**第一节：Hadoop的安装**

1. 预备知识

本文假设你已经对虚拟机软件、Linux等有初步的了解，如果你不了解这方面的内容，请首先学习马士兵Linux教程

1. 下载软件
   1. 下载VirtualBox <http://download.virtualbox.org/virtualbox/5.1.18/VirtualBox-5.1.18-114002-Win.exe>
   2. 下载CentOS <http://isoredirect.centos.org/centos/7/isos/x86_64/CentOS-7-x86_64-DVD-1611.iso>
   3. 下载XShell与XFtp 请在百度自己搜索与安装
   4. 下载jdk <http://download.oracle.com/otn-pub/java/jdk/8u121-b13/e9e7ea248e2c4826b92b3f075a80e441/jdk-8u121-linux-x64.rpm>
   5. 下载hadoop2.7.3 -> <http://mirror.bit.edu.cn/apache/hadoop/common/hadoop-2.7.3/hadoop-2.7.3.tar.gz>
2. 安装VirtualBox

设定VirtualBox虚拟网卡的IP地址

1. 设置虚拟机中的网络设置
   1. 在虚拟机中选用host-only网络
   2. vi /etc/sysconfig/network   
      NETWORKING=yes  
      GATEWAY=192.168.56.1
   3. vi /etc/sysconfig/network-sripts/ifcfg-enp0s3 TYPE=Ethernet IPADDR=192.168.56.100 NETMASK=255.255.255.0
   4. 修改主机名hostnamectl set-hostname master (主机名千万不能有下划线！)
   5. 重启网络service network restart
   6. 互相ping，看是否测试成功，若不成功，注意防火墙的影响。关闭windows或虚拟机的防火墙。systemctl stop firewalld system disable firewalld
2. 使用XShell登陆

检查ssh服务状态systemctl status sshd (service sshd status)，验证使用XShell是否能登陆成功。

1. 将hadoop和jdk上传到虚拟机
2. 安装JDK rpm -ivh ./xxxxx.jdk，验证rpm -qa | grep jdk，在命令行中敲java命令，确认jdk已经安装完成

jdk默认安装在/usr/java目录下

1. 安装hadoop

cd /usr/local  
tar –xvf ./hadoop-2.7.2.tar.gz  
把目录修改为hadoop mv hadoop-2... hadoop  
修改hadoop-env.sh  
vi /usr/local/hadoop/etc/hadoop/hadoop-env.sh  
修改export JAVA\_HOME 语句为 export JAVA\_HOME=/usr/java/default  
把/usr/hadoop/bin和/usr/hadoop/sbin设到PATH中  
vi /etc/profile  
追加 export PATH=$PATH:/usr/hadoop/bin:/usr/hadoop/sbin  
source etc/profile  
测试hadoop命令是否可以直接执行，任意目录下敲hadoop

1. 关闭虚拟机，复制3份

分别修改虚拟机的ip和hostname，确认互相能够ping通，用ssh登陆，同时修改所有虚拟机的/etc/hosts，确认使用名字可以ping通

**第二节 Hadoop测试**

1. 启动集群 1.在master上启动hadoop-daemon.sh start namenode

2.在slave上启动hadoop-daemon.sh start datanode

3.用jps指令观察执行结果

4.用hdfs dfsadmin -report观察集群配置情况

5.hadoop fs -rm /filename

6.通过http://192.168.56.100:50070/web界面观察集群运行情况

7.用hadoop-daemon.sh stop ...手工关闭集群

2. 对集群进行集中管理 1.修改master上/etc/hadoop/slaves文件，每一个slave占一行 slave1

slave2

slave3

2.使用start-dfs.sh启动集群，并观察结果

3.使用stop-dfs.sh停止集群

4.配置免密SSH远程登陆 ssh slave1

输入密码

exit

#此时需要密码才能登陆

cd

ls -la

cd .ssh

ssh-keygen -t rsa (四个回车)

#会用rsa算法生成私钥id\_rsa和公钥id\_rsa.pub

ssh-copy-id slaveX

再次ssh slave1

此时应该不再需要密码

5.重新使用start-dfs.sh启动集群

3. 修改windows机上C:\Windows\System32\drivers\etc\hosts文件，可以通过名字访问集群web界面

4. 使用hdfs dfs 或者hadoop fs命令对文件进行增删改查的操作 1.hadoop fs -ls /

2.hadoop fs -put file /

3.hadoop fs -mkdir /dirname

4.hadoop fs -text /filename

5.hadoop fs -rm /filename

5.通过网页观察文件情况

6.将hdfs-site.xml的replication值设为2，通过网页观察分块情况

7.设定dfs.namenode.heartbeat.recheck-interval为10000，然后停掉其中一台slave，观察自动复制情况

8.启动停掉的slave节点，通过网页观察datanode的动态添加

9.添加新的节点，体会hadoop的横向扩展 1.启动slave4，关闭防火墙，修改hostname

2.修改etc/hosts，加入对自己以及其他机器的解析，重启网络

3.在master中设定对于slave4的解析

4.启动slave4的datanode，通过网页进行观察

5.hdfs dfsadmin –shutdownDatanode slave4:50020

10. 通过java程序访问hdfs，就把HDFS集群当成一个大的系统磁盘就行了！

11. 从HDFS中读取文件 URL.setURLStreamHandlerFactory(new FsUrlStreamHandlerFactory());

InputStream in = new URL("hdfs://192.168.56.100:9000/test.data").openStream();

IOUtils.copyBytes(in, System.out, 4096, true);

12. 学习使用FileSystem类 Configuration conf = new Configuration();

conf.set("fs.defaultFS", "hdfs://192.168.56.100:9000");

FileSystem fileSystem = FileSystem.get(conf);

boolean b = fileSystem.exists(new Path("/hello"));

System.out.println(b);

boolean success = fileSystem.mkdirs(new Path("/mashibing"));

System.out.println(success);

success = fileSystem.delete(new Path("/mashibing"), true);

System.out.println(success);

FSDataOutputStream out = fileSystem.create(new Path("/test.data"), true);

FileInputStream fis = new FileInputStream("c:/test/core-site.xml");

IOUtils.copyBytes(fis, out, 4096, true);

FileStatus[] statuses = fileSystem.listStatus(new Path("/"));

//System.out.println(statuses.length);

for(FileStatus status : statuses) {

System.out.println(status.getPath());

System.out.println(status.getPermission());

System.out.println(status.getReplication());

}

13.思考百度网盘的实现方式？

14.core-site做什么用？hdfs-site做什么用？

第三节：java编写hdfs程序

1. hdfs系统会把用到的数据存储在core-site.xml中由hadoop.tmp.dir指定，而这个值默认位于/tmp/hadoop-${user.name}下面， 由于/tmp目录在系统重启时候会被删除，所以应该修改目录位置。 修改core-site.xml(在所有站点上都修改)

<property>

<name>hadoop.tmp.dir</name>

<value>/var/hadoop</value>

</property>

2.hdfs namenode -format

3. 通过java程序访问hdfs，就把HDFS集群当成一个大的系统磁盘就行了！

4. windows上的权限系统和linux上的权限系统，测试期间为了简单起见可以关闭权限检查 在namenode的hdfs-site.xml上，添加配置：

<property>

<name>dfs.permissions.enabled</name>

<value>false</value>

</property>

重新启动namenode: hadoop-daemon.sh stop namenode, hadoop-daemon.sh start namenode

5. 从HDFS中读取文件

URL.setURLStreamHandlerFactory(new FsUrlStreamHandlerFactory());

InputStream in = new URL("hdfs://192.168.56.100:9000/test.data").openStream();

IOUtils.copyBytes(in, System.out, 4096, true);

6. 学习使用FileSystem类

Configuration conf = new Configuration();

conf.set("fs.defaultFS", "hdfs://192.168.56.100:9000");

FileSystem fileSystem = FileSystem.get(conf);

boolean b = fileSystem.exists(new Path("/hello"));

System.out.println(b);

boolean success = fileSystem.mkdirs(new Path("/mashibing"));

System.out.println(success);

success = fileSystem.delete(new Path("/mashibing"), true);

System.out.println(success);

FSDataOutputStream out = fileSystem.create(new Path("/test.data"), true);

FileInputStream fis = new FileInputStream("c:/test/core-site.xml");

IOUtils.copyBytes(fis, out, 4096, true);

FileStatus[] statuses = fileSystem.listStatus(new Path("/"));

//System.out.println(statuses.length);

for(FileStatus status : statuses) {

System.out.println(status.getPath());

System.out.println(status.getPermission());

System.out.println(status.getReplication());

}

7.思考百度网盘的实现方式？

8.core-site做什么用？hdfs-site做什么用？

**第四节：yarn安装和测试**

1.配置计算调度系统Yarn和计算引擎Map/Reduce

2. namenode上配置mapred-site.xml

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

3. yarn-site.xml的配置

<property>

<name>yarn.resourcemanager.hostname</name>

<value>master</value>

</property>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

4.启动yarn集群start-yarn.sh

5.jps观察启动结果

6.可以使用yarn-daemon.sh单独启动resourcemanager和nodemanager

7.通过网页http://master:8088/观察yarn集群

8.find /usr/local/hadoop -name \*example\*.jar 查找示例文件

9.通过hadoop jar xxx.jar wordcount /input /output来运行示例程序

10.通过网页来观察该job的运行情况

11. hadoop job -kill job\_id --> mapred job -kill job\_id mapred job -list

12. 经验：yarn-site如果是集中启动，其实只需要在管理机上配置一份即可，但是如果单独启动，需要每台机器一份，在网页上可以看到当前机器的配置，以及这个配置的来源

**第五节：java编写mapreduce程序**

首页

1.java开发map\_reduce程序

2. 配置系统环境变量HADOOP\_HOME，指向hadoop安装目录（如果你不想招惹不必要的麻烦，不要在目录中包含空格或者中文字符）

把HADOOP\_HOME/bin加到PATH环境变量（非必要，只是为了方便）

3. 如果是在windows下开发，需要添加windows的库文件 1.把盘中共享的bin目录覆盖HADOOP\_HOME/bin

2.如果还是不行，把其中的hadoop.dll复制到c:\windows\system32目录下，可能需要重启机器

4.建立新项目，引入hadoop需要的jar文件

5. 代码WordMapper：

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 WordMapper extends Mapper<LongWritable,Text, Text, IntWritable> {

@Override

protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, IntWritable>.Context context)

throws IOException, InterruptedException {

String line = value.toString();

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

for(String word : words) {

context.write(new Text(word), new IntWritable(1));

}

}

}

6. 代码WordReducer：

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.Reducer;

public class WordReducer extends Reducer<Text, IntWritable, Text, LongWritable> {

@Override

protected void reduce(Text key, Iterable<IntWritable> values,

Reducer<Text, IntWritable, Text, LongWritable>.Context context) throws IOException, InterruptedException {

long count = 0;

for(IntWritable v : values) {

count += v.get();

}

context.write(key, new LongWritable(count));

}

}

7. 代码Test：

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

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 Test {

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf);

job.setMapperClass(WordMapper.class);

job.setReducerClass(WordReducer.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(IntWritable.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(LongWritable.class);

FileInputFormat.setInputPaths(job, "c:/bigdata/hadoop/test/test.txt");

FileOutputFormat.setOutputPath(job, new Path("c:/bigdata/hadoop/test/out/"));

job.waitForCompletion(true);

}

}

8. 把hdfs中的文件拉到本地来运行

FileInputFormat.setInputPaths(job, "hdfs://master:9000/wcinput/");

FileOutputFormat.setOutputPath(job, new Path("hdfs://master:9000/wcoutput2/"));

注意这里是把hdfs文件拉到本地来运行，如果观察输出的话会观察到jobID带有local字样

同时这样的运行方式是不需要yarn的(自己停掉yarn服务做实验)

9. 在远程服务器执行

conf.set("fs.defaultFS", "hdfs://master:9000/");

conf.set("mapreduce.job.jar", "target/wc.jar");

conf.set("mapreduce.framework.name", "yarn");

conf.set("yarn.resourcemanager.hostname", "master");

conf.set("mapreduce.app-submission.cross-platform", "true");

FileInputFormat.setInputPaths(job, "/wcinput/");

FileOutputFormat.setOutputPath(job, new Path("/wcoutput3/"));

如果遇到权限问题，配置执行时的虚拟机参数-DHADOOP\_USER\_NAME=root

10.也可以将hadoop的四个配置文件拿下来放到src根目录下，就不需要进行手工配置了，默认到classpath目录寻找

11.或者将配置文件放到别的地方，使用conf.addResource(.class.getClassLoader.getResourceAsStream)方式添加，不推荐使用绝对路径的方式

12. 建立maven-hadoop项目：

<project xmlns="http://maven.apache.org/POM/4.0.0" xsi:schemalocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<modelversion>4.0.0</modelversion>

<groupid>mashibing.com</groupid>

<artifactid>maven</artifactid>

<version>0.0.1-SNAPSHOT</version>

<name>wc</name>

<description>hello mp</description>

<properties>

<project.build.sourceencoding>UTF-8</project.build.sourceencoding>

<hadoop.version>2.7.3</hadoop.version>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

</dependency>

<dependency>

<groupId>org.apache.hadoop</groupId>

<artifactId>hadoop-client</artifactId>

<version>${hadoop.version}</version>

</dependency>

<dependency>

<groupId>org.apache.hadoop</groupId>

<artifactId>hadoop-common</artifactId>

<version>${hadoop.version}</version>

</dependency>

<dependency>

<groupId>org.apache.hadoop</groupId>

<artifactId>hadoop-hdfs</artifactId>

<version>${hadoop.version}</version>

</dependency>

</dependencies>

</project>

13. 配置log4j.properties，放到src/main/resources目录下

log4j.rootCategory=INFO, stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=[QC] %p [%t] %C.%M(%L) | %m%n

第六节：hive数据挖掘：

1.Hive入门

2. 解压Hive，到/usr/local目录，将解压后的目录名mv为hive

设定环境变量HADOOP\_HOME，HIVE\_HOME，将bin目录加入到PATH中

3. 1.cd /usr/local/hive/conf

2.cp hive-default.xml.template hive-site.xml

3.修改hive.metastore.schema.verification，设定为false

4.创建/usr/local/hive/tmp目录，替换${system:java.io.tmpdir}为该目录

5.替换${system:user.name}为root

4.schematool -initSchema -dbType derby

会在当前目录下简历metastore\_db的数据库。

注意！！！下次执行hive时应该还在同一目录，默认到当前目录下寻找metastore。

遇到问题，把metastore\_db删掉，重新执行命令

实际工作环境中，经常使用mysql作为metastore的数据

5.启动hive

6.观察hadoop fs -ls /tmp/hive中目录的创建

7. 1.show databases;

2.use default;

3.create table doc(line string);

4.show tables;

5.desc doc;

6.select \* from doc;

7.drop table doc;

8.观察hadoop fs -ls /user

9.启动yarn

10. 1.load data inpath '/wcinput' overwrite into table doc;

2.select \* from doc;

3.select split(line, ' ') from doc;

4.select explode(split(line, ' ')) from doc;

5.select word, count(1) as count from (select explode(split(line, ' ')) as word from doc) w group by word;

6.select word, count(1) as count from (select explode(split(line, ' ')) as word from doc) w group by word order by word;

7.create table word\_counts as select word, count(1) as count from (select explode(split(line, ' ')) as word from doc) w group by word order by word;

8.select \* from word\_counts;

9.dfs -ls /user/hive/...

11. 使用sougou搜索日志做实验

12. 将日志文件上传的hdfs系统，启动hive

13. create table sougou (qtime string, qid string, qword string, url string) row format delimited fields terminated by ',';

14. load data inpath '/sougou.dic' into table sougou;

15.select count(\*) from sougou;

16.create table sougou\_results as select keyword, count(1) as count from (select qword as keyword from sougou) t group by keyword order by count desc;

17.select \* from sougou\_results limit 10;

第六节：spark入门

下载

1. 地址spark.apache.org

安装

1. 复制一台单独的虚拟机，名c

2. 修改其ip，192.168.56.200

3. 修改其hostname为c，hostnamectl set-hostname c

4. 修改/etc/hosts加入对本机的解析

5. 重启网络服务 systemctl restart network

6. 上传spark安装文件到root目录

7. 解压spark到/usr/local下，将其名字修改为spark

本地运行模式

使用spark-submit提交job

1. cd /usr/local/spark

2. ./bin/spark-submit --class org.apache.spark.examples.SparkPi ./examples/jars/spark-examples\_2.11-2.1.0.jar 10000

使用spark-shell进行交互式提交

1. 创建root下的文本文件hello.txt

2. ./bin/spark-shell

3. 再次连接一个terminal，用jps观察进程，会看到spark-submit进程

4. sc

5. sc.textFile("/root/hello.txt")

6. val lineRDD = sc.textFile("/root/hello.txt")

7. lineRDD.foreach(println)

8. 观察网页端情况

9. val wordRDD = lineRDD.flatMap(line => line.split(" "))

10. wordRDD.collect

11. val wordCountRDD = wordRDD.map(word => (word,1))

12. wordCountRDD.collect

13. val resultRDD = wordCountRDD.reduceByKey((x,y)=>x+y)

14. resultRDD.collect

15. val orderedRDD = resultRDD.sortByKey(false)

16. orderedRDD.collect

17. orderedRDD.saveAsTextFile("/root/result")

18. 观察结果

19. 简便写法：sc.textFile("/root/hello.txt").flatMap(\_.split(" ")).map((\_,1)).reduceByKey(\_+\_).sortByKey().collect

使用local模式访问hdfs数据

1. start-dfs.sh

2. spark-shell执行：sc.textFile("hdfs://192.168.56.100:9000/hello.txt").flatMap(\_.split(" ")).map((\_,1)).reduceByKey(\_+\_).sortByKey().collect （可以把ip换成master，修改/etc/hosts）

3. sc.textFile("hdfs://192.168.56.100:9000/hello.txt").flatMap(\_.split(" ")).map((\_,1)).reduceByKey(\_+\_).sortByKey().saveAsTextFile("hdfs://192.168.56.100:9000/output1")

spark standalone模式

1. 在master和所有slave上解压spark

2. 修改master上conf/slaves文件，加入slave

3. 修改conf/spark-env.sh，export SPARK\_MASTER\_HOST=master

4. 复制spark-env.sh到每一台slave

5. cd /usr/local/spark

6. ./sbin/start-all.sh

7. 在c上执行：./bin/spark-shell --master spark://192.168.56.100:7077 (也可以使用配置文件)

8. 观察http://master:8080

spark on yarn模式