Preemptive, Low Latency Datacenter Scheduling via Lightweight Virtualization 实验报告

1 实验目的

论文提出了通过轻量级虚拟化技术实现非完全抢占任务调度从而降低数据中心任务调度中的时延的方法，并且做了一些实验验证了其方法是当前最好的。本实验目的在于重现论文中的方法及其实验，从而对该论文有更好的理解。

2 论文实验概述

2.1 实验配置

- 硬件配置

26 个结点集群；每个结点上 32 个核，128G 内存；10Gbps 网络带宽；RAID-5 HDDs

- 软件配置

-Hadoop-2.7.1，Docker-1.12.1

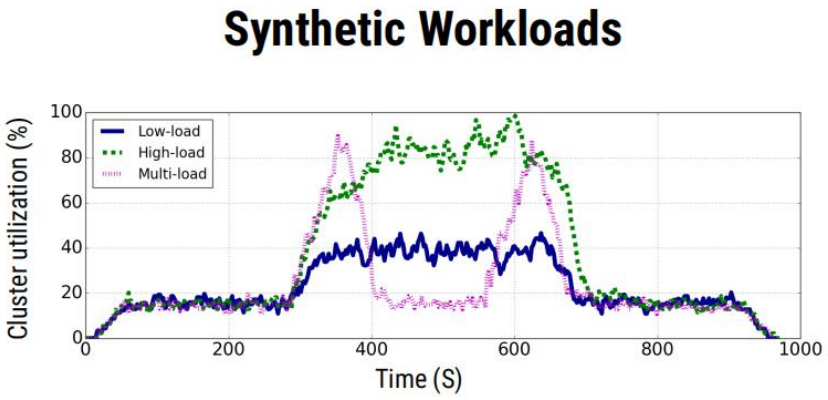
- 集群配置

2 个队列：长作业队列与短作业队列分别占 95%和 5%

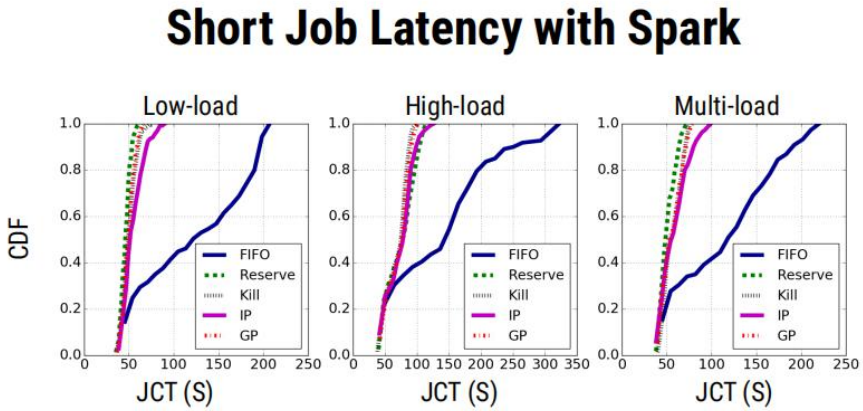
调度器：FIFO(非抢占式），预留（60%性能为短作业）,Kill,IP，GP

负载：Spark-SQL 用于短作业，HiBench 用于长作业。

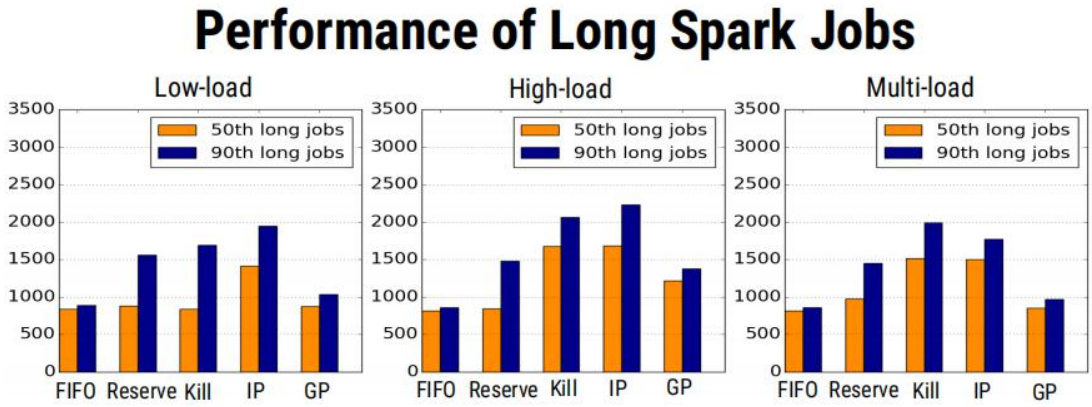
2.2 结果图示



显示了高、低并且多次的短作业；长作业持续占有集群 80%的性能。

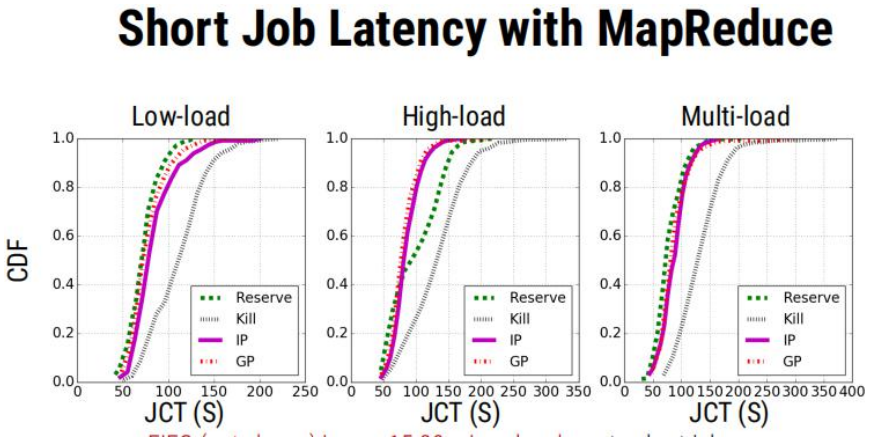


- 由于无法抢占长作业，FIFO 是最坏的方案。
- 由于高负荷下保留容量不足，储备表现不佳。
- 由于较少的资源回收时间或交换，GP 比 IP 更好。

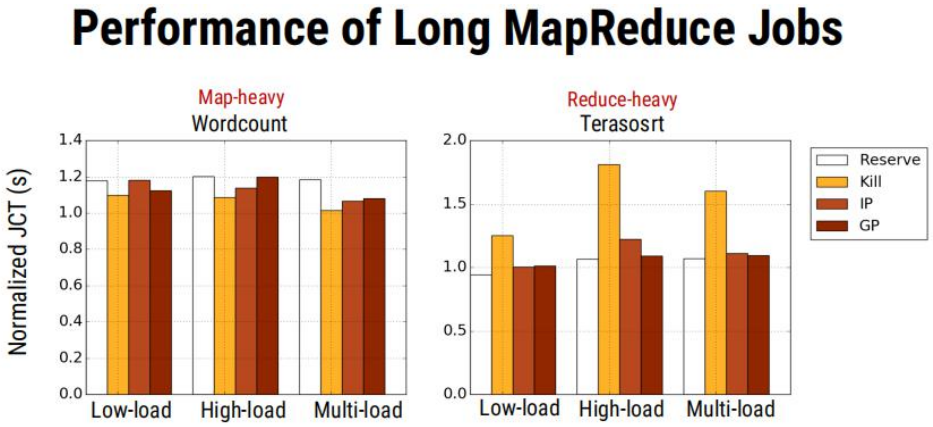


- FIFO 可以作为长作业的性能参考。

- GP 平均比杀死式的方法实现了 60%的性能提高。
- 对于 Spark jobs，IP 有很大的开销。



- 重新提交杀死的 MapReduce 作业会阻止短时间的作业。
- IP 和 GP 实现了相似的性能水平。



- 对于 map-heavy 工作负载，Kill 方式实现得很好。
- IP 和 GP 显示了对于 MapReduce 工作相似的性能水平。

2.3 实验总结

1. 数据密集型集群计算缺乏有效的任务抢占机制。
2. BIG-C 是一种简单而有效的方法来实现抢先式集群调度。

3 实验重现过程

3.1 硬件配置

VMware 虚拟机下安装 ubuntu 16.04 LTS

3.2 软件配置

3.2.1 配置 ssh 登录

```
sudo apt-get update
sudo apt-get install ssh
/etc/init.d/ssh start
```

xshell 登录虚拟机 linxu

linux 系统下 ifconfig 查看 ip 地址：

```
ens33     Link encap:Ethernet  HWaddr 00:0c:29:cf:99:9d
          inet addr:192.168.227.132  Bcast:192.168.227.255  Mask:255.255.255.
          inet6 addr: fe80::641e:56c7:d291:8143/64 Scope:Link
```

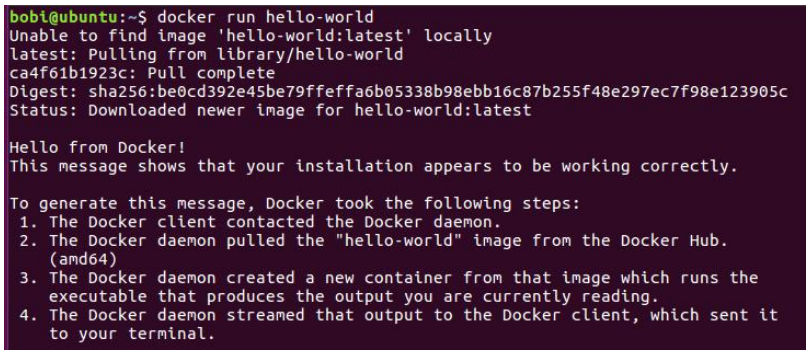
xshell 连接虚拟机

3.2.2 安装 docker

```
sudo apt-get update
sudo apt-get install apt-transport-https \
ca-certificates curl software-properties-common
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository \
  "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
       $(lsb_release -cs) stable"
sudo apt-get update
sudo apt-get install docker-ce
```

验证安装成功



3.2.3 编译实验源码 big-c-master

作者已将 big-c-master 源码发布在 github 上，该项目是一个 hadoop 源码，其中也给出了 build.txt 作为编译源码所需要的环境，下面搭建所需环境。

build.txt 环境要求：

```
* Unix System
* JDK 1.7+
* Maven 3.0 or later
* Findbugs 1.3.9 (if running findbugs)
* ProtocolBuffer 2.5.0
* CMake 2.6 or newer (if compiling native code), must be 3.0 or newer on Mac
* Zlib devel (if compiling native code)
* openssl devel ( if compiling native hadoop-pipes and to get the best HDFS encryption performance )
* Jansson C XML parsing library ( if compiling libwebhdfs )
* Linux FUSE (Filesystem in Userspace) version 2.6 or above ( if compiling
```

fuse_dfs)

* Internet connection for first build (to fetch all Maven and Hadoop dependencies)

1. 配置 java 环境

官网上已经没有 java1.7 了，故文档中用 apt-get 方式安装 jdk7 不能用，于是自己下载了 java1.8,然后配置路径。

在/usr/local 目录下创建 java 文件夹。

mkdir java

tar -xzvf jdk-8u151-linux-x64.tar.gz

vim /etc/profile

```
#set java environment
export JAVA_HOME=/usr/local/java/jdk1.8.0_151
export JRE_HOME=${JAVA_HOME}/jre
export CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib
export PATH=${JAVA_HOME}/bin:$PATH
```

Java -version

```
bobi@ubuntu:/usr/local/java$ java -version
java version "1.8.0_151"
Java(TM) SE Runtime Environment (build 1.8.0_151-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, mixed mode)
```

2. 配置 Maven

同样，论文构建文件中的方法不可取，于是便自己下载 Maven 安装包,版本选择为 maven-3.5.2。

在/usr/local 文件夹下解压压缩包。

tar -xzvf apache-maven-3.5.2-bin.tar.gz

vim /etc/profile

```
#set maven environment
export MAVEN_HOME=/usr/local/apache-maven-3.5.2
export PATH=${MAVEN_HOME}/bin:$PATH
```

Source /etc/profile

```
bobi@ubuntu:/usr/local$ mvn -version
Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T00:58:13-07:00)
Maven home: /usr/local/apache-maven-3.5.2
Java version: 1.8.0_151, vendor: Oracle Corporation
Java home: /usr/local/java/jdk1.8.0_151/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "4.10.0-38-generic", arch: "amd64", family: "unix"
```

3.配置 ProtocolBuffer 2.5.0

查看网上资料说，直接用 sudo apt-get 安装的是最新的 3.0，这在后面安装的

时候冲突，于是自己下载了 2.5 的安装包。

```
tar -xzvf protobuf-2.5.0.tar.gz
cd protobuf-2.5.0
make&&make install
```

然后 protoc -version 出现了一个问题 protoc: error while loading shared libraries: libprotoc.so.8: cannot open shared

解决办法：

创建文件/etc/ld.so.conf.d/libprotobuf.conf

添加内容

/usr/local/lib

然后

sudo ldconfig

问题解决

protoc --version

```
bobi@ubuntu:~$ protoc --version
libprotoc 2.5.0
```

4.安装本地必要库

```
sudo apt-get -y install build-essential autoconf automake libtool cmake zlib1g-dev pkg-config libssl-dev
```

```
autoconf is already the newest version (2.69-9).
automake is already the newest version (1:1.15-4ubuntu1).
build-essential is already the newest version (12.1ubuntu2).
libtool is already the newest version (2.4.6-0.1).
pkg-config is already the newest version (0.29.1-0ubuntu1).
cmake is already the newest version (3.5.1-1ubuntu3).
libssl-dev is already the newest version (1.0.2g-1ubuntu4.9).
zlib1g-dev is already the newest version (1:1.2.8.dfsg-2ubuntu4.1).
```

由以上，编译源码环境已经全部搭建好，然后编译 big-c-master。

```
mvn package -Pdist,native -DskipTests  –Dtar
```

经过 1 个小时的编译，终于失败了。

```
[INFO] Apache Hadoop Main ................................ SUCCESS [01:11 min]
[INFO] Apache Hadoop Project POM ......................... SUCCESS [ 33.429 s]
[INFO] Apache Hadoop Annotations ......................... SUCCESS [ 31.011 s]
[INFO] Apache Hadoop Assemblies .......................... SUCCESS [  0.144 s]
[INFO] Apache Hadoop Project Dist POM .................... SUCCESS [ 20.781 s]
[INFO] Apache Hadoop Maven Plugins ....................... SUCCESS [ 25.564 s]
[INFO] Apache Hadoop MiniKDC ............................. SUCCESS [04:30 min]
[INFO] Apache Hadoop Auth ................................ SUCCESS [08:05 min]
[INFO] Apache Hadoop Auth Examples ....................... SUCCESS [  9.526 s]
[INFO] Apache Hadoop Common .............................. FAILURE [54:43 min]
```

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 01:10 h
[INFO] Finished at: 2017-11-20T04:28:30-08:00
[INFO] Final Memory: 61M/433M
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.17:test (default-test) on project hadoop-common: Ther
e are test failures.
[ERROR]
[ERROR] Please refer to /home/bobi/big-c-master/hadoop-common-project/hadoop-common/target/surefire-reports for the individual test r
esults.
[ERROR] -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException
[ERROR]
[ERROR] After correcting the problems, you can resume the build with the command
[ERROR]   mvn <goals> -rf :hadoop-common
bobi@ubuntu:~/big-c-master$
```

为了判断是否是我环境搭建有问题，我又下载了官方的 hadoop 源码，版本为 hadoop-2.7.4，因为官网上没看到有 2.7.1 的版本了，而查看编译环境也是一样的，编译官方源码，幸运地编译成功了。

```
[INFO] Apache Hadoop Rumen ............................... SUCCESS [  3.240 s]
[INFO] Apache Hadoop Gridmix ............................. SUCCESS [  2.618 s]
[INFO] Apache Hadoop Data Join ........................... SUCCESS [  1.565 s]
[INFO] Apache Hadoop Ant Tasks ........................... SUCCESS [  1.912 s]
[INFO] Apache Hadoop Extras .............................. SUCCESS [  1.658 s]
[INFO] Apache Hadoop Pipes ............................... SUCCESS [  7.183 s]
[INFO] Apache Hadoop OpenStack support ................... SUCCESS [  2.955 s]
[INFO] Apache Hadoop Amazon Web Services support ......... SUCCESS [01:17 min]
[INFO] Apache Hadoop Azure support ....................... SUCCESS [ 11.210 s]
[INFO] Apache Hadoop Client .............................. SUCCESS [  6.139 s]
[INFO] Apache Hadoop Mini-Cluster ........................ SUCCESS [  0.961 s]
[INFO] Apache Hadoop Scheduler Load Simulator ............ SUCCESS [  3.353 s]
[INFO] Apache Hadoop Tools Dist .......................... SUCCESS [  5.546 s]
[INFO] Apache Hadoop Tools ............................... SUCCESS [  0.023 s]
[INFO] Apache Hadoop Distribution ........................ SUCCESS [ 31.439 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 12:39 min
[INFO] Finished at: 2017-11-20T05:50:27-08:00
[INFO] Final Memory: 221M/572M
[INFO] ------------------------------------------------------------------------
bobi@ubuntu:~/hadoop-2.7.4-src$
```

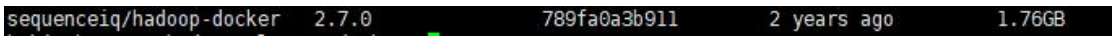
这么就很纠结了，感觉不能还原实验了，便尝试用 docker 搭建一下 hadoop 集群环境。

3.2.4 搭建 hadoop 集群

我开始准备尝试从基本的 ubuntu 系统配置 hadoop 环境，首先从官方 docker 镜像中 pull 下 ubuntu 镜像，然后在该镜像中配置 java 环境，配置 ssh 无密码登录。然后配置 hadoop 集群文件，在配置 hadoop 集群中，要修改 slave 文件及 hadoop-env.sh,core-site.xml,mapred-site.xml,hdfs-site.xml 等文件。然后由该镜像生

成 namenode 和 datanode。最后配置网络 ip。

这个过程，又要重新配置一次环境，我个人感觉与之前许多工作冗余。而论文中使用容器搭建结点是用的官方的镜像，于是我也尝试使用如下图镜像，然后修改其配置文件。



主要步骤为：

修改主机名

vi /etc/hostname

创立文件夹

mkdir tmp //hadoop 临时目录

mkdir datanode //datanode 的存放目录

mkdir namenode //namenode 存放目录

修改主结点为 master,分结点为 slave1,slave2

Hadoop-env.sh 其中 java 和 hadoop 路径已是配置好，不用修改

1)Core-site.xml 中

```
<configuration>
    <property>
        <name>hadoop.tmp.dir</name>
        <value>/usr/local/hadoop-2.7.0/tmp</value>
    </property>

    <property>
        <name>fs.default.name</name>
        <value>hdfs://master:9000</value>
    </property>
</configuration>
```

2) hdfs-site.xml

```
<configuration>
 <property>
```

```
        <name>dfs.replication</name>
        <value>2</value>
    </property>

    <property>
        <name>dfs.namenode.name.dir</name>
        <value>/usr/local/hadoop-2.7.0/namenode</value>
    </property>

    <property>
        <name>dfs.datanode.data.dir</name>
        <value>/usr/local/hadoop-2.7.0/datanode</value>
    </property>
</configuration>
```

3) mapred-site.xml 配置

```
<configuration>
    <property>
        <name>mapred.job.tracker</name>
        <value>master:9001</value>
    </property>
</configuration>
```

然后将配置好的 hadoop 文件分别替换掉原有镜像的 hadoop 文件,重启三个容器

执行 sbin/start-all.sh，后执行 jps

```
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn--resourcemanager-53e9ce5bcc5d.out
slave2: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-root-nodemanager-2bf5af6c86d0.out
slave1: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-root-nodemanager-cdb421047734.out
bash-4.1# jps
127 NameNode
247 DataNode
4753 Jps
407 SecondaryNameNode
658 NodeManager
```

```
bash-4.1# pwd
/usr/local/hadoop-2.7.0
bash-4.1# jps
559 ResourceManager
1432 Jps
407 SecondaryNameNode
bash-4.1# ls
```

好吧，依旧失败了，主结点中仍然运行有 datanode 进程，而数据结点没有 datanode 进程，说明仍然是单机伪分布式结构。

最后，使用 KiwenLau 的基于 docker 搭建 hadoop 的项目博客，他已将 Hadoop 环境配置好，只需要运行脚本文件就行，使用他的 docker 镜像搭建了一个 3 机 hadoop 并成功，过程如下。

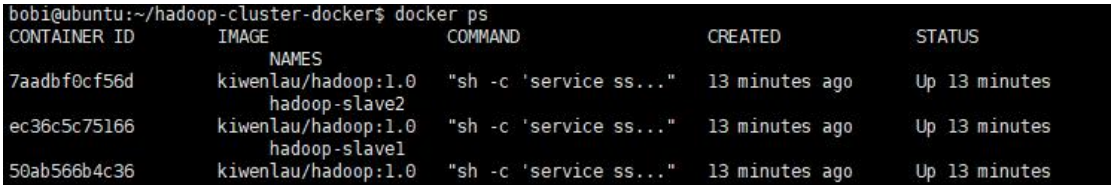
```
sudo docker pull kiwenlau/hadoop:1.0
git clone https://github.com/kiwenlau/hadoop-cluster-docker
sudo docker network create --driver=bridge hadoop
cd hadoop-cluster-docker
```

```
bobi@ubuntu:~/hadoop-cluster-docker$ ./start-container.sh
start hadoop-master container...
start hadoop-slave1 container...
start hadoop-slave2 container...
```

```
root@hadoop-master:~# ./start-hadoop.sh


Starting namenodes on [hadoop-master]
hadoop-master: Warning: Permanently added 'hadoop-master,172.18.0.2' (ECDSA) to the list of known hosts.
hadoop-master: starting namenode, logging to /usr/local/hadoop/logs/hadoop-root-namenode-hadoop-master.out
hadoop-slave2: Warning: Permanently added 'hadoop-slave2,172.18.0.4' (ECDSA) to the list of known hosts.
hadoop-slave1: Warning: Permanently added 'hadoop-slave1,172.18.0.3' (ECDSA) to the list of known hosts.
hadoop-slave2: starting datanode, logging to /usr/local/hadoop/logs/hadoop-root-datanode-hadoop-slave2.out
hadoop-slave1: starting datanode, logging to /usr/local/hadoop/logs/hadoop-root-datanode-hadoop-slave1.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: Warning: Permanently added '0.0.0.0' (ECDSA) to the list of known hosts.
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-root-secondarynamenode-hadoop-master.out


starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn--resourcemanager-hadoop-master.out
hadoop-slave1: Warning: Permanently added 'hadoop-slave1,172.18.0.3' (ECDSA) to the list of known hosts.
hadoop-slave2: Warning: Permanently added 'hadoop-slave2,172.18.0.4' (ECDSA) to the list of known hosts.
hadoop-slave1: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-root-nodemanager-hadoop-slave1.out
hadoop-slave2: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-root-nodemanager-hadoop-slave2.out
```

运行自带的 wordcount 程序：

```
input file1.txt:
Hello Hadoop

input file2.txt:
Hello Docker

wordcount output:
Docker  1
Hadoop  1
Hello   2
```



则一个三节点的 hadoop 集群搭建成功。

4 实验心得及总结

本次实验并没有很好的完成，但从中我还是能学到很多东西，学会了搭建数据中心集群，也磨炼了我的耐心，希望以后好好学习，能更好地完成实验。