hadoop、spark搭建

2022年11月8日 14:54

首先：

安装jdk

```
sudo apt-get install default-jdk
```

安装完后查找jdk位置：

```
which java
        #/usr/bin/java
file /usr/bin/java
        #/etc/alternatives/java
file /etc/alternatives/java
        #/usr/lib/jvm/java-11-openjdk-amd64/bin/java
```

则JDK的目录为/usr/lib/jvm/java-11-openjdk-amd64

开启ssh

```
sudo service ssh start
```

可能出现permission denied

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

hadoop：(以用户的~为开始目录)

首先下载安装包

```
sudo wget
https://mirrors.ustc.edu.cn/apache/hadoop/common/hadoop-3.2.3/hadoop-3.2.3.tar.gz
```

解压并配置环境变量

```
tar zxvf hadoop-3.2.3.tar.gz && mv hadoop-3.2.3 hadoop3
echo "export HADOOP_HOME=/home/huashuimu/hadoop3" >> ~/.bashrc
echo "export PATH=/home/huashuimu/hadoop3/bin:$PATH" >> ~/.bashrc
source ~/.bashrc
```

修改配置文件（hadoop-env.sh）

```
cd /home/huashuimu/hadoop3/etc/hadoop/
vim hadoop-env.sh
:/export JAVA_HOME
```

去掉注释修改为自己的JDK目录

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

配置Hadoop文件（core-site） `vim $HADOOP_HOME/etc/hadoop/core-site.xml`

```
<configuration>
    <property>
<name>fs.defaultFS</name>
<value>hdfs://localhost:9000</value>
</property>
<property>
<description>A base for other temporary directories.</description>
</property>
    <property>
<name>hadoop.proxyuser.wzt.hosts</name>
<value>*</value>
    </property>
    <property>
        <name>hadoop.proxyuser.wzt.groups</name>
<value>*</value>
    </property>

</configuration>
```

配置Hadoop文件（hdfs-site) vim $HADOOP_HOME/etc/hadoop/hdfs-site.xml

```
<configuration>
    <property>
        <name>dfs.replication</name>
<value>1</value>
 </property>
<property>
        <name>dfs.webhdfs.enabled</name>
<value>true</value>                                              </property>
<property>
        <name>dfs.permissions.enabled</name>
<value>false</value>
</property>
</configuration>
```

配置Hadoop文件（yarn-site) vim $HADOOP_HOME/etc/hadoop/yarn-site.xml

```
<configuration>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>

    <property>
        <name>yarn.nodemanager.local-dirs</name>
        <value>/tmp/yarn-local-dirs</value>
    </property>
    <property>
        <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
        <value>org.apache.hadoop.mapred.ShuffleHandler</value>
    </property>

    <property>
        <name>yarn.nodemanager.hostname</name>
        <value>127.0.0.1</value>
    </property>

    <property>
        <name>yarn.nodemanager.disk-health-checker.enable</name>
        <value>false</value>
    </property>

    <property>
        <name>yarn.acl.enable</name>
        <value>0</value>
   </property>

    <property>
        <name>yarn.scheduler.maximum-allocation-mb</name>
        <value>2048</value>
    </property>
```

```
</configuration>
```

spark：(以用户的~为开始目录，要先安装Hadoop)

首先下载安装包

```
sudo wget https://archive.apache.org/dist/spark/spark-3.1.2/spark-3.1.2-bin-hadoop3.2.tgz
```

解压安装包并配置环境变量

```
tar zxvf spark-3.1.2-bin-hadoop3.2.tgz && mv spark-3.1.2-bin-hadoop3.2 spark3
echo "export SPARK_HOME=/home/huashuimu/spark3" >> ~/.bashrc
```

修改配置文件

```
cd /home/huashuimu/spark3/conf/
cp spark-env.sh.template spark-env.sh
vim spark-env.sh

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export HADOOP_HOME=/home/huashuimu/hadoop3
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export YARN_CONF_DIR=$HADOOP_HOME/etc/hadoop
```

修改yarn配置文件

```
cd /home/huashuimu/hadoop3/etc/hadoop/
vim yarn-site.xml
```

原来基础上加上：

```
<property>
        <name>yarn.nodemanager.aux-services.mapreduce_shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>

 <property>
        <name>yarn.scheduler.maximum-allocation-mb</name>
        <value>1024</value>
</property>

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

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

运行hadoop

```
$HADOOP_HOME/sbin/start-all.sh
```

如果显示localhost: datanode is running as process 10474. Stop it first and ensure /tmp/hadoop-huashuimu-datanode.pid file is empty before retry.执行：

```
cat /tmp/hadoop-huashuimu-* > kill
ls /tmp/ |grep hadoop-huashuimu-
> hadoop-huashuimu-datanode.pid
> hadoop-huashuimu-namenode.pid
> hadoop-huashuimu-nodemanager.pid
> hadoop-huashuimu-resourcemanager.pid
> hadoop-huashuimu-secondarynamenode.pid
```

查看各个部分运行端口

```
jps
```

数据集上传

```
hdfs dfs -mkdir -p input  #创建input文件夹
hdfs dfs -put ~/wordcount/sample.txt input/  #将数据放到创建的文件夹
hdfs dfs -cat input/sample.txt  #查看文件内容
```

在hadoop的文件系统上操作

hadoop fs -command [parameter] 或 hdfs dfs -command [parameter]

例如：

```
hadoop fs -ls
```

运行程序

```
hadoop jar WordCount.jar input/sample.txt ./result
```

spark（会出错）

运行wordcount

```
cd wordcount
${SPARK_HOME}/bin/spark-submit --master yarn --name "job" --deploy-mode client driver-memory 2g --driver-cores 2 --executor-memory 4g --executor-cores 4 --num-executors 15 --class org.example.WordCount ./WordCount.jar
```

Map、Reduce编程

2022年11月8日 21:40

首先创建maven项目，maven的下载、设置
https://blog.csdn.net/weixin_64987028/article/details/123641226

创建完项目后：
在src/java下创建package：com.mapreduce.wordcount，这个就是最终被打包的
在com.mapreduce.wordcount下分别创建：
WordCountDriver、WordCountMapper、WordCountReducer
在src/main/resources创建log4j.properties，用于设置日志相关参数

设置基本参数：
pom.xml：项目最基本的设置

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.test</groupId>
  <artifactId>MapReduce</artifactId>
  <version>1.0-SNAPSHOT</version>

  <dependencies>  //最重要部分，设置项目依赖，自动下载缺少的
    <dependency>
      <groupId>org.apache.hadoop</groupId>
      <artifactId>hadoop-client</artifactId>
      <version>3.1.3</version>
    </dependency>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
    </dependency>
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-log4j12</artifactId>
      <version>1.7.30</version>
    </dependency>
  </dependencies>

  <build>  //构造jar相关参数，这里设置了main类
    <finalName>WordCount</finalName>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-jar-plugin</artifactId>
        <configuration>
```

```
                    <archive>
                        <manifest>
                            <mainClass>
com.mapreduce.wordcount.WordCountDriver</mainClass>
                        </manifest>
                    </archive>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

log4j.properties：日志相关参数，保持默认即可

```
log4j.rootLogger=INFO,stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender。stdout.layout.ConversionPattern=%d %p [%c] - %m%n
log4j.appender.logfile=org.apache.1og4j.FileAppender
log4j.appender.logfile.File=target/spring.log
log4j.appender.logfile.layout=org.apache.log4j.PatternLayout
log4j.appender.logfile.layout.ConversionPattern=%d %p [%c] - %m%n
```

代码：

WordCountDriver：主类，用于与集群交互

```
package com.mapreduce.wordcount;
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCountDriver{
    public static void main(String[] args)throws
IOException,ClassNotFoundException,InterruptedException{

        //1获取配置信息以及获取job对象
        Configuration conf=new Configuration();
        Job job=Job.getInstance(conf);

        //2关联本Driver程序的jar
        job.setJarByClass(WordCountDriver.class);

        //3关联Mapper和Reducer的jar
        job.setMapperClass(WordCountMapper.class);
        job.setReducerClass(WordCountReducer.class);

        //4设置Mapper输出的Kv类型
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);

        //5设置最终输出kv类型
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        //6设置输入和输出路径
        FileInputFormat.setInputPaths(job,new Path(args[0]));
        FileOutputFormat.setOutputPath(job,new Path(args[1]));

        //7提交job
        boolean result=job.waitForCompletion(true);
        System.exit(result ? 0:1);
    }
}
```

WordCountMapper：执行map操作的类

```
package com.mapreduce.wordcount;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class WordCountMapper extends
Mapper<LongWritable,Text,Text,IntWritable> {
    Text k = new Text();
    IntWritable v = new IntWritable(1);

    @Override
    protected void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {

        String line = value.toString();

        String[] words = line.split(" ");

        for (String word : words) {
            k.set(word);
            context.write(k, v);
        }
    }
}
```

WordCountReducer：执行reduce操作的类

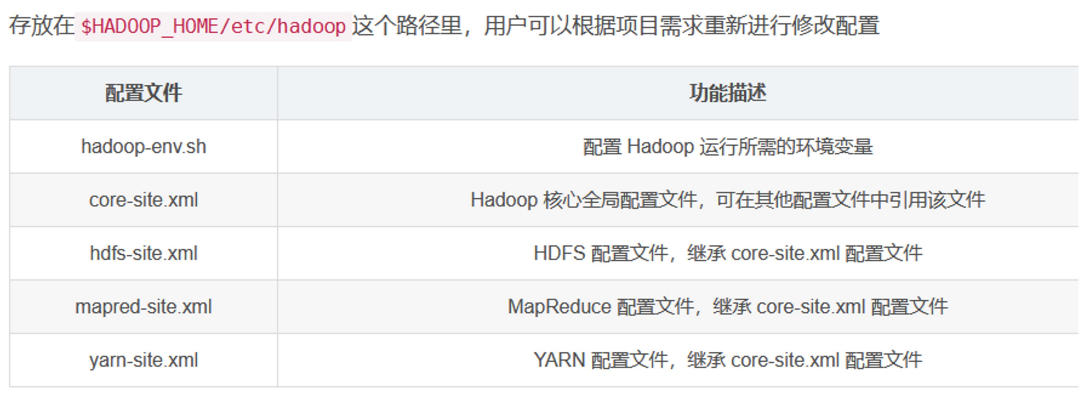
```
package com.mapreduce.wordcount;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
```

```
public class WordCountReducer extends
Reducer<Text,IntWritable,Text,IntWritable> {
    int sum;
    IntWritable v = new IntWritable();

    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Context
context) throws IOException, InterruptedException {

        sum = 0;
        for (IntWritable count : values) {
            sum += count.get();
        }
        v.set(sum);
        context.write(key, v);
    }
}
```

hadoop配置详解

2022年11月8日　　21:50

主要配置文件：



存放在 `$HADOOP_HOME/etc/hadoop` 这个路径里，用户可以根据项目需求重新进行修改配置

配置文件	功能描述
hadoop-env.sh	配置 Hadoop 运行所需的环境变量
core-site.xml	Hadoop 核心全局配置文件，可在其他配置文件中引用该文件
hdfs-site.xml	HDFS 配置文件，继承 core-site.xml 配置文件
mapred-site.xml	MapReduce 配置文件，继承 core-site.xml 配置文件
yarn-site.xml	YARN 配置文件，继承 core-site.xml 配置文件

hadoop-env.sh:

这个文件唯一要修改的地方，就是jdk的目录

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

core-site.xml

```
<property>
        <name>hadoop.proxyuser.userA.hosts</name>
        <value>*</value>
 </property>
<property>
        <name>hadoop.proxyuser.userA.users</name>
        <value>user1,user2</value>
 </property>
```

允许用户A在任意主机上代理用户1和用户2

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

设置namenode的hdfs协议的文件系统的通讯地址（web访问hdfs文件系统地址）

hdfs-site.xml

```
<property>
    <name>dfs.namenode.name.dir</name>
    <value>/usr/local/hadoop-2.8.3/data/name</value>
 </property>
```

设置namenode数据的存放地址，也就是hdfs系统中文件的元数据存储地址

```
  <property>
    <name>dfs.namenode.http-address</name>
    <value>node01:50070</value>
  </property>
```

namenode的访问地址

```
  <property>
     <name>dfs.datanode.data.dir</name>
     <value>/usr/local/hadoop-2.8.3/data/data</value>
  </property>
```
datanode的数据存放地址，也就是block块存放的地址

```
  <property>
     <name>dfs.replication</name>
     <value>3</value>
  </property>
```
副本的数目，上传一个文件，分割为block块后（或就一个block），每个block冗余副本数

```
  <property>
           <name>dfs.permissions</name>
           <value>false</value>
  </property>
```
HDFS文件的控制权限

```
  <property>
     <name>dfs.blocksize</name>
     <value>134217728</value>
  </property>
```
block块的大小

```
  <property>
     <name>dfs.secondary.http.address</name>
     <value>hadoop01:50090</value>
  </property>
  <property>
     <name>dfs.secondary.http.address</name>
     <value>hadoop01:50090</value>
  </property>
```
第二namenode访问地址

```
  <property>
     <name>dfs.webhdfs.enabled</name>
     <value>true</value>
  </property>
```
开启hdfs的web访问接口

```
<property>
        <name>dfs.namenode.edits.dir</name>
        <value>file:///export/servers/hadoop-2.7.5/hadoopDatas/nn/edits</value>
</property>
```
元数据操作日志的存放位置 edits的存放位置

```
<property>
        <name>dfs.namenode.checkpoint.dir</name>
        <value>file:///export/servers/hadoop-2.7.5/hadoopDatas/snn/name</value>
</property>
```
元数据检查点保存的位置

mapred-site.xml

```
<property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
</property>
```
基于资源管理器yarn的map-reduce

```
<property>
    <name>mapreduce.job.ubertask.enable</name>
    <value>true</value>
</property>
```
开启map-reduce小任务模式，调优

```
<property>
    <name>mapreduce.jobhistory.address</name>
    <value>hadoop01:10020</value>
</property>
<property>
    <name>mapreduce.jobhistory.webapp.address</name>
    <value>hadoop02:19888</value>
</property>
```
jobhistory的访问地址

yarn-site.xml

```
<property>
    <name>yarn.resourcemanager.hostname</name>
    <value>hadoop01</value>
</property>
```
资源管理器运行在哪台主机上

```
<property>
    <name>yarn.resourcemanager.address</name>
    <value>hadoop01:8032</value>
</property>
<property>
    <name>yarn.resourcemanager.resource-tracker.address</name>
    <value>hadoop01:8031</value>
</property>
```
资源管理器的IPC通信地址

```
<property>
    <name>yarn.resourcemanager.scheduler.address</name>
    <value>hadoop01:8030</value>
</property>
```
资源管理器调度程序的IPC通讯地址

```
<property>
    <name>yarn.resourcemanager.webapp.address</name>
    <value>singlehost:8088</value>
</property>
```
资源管理器的http通讯地址

```
<property>
    <name>yarn.resourcemanager.admin.address</name>
```

```
        <value>hadoop01:8033</value>
    </property>
```
资源管理器的IPC的管理地址

```
   <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
```
nodemanager的通信方式

```
    <property>
        <name>yarn.log-aggregation-enable</name>
        <value>true</value>
    </property>
    <property>
        <name>yarn.log-aggregation.retain-seconds</name>
        <value>604800</value>
    </property>
```
日志聚合模式，以及保存时长