

云计算技术课程实验报告

pangbo

51903091xxxx

I. 实验简介

Hadoop 与 Spark 是两个能够让用户轻松架构和使用的分布式计算框架，本实验将基于 Hadoop 与 Spark 框架学习分布式框架的使用并解决实际问题。

II. 云服务环境准备

A. 云服务器购买

登录华为云，选择购买弹性云服务器 ECS，购买配置如图1。

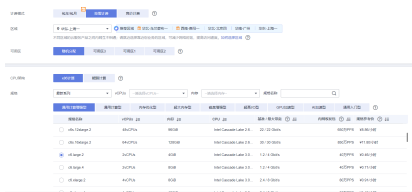


图 1. 云服务器配置选择

使用相同配置购买三台服务器，分别命名为 master、slave01、slave02。为方便后续配置，我分别将三台主机的内网 ip 地址设置为 192.168.0.2、192.168.0.3、192.168.0.4。

B. 集群网络配置

为三台主机修改 hosts 文件，将主机名与对应内网 ip 对应，便于后续快速连接主机。使用指令 `sudo nano /etc/hosts` 打开文本编辑器，修改后内容如图2所示。需要注意的是，为了避免后续实验中出现主机名解析混淆，最好将 hosts 文件中原有项全部删除或注释。

为了方便 master 主机连接 slave 主机，可以为 master 创建公私钥对，将 master 的公钥保存在 slave 主机之后，master 主机通过 ssh 连接 slave 主机无需再输入密码。在 slave 主机上编辑 `~/.ssh/authorized_keys` 文件，追加 master 主机与个人电脑公钥，便于后续实验操作，如图3所示。

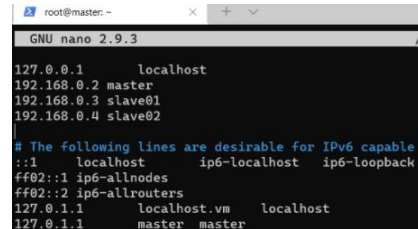


图 2. hosts 文件修改

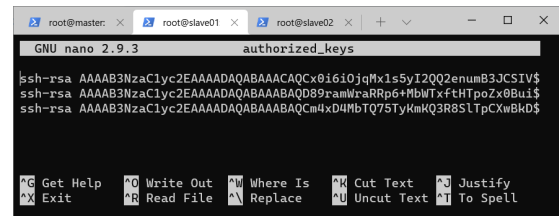


图 3. ssh 公钥配置

尝试在三台主机间使用 ping 或 ssh，均工作正常，集群网络配置完成。

C. 安装 JDK

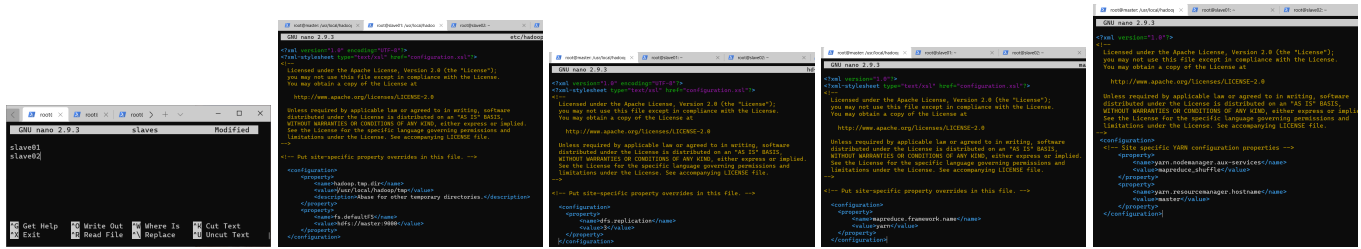
使用 ubuntu 自带的包管理器安装 jdk。

```
sudo apt-get install default-jdk
echo export JAVA_HOME=/usr/lib/jvm/
default-java >> ~/.bashrc
source ~/.bashrc
```

D. 安装 Hadoop

我选择了 Hadoop2.10.1 版本进行实验，在 Hadoop 官网找到二进制文件压缩包，下载并解压。

```
wget https://downloads.apache.org/hadoop/
common/hadoop-2.10.1/hadoop-2.10.1.tar.
gz
sudo tar -zxf ~/hadoop-2.10.1.tar.gz -C /usr/
local
```



(a) slaves (b) core-site.xml (c) hdfs-site.xml (d) mapred-site.xml (e) yarn-site.xml

图 4. Hadoop 配置文件修改

echo export HADOOP_HOME=/usr/local/

hadoop >> ~/.bashrc

echo export PATH=\$PATH:

\$HADOOP_HOME/bin:\$HADOOP_HOME

/sbin >> ~/.bashrc

source ~/.bashrc

以上命令将 Hadoop 解压到 `/usr/local/hadoop` 目录下，也完成了相关环境变量设置。为了配置集群，我们还需要修改 Hadoop 配置文件，这些文件位于 `$HADOOP_HOME/etc/hadoop` 目录下，需要修改的文件包括 `slaves`、`core-site.xml`、`hdfs-site.xml`、`mapred-site.xml`、`yarn-site.xml`。配置文件修改如图4所示。

值得提出的是，`core-site.xml` 文件的设置与参考资料不完全一致，由于相关基础依赖库版本不同，`file:/` 前缀不再被支持，应当在配置文件中移除。

在启动 Hadoop 之前，还需要修改 `$HADOOP_HOME/etc/hadoop/hadoop-env.sh` 文件，该文件存在语法问题，导致 Hadoop 无法正确获得 java 安装目录，在该文件中找到 `JAVA_HOME` 一行，直接将 java 安装目录硬编码入该处。

完成修改后，将配置完成的 Hadoop 发送到其余主机并解压安装。之后便可以顺利启动 Hadoop 集群，master 主机执行 `jps` 指令后输出如图5所示。

```
root@master:/usr/local/hadoop# jps
866 WrapperSimpleApp
27464 Jps
26445 NameNode
26717 SecondaryNameNode
24638 ResourceManager
root@master:/usr/local/hadoop#
```

图 5. master 主机启动 Hadoop 后 jps 结果

E. 安装 Spark

我选择使用 Spark2.1.0 版本进行本实验。从项目主页下载不包含 Hadoop 的版本，解压并设置环境变量。

sudo tar -zxvf ~/spark-2.1.0-bin-without-

hadoop.tgz -C /usr/local/

cd /usr/local

sudo mv ./spark-2.1.0-bin-without-hadoop/ ./

spark

sudo chown -R hadoop ./spark

echo export SPARK_HOME=/usr/local/spark

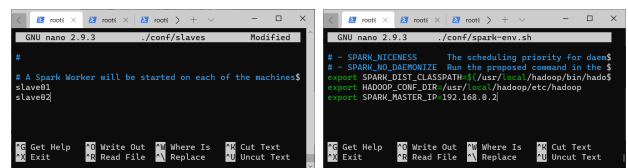
>> ~/.bashrc

echo export PATH=\$PATH:\$SPARK_HOME/

bin:\$SPARK_HOME/sbin >> ~/.bashrc

source ~/.bashrc

Spark 的配置文件位于目录 `$SPARK_HOME/conf` 下，需要修改的配置文件为 `slaves` 与 `spark-env.sh`。修改后如图6所示。

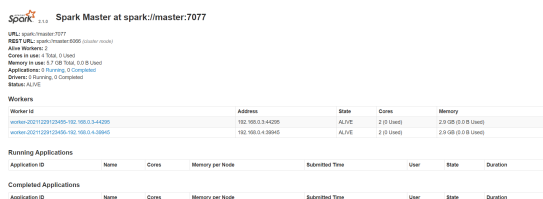


(a) slaves

(b) spark-env.sh

图 6. spark 配置文件修改

将配置完成的 spark 打包传输到其他主机，解压并配置环境变量，在 master 主机上启动主节点，其余主机启动从节点。打开 spark 的管理网页，如图7所示，可以看到两个 worker 均已就绪。



Spark Master at spark://master:7077

URL: spark://master:7077
 REST URL: spark://master:8080 (available locally)
 Also Known As:
 Core is on 1 Task (1 Used)
 Memory is on 17.02 TB (17.02 TB Used)
 Applications: 0 Running, 0 Completed
 Drivers: 0 Starting, 0 Completed
 Status: ALIVE

Workers	Address	State	Cores	Memory
Worker-01	192.168.0.3:4000	ALIVE	2 (1 Used)	2.0 GB (0.0 GB Used)
Worker-02	192.168.0.4:4000	ALIVE	2 (1 Used)	2.0 GB (0.0 GB Used)

Running Applications	Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
Completed Applications	Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration

图 7. spark 管理网页

F. 安装 sbt

下载 sbt1.3.8 压缩包，将压缩包文件解压到/usr/local/sbt目录下，配置相关环境变量。按照参考资料修改/usr/local/sbt/sbt文件后，执行命令./sbt sbtVersion，等待相关依赖下载完毕即可完成配置。

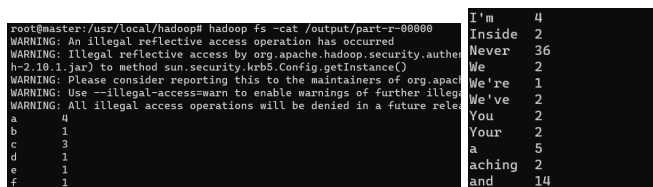
III. 运行 HADOOP 自带 WORDCOUNT 样例

为测试 Hadoop 是否正常运行，我们首先运行 Hadoop 自带的 wordcount 样例，该样例会对输入文件中不同单词进行计数。为进行测试，我首先在本地准备了一个测试文本文件input.txt，该文件包括若干由空格或换行符分隔的单字母。

启动集群之后，执行下列命令。

```
hadoop fs -mkdir /input
hadoop fs -put input.txt /input
hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.10.1.jar wordcount /input /output
hadoop fs -cat /output/part-r-00000
```

输出如图8(a)所示，之后我又将输入文件内容替换为《never gonna give you up》歌词，重新运行程序，部分输出结果如图8(b)所示。



```
root@master:/usr/local/hadoop# hadoop fs -cat /output/part-r-00000
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication-2.10.1.jar to method sun.security.krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.security.authentication-2.10.1.jar
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
a 4
b 1
c 3
d 1
e 1
f 1
I'm 4
Inside 2
Never 36
We 2
We're 1
We've 2
You 2
Your 2
a 5
aching 2
and 14
```

(a) (b)

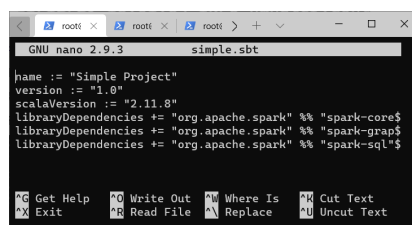
图 8. wordcount 运行结果

IV. 运行 CONNECTED COMPONENT 样例

首先从 github 上下载实验数据文件 user.txt、followers.txt 与样例程序代码 ConnectedComponentExample.scala。执行下列命令，将数据文件上传到分布式文件系统。

```
hadoop fs -put user.txt /user/root/data/graphx/
user.txt
hadoop fs -put followers.txt /user/root/data/graphx/followers.txt
```

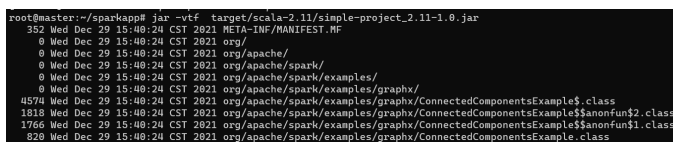
创建目录 sparkapp，该目录是我们后续编译打包项目的根目录。将样例程序代码文件复制到./src/main/scala/目录中。同时在项目根目录创建文件 simple.sbt，该文件会指定本项目的版本信息以及所依赖的库，本实验中，我们使用的 simple.sbt 内容如图9所示。



```
GNU nano 2.9.3 simple.sbt
name := "Simple Project"
version := "1.0"
scalaVersion := "2.11.8"
libraryDependencies += "org.apache.spark" %% "spark-core"
libraryDependencies += "org.apache.spark" %% "spark-graphx"
libraryDependencies += "org.apache.spark" %% "spark-sql"
```

图 9. simple.sbt 文件内容

执行命令/usr/local/sbt/sbt package，对项目进行打包，初次运行需要下载相关依赖库，等待依赖处理完毕，项目被打包到 target/scala-2.11/simple-project_2.11-1.0.jar 中。通过jar -vtf指令，可以查看 jar 文件内包含的 class，执行结果如图10所示，可以看到样例程序已经成功打包。



```
root@master:~/sparkapp# jar -vtf target/scala-2.11/simple-project_2.11-1.0.jar
352 Wed Dec 29 15:40:24 CST 2021 META-INF/MANIFEST.MF
0 Wed Dec 29 15:40:24 CST 2021 org/
0 Wed Dec 29 15:40:24 CST 2021 org/apache/
0 Wed Dec 29 15:40:24 CST 2021 org/apache/spark/
0 Wed Dec 29 15:40:24 CST 2021 org/apache/spark/examples/
0 Wed Dec 29 15:40:24 CST 2021 org/apache/spark/examples/graphx/
4574 Wed Dec 29 15:40:24 CST 2021 org/apache/spark/examples/graphx/ConnectedComponentsExample$.class
1818 Wed Dec 29 15:40:24 CST 2021 org/apache/spark/examples/graphx/ConnectedComponentsExample$.anonfun$2.class
1766 Wed Dec 29 15:40:24 CST 2021 org/apache/spark/examples/graphx/ConnectedComponentsExample$.anonfun$1.class
820 Wed Dec 29 15:40:24 CST 2021 org/apache/spark/examples/graphx/ConnectedComponentsExample.class
```

图 10. simple-project_2.11-1.0.jar 中包含的 class

将打包后文件提交到集群，提交命令如下，执行结果如图11所示。

```
/usr/local/spark/bin/spark-submit --class "
org.apache.spark.examples.graphx.
```

ConnectedComponentsExample” target/scala
-2.11/simple-project_2.11-1.0.jar

```
21/12/29 15:42:12 INFO sche
(justinbieber,1)
(matei_zaharia,3)
(ladygaga,1)
(BarackObama,1)
(jeresig,3)
(odersky,3)
```

图 11. Connected Component 样例执行结果

V. 使用 GRAPHX API 执行 PAGERANK 算法

与上一实验类似，首先下载数据集文件 wiki-Vote.txt 并将其上传到分布式文件系统。PageRank 程序可以参考 spark 官方提供的 PageRankExample.scala 样例程序实现，该实例程序给出了 pageRank 算法 API 的调用方法。我们通过修改该程序，实现实验要求的功能。

实验要求可以分为两部分实现，pageRank 计算与页面权重排序。第一部分使用与样例程序相同的方式调用 API 即可，pageRank 算法 API 会返回节点 id 及其权重的键值对，根据键值对第二个元素值降序排序后，选择前二十个成员输出即可实现实验要求。

代码打包、提交过程与上一实验类似，此处不再赘述，代码内容与执行方法可参见代码的 README 文件。提交到集群执行之后，输出如图12所示，程序输出了权重最高的 20 个节点编号及其权重。

```
21/12/29 18:06:10 INFO scheduler.DAGScheduler: Job 29 finish
21/12/29 18:06:10 INFO scheduler.TaskSetManager: Finished tas
21/12/29 18:06:10 INFO scheduler.TaskSchedulerImpl: Removed T
(4037, 13.687699331078953)
(15, 10.932666178472552)
(6634, 10.656272834035363)
(2625, 9.755493724371089)
(2398, 7.7580795368983085)
(2478, 7.4988349973549)
(2237, 7.417371235363198)
(4191, 6.737652594679844)
(7553, 6.446124253989366)
(5284, 6.387824930252768)
(2328, 6.858582278677398)
(1186, 6.047484088321153)
(1297, 5.78894778260847)
(4335, 5.753966688851237)
(7620, 5.748883767693301)
(5412, 5.7089379429240885)
(7632, 5.66775866789827)
(4875, 5.56697758206254)
(6946, 5.372689389210791)
(3352, 5.308889625474641)
21/12/29 18:06:10 INFO server.ServerConnector: Stopped Server
21/12/29 18:06:10 INFO handler.ContextHandler: Stopped o.s.j
21/12/29 18:06:10 INFO handler.ContextHandler: Stopped o.s.j
21/12/29 18:06:10 INFO handler.ContextHandler: Stopped o.s.j
21/12/29 18:06:10 INFO handler.ContextHandler: Stopped o.s.j
```

图 12. pageRank 程序执行结果

VI. 实验中遇到的问题与解决方式

在实验的过程中，常会出现参考资料未提及的意外情况。对此，我需要在网上查找资料并自行阅读部分代

码以排除故障，而这样的过程加深了我对相关技术的理解，带给我很大收获，故特地将本实验中遇到的问题与解决方法记录于此。

1) Hadoop 启动时，找不到 JAVA_HOME。

该错误由脚本文件 hadoop-env.sh 设计错误导致，脚本不能顺利读取环境变量。通过修改脚本内 JAVA_HOME 定义代码，将其改为硬编码 java 安装目录，可以解决此问题。

2) 初始化 hdfs 时，出现地址解析错误。

该问题由 java 相关库不再支持file:/前缀引起，将配置文件 core-site.xml 中相应前缀删除即可解决本问题。

3) spark-shell 无法启动，报错 Failed to initialize compiler。

该问题由 spark 与 jdk 不兼容造成，spark-shell 需要 jdk-8 才能运行。为此我安装了 jdk-8 并在环境变量 PATH 中将 jdk-8 安装目录置于默认版本 jdk 之前，之后 spark 便可在 jdk-8 中正常运行。

4) spark 管理页面不显示节点。

检查日志后发现 slave 主机无法连接到 master 主机上的管理进程，造成问题的根本原因是管理进程在错误网络接口上监听端口，将 hosts 文件中默认规则注释后问题解决。

VII. 总结

通过本实验，我学习到了 Hadoop 与 Spark 框架环境的搭建与使用，与我预期不同的是，本实验出现最多意外状况与花费最多精力的部分是运行环境的配置。在完成环境配置之后，一切都变得顺利起来，分布式框架提供了相当强大的功能，在框架的帮助下，我非常顺利的完成了后续编程实验。

在本实验之前，我对分布式框架的了解还仅限于“听说过”，我本以为将任务运行在分布式节点上需要对程序进行相当多的改动。但在完成本实验之后，我发现现在分布节点上运行任务并非如此困难，多亏分布式计算框架，我们可以快速拆分任务并执行。无疑，分布式计算框架推动了云技术的发展。

除了分布式计算框架，本实验还让我接触到了 scala 与商业云计算平台，在完成实验过程中的不断探索提升了我的综合能力。在此，我想向精心设计本次实验的老师与助教表示由衷的感谢。