arquivo `hadoop-mapreduce-examples-3.3.4.jar`:

Exemplo	**Descrição**
aggregatewordcount	Conta as palavras nos arquivos de entrada.
aggregatewordhist	Computa o histograma das palavras nos arquivos de entrada.
bbp	Usa Bailey-Borwein-Plouffe para computar os dígitos exatos de Pi.
dbcount	Conta os logs de pageview armazenados em um banco de dados.
distbbp	Usa uma fórmula do tipo BBP para computar os bits exatos de Pi.
grep	Conta as correspondências de uma regex na entrada.
join	Faz a união de conjuntos de dados classificados e particionados igualmente.
multifilewc	Conta palavras de vários arquivos.
pentomino	Um programa para organizar peças lado a lado visando a encontrar soluções para problemas de pentomino.
pi	Estima Pi usando um método semelhante ao de Monte Carlo.
randomtextwriter	Grava 10 GB de dados de texto aleatórios por nó.
randomwriter	Grava 10 GB de dados aleatórios por nó.
secondarysort	Define uma classificação secundária para a fase de redução.
sort	Classifica os dados gravados pelo gravador aleatório.
sudoku	Um solucionador de sudoku.
teragen	Gera dados para o terasort.
terasort	Executa o terasort.
teravalidate	Verificação dos resultados do terasort.
wordcount	Conta as palavras nos arquivos de entrada.
wordmean	Conta o tamanho médio das palavras nos arquivos de entrada.
wordmedian	Conta o tamanho mediano das palavras nos arquivos de entrada.
wordstandarddeviation	Conta o desvio padrão do tamanho das palavras nos arquivos de entrada.

Fonte: https://learn.microsoft.com/pt-br/azure/hdinsight/hadoop/apache-hadoop-run-samples-linux