UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO

**Roteiro de Aula Prática – Implantação Apache Hadoop**

DISCIPLINA: DCA0132 – ENGENHARIA DE DADOS

PROFESSOR: Carlos Manuel Dias Viegas

Esta prática consiste em uma introdução básica à implantação de uma solução Apache Hadoop. Será apresentado o passo a passo para a instalação, configuração e execução de tarefas. Entretanto, este roteiro serve apenas como guia para o entendimento das configurações necessárias para utilizar o Hadoop.

**ATENÇÃO**: Este roteiro é uma sugestão para a implantação do Apache Hadoop nativamente em uma máquina linux. Entretanto, nesta prática será utilizada uma solução baseada em containers Docker, na qual o Apache Hadoop já está devidamente configurado. Portanto, este roteiro não precisa ser seguido, bastando apenas fazer o download dos arquivos no repositório do github:

1. Os recursos Docker para a implantação do Apache Hadoop estão disponíveis em:

<https://github.com/cmdviegas/docker-hadoop-cluster>

1. Ao inicializar o cluster, proceda com os testes descritos nos passos 15 a 18.

**Requisitos**

* Distribuição Linux com máquina virtual Java (JDK versão 8).

**Instalação e Configuração**

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

1. Instância única (*standalone*), no qual apenas um nó executa as tarefas, sem que seja necessário fazer qualquer configuração adicional (padrão);
2. Pseudo-distribuído (*pseudo-distributed*), no qual um único nó pode executar múltiplos processos java de maneira a simular o processamento em paralelo;
3. Complemente 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.

As instruções contidas neste roteiro permitem a implantação do Hadoop no modo *fully-distributed*, criando um cluster formado por um nó mestre e 4 nós escravos.

Antes de iniciar o passo a passo, pressupõe-se que o Hadoop foi previamente baixado e extraído em alguma pasta.

1. No nó que será o mestre, deve-se configurar o *hostname* em que o *NameNode* será executado. Além disso, definimos o usuário padrão como sendo root para que seja possível utilizar a interface web com as devidas permissões. A porta 9000 é padrão do Hadoop.

Arquivo: **~/hadoop-3.4.0/etc/hadoop/core-site.xml**

|  |
| --- |
| <configuration>  <property>  <name>fs.default.name</name>  <value>hdfs://**node-master**:**9000**</value>  </property>  <property>  <name>hadoop.http.staticuser.user</name>  <value>**root**</value>  </property>  </configuration> |

Em seguida, configurar o HDFS para indicar os diretórios em que o *NameNode* e os *DataNodes* armazenarão os dados. Além disso, o parâmetro dfs.replication indica a quantidade de nós que terão os dados replicados. Este valor nunca deve exceder a quantidade de nós escravos.

Arquivo: **~/hadoop-3.4.0/etc/hadoop/hdfs-site.xml**

|  |
| --- |
| <configuration>  <property>  <name>dfs.namenode.name.dir</name>  <value>/root/data/nameNode</value>  </property>  <property>  <name>dfs.datanode.data.dir</name>  <value>/root/data/dataNode</value>  </property>  <property>  <name>dfs.replication</name>  <value>2</value>  </property>  </configuration> |

1. O Yarn será o escalonador de tarefas padrão, atuando no lugar do MapReduce. Para essa finalidade, é necessário configurá-lo e especificar a quantidade de memória RAM a ser alocada para cada tarefa. Além disso, é necessário definir as variáveis do ambiente do MapReduce para que as classes (em Java) possam ser localizadas.

Arquivo: **~/hadoop-3.4.0/etc/hadoop/mapred-site.xml**

|  |
| --- |
| <configuration>  <property>  <name>mapreduce.framework.name</name>  <value>yarn</value>  </property>  <property>  <name>mapreduce.application.classpath</name>  <value>$HADOOP\_MAPRED\_HOME/share/hadoop/mapreduce/\*, $HADOOP\_MAPRED\_HOME/share/hadoop/mapreduce/lib/\*</value>  </property>  <property>  <name>yarn.app.mapreduce.am.env</name>  <value>HADOOP\_MAPRED\_HOME=${HADOOP\_HOME}</value>  </property>  <property>  <name>mapreduce.map.env</name>  <value>HADOOP\_MAPRED\_HOME=${HADOOP\_HOME}</value>  </property>  <property>  <name>mapreduce.reduce.env</name>  <value>HADOOP\_MAPRED\_HOME=${HADOOP\_HOME}</value>  </property>  <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>  </configuration> |

Os três últimos parâmetros definem a quantidade de memória RAM alocada para cada tarefa. No total são 512MB de RAM, sendo 256MB para map()e 256MB para reducer().

1. Configurar os parâmetros do Yarn, tais como quantidade de memória alocada para o escalonador, qual nó irá executar o *ResourceManager* e quais as portas de serviço, variáveis ambiente, entre outros.

Arquivo: **~/hadoop-3.4.0/etc/hadoop/yarn-site.xml**

|  |
| --- |
| <configuration>  <property>  <name>yarn.resourcemanager.hostname</name>  <value>node-master</value>  </property>  <property>  <name>yarn.nodemanager.aux-services</name>  <value>mapreduce\_shuffle</value>  </property>  <property>  <name>yarn.nodemanager.resource.memory-mb</name>  <value>1024</value>  </property>  <property>  <name>yarn.scheduler.maximum-allocation-mb</name>  <value>1024</value>  </property>  <property>  <name>yarn.scheduler.minimum-allocation-mb</name>  <value>256</value>  </property>  <property>  <name>yarn.nodemanager.vmem-check-enabled</name>  <value>false</value>  </property>  <property>  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>  <value>org.apache.hadoop.mapred.ShuffleHandler</value>  </property>  <property>  <name>yarn.scheduler.capacity.maximum-am-resource-percent</name>  <value>0.95</value>  </property>  <property>  <name>yarn.application.classpath</name>  <value>$HADOOP\_CONF\_DIR,$HADOOP\_COMMON\_HOME/share/hadoop/common/\*,$HADOOP\_COMMON\_HOME/share/hadoop/common/lib/\*,$HADOOP\_HDFS\_HOME/share/hadoop/hdfs/\*,$HADOOP\_HDFS\_HOME/share/hadoop/hdfs/lib/\*,$HADOOP\_CONF\_DIR/\*,$HADOOP\_YARN\_HOME/share/hadoop/yarn/\*,$HADOOP\_YARN\_HOME/share/hadoop/yarn/lib/\*</value>  </property>  </configuration> |

1. Definir a localização do JAVA no ambiente HADOOP e especificar propriedades adicionais para que o HADOOP não apresente *warnings* sobre bibliotecas não nativas do JAVA.

Arquivo: **~/hadoop-3.4.0/etc/hadoop/hadoop-env.sh**

**(linha 54 e 90)**

|  |
| --- |
| export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre/  export HADOOP\_OPTS="$HADOOP\_OPTS -XX:-PrintWarnings -Djava.net.preferIPv4Stack=true -Djava.library.path=$HADOOP\_COMMON\_LIB\_NATIVE\_DIR" |

1. Definir quem são os nós que irão executar as tarefas, editando o arquivo workers.

Arquivo: **~/hadoop-3.4.0/etc/hadoop/workers**

|  |
| --- |
| slave1  slave2  node-master |

(remover o arquivo **slaves**, caso exista)

1. Definir as variáveis do ambiente do sistema Linux. Estas variáveis indicam a localização da instalação do Hadoop e seus arquivos de configuração e exemplo.

Arquivo: **~/.bashrc**

**(ao final do arquivo acrescentar)**

|  |
| --- |
| export HADOOP\_HOME "/root/hadoop"  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 "$HADOOP\_OPTS -XX:-PrintWarnings -Djava.net.preferIPv4Stack=true -Djava.library.path=$HADOOP\_COMMON\_LIB\_NATIVE\_DIR"  export LD\_LIBRARY\_PATH "$HADOOP\_COMMON\_LIB\_NATIVE\_DIR"  export JAVA\_HOME "/usr/lib/jvm/java-8-openjdk-amd64/jre/"  export \_JAVA\_OPTIONS "-Xmx2048m"  export PATH "$PATH:$HADOOP\_HOME/sbin:$HADOOP\_HOME/bin:$JAVA\_HOME/bin"  export HDFS\_NAMENODE\_USER "root"  export HDFS\_DATANODE\_USER "$HDFS\_NAMENODE\_USER"  export HDFS\_SECONDARYNAMENODE\_USER "$HDFS\_NAMENODE\_USER"  export YARN\_RESOURCEMANAGER\_USER "$HDFS\_NAMENODE\_USER"  export YARN\_NODEMANAGER\_USER "$HDFS\_NAMENODE\_USER" |

1. Configurar o arquivo de hosts para que os nós possam ser reconhecidos pelo nome (em vez de IP).

Arquivo: **/etc/hosts**

|  |  |
| --- | --- |
|  |  |
| 172.18.0.2 | node-master |
| 172.18.0.3 | slave1 |
| 172.18.0.4 | slave2 |
|  |  |

1. Neste momento já temos o Apache Hadoop configurado no mestre (node-master). Entretanto, faz-se necessário copiar a pasta de instalação Hadoop para cada um dos nós escravos. Cada escravo deve ser configurado para que possua o seu hostname conforme tabela do item 7 acima. Isto é possível editando o arquivo /etc/hostname. Além disso, deve ser definido o IP também conforme o item 7 acima.
2. Apenas no **Node-Master**: gerar e distribuir as chaves de autenticação para o usuário root
   1. Para gerar as chaves (com 4096 bits)

~$ ssh-keygen -b 4096

* 1. Inserir na lista de chaves autorizadas

~$ cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys

~$ chmod 600 ~/.ssh/authorized\_keys

* 1. Copiar para cada um dos nós escravos

~$ ssh-copy-id -i ~/.ssh/id\_rsa.pub root@slave1

~$ ssh-copy-id -i ~/.ssh/id\_rsa.pub root@slave2

(Verificar se o node-master consegue realizar uma conexão via ssh (sem solicitar uma senha) para cada um dos slaves, por exemplo: ~$ ssh slave1)

Em caso de ainda necessitar digitar a senha, digite no terminal: ~$ ssh-add

1. Se tudo estiver correto, apenas no **Node-Master** devemos proceder à preparação do sistema de arquivos HDFS:

~$ hdfs namenode –format

1. A partir deste momento, o sistema está pronto para ser executado. Será executado o HDFS e em seguida o Yarn (em um só comando).

- No Node-Master:

~$ start-dfs.sh && start-yarn.sh

(este comando automaticamente iniciará os processos nos escravos)

(deve-se aguardar em torno de 30 a 60 segundos para que os nós formem o cluster)

1. Para verificar a formação do cluster no **Node-Master**:

Via terminal: ~$ hdfs dfsadmin -report

Ou

Via web: http://node-master:9870

(verificar se existem 5 Live Nodes)

1. Para verificar a execução do Yarn no **Node-Master**:

Via terminal: ~$ yarn node -list

Ou

Via web: http://node-master:8088

(verificar se existem 5 nós ativos)

1. Lista de comandos úteis do HDFS:

|  |
| --- |
| hdfs dfs -ls : Lista um diretório  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 |

Referência: <https://hadoop.apache.org/docs/r3.0.0/hadoop-project-dist/hadoop-common/FileSystemShell.html>

1. Uma vez que o cluster está operacional, vamos criar as pastas e copiar arquivos para o HDFS:
   1. Criar as pastas no HDFS:

~$ hdfs dfs -mkdir -p /user/root/input

* 1. Copiar arquivos para a pasta *input*

~$ hdfs dfs -put ${HADOOP\_HOME}/etc/hadoop/\*.xml input

1. Executar uma tarefa com o Yarn e monitorá-la pela interface web:

~$ yarn jar ${HADOOP\_HOME}/share/hadoop/mapreduce/hadoop-mapreduce-examples-\*.jar grep input output 'dfs[a-z.]+'

(esta é a mesma tarefa de procurar palavras que começam com dfs\*)

1. Exemplo: Cálculo do Pi

O exemplo do pi usa um método estatístico (quasi-Monte Carlo) para estimar o valor de pi. Pontos são colocados aleatoriamente em um quadrado de unidade. O quadrado também contém um círculo. A probabilidade de que os pontos caiam dentro do círculo é igual à área do círculo, pi/4. O valor de pi pode ser estimado do valor de 4R. R é a proporção do número de pontos que estão dentro do círculo em relação ao número total de pontos que estão dentro do quadrado. Quanto maior a amostra de pontos usados, melhor será a estimativa.

Use o seguinte comando para executar o exemplo. O comando usa 16 mapas com 100.000 amostras cada um para estimar o valor de pi:

~$ yarn jar ${HADOOP\_HOME}/share/hadoop/mapreduce/hadoop-mapreduce-examples-\*.jar pi 16 100000

O valor retornado por este comando deve ser semelhante a 3,14159155000000000000. Para referência, as 10 primeiras casas decimais de pi são 3,1415926535.

1. Experimente utilizar os arquivos mapper e reducer desenvolvidos em python como scripts de processamento dos dados no Yarn. Deve-se criar um arquivo input.txt no HDFS contendo as palavras que se deseja contar.

~$ yarn jar ${HADOOP\_HOME}/share/hadoop/tools/lib/hadoop-streaming\*.jar -files mapper.py,reducer.py -mapper mapper.py -reducer reducer.py -input input.txt -output output