电子科技大学

课　程　报　告

词频统计

学　　院：信息与软件工程学院

学生姓名：杨庆

学　　号：201822090316

指导教师：林迪

1 实验目的

（1）理解 MapReduce、等 Spark 思想；

（2）了解大数据整个体系框架；

（3）综合运用所学知识，编写相关项目案例 WordCount 词频统计

2 实验内容

（1）安装部署 Hadoop、HDFS、MapReduce、Spark；

（2）分别在 MapReduce 及 Spark 执行词频统计；

（3）比较 MapReduce 及 Spark 执行效率异同；

（4）从软件体系架构角度解释分析实验结果。

3 Hadoop 介绍

3.1 Hadoop 产生背景

Hadoop 最早起源于 Nutch。Nutch 的设计目标是构建一个大型的全网搜索引擎，包括网页抓取、索引、查询等功能，但随着抓取网页数量的增加，遇到了严重的可扩展性问题——如何解决数十亿网页的存储和索引问题。

2003 年、2004 年谷歌发表的两篇论文为该问题提供了可行的解决方案。

（1）分布式文件系统（GFS），可用于处理海量网页的存储

（2）分布式计算框架 MapReduce，可用于处理海量网页的索引计算问题。

Nutch 的开发人员完成了相应的开源实现 HDFS 和 MapReduce，并从 Nutch 中剥离成为独立项目 Hadoop，到 2008 年 1 月，Hadoop 成为 Apache 顶级项目，迎来了它的快速发展期。

3.2 Hadoop 简介

Hadoop 是 Apache 旗下的一套开源软件平台，Hadoop 是利用服务器集群对数据进行存储，根据用户的自定义业务逻辑，对海量数据进行分布式计算。广义上来说，Hadoop 通常是指一个更广泛的概念——Hadoop 生态圈。

Hadoop 解决了海量数据的存储（HDFS）、海量数据的技术（MapReduce）、资源调度（YARN）等问题。其中重点组件包括：

（1）HDFS：分布式文件系统；

（2）MapReduce：分布式运算程序开发框架；

（3）YARN:资源调度系统。

（4）ZOOKEEPER：分布式协调服务基础组件；

（5）HIVE：SQL 数据仓库工具；

（6）HBASE：基于 Hadoop 的分布式海量数据库；

（7）Sqoop：数据迁移工具；

（8）Flume：日志数据采集框架；

3. 3　Hadoop 架构

3. 3. 1　分布式架构简介

1.单机的问题

（1）存储能力有限；

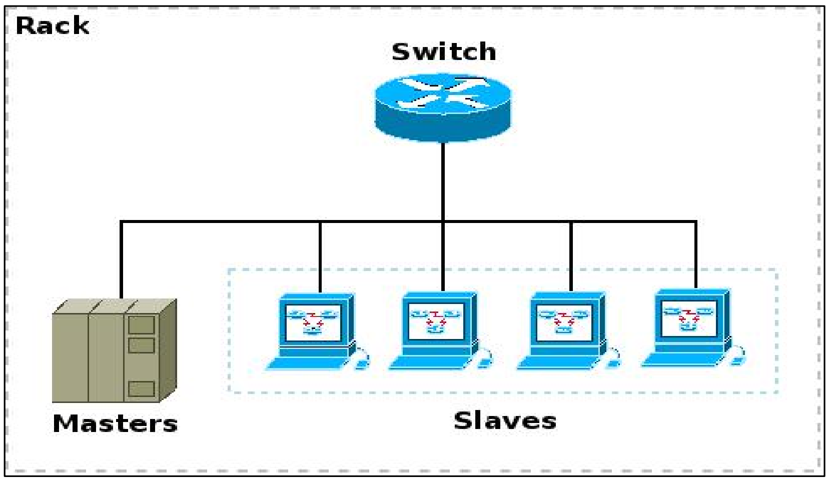
（2）计算能力有限；

（3）有单点故障等。

2.分布式架构解决了单机的问题

3.经典分布式主从架构（Master-Slave）

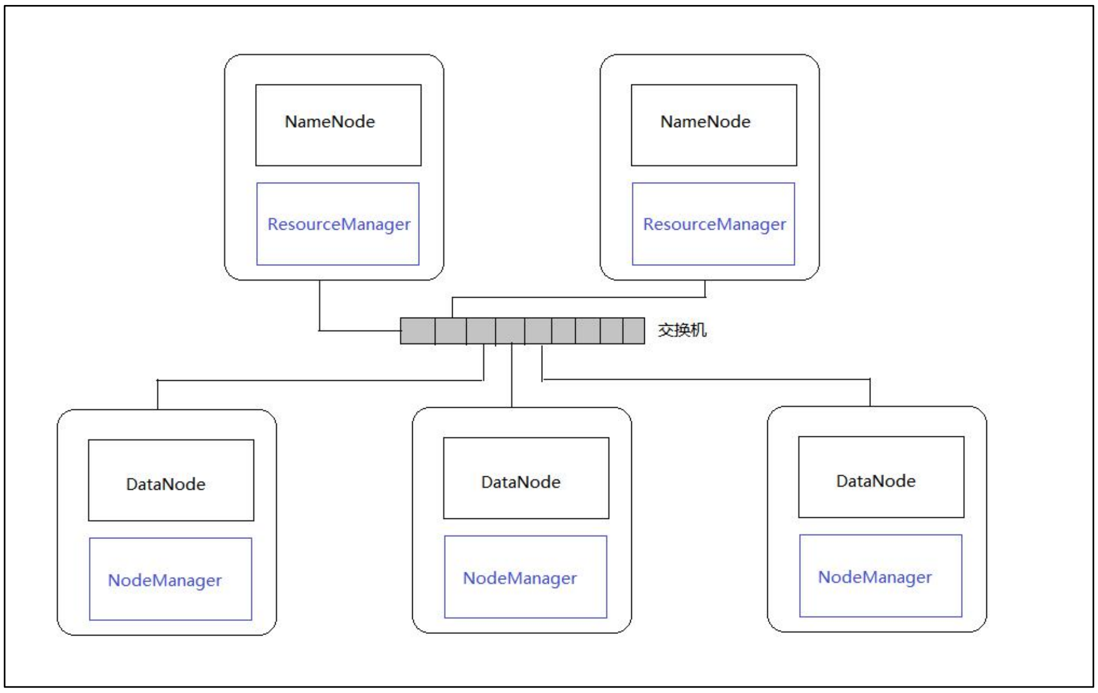
4.Master 负责管理，且可以有多个，防止单点故障的发生。Slave 负责干活，Slave 有多个，并且可以动态的添加或移除。



3.3.2 Hadoop2.0

（1）HDFS ：NameNode（老大） DataNode（小弟）

（2）YARN ：ResourceManager（老大） NodeManager（小组长）



3.3.3 伪分布式架构

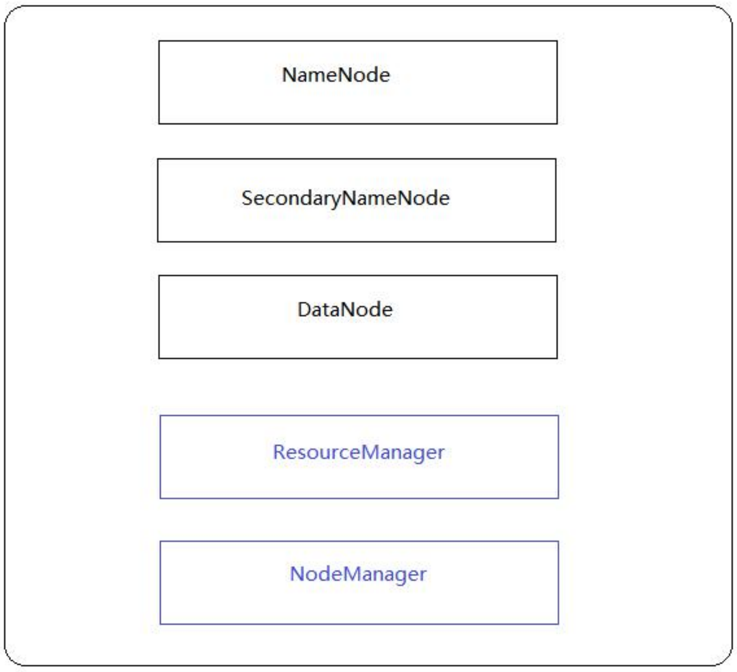
（1）NameNode：HDFS 的管理节点，负责 DataNode 的管理和元数据管理；

（2）SecondaryNameNode：NameNode 的一个助理，帮助 NameNode 管理元数据，防止元数据丢失；

（3）DataNode：负责数据存储；

（4）ResourceManager：YARN 的管理节点，负责 NodeManager 的管理、任务调度等；

（5）NodeManager：YARN 的节点管理器，负责向 ResourceManager 汇报当前节点的状态和启动计算任务进程（YarnChild）并监控 YarnChild。



4　实验过程

4.1　ubuntu16.04 配置 Hadoop 伪分布式

4.1.1 实验环境

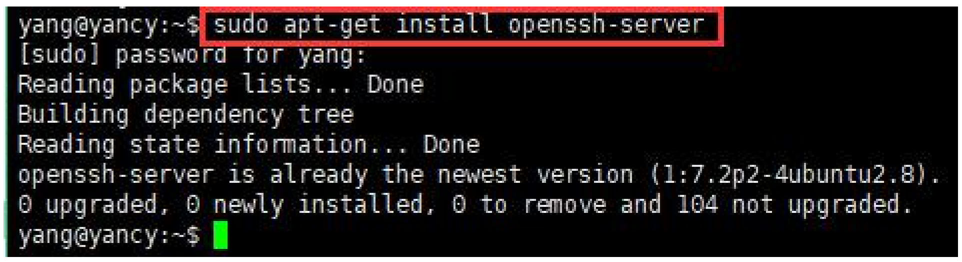
（1）操作系统：Ubuntu16.04

（2）Java 环境：jdk1.8.0_181

（3）Hadoop 版本：hadoop-2.7.6

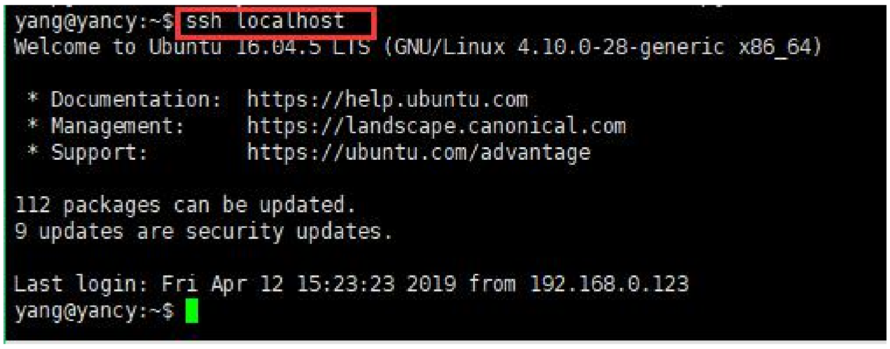
4.1.2 SSH 免密码登录

（1）输入：sudo apt-get install openssh-server，安装 SSH server；



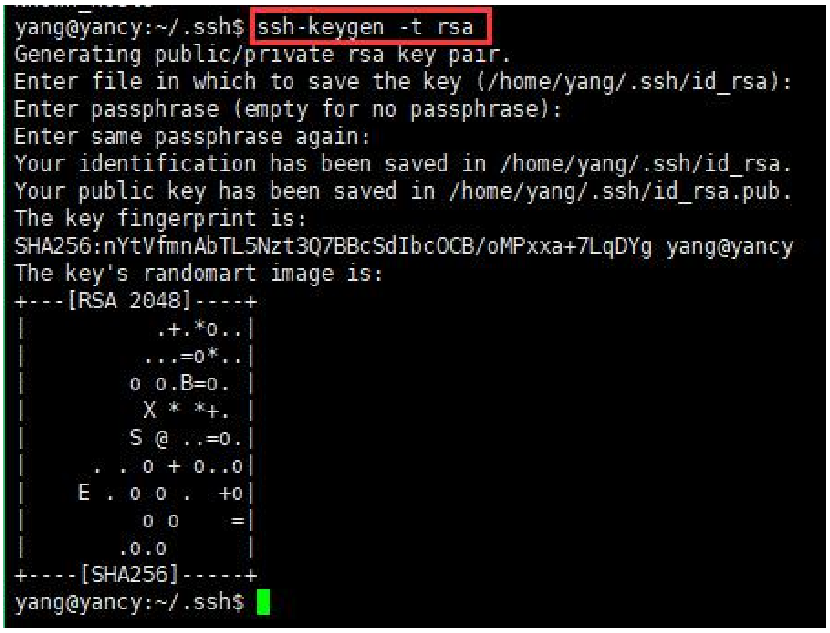
```
yang@yancy:~$ sudo apt-get install openssh-server
[sudo] password for yang:
Reading package lists... Done
Building dependency tree
Reading state information... Done
openssh-server is already the newest version (1:7.2p2-4ubuntu2.8).
0 upgraded, 0 newly installed, 0 to remove and 104 not upgraded.
yang@yancy:~$
```

（2）输入：cd ~/.ssh/，如果没法进入该目录，执行一次 ssh localhost；



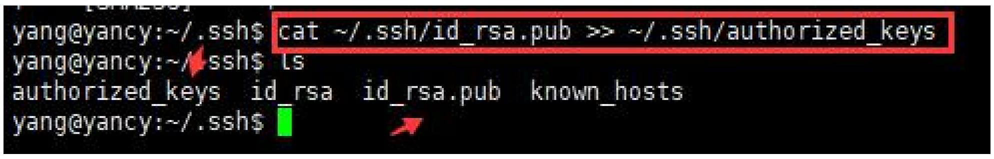
```
yang@yancy:~$ ssh localhost
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.10.0-28-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

112 packages can be updated.
9 updates are security updates.

Last login: Fri Apr 12 15:23:23 2019 from 192.168.0.123
yang@yancy:~$
```

（3）输入：ssh-keygen -t rsa，三次回车后，该目录下将会产生 id_rsa，id_rsa.pub 文件；



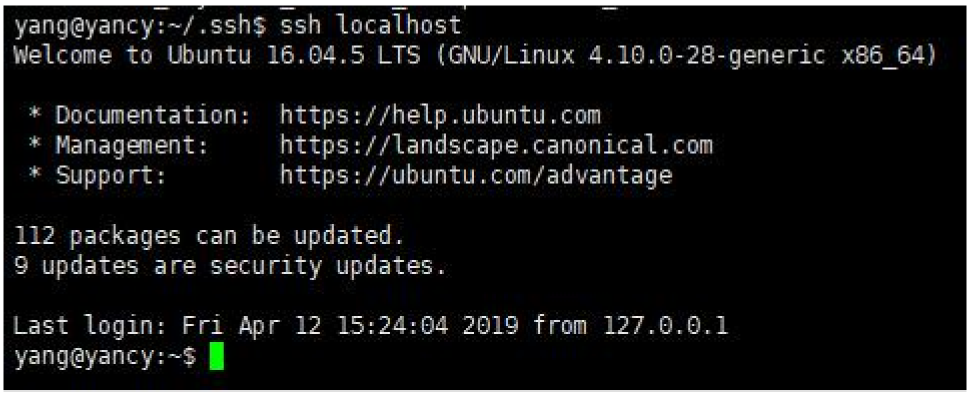
```
yang@yancy:~/.ssh$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/yang/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/yang/.ssh/id_rsa.
Your public key has been saved in /home/yang/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:nYtVfmnAbTL5Nzt3Q7BBcSdIbcOCB/oMPxxa+7LqDYg yang@yancy
The key's randomart image is:
+---[RSA 2048]----+
|          .+.*o..|
|         ...=o*..|
|        o o.B=o. |
|         X * *+. |
|        S @ ..=o.|
|     . . o + o..o|
|    E . o o .  +o|
|         o o    =|
|       .o.o      |
+----[SHA256]-----+
yang@yancy:~/.ssh$
```

（4）输入：cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys 加入授权；



```
yang@yancy:~/.ssh$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
yang@yancy:~/.ssh$ ls
authorized_keys  id_rsa  id_rsa.pub  known_hosts
yang@yancy:~/.ssh$
```

（5）输入：ssh localhost，如果不提示输入密码则 SSH 无密登陆配置成功；



```
yang@yancy:~/.ssh$ ssh localhost
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.10.0-28-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

112 packages can be updated.
9 updates are security updates.

Last login: Fri Apr 12 15:24:04 2019 from 127.0.0.1
yang@yancy:~$
```

4.1.3 安装 java1.8.0_181

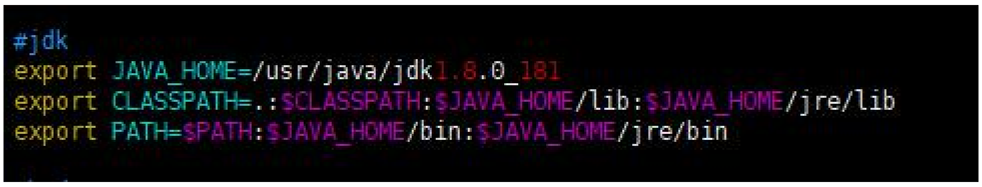
（1）https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html 下载 jdk-8u181-linux-x64.tar.gz，输入：sudo tar zxvf jdk-8u201-linux-x64.tar.gz -C /usr/java/jdk1.8.0_181 将 jdk-8u181-linux-x64.tar.gz 解压到/usr/java/目录下；

（2）输入：sudo vim ~/.bashrc 配置环境变量，在最后添加三行：

export JAVA_HOME=/usr/java/jdk1.8.0_181

export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JAVA_HOME/jre/lib

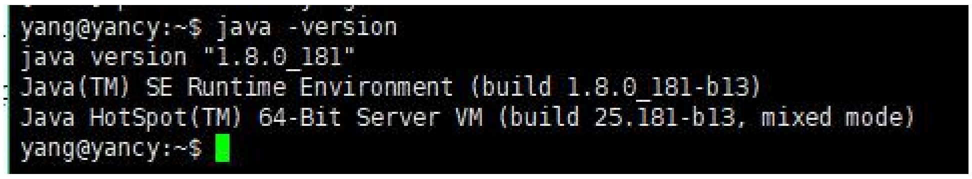
export PATH=$PATH:$JAVA_HOME/bin:$JAVA_HOME/jre/bin



```
#jdk
export JAVA_HOME=/usr/java/jdk1.8.0_181
export CLASSPATH=.:$CLASSPATH:$JAVA_HOME/lib:$JAVA_HOME/jre/lib
export PATH=$PATH:$JAVA_HOME/bin:$JAVA_HOME/jre/bin
```

（3）输入：source ~/.bashrc，使新配置的环境变量生效；

（4）输入：java -version，查看 Java 版本，检测是否安装成功；



```
yang@yancy:~$ java -version
java version "1.8.0_181"
Java(TM) SE Runtime Environment (build 1.8.0_181-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.181-b13, mixed mode)
yang@yancy:~$
```

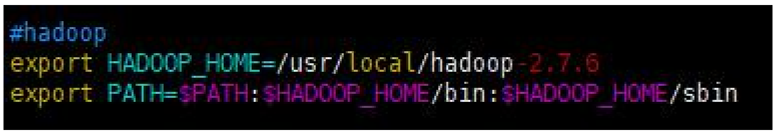
4.1.4 安装 hadoop-2.7.6

（1）在 https://mirrors.tuna.tsinghua.edu.cn/apache/hadoop/common/ 下载 hadoop-2.7.6.tar.gz，输入：sudo tar zxvf hadoop-2.7.6.tar.gz -C /usr/local/hadoop-2.7.6 将 hadoop-2.7.6.tar.gz 解压到/usr/local/目录下；

（2）输入：sudo vim ~/.bashrc 添加如下两行，然后输入 source ~./bashrc；

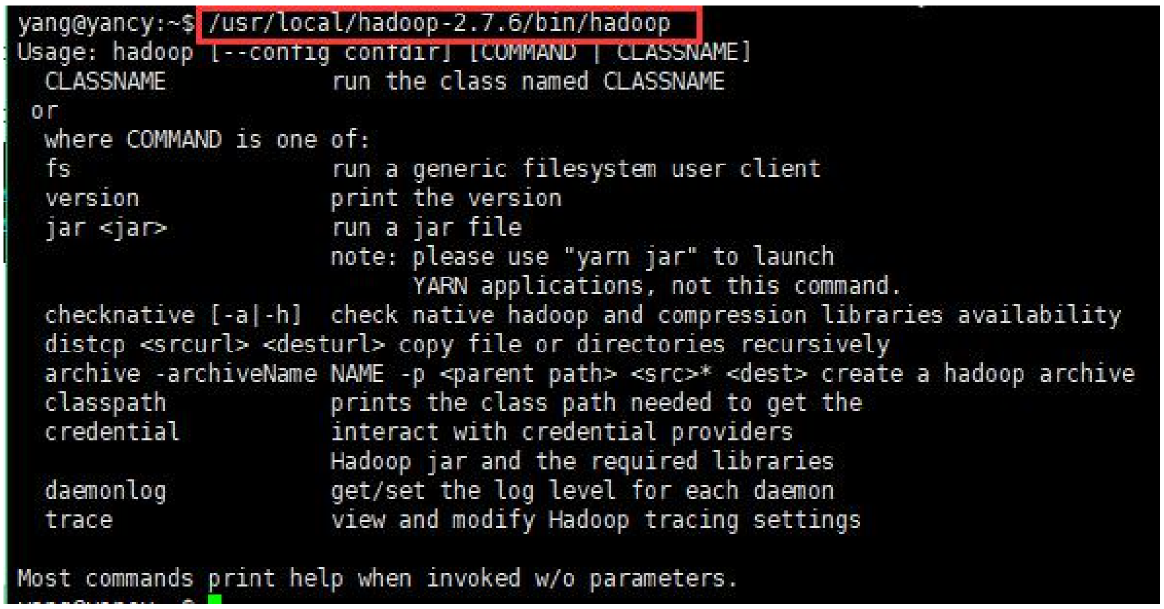
export HADOOP_HOME=/usr/local/hadoop-2.7.6

export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin



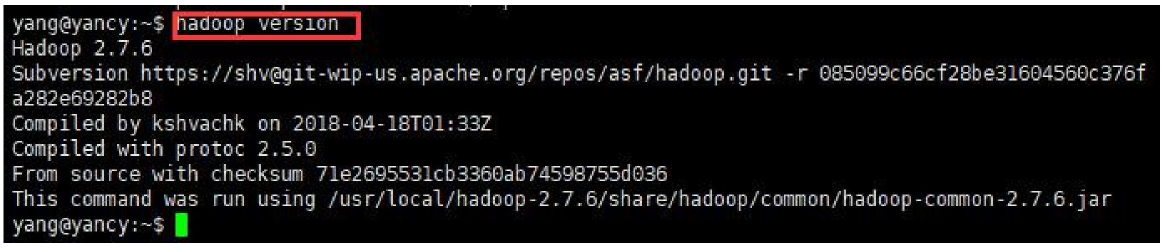
```
#hadoop
export HADOOP_HOME=/usr/local/hadoop-2.7.6
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

（3）输入：/usr/local/hadoop-2.7.6/bin/hadoop 查看 hadoop 命令相关使用信息；



```
yang@yancy:~$ /usr/local/hadoop-2.7.6/bin/hadoop
Usage: hadoop [--config confdir] [COMMAND | CLASSNAME]
  CLASSNAME            run the class named CLASSNAME
 or
  where COMMAND is one of:
  fs                   run a generic filesystem user client
  version              print the version
  jar <jar>            run a jar file
                       note: please use "yarn jar" to launch
                             YARN applications, not this command.
  checknative [-a|-h]  check native hadoop and compression libraries availability
  distcp <srcurl> <desturl> copy file or directories recursively
  archive -archiveName NAME -p <parent path> <src>* <dest> create a hadoop archive
  classpath            prints the class path needed to get the
  credential           interact with credential providers
                       Hadoop jar and the required libraries
  daemonlog            get/set the log level for each daemon
  trace                view and modify Hadoop tracing settings

Most commands print help when invoked w/o parameters.
```

（4）输入：hadoop version 查看版本。



```
yang@yancy:~$ hadoop version
Hadoop 2.7.6
Subversion https://shv@git-wip-us.apache.org/repos/asf/hadoop.git -r 085099c66cf28be31604560c376f
a282e69282b8
Compiled by kshvachk on 2018-04-18T01:33Z
Compiled with protoc 2.5.0
From source with checksum 71e2695531cb3360ab74598755d036
This command was run using /usr/local/hadoop-2.7.6/share/hadoop/common/hadoop-common-2.7.6.jar
yang@yancy:~$
```

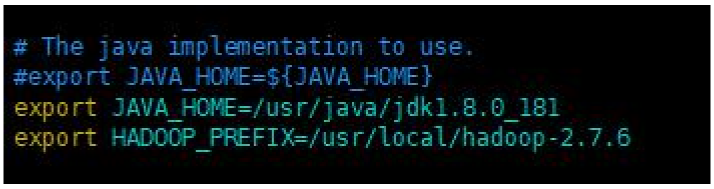
4.1.5 伪分布式配置

Hadoop 可以在单节点上以伪分布式的方式运行，Hadoop 进程以分离的 Java 进程来运行，节点既作为 NameNode 也作为 DataNode，同时，读取的是 HDFS 中的文件。Hadoop 的配置文件位于 /usr/local/hadoop-2.7.6/etc/hadoop/ 中，伪分布式需要修改 2 个配置文件 core-site.xml 和 hdfs-site.xml 。Hadoop 的配置文件是 xml 格式，每个配置以声明 property 的 name 和 value 的方式来实现。此处我们另外修改了配置文件。

（1）JAVA_HOME 位于/usr/java/jdk1.8.0_181，Hadoop 在/usr/local/hadoop-2.7.6。输入：sudo vim /usr/local/hadoop-2.7.6/etc/hadoop/hadoop-env.sh 添加两行参数：

export JAVA_HOME=/usr/java/jdk1.8.0_181

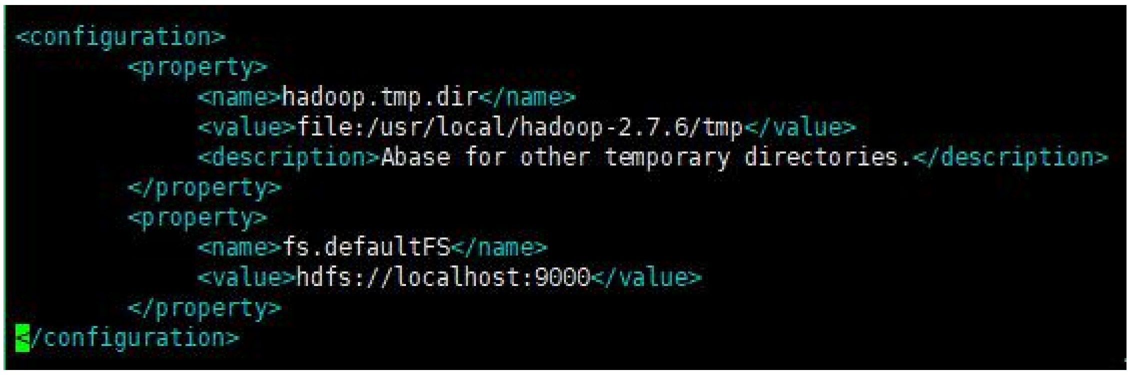
export HADOOP_PREFIX=/usr/local/hadoop-2.7.6



```
# The java implementation to use.
#export JAVA_HOME=${JAVA_HOME}
export JAVA_HOME=/usr/java/jdk1.8.0_181
export HADOOP_PREFIX=/usr/local/hadoop-2.7.6
```

（2）输入：sudo vim /usr/local/hadoop-2.7.6/etc/hadoop/core-site.xml 修改 core-site.xml 文件添加如下内容：

```
<configuration>
        <!-- 配置 Hadoop 运行时产生数据的存储目录，不是临时的数据 -->
        <property>
              <name>hadoop.tmp.dir</name>
              <value>file:/usr/local/hadoop-2.7.6/tmp</value>
              <description>Abase for other temporary directories.</description>
        </property>
        <!-- 配置 hdfs 的 Namenode（老大）的地址 -->
        <property>
              <name>fs.defaultFS</name>
              <value>hdfs://localhost:9000</value>
        </property>
</configuration>
```

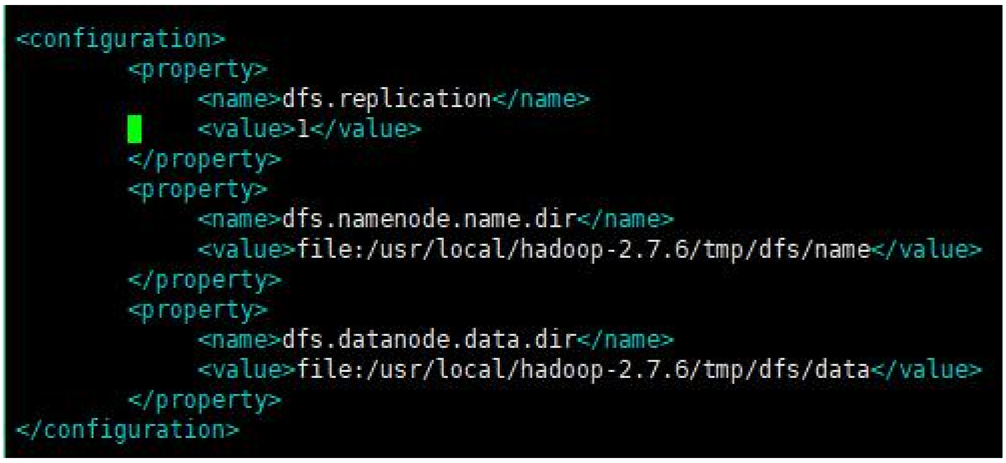


```
<configuration>
        <property>
             <name>hadoop.tmp.dir</name>
             <value>file:/usr/local/hadoop-2.7.6/tmp</value>
             <description>Abase for other temporary directories.</description>
        </property>
        <property>
             <name>fs.defaultFS</name>
             <value>hdfs://localhost:9000</value>
        </property>
</configuration>
```

（3）输入：sudo vim /usr/local/hadoop-2.7.6/etc/hadoop/hdfs-site.xml 修改配置文件 hdfs-site.xml 添加如下内容：

```
<configuration>
        <!-- 指定 HDFS 存储数据的副本数量 -->
        <property>
              <name>dfs.replication</name>
              <value>1</value>
        </property>
        <property>
```

```
            <name>dfs.namenode.name.dir</name>
            <value>file:/usr/local/hadoop-2.7.6/tmp/dfs/name</value>
        </property>
        <property>
            <name>dfs.datanode.data.dir</name>
            <value>file:/usr/local/hadoop-2.7.6/tmp/dfs/data</value>
        </property>
</configuration>
```



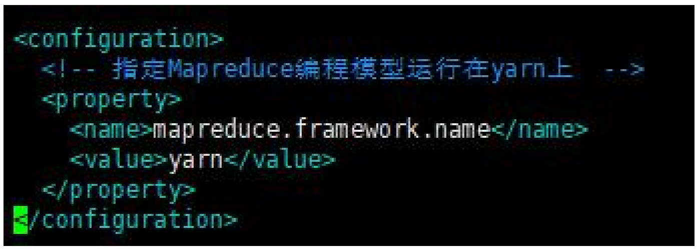
```
<configuration>
        <property>
             <name>dfs.replication</name>
             <value>1</value>
        </property>
        <property>
             <name>dfs.namenode.name.dir</name>
             <value>file:/usr/local/hadoop-2.7.6/tmp/dfs/name</value>
        </property>
        <property>
             <name>dfs.datanode.data.dir</name>
             <value>file:/usr/local/hadoop-2.7.6/tmp/dfs/data</value>
        </property>
</configuration>
```

（4）将 mapred-site.xml.template 重命名为 mapred-site.xml，

输入：sudo mv mapred-site.xml.template mapred-site.xml

然后输入：sudo vim /usr/local/hadoop-2.7.6/etc/hadoop/mapred-site.xml 修改配置文件 mapred-site.xml 添加如下内容：

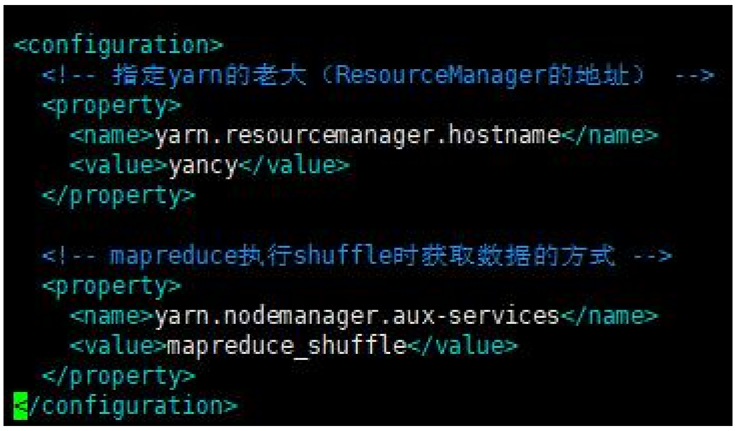
```
<configuration>
  <!--  指定 Mapreduce 编程模型运行在 yarn 上    -->
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```



```
<configuration>
  <!-- 指定Mapreduce编程模型运行在yarn上  -->
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

（5）输入：sudo vim /usr/local/hadoop-2.7.6/etc/hadoop/yarn-site.xml 修改配置文件 yarn-site.xml 添加如下内容：

```
<configuration>
  <!--  指定 yarn 的老大（ResourceManager 的地址）  -->
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>yancy</value>
  </property>

  <!-- mapreduce 执行 shuffle 时获取数据的方式  -->
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
```



```
<configuration>
  <!-- 指定yarn的老大（ResourceManager的地址） -->
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>yancy</value>
  </property>

  <!-- mapreduce执行shuffle时获取数据的方式 -->
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
```

4.1.6 运行 Hadoop

Hadoop 的运行方式是由配置文件决定的（运行 Hadoop 时会读取配置文件），因此如果需要从伪分布式模式切换回非分布式模式，需要删除 core-site.xml 中的配

置项。此外，伪分布式虽然只需要配置 fs.defaultFS 和 dfs.replication 就可以运行（可参考官方教程），不过若没有配置 hadoop.tmp.dir 参数，则默认使用的临时目录为 /tmp/hadoo-hadoop，而这个目录在重启时有可能被系统清理掉，导致必须重新执行 format 才行。所以我们进行了设置，同时也指定 dfs.namenode.name.dir 和 dfs.datanode.data.dir，否则在接下来的步骤中可能会出错。

在 Hadoop 安装包目录下有几个比较重要的目录：

1）sbin：启动或停止 Hadoop 相关服务的脚本；

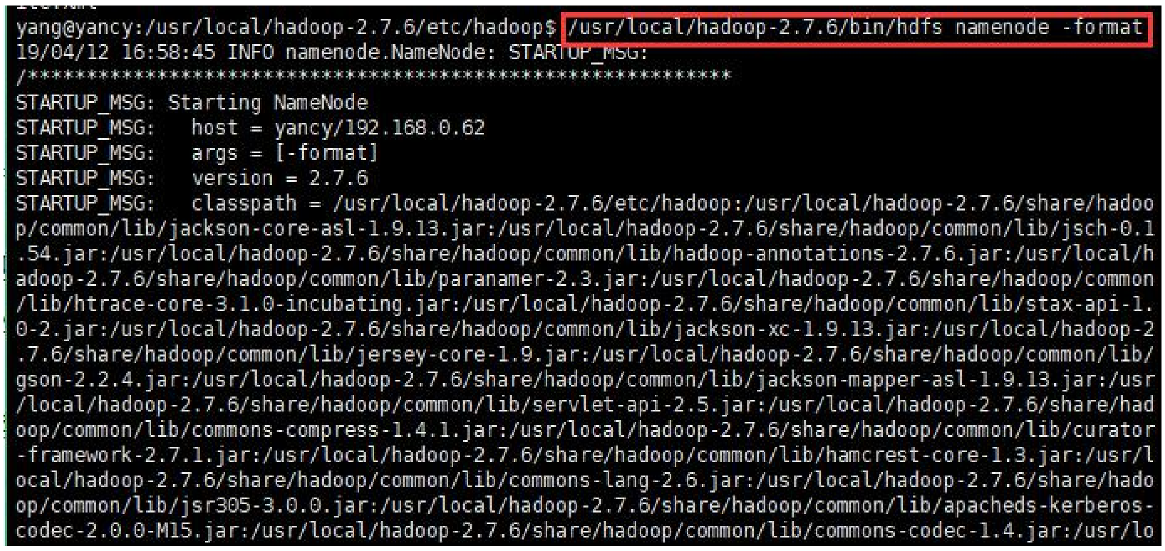
2）bin：对 Hadoop 相关服务（HDFS,YARN）进行操作的脚本；

3）etc：Hadoop 的配置文件目录；

4）share：Hadoop 的依赖 jar 包和文档，文档可以被删掉；

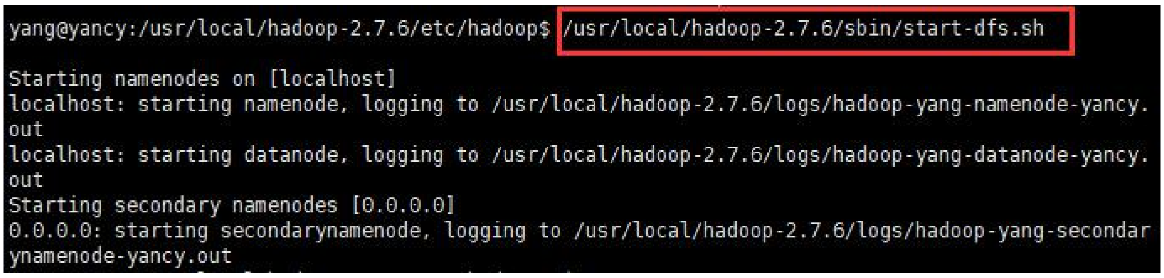
5）lib：Hadoop 的本地库（对数据进行压缩解压缩功能的）。

（1）输入：/usr/local/hadoop-2.7.6/bin/hdfs namenode -format 执行 NameNode 的格式化；



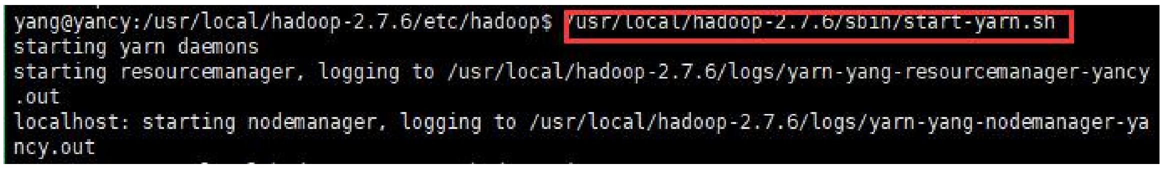
```
yang@yancy:/usr/local/hadoop-2.7.6/etc/hadoop$ /usr/local/hadoop-2.7.6/bin/hdfs namenode -format
19/04/12 16:58:45 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = yancy/192.168.0.62
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.7.6
STARTUP_MSG:   classpath = /usr/local/hadoop-2.7.6/etc/hadoop:/usr/local/hadoop-2.7.6/share/hadoo
p/common/lib/jackson-core-asl-1.9.13.jar:/usr/local/hadoop-2.7.6/share/hadoop/common/lib/jsch-0.1
.54.jar:/usr/local/hadoop-2.7.6/share/hadoop/common/lib/hadoop-annotations-2.7.6.jar:/usr/local/h
adoop-2.7.6/share/hadoop/common/lib/paranamer-2.3.jar:/usr/local/hadoop-2.7.6/share/hadoop/common
/lib/htrace-core-3.1.0-incubating.jar:/usr/local/hadoop-2.7.6/share/hadoop/common/lib/stax-api-1.
0-2.jar:/usr/local/hadoop-2.7.6/share/hadoop/common/lib/jackson-xc-1.9.13.jar:/usr/local/hadoop-2
.7.6/share/hadoop/common/lib/jersey-core-1.9.jar:/usr/local/hadoop-2.7.6/share/hadoop/common/lib/
gson-2.2.4.jar:/usr/local/hadoop-2.7.6/share/hadoop/common/lib/jackson-mapper-asl-1.9.13.jar:/usr
/local/hadoop-2.7.6/share/hadoop/common/lib/servlet-api-2.5.jar:/usr/local/hadoop-2.7.6/share/had
oop/common/lib/commons-compress-1.4.1.jar:/usr/local/hadoop-2.7.6/share/hadoop/common/lib/curator
-framework-2.7.1.jar:/usr/local/hadoop-2.7.6/share/hadoop/common/lib/hamcrest-core-1.3.jar:/usr/l
ocal/hadoop-2.7.6/share/hadoop/common/lib/commons-lang-2.6.jar:/usr/local/hadoop-2.7.6/share/hado
op/common/lib/jsr305-3.0.0.jar:/usr/local/hadoop-2.7.6/share/hadoop/common/lib/apacheds-kerberos-
codec-2.0.0-M15.jar:/usr/local/hadoop-2.7.6/share/hadoop/common/lib/commons-codec-1.4.jar:/usr/lo
```

（2）输入：/usr/local/hadoop-2.7.6/sbin/start-dfs.sh 启动 NameNode 和 DataNode 进程，并查看启动结果；



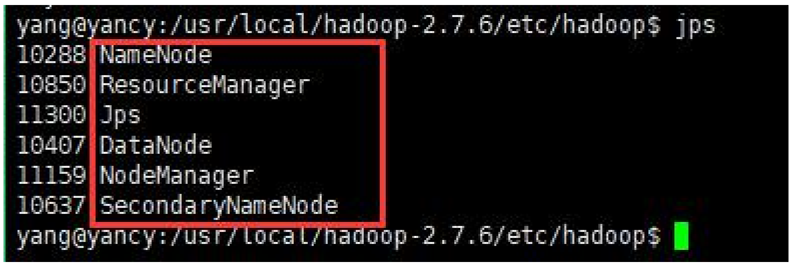
```
yang@yancy:/usr/local/hadoop-2.7.6/etc/hadoop$ /usr/local/hadoop-2.7.6/sbin/start-dfs.sh

Starting namenodes on [localhost]
localhost: starting namenode, logging to /usr/local/hadoop-2.7.6/logs/hadoop-yang-namenode-yancy.
out
localhost: starting datanode, logging to /usr/local/hadoop-2.7.6/logs/hadoop-yang-datanode-yancy.
out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop-2.7.6/logs/hadoop-yang-secondar
ynamenode-yancy.out
```

（3）输入：/usr/local/hadoop-2.7.6/sbin/start-yarn.sh 启动 ResourceManager 和 NodeManager；



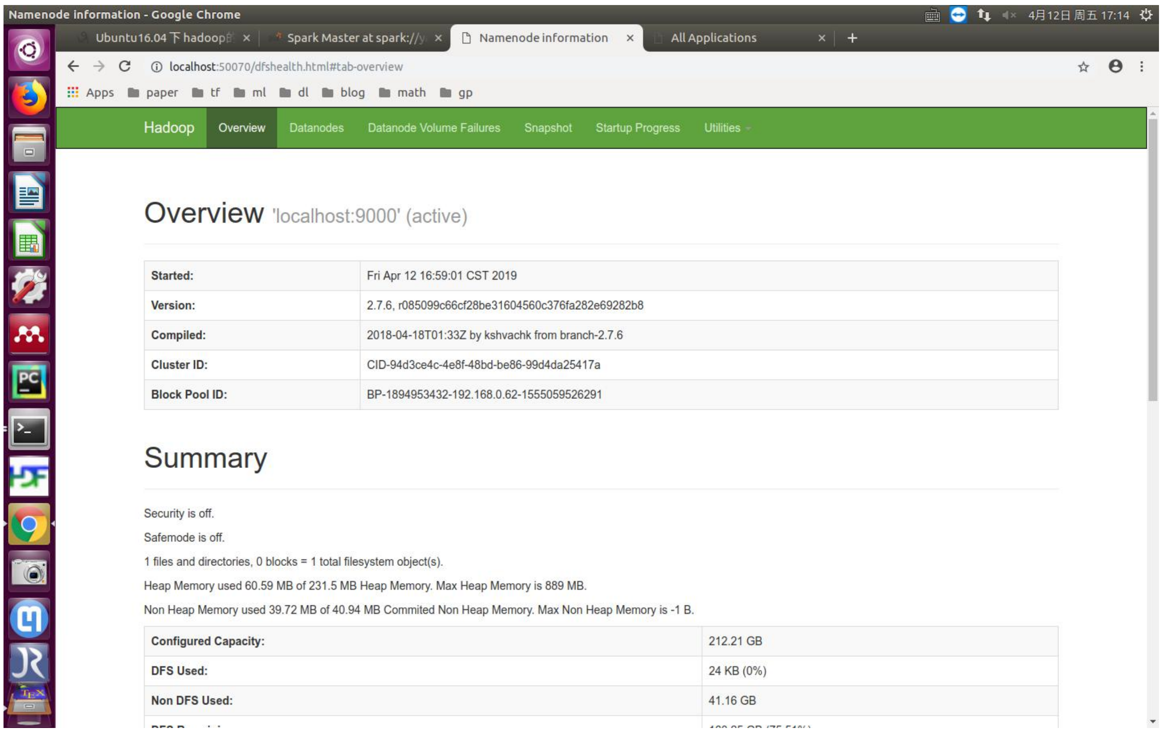
```
yang@yancy:/usr/local/hadoop-2.7.6/etc/hadoop$ /usr/local/hadoop-2.7.6/sbin/start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop-2.7.6/logs/yarn-yang-resourcemanager-yancy
.out
localhost: starting nodemanager, logging to /usr/local/hadoop-2.7.6/logs/yarn-yang-nodemanager-ya
ncy.out
```

（4）输入：jps，判断是否成功启动，若成功启动则会列出如下进程: “NameNode”、”DataNode”、“SecondaryNameNode”、“ResourceManager”和 NodeManager；



```
yang@yancy:/usr/local/hadoop-2.7.6/etc/hadoop$ jps
10288 NameNode
10850 ResourceManager
11300 Jps
10407 DataNode
11159 NodeManager
10637 SecondaryNameNode
yang@yancy:/usr/local/hadoop-2.7.6/etc/hadoop$
```

（5）访问 HDFS 的管理界面：在浏览器访问 http://localhost:50070 查看 NameNode 和 DataNode 的相关信息，还可以在线查看 HDFS 中的文件；



Overview 'localhost:9000' (active)

Started:	Fri Apr 12 16:59:01 CST 2019
Version:	2.7.6, r085099c66cf28be31604560c376fa282e69282b8
Compiled:	2018-04-18T01:33Z by kshvachk from branch-2.7.6
Cluster ID:	CID-94d3ce4c-4e8f-48bd-be86-99d4da25417a
Block Pool ID:	BP-1894953432-192.168.0.62-1555059526291

Summary

Security is off.

Safemode is off.

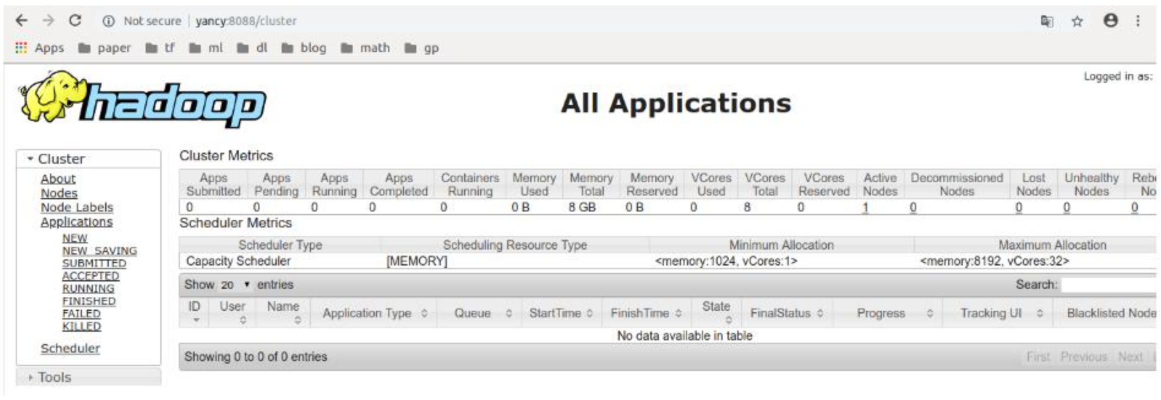
1 files and directories, 0 blocks = 1 total filesystem object(s).

Heap Memory used 60.59 MB of 231.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 39.72 MB of 40.94 MB Commited Non Heap Memory. Max Non Heap Memory is -1 B.

Configured Capacity:	212.21 GB
DFS Used:	24 KB (0%)
Non DFS Used:	41.16 GB

（6）访问 YARN 的管理界面：在浏览器访问 http://yancy:8088 查看 Cluster 相关信息。



Not secure | yancy:8088/cluster

Apps paper tf ml dl blog math gp

Logged in as:

All Applications

Cluster
About
Nodes
Node Labels
Applications
NEW
NEW_SAVING
SUBMITTED
ACCEPTED
RUNNING
FINISHED
FAILED
KILLED
Scheduler
Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebo No
0	0	0	0	0	0 B	8 GB	0 B	0	8	0	1	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:32>

Show 20 entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI	Blacklisted Node
No data available in table											

Showing 0 to 0 of 0 entries First Previous Next

4.2 ubuntu16.04 配置 spark

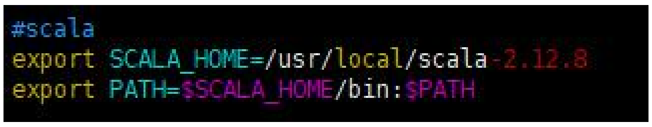
4.2.1 ubuntu16.04 安装 scala-2.12.8

（1）输入：sudo tar -xzvf scala-2.12.8.tgz -C /usr/local，解压 scala 到/usr/local。网址为：https://www.scala-lang.org/download/；

（2）输入：sudo vim ~/.bashrc，在最后添加下面内容：

export SCALA_HOME=/usr/local/scala-2.12.8

export PATH=$SCALA_HOME/bin:$PATH



```
#scala
export SCALA_HOME=/usr/local/scala-2.12.8
export PATH=$SCALA_HOME/bin:$PATH
```

（3）输入：source ~/.bashrc，使新配置的环境变量生效；

（4）输入：scala -version 查看版本。

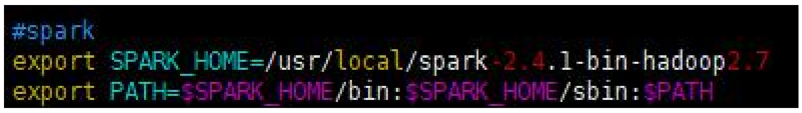
4.2.2 ubuntu16.04 安装 spark-2.4.1-bin-hadoop2.7

（1）输入：sudo tar -zxf spark-2.4.1-bin-hadoop2.7.tgz -C /usr/local，解压下载的 spark 文件。网址为：http://spark.apache.org/downloads.html

（2）输入：sudo vim ~/.bashrc，在最后添加下面内容：

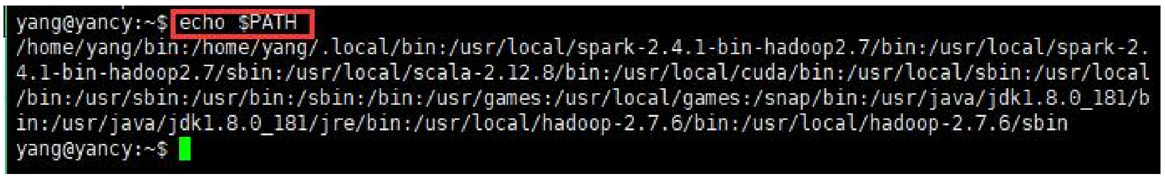
export SPARK_HOME=/usr/local/spark-2.4.1-bin-hadoop2.7

export PATH=$SPARK_HOME/bin:$SPARK_HOME/sbin:$PATH



```
#spark
export SPARK_HOME=/usr/local/spark-2.4.1-bin-hadoop2.7
export PATH=$SPARK_HOME/bin:$SPARK_HOME/sbin:$PATH
```

（3）输入：source ~/.bashrc，使新配置的环境变量生效；



```
yang@yancy:~$ echo $PATH
/home/yang/bin:/home/yang/.local/bin:/usr/local/spark-2.4.1-bin-hadoop2.7/bin:/usr/local/spark-2.
4.1-bin-hadoop2.7/sbin:/usr/local/scala-2.12.8/bin:/usr/local/cuda/bin:/usr/local/sbin:/usr/local
/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/usr/java/jdk1.8.0_181/b
in:/usr/java/jdk1.8.0_181/jre/bin:/usr/local/hadoop-2.7.6/bin:/usr/local/hadoop-2.7.6/sbin
yang@yancy:~$
```

（4）拷贝配置文件：

cd /usr/local/spark-2.4.1-bin-hadoop2.7

sudo cp ./conf/spark-env.sh.template ./conf/spark-env.sh

（5）输入：sudo vim /usr/local/spark-2.4.1-bin-hadoop2.7/conf/spark-env.sh，修改配置文件，添加下面一行：

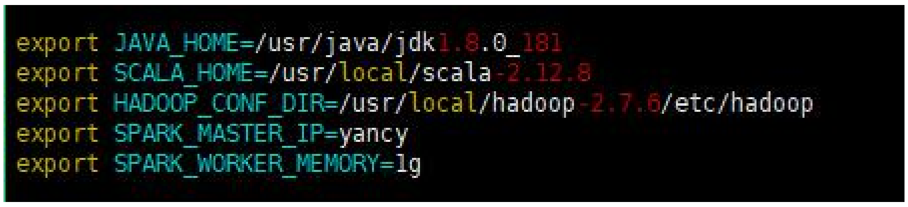
export JAVA_HOME=/usr/java/jdk1.8.0_181

export SCALA_HOME=/usr/local/scala-2.12.8

export HADOOP_CONF_DIR=/usr/local/hadoop-2.7.6/etc/hadoop

export SPARK_MASTER_IP=yancy

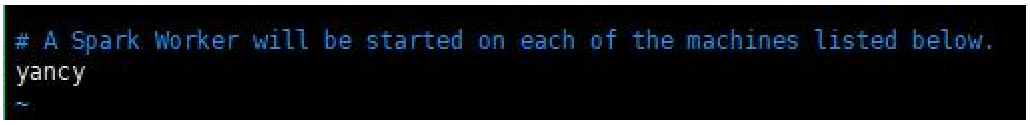
export SPARK_WORKER_MEMORY=1g



```
export JAVA_HOME=/usr/java/jdk1.8.0_181
export SCALA_HOME=/usr/local/scala-2.12.8
export HADOOP_CONF_DIR=/usr/local/hadoop-2.7.6/etc/hadoop
export SPARK_MASTER_IP=yancy
export SPARK_WORKER_MEMORY=1g
```

（6）输入：sudo cp /usr/local/spark-2.4.1-bin-hadoop2.7/conf/slaves.template slaves，将 slaves.template 重命名为 slaves；

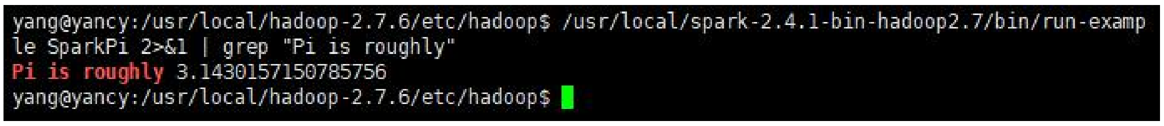
（7）输入：sudo vim /usr/local/spark-2.4.1-bin-hadoop2.7/conf/slaves，将 slaves 中的 localhost 修改为主机名，我的是 yancy。



```
# A Spark Worker will be started on each of the machines listed below.
yancy
~
```

（8）运行简单示例：

/usr/local/spark-2.4.1-bin-hadoop2.7/bin/run-example SparkPi 2>&1 | grep "Pi is roughly"



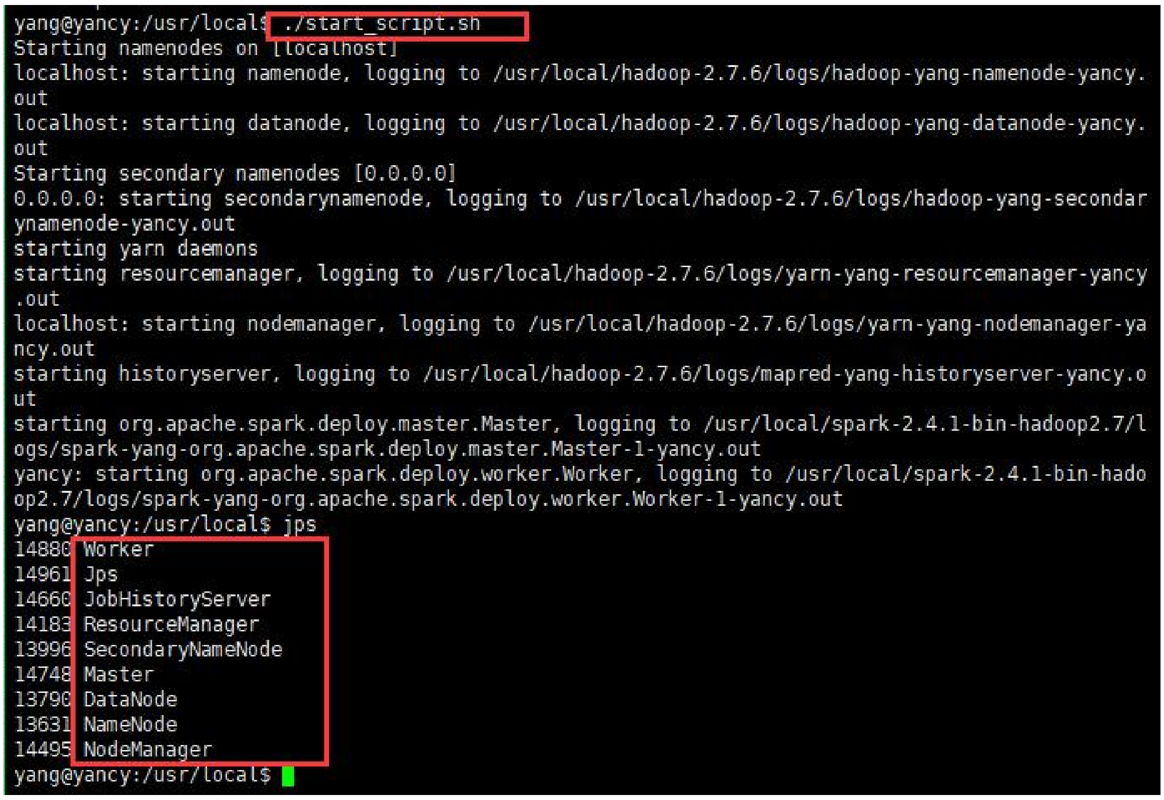
```
yang@yancy:/usr/local/hadoop-2.7.6/etc/hadoop$ /usr/local/spark-2.4.1-bin-hadoop2.7/bin/run-examp
le SparkPi 2>&1 | grep "Pi is roughly"
Pi is roughly 3.1430157150785756
yang@yancy:/usr/local/hadoop-2.7.6/etc/hadoop$
```

（9）输入；sudo chown -R yang:yang spark-2.4.1-bin-hadoop2.7/，修改权限；

（10）输入：/usr/local/spark-2.4.1-bin-hadoop2.7/sbin/start-all.sh，启动 Spark；

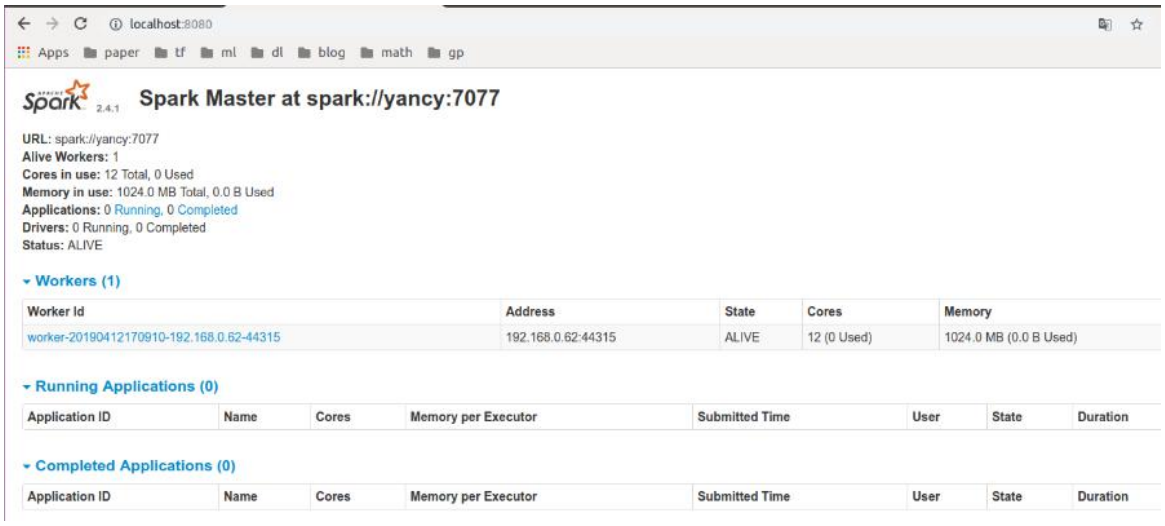
（11）编写启动脚本 start_script.sh 启动 Hadoop 以及 Spark：

```
#!/bin/bash
start-dfs.sh #  启动 Hadoop
start-yarn.sh #  启动 Yarn
mr-jobhistory-daemon.sh start historyserver #  启动历史服务器,以便在 Web 中查看任务运行情况。
/usr/local/spark-2.4.1-bin-hadoop2.7/sbin/start-all.sh #  启动 Spark
```



```
yang@yancy:/usr/local$ ./start_script.sh
Starting namenodes on [localhost]
localhost: starting namenode, logging to /usr/local/hadoop-2.7.6/logs/hadoop-yang-namenode-yancy.
out
localhost: starting datanode, logging to /usr/local/hadoop-2.7.6/logs/hadoop-yang-datanode-yancy.
out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop-2.7.6/logs/hadoop-yang-secondar
ynamenode-yancy.out
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop-2.7.6/logs/yarn-yang-resourcemanager-yancy
.out
localhost: starting nodemanager, logging to /usr/local/hadoop-2.7.6/logs/yarn-yang-nodemanager-ya
ncy.out
starting historyserver, logging to /usr/local/hadoop-2.7.6/logs/mapred-yang-historyserver-yancy.o
ut
starting org.apache.spark.deploy.master.Master, logging to /usr/local/spark-2.4.1-bin-hadoop2.7/l
ogs/spark-yang-org.apache.spark.deploy.master.Master-1-yancy.out
yancy: starting org.apache.spark.deploy.worker.Worker, logging to /usr/local/spark-2.4.1-bin-hado
op2.7/logs/spark-yang-org.apache.spark.deploy.worker.Worker-1-yancy.out
yang@yancy:/usr/local$ jps
14880 Worker
14961 Jps
14660 JobHistoryServer
14183 ResourceManager
13996 SecondaryNameNode
14748 Master
13790 DataNode
13631 NameNode
14495 NodeManager
yang@yancy:/usr/local$
```

（12）通过 WEB 页面查看：浏览器中输入地址：localhost:8080。



Spark Master at spark://yancy:7077

URL: spark://yancy:7077
Alive Workers: 1
Cores in use: 12 Total, 0 Used
Memory in use: 1024.0 MB Total, 0.0 B Used
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory
worker-20190412170910-192.168.0.62-44315	192.168.0.62:44315	ALIVE	12 (0 Used)	1024.0 MB (0.0 B Used)

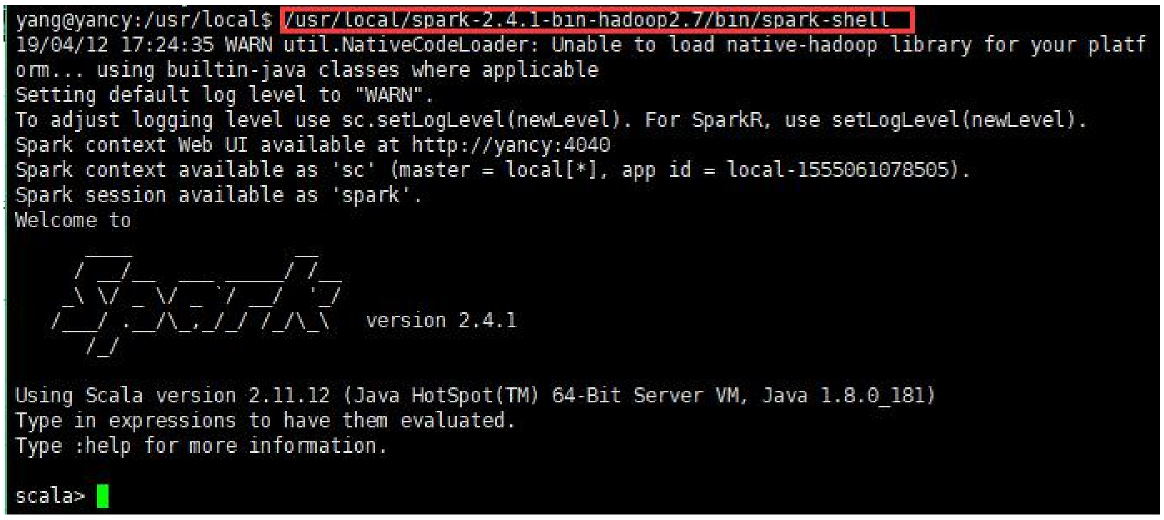
Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration

（13）输　入　：　/usr/local/spark-2.4.1-bin-hadoop2.7/bin/spark-shell　，　启　动 SparkContext。



```
yang@yancy:/usr/local$ /usr/local/spark-2.4.1-bin-hadoop2.7/bin/spark-shell
19/04/12 17:24:35 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platf
orm... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://yancy:4040
Spark context available as 'sc' (master = local[*], app id = local-1555061078505).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.1
      /_/

Using Scala version 2.11.12 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_181)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

（14）编写停止脚本 stop_script.sh 停止 Hadoop 以及 Spark：

#!/bin/bash

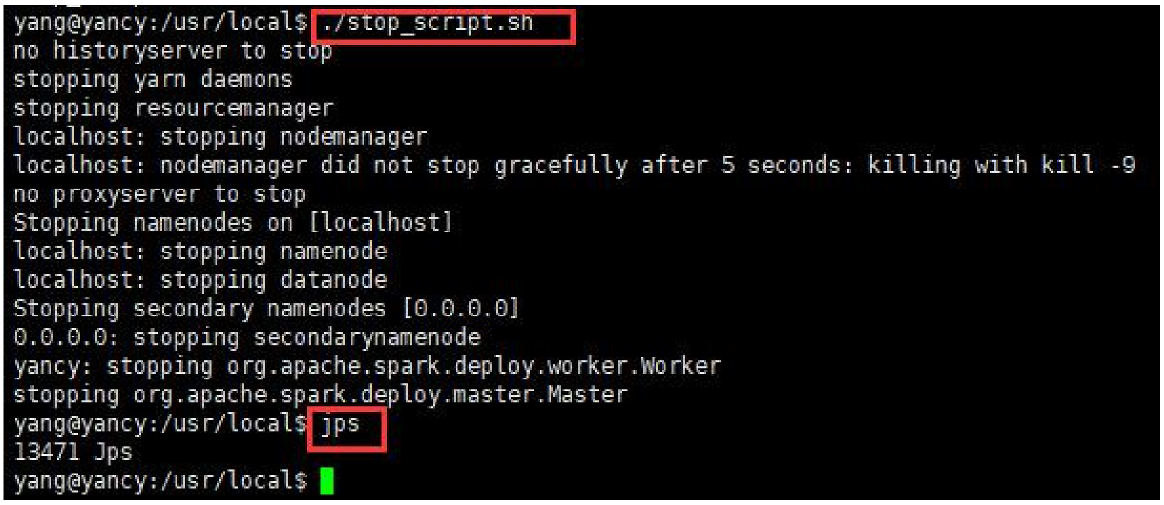
mr-jobhistory-daemon.sh stop historyserver # 停止历史服务器

stop-yarn.sh # 停止 Yarn

stop-dfs.sh # 停止 Hadoop

#/usr/local/hadoop/sbin/stop-all.sh # 停止 Hadoop 以及 yarn

/usr/local/spark-2.4.1-bin-hadoop2.7/sbin/stop-all.sh # 停止 Spark



```
yang@yancy:/usr/local$ ./stop_script.sh
no historyserver to stop
stopping yarn daemons
stopping resourcemanager
localhost: stopping nodemanager
localhost: nodemanager did not stop gracefully after 5 seconds: killing with kill -9
no proxyserver to stop
Stopping namenodes on [localhost]
localhost: stopping namenode
localhost: stopping datanode
Stopping secondary namenodes [0.0.0.0]
0.0.0.0: stopping secondarynamenode
yancy: stopping org.apache.spark.deploy.worker.Worker
stopping org.apache.spark.deploy.master.Master
yang@yancy:/usr/local$ jps
13471 Jps
yang@yancy:/usr/local$
```

4.3　ubuntu16.04 配置 Hadoop 全分布式集群

搭建完全分布式集群需准备三台主机，一个主节点 yancy 和两个从节点 Slave、Slave2，首先需要对主机名进行修改。

4.3.1 配置 hosts 文件

（1）输入：sudo vim /etc/hostname，修改主机名为 yancy；

（2）输入：sudo vim /etc/hosts，添加各主机 IP 地址如下：

```
192.168.0.62     yancy
192.168.0.104    Slave1
192.168.0.50     Slave2
```

（3）其它两台从节点也都要修改 hostname 和 hosts 文件。配置完 hosts 后三台主机就可以进行通信了，可以互相 ping 通，是可以 ping 通的。

4.3.2 SSH 免密码登录

（1）输入：dpkg --list|grep ssh，查看安装的 openssh-server；

（2）输入：ssh-keygen -t rsa，三次回车后，该目录下将会产生 id_rsa，id_rsa.pub 文件；

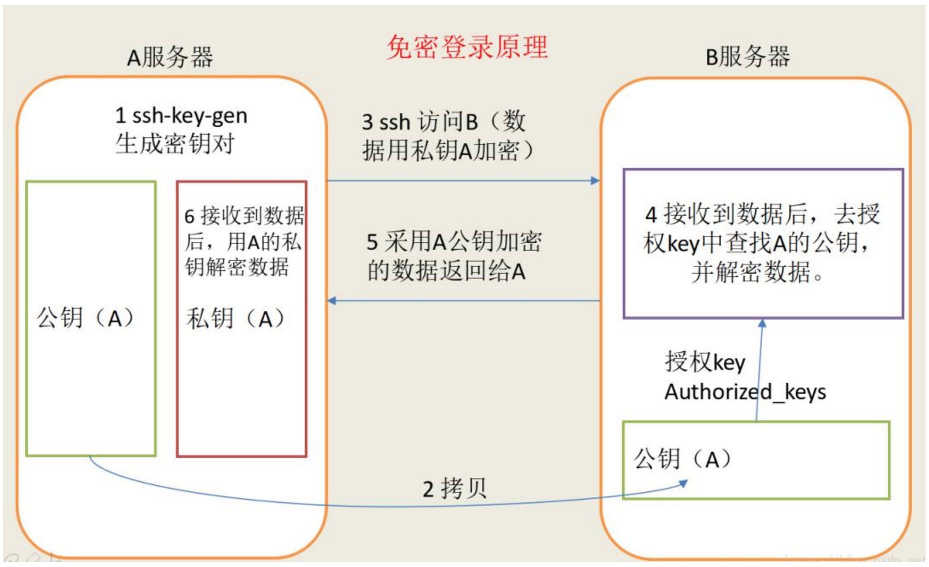
（3）输入：cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys 加入授权；

（4）输入：scp /home/yang/.ssh/id_rsa.pub qxxhemu@Slave1:~/.ssh/，将公钥复制到其他从机，或 ssh-copy-id -i ~/.ssh/id_rsa.pub Slave1；

（5）输入：scp /home/yang/.ssh/id_rsa.pub long@Slave2:~/.ssh/

（6）输入：cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys 加入授权；

（7）输入：ssh qxxhemu@Slave1，第一次需要密码，之后 exit 退出，再 ssh qxxhemu@Slave1 就不需要密码登录成功。


免密登录原理
A服务器
B服务器
1 ssh-key-gen
生成密钥对
3 ssh 访问B（数据用私钥A加密）
6 接收到数据后，用A的私钥解密数据
公钥（A）
私钥（A）
5 采用A公钥加密的数据返回给A
4 接收到数据后，去授权key中查找A的公钥，并解密数据。
授权key
Authorized_keys
公钥（A）
2 拷贝

4.3.3 修改 slaves 文件

（1）输入：sudo vim /usr/local/hadoop-2.7.6/etc/hadoop/slaves，添加内容：Slave1 和 Slave2；

4.3.4 配置 Hadoop 及 spark 文件

其余步骤类似，不再详述。

4.4 WordCount 词频统计

4.4.1 MapReduce 词频统计

下面为一些 HDFS 常用命令：

hadoop fs -mkdir /tmp/input 在 HDFS 上目录/tmp/input；

hadoop fs -put input1.txt /tmp/input 把本地文件 input1.txt 传到 HDFS 的/tmp/input 目录下；

hadoop fs -get input1.txt /tmp/input/input1.txt 把 HDFS 文件拉到本地；

hadoop fs -ls /tmp/output 列出 HDFS 的目录/tmp/output；

hadoop fs -cat /tmp/ouput/output1.txt 查看 HDFS 上文件/tmp/ouput/output1.txt；

hadoop fs -rmr /tmp/intput 删除 HDFS 上的目录/tmp/intput；

hadoop dfsadmin -report 查看 HDFS 状态，比如每个 DataNode 的情况；

hadoop dfsadmin -safemode leave 离开安全模式；

hadoop dfsadmin -safemode enter 进入安全模式。

WordCount 词频统计如下步骤：

（1）输入：start-all.sh，启动 HDFS；

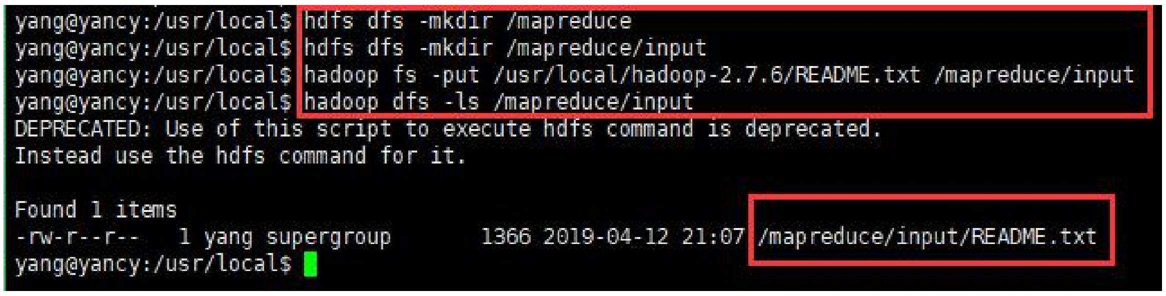
（2）输入：hadoop dfs -ls /，查看 HDFS 下面包含的文件目录，第一次运行 hdfs 什么文件都没有；

（3）输入：hdfs dfs -mkdir -p /mapreduce/input，在 HDFS 中创建一个文件目录/mapreduce/input；

（4）输入：hadoop fs -put /usr/local/hadoop-2.7.6/README.txt /mapreduce/input，将/usr/local/hadoop-2.7.6/README.txt 上传至/mapreduce/input 中；

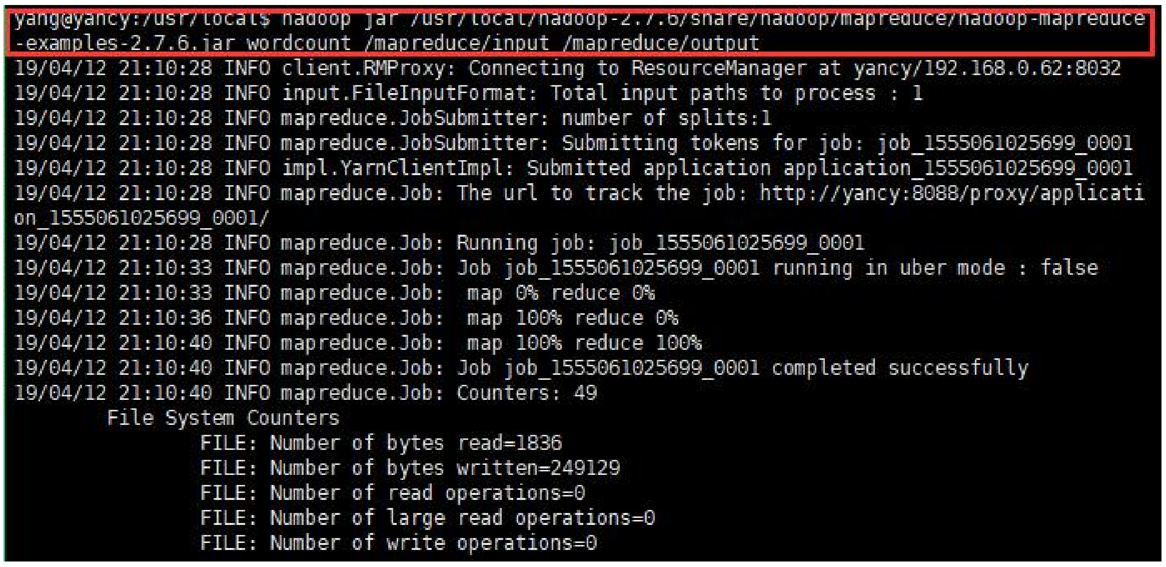
（5）输入：hadoop dfs -ls /mapreduce/input 查看/mapreduce/input 下多了一个

README.txt；

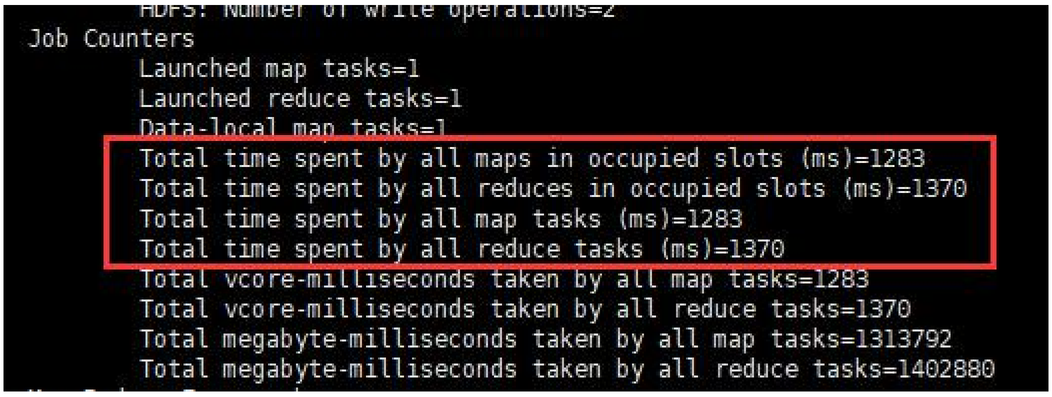


```
yang@yancy:/usr/local$ hdfs dfs -mkdir /mapreduce
yang@yancy:/usr/local$ hdfs dfs -mkdir /mapreduce/input
yang@yancy:/usr/local$ hadoop fs -put /usr/local/hadoop-2.7.6/README.txt /mapreduce/input
yang@yancy:/usr/local$ hadoop dfs -ls /mapreduce/input
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Found 1 items
-rw-r--r--   1 yang supergroup       1366 2019-04-12 21:07 /mapreduce/input/README.txt
yang@yancy:/usr/local$
```

（6）执行如下命令运行 wordcount 并将结果输到 output： hadoop jar /usr/local/hadoop-2.7.6/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.6.jar wordcount /mapreduce/input /mapreduce/output；

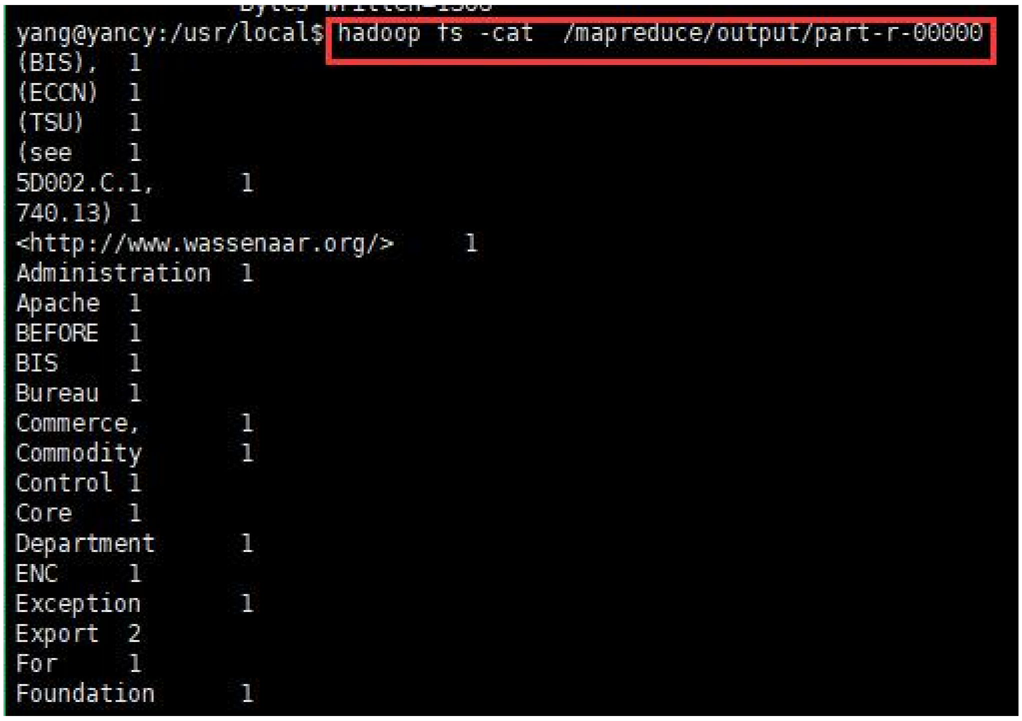


```
yang@yancy:/usr/local$ hadoop jar /usr/local/hadoop-2.7.6/share/hadoop/mapreduce/hadoop-mapreduce
-examples-2.7.6.jar wordcount /mapreduce/input /mapreduce/output
19/04/12 21:10:28 INFO client.RMProxy: Connecting to ResourceManager at yancy/192.168.0.62:8032
19/04/12 21:10:28 INFO input.FileInputFormat: Total input paths to process : 1
19/04/12 21:10:28 INFO mapreduce.JobSubmitter: number of splits:1
19/04/12 21:10:28 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1555061025699_0001
19/04/12 21:10:28 INFO impl.YarnClientImpl: Submitted application application_1555061025699_0001
19/04/12 21:10:28 INFO mapreduce.Job: The url to track the job: http://yancy:8088/proxy/applicati
on_1555061025699_0001/
19/04/12 21:10:28 INFO mapreduce.Job: Running job: job_1555061025699_0001
19/04/12 21:10:33 INFO mapreduce.Job: Job job_1555061025699_0001 running in uber mode : false
19/04/12 21:10:33 INFO mapreduce.Job:  map 0% reduce 0%
19/04/12 21:10:36 INFO mapreduce.Job:  map 100% reduce 0%
19/04/12 21:10:40 INFO mapreduce.Job:  map 100% reduce 100%
19/04/12 21:10:40 INFO mapreduce.Job: Job job_1555061025699_0001 completed successfully
19/04/12 21:10:40 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=1836
                FILE: Number of bytes written=249129
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
```

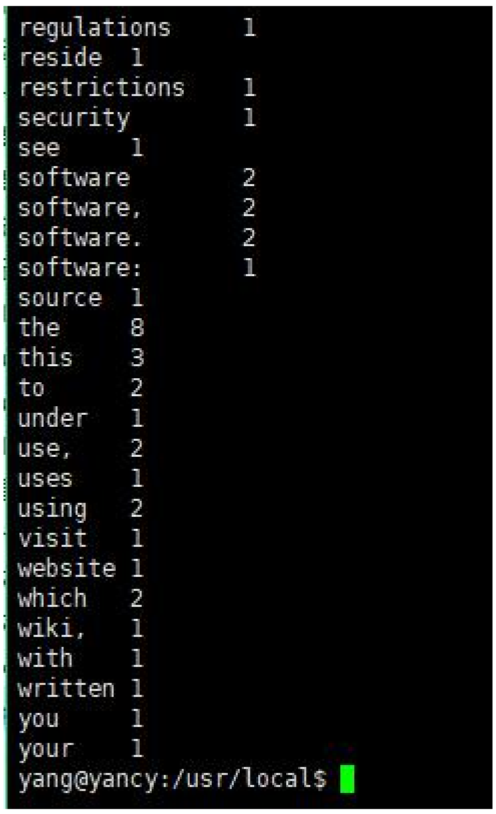


```
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=1283
                Total time spent by all reduces in occupied slots (ms)=1370
                Total time spent by all map tasks (ms)=1283
                Total time spent by all reduce tasks (ms)=1370
                Total vcore-milliseconds taken by all map tasks=1283
                Total vcore-milliseconds taken by all reduce tasks=1370
                Total megabyte-milliseconds taken by all map tasks=1313792
                Total megabyte-milliseconds taken by all reduce tasks=1402880
```

（7）执行成功后 output 目录下会生成两个文件：一个是_SUCCESS 成功标志的文件，里面没有内容，另一个是 part-r-00000 ，通过以下命令查看执行的结果：hadoop fs -cat /mapreduce/output/part-r-00000。



```
yang@yancy:/usr/local$ hadoop fs -cat  /mapreduce/output/part-r-00000
(BIS),  1
(ECCN)  1
(TSU)   1
(see    1
5D002.C.1,      1
740.13) 1
<http://www.wassenaar.org/>     1
Administration  1
Apache  1
BEFORE  1
BIS     1
Bureau  1
Commerce,       1
Commodity       1
Control 1
Core    1
Department      1
ENC     1
Exception       1
Export  2
For     1
Foundation      1
```



```
regulations     1
reside  1
restrictions    1
security        1
see     1
software        2
software,       2
software.       2
software:       1
source  1
the     8
this    3
to      2
under   1
use,    2
uses    1
using   2
visit   1
website 1
which   2
wiki,   1
with    1
written 1
you     1
your    1
yang@yancy:/usr/local$
```

4. 4. 2 Spark 词频统计

（1）输入：/usr/local/spark-2.4.1-bin-hadoop2.7/bin/spark-shell，启动 spark shell；

（2）或者：/usr/local/spark-2.4.1-bin-hadoop2.7/bin/spark-shell \

> --master spark://yancy:7077 \

> --executor-memory 500m

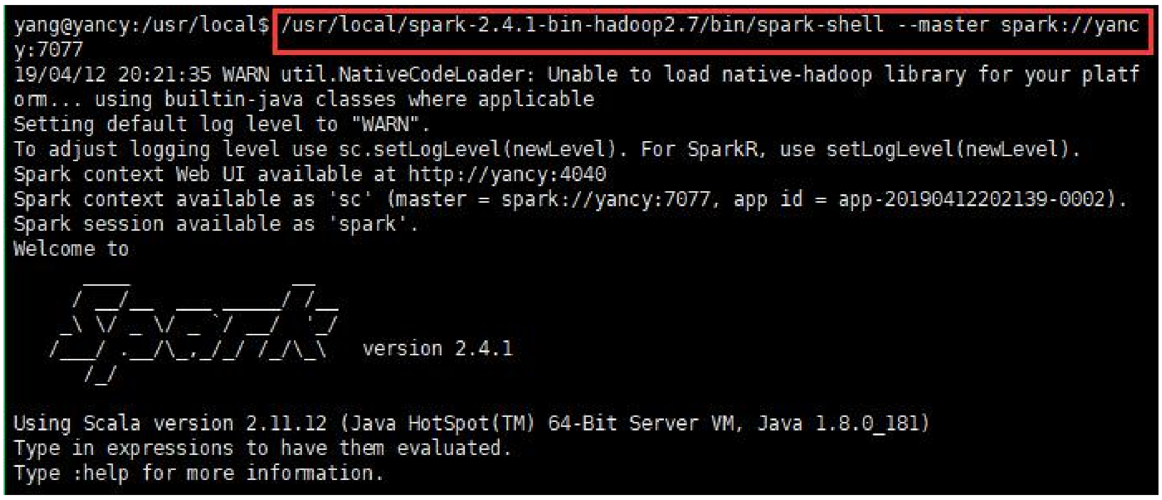
> --total-executor-cores 1

参数说明：

--master spark://yancy:7077 ：指定 Master 的地址；

--executor-memory 500m ：指定每个 worker 可用内存为 500m；

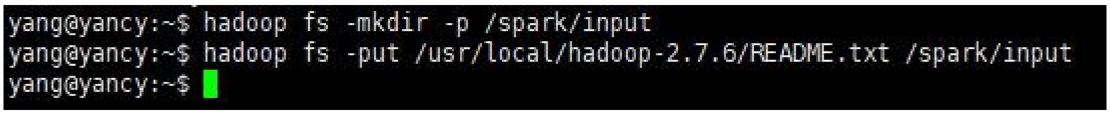
--total-executor-cores 1 ：指定整个集群使用的 CPU 核数为 1 个；



```
yang@yancy:/usr/local$ /usr/local/spark-2.4.1-bin-hadoop2.7/bin/spark-shell --master spark://yanc
y:7077
19/04/12 20:21:35 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platf
orm... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://yancy:4040
Spark context available as 'sc' (master = spark://yancy:7077, app id = app-20190412202139-0002).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.1
      /_/

Using Scala version 2.11.12 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_181)
Type in expressions to have them evaluated.
Type :help for more information.
```

（3）输入：hadoop fs -mkdir -p /spark/input，创建/spark/input 文件夹；

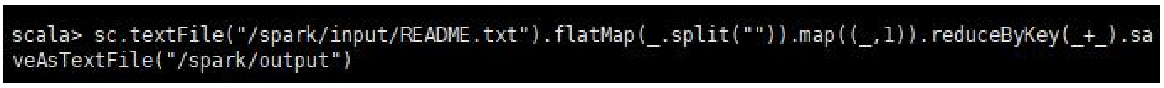
（4）输入：hadoop fs -put /usr/local/hadoop-2.7.6/README.txt /spark/input，将/usr/local/hadoop-2.7.6/README.txt 上传至/spark/input 中；



```
yang@yancy:~$ hadoop fs -mkdir -p /spark/input
yang@yancy:~$ hadoop fs -put /usr/local/hadoop-2.7.6/README.txt /spark/input
yang@yancy:~$ 
```

（5）在 spark shell 中用 scala 编写 spark 程序，按空格分割数据，输入：

sc.textFile("/spark/input/README.txt").flatMap(_.split("")).map((_,1)).reduceByKey(_+_).saveAsTextFile("/spark/output")；



```
scala> sc.textFile("/spark/input/README.txt").flatMap(_.split("")).map((_,1)).reduceByKey(_+_).sa
veAsTextFile("/spark/output")
```

说明：

sc 是 SparkContext 对象，该对象是提交 spark 程序的入口；

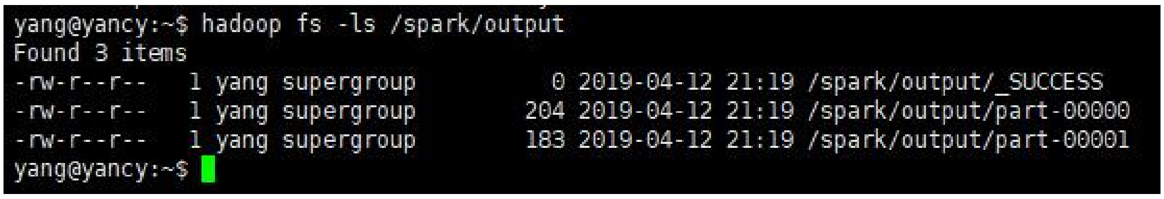
textFile("/spark/input/README.txt") 是从 hdfs 中读取数据；

flatMap(_.split(" ")) 先 map 再压平；

map((_,1)) 将单词和 1 构成元组；

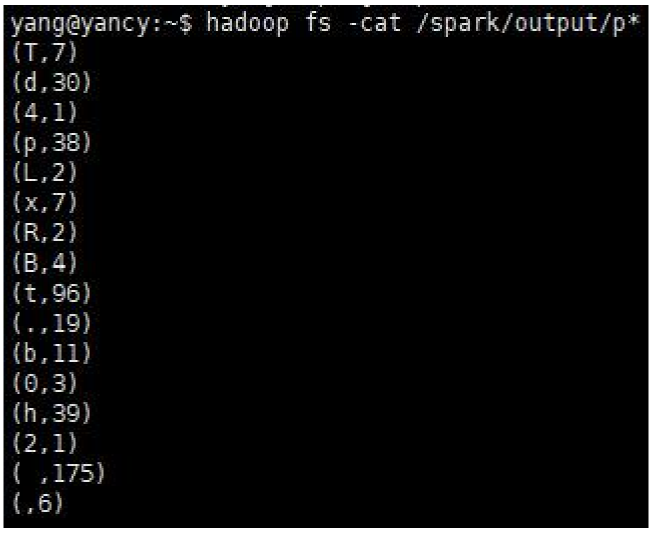
reduceByKey(_+_) 按照 key 进行 reduce，并将 value 累加；

saveAsTextFile("/spark/output") 将结果写入到 hdfs 中。



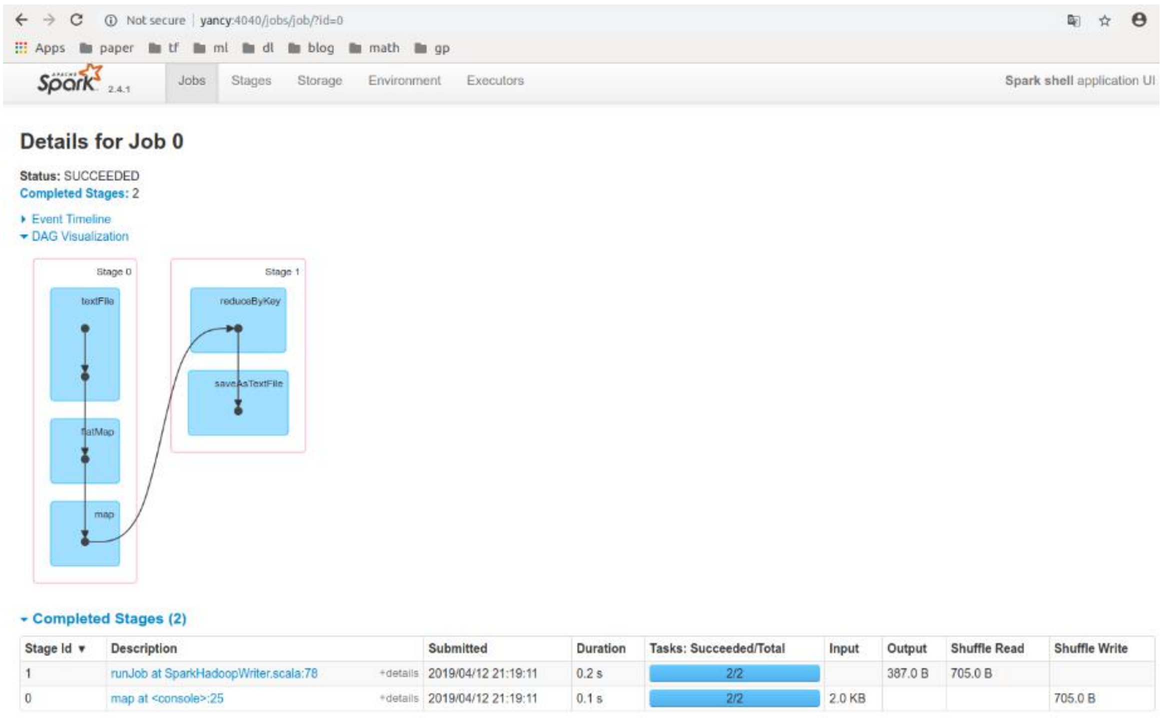
```
yang@yancy:~$ hadoop fs -ls /spark/output
Found 3 items
-rw-r--r--   1 yang supergroup          0 2019-04-12 21:19 /spark/output/_SUCCESS
-rw-r--r--   1 yang supergroup        204 2019-04-12 21:19 /spark/output/part-00000
-rw-r--r--   1 yang supergroup        183 2019-04-12 21:19 /spark/output/part-00001
yang@yancy:~$ 
```

（6）输入：hadoop fs -cat /spark/output/p*，查看 hdfs 的执行结果：



```
yang@yancy:~$ hadoop fs -cat /spark/output/p*
(T,7)
(d,30)
(4,1)
(p,38)
(L,2)
(x,7)
(R,2)
(B,4)
(t,96)
(.,19)
(b,11)
(0,3)
(h,39)
(2,1)
( ,175)
(,6)
```

（7）在浏览器输入：yancy:4040，查看 spark UI 。



Details for Job 0

Status: SUCCEEDED
Completed Stages: 2

▸ Event Timeline
▾ DAG Visualization

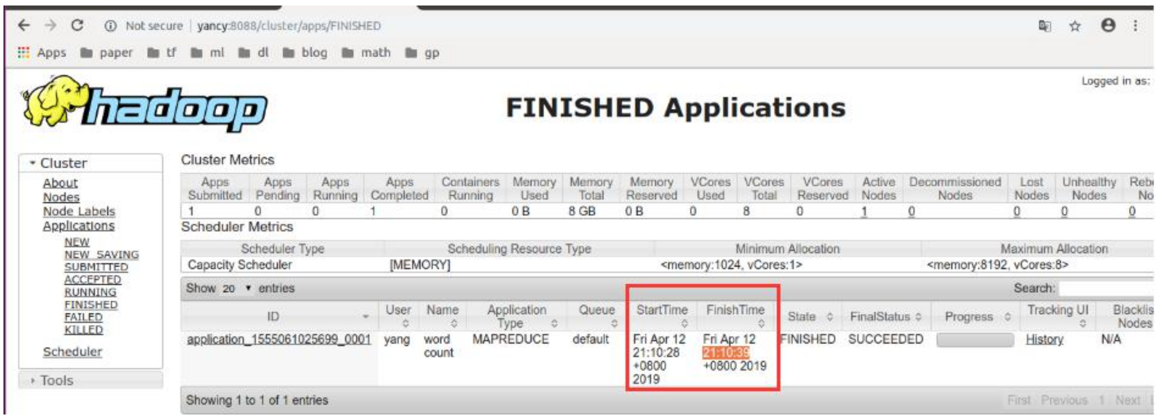
▾ Completed Stages (2)

Stage Id ▾	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	runJob at SparkHadoopWriter.scala:78	+details	2019/04/12 21:19:11	0.2 s	2/2		387.0 B	705.0 B	
0	map at <console>:25	+details	2019/04/12 21:19:11	0.1 s	2/2	2.0 KB			705.0 B

4.5 WordCount 词频统计

4.5.1 MapReduce 执行效率

如下图所示，从图中可以观察到，MapReduce 执行词频统计从开始时间为 21:10:28，结束时间为 21:10:39，大约花费 11 秒运算效率不太高。



FINISHED Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebo No
1	0	0	1	0	0 B	8 GB	0 B	0	8	0	1	0	0	0	0

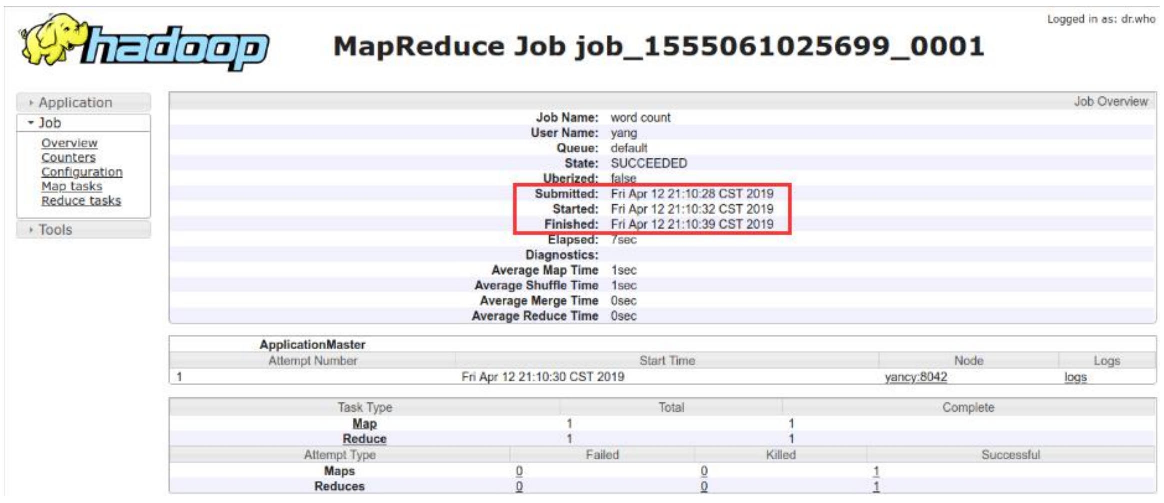
Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:8>

Show 20 entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI	Blacklis Nodes
application_1555061025699_0001	yang	word count	MAPREDUCE	default	Fri Apr 12 21:10:28 +0800 2019	Fri Apr 12 21:10:39 +0800 2019	FINISHED	SUCCEEDED		History	N/A

Showing 1 to 1 of 1 entries First Previous 1 Next



MapReduce Job job_1555061025699_0001

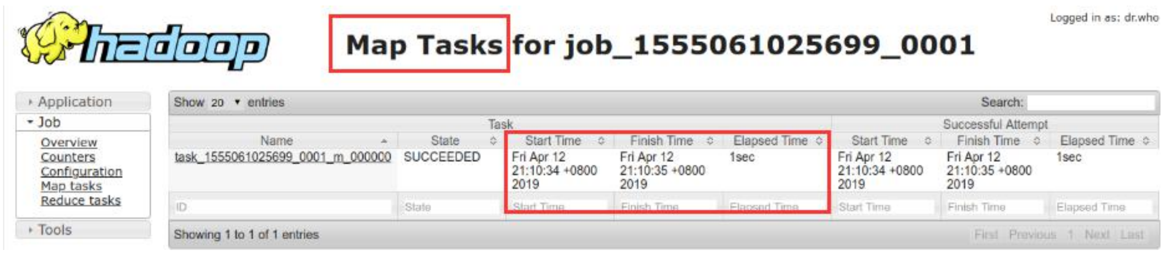
Job Overview

Job Name:	word count
User Name:	yang
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Fri Apr 12 21:10:28 CST 2019
Started:	Fri Apr 12 21:10:32 CST 2019
Finished:	Fri Apr 12 21:10:39 CST 2019
Elapsed:	7sec
Diagnostics:	
Average Map Time	1sec
Average Shuffle Time	1sec
Average Merge Time	0sec
Average Reduce Time	0sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Fri Apr 12 21:10:30 CST 2019	yancy:8042	logs

Task Type	Total	Complete	
Map	1	1	
Reduce	1	1	
Attempt Type	Failed	Killed	Successful
Maps	0	0	1
Reduces	0	0	1

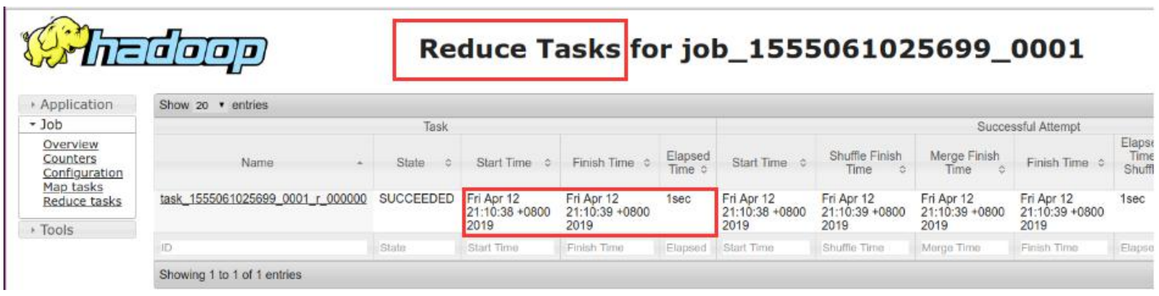


Map Tasks for job_1555061025699_0001

Show 20 entries Search:

Task					Successful Attempt		
Name	State	Start Time	Finish Time	Elapsed Time	Start Time	Finish Time	Elapsed Time
task_1555061025699_0001_m_000000	SUCCEEDED	Fri Apr 12 21:10:34 +0800 2019	Fri Apr 12 21:10:35 +0800 2019	1sec	Fri Apr 12 21:10:34 +0800 2019	Fri Apr 12 21:10:35 +0800 2019	1sec
ID	State	Start Time	Finish Time	Elapsed Time	Start Time	Finish Time	Elapsed Time

Showing 1 to 1 of 1 entries First Previous 1 Next Last

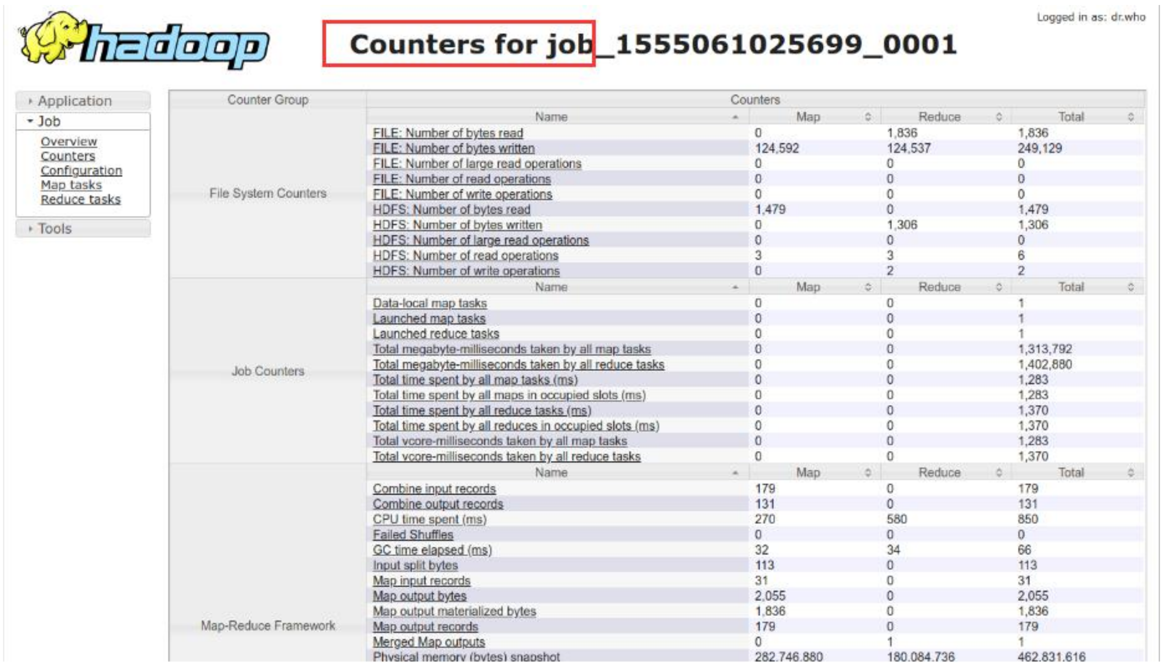


Reduce Tasks for job_1555061025699_0001

Show 20 entries

Task					Successful Attempt				
Name	State	Start Time	Finish Time	Elapsed Time	Start Time	Shuffle Finish Time	Merge Finish Time	Finish Time	Elapse Time Shuffl
task_1555061025699_0001_r_000000	SUCCEEDED	Fri Apr 12 21:10:38 +0800 2019	Fri Apr 12 21:10:39 +0800 2019	1sec	Fri Apr 12 21:10:38 +0800 2019	Fri Apr 12 21:10:39 +0800 2019	Fri Apr 12 21:10:39 +0800 2019	Fri Apr 12 21:10:39 +0800 2019	1sec
ID	State	Start Time	Finish Time	Elapsed	Start Time	Shuffle Time	Merge Time	Finish Time	Elapse

Showing 1 to 1 of 1 entries

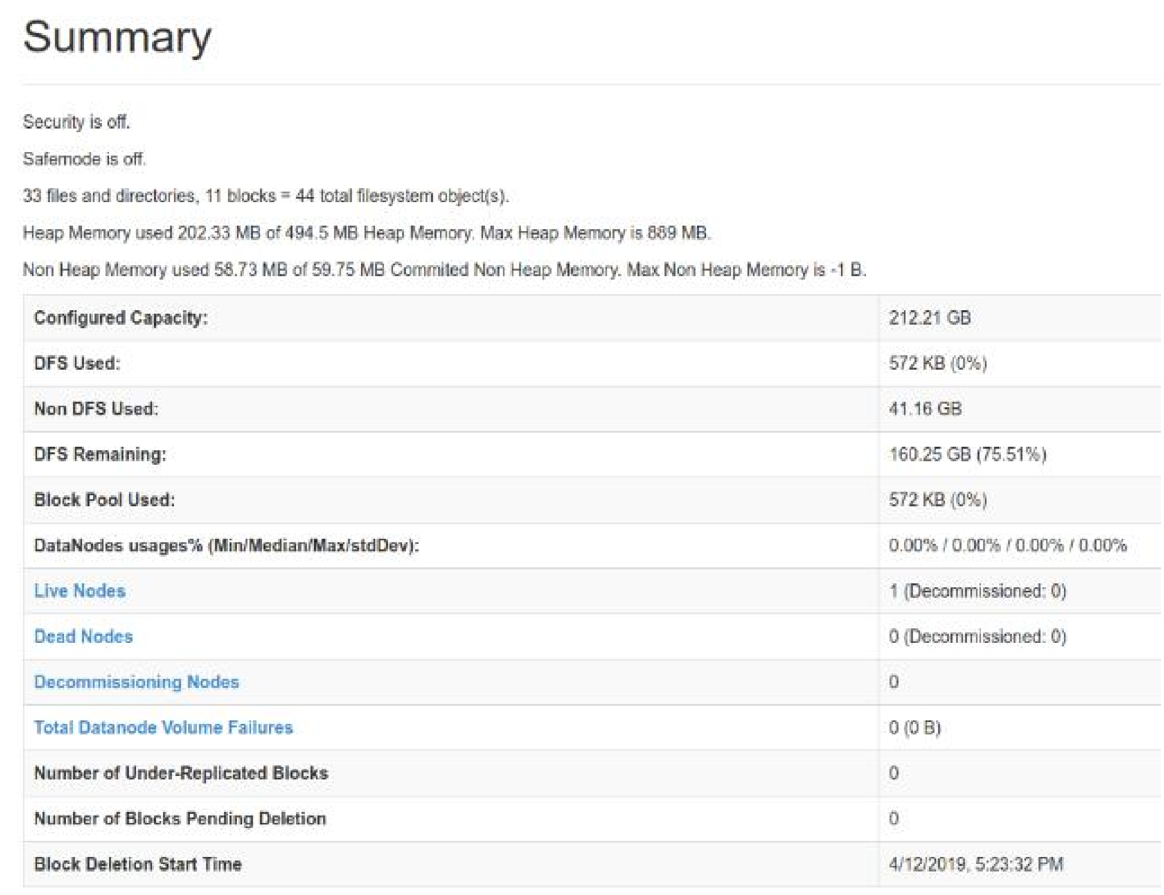


Logged in as: dr.who

Counters for job_1555061025699_0001

- Application
- Job
  - Overview
  - Counters
  - Configuration
  - Map tasks
  - Reduce tasks
- Tools

Counter Group	Name	Map	Reduce	Total
File System Counters	FILE: Number of bytes read	0	1,836	1,836
	FILE: Number of bytes written	124,592	124,537	249,129
	FILE: Number of large read operations	0	0	0
	FILE: Number of read operations	0	0	0
	FILE: Number of write operations	0	0	0
	HDFS: Number of bytes read	1,479	0	1,479
	HDFS: Number of bytes written	0	1,306	1,306
	HDFS: Number of large read operations	0	0	0
	HDFS: Number of read operations	3	3	6
	HDFS: Number of write operations	0	2	2
	Name	Map	Reduce	Total
Job Counters	Data-local map tasks	0	0	1
	Launched map tasks	0	0	1
	Launched reduce tasks	0	0	1
	Total megabyte-milliseconds taken by all map tasks	0	0	1,313,792
	Total megabyte-milliseconds taken by all reduce tasks	0	0	1,402,880
	Total time spent by all map tasks (ms)	0	0	1,283
	Total time spent by all maps in occupied slots (ms)	0	0	1,283
	Total time spent by all reduce tasks (ms)	0	0	1,370
	Total time spent by all reduces in occupied slots (ms)	0	0	1,370
	Total vcore-milliseconds taken by all map tasks	0	0	1,283
	Total vcore-milliseconds taken by all reduce tasks	0	0	1,370
	Name	Map	Reduce	Total
Map-Reduce Framework	Combine input records	179	0	179
	Combine output records	131	0	131
	CPU time spent (ms)	270	580	850
	Failed Shuffles	0	0	0
	GC time elapsed (ms)	32	34	66
	Input split bytes	113	0	113
	Map input records	31	0	31
	Map output bytes	2,055	0	2,055
	Map output materialized bytes	1,836	0	1,836
	Map output records	179	0	179
	Merged Map outputs	0	1	1
	Physical memory (bytes) snapshot	282,746,880	180,084,736	462,831,616



Summary

Security is off.

Safemode is off.

33 files and directories, 11 blocks = 44 total filesystem object(s).

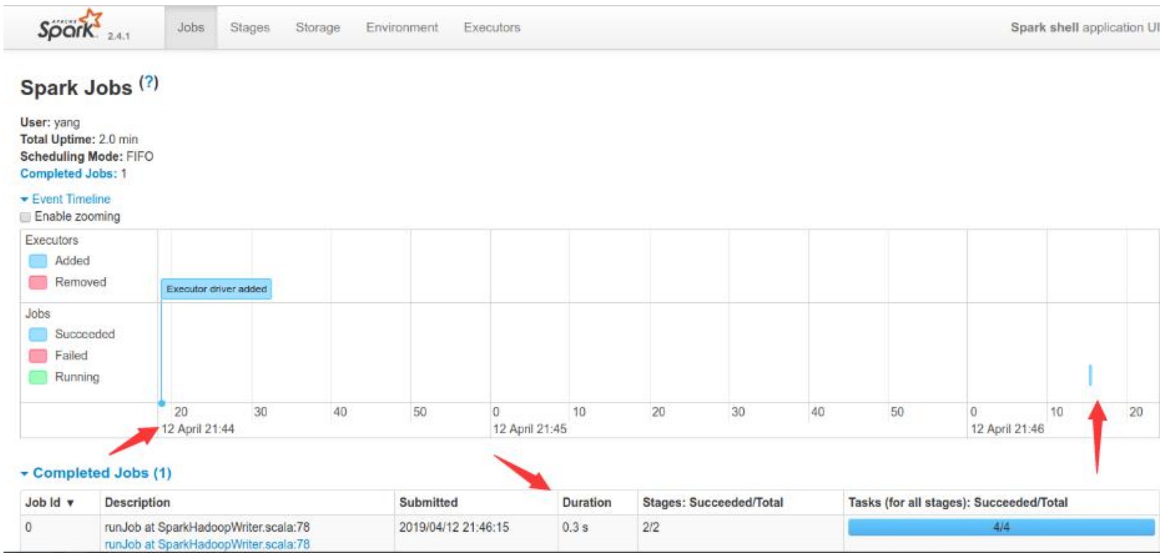
Heap Memory used 202.33 MB of 494.5 MB Heap Memory. Max Heap Memory is 889 MB.

Non Heap Memory used 58.73 MB of 59.75 MB Commited Non Heap Memory. Max Non Heap Memory is -1 B.

Configured Capacity:	212.21 GB
DFS Used:	572 KB (0%)
Non DFS Used:	41.16 GB
DFS Remaining:	160.25 GB (75.51%)
Block Pool Used:	572 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	1 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	0
Number of Blocks Pending Deletion	0
Block Deletion Start Time	4/12/2019, 5:23:32 PM

4.5.2 Spark 执行效率

如下图所示，从图中可以观察到，Spark 执行词频统计几乎瞬间完成，运算效率及消耗的硬件资源都相对 MapReduce 少。Spark 运行效率和成本相对于 MapReduce 方式减少非常明显。



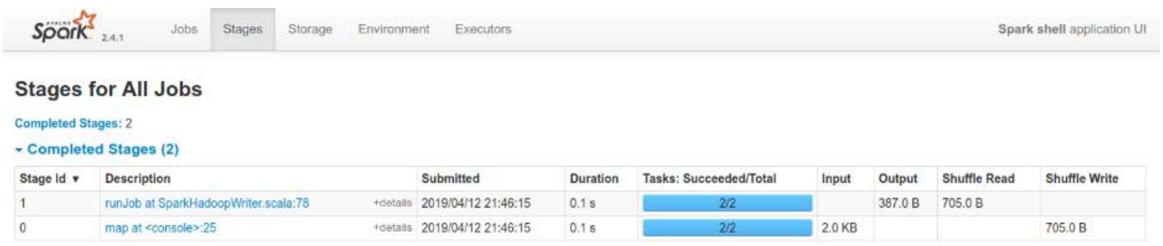
Spark Jobs (?)

User: yang
Total Uptime: 2.0 min
Scheduling Mode: FIFO
Completed Jobs: 1

▾ Event Timeline
☐ Enable zooming

▾ Completed Jobs (1)

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	runJob at SparkHadoopWriter.scala:78 runJob at SparkHadoopWriter.scala:78	2019/04/12 21:46:15	0.3 s	2/2	4/4

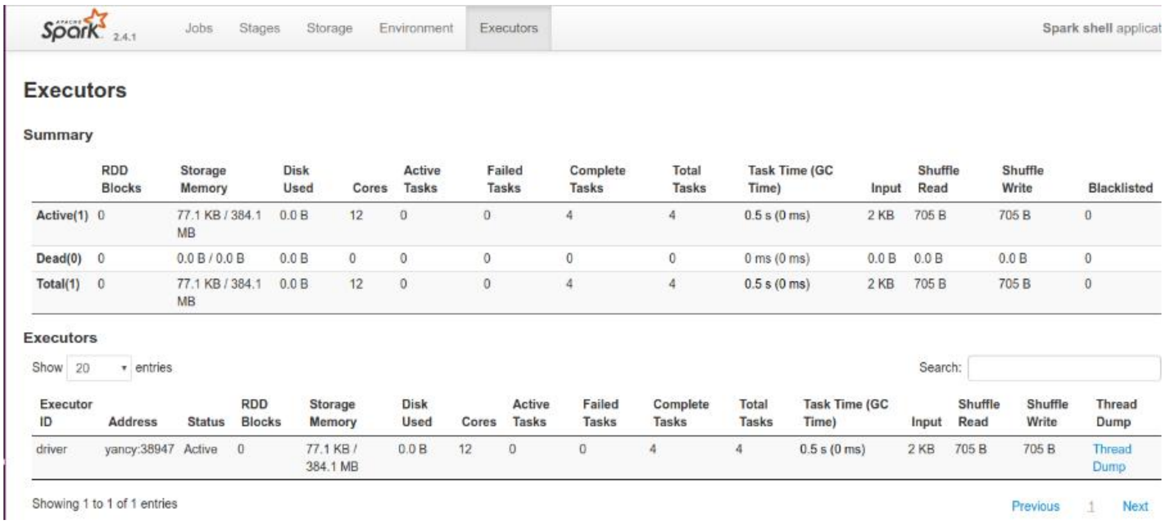


Stages for All Jobs

Completed Stages: 2

▾ Completed Stages (2)

Stage Id ▾	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	runJob at SparkHadoopWriter.scala:78	+details	2019/04/12 21:46:15	0.1 s	2/2		387.0 B	705.0 B	
0	map at <console>:25	+details	2019/04/12 21:46:15	0.1 s	2/2	2.0 KB			705.0 B



Executors

Summary

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Blacklisted
Active(1)	0	77.1 KB / 384.1 MB	0.0 B	12	0	0	4	4	0.5 s (0 ms)	2 KB	705 B	705 B	0
Dead(0)	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B	0
Total(1)	0	77.1 KB / 384.1 MB	0.0 B	12	0	0	4	4	0.5 s (0 ms)	2 KB	705 B	705 B	0

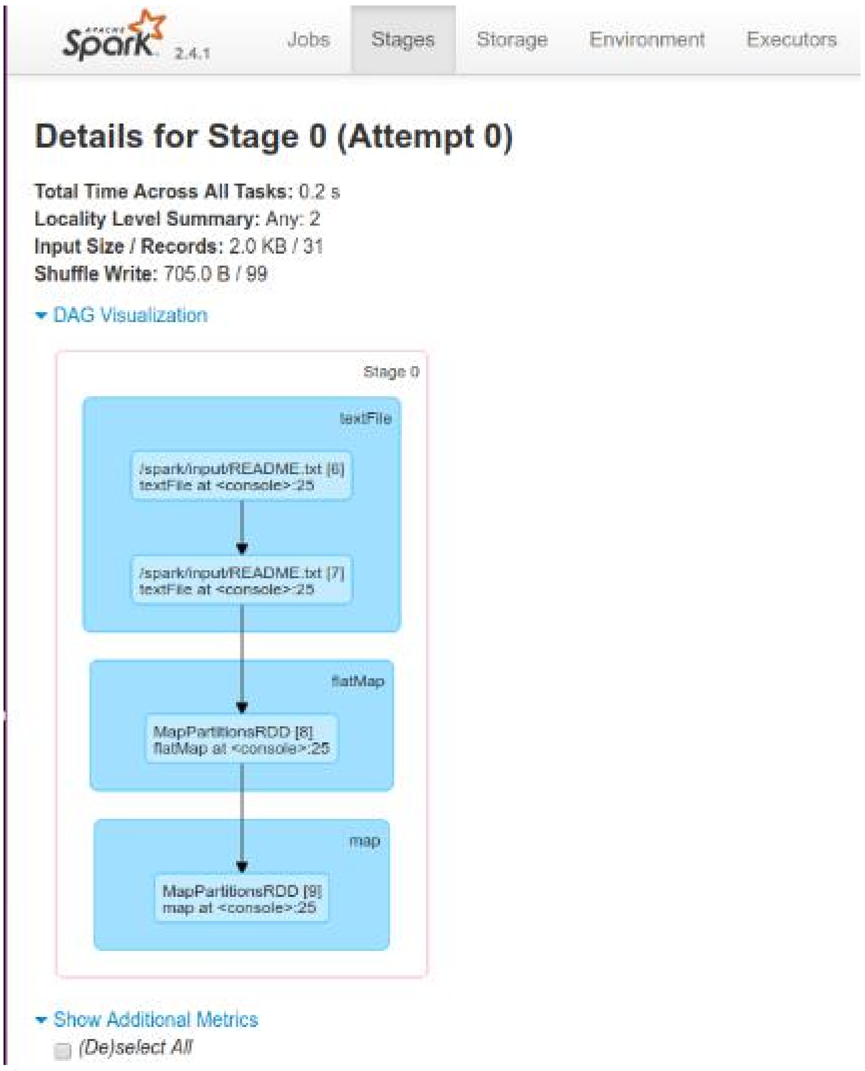
Executors

Show 20 ▾ entries Search:

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Thread Dump
driver	yancy:38947	Active	0	77.1 KB / 384.1 MB	0.0 B	12	0	0	4	4	0.5 s (0 ms)	2 KB	705 B	705 B	Thread Dump

Showing 1 to 1 of 1 entries

Previous 1 Next



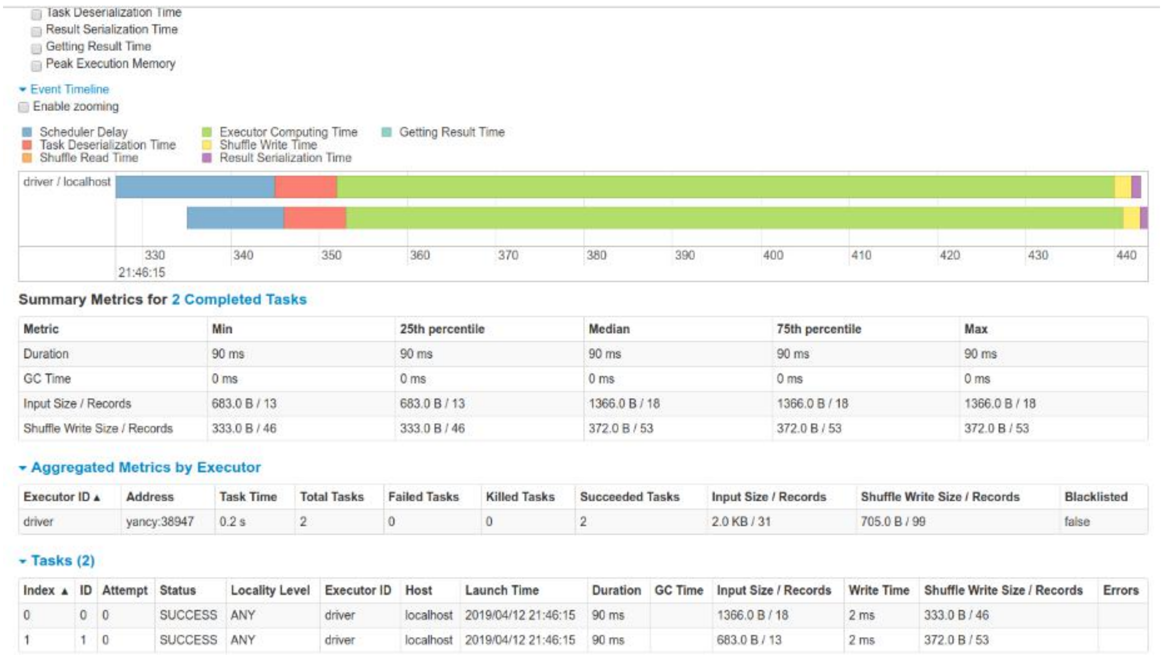
Details for Stage 0 (Attempt 0)

Total Time Across All Tasks: 0.2 s
Locality Level Summary: Any: 2
Input Size / Records: 2.0 KB / 31
Shuffle Write: 705.0 B / 99

▾ DAG Visualization

▾ Show Additional Metrics

☐ *(De)select All*



☐ Task Deserialization Time
☐ Result Serialization Time
☐ Getting Result Time
☐ Peak Execution Memory

▾ Event Timeline

☐ Enable zooming

Summary Metrics for 2 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	90 ms	90 ms	90 ms	90 ms	90 ms
GC Time	0 ms	0 ms	0 ms	0 ms	0 ms
Input Size / Records	683.0 B / 13	683.0 B / 13	1366.0 B / 18	1366.0 B / 18	1366.0 B / 18
Shuffle Write Size / Records	333.0 B / 46	333.0 B / 46	372.0 B / 53	372.0 B / 53	372.0 B / 53

▾ **Aggregated Metrics by Executor**

Executor ID ▴	Address	Task Time	Total Tasks	Failed Tasks	Killed Tasks	Succeeded Tasks	Input Size / Records	Shuffle Write Size / Records	Blacklisted
driver	yancy:38947	0.2 s	2	0	0	2	2.0 KB / 31	705.0 B / 99	false

▾ **Tasks (2)**

Index ▴	ID	Attempt	Status	Locality Level	Executor ID	Host	Launch Time	Duration	GC Time	Input Size / Records	Write Time	Shuffle Write Size / Records	Errors
0	0	0	SUCCESS	ANY	driver	localhost	2019/04/12 21:46:15	90 ms		1366.0 B / 18	2 ms	333.0 B / 46	
1	1	0	SUCCESS	ANY	driver	localhost	2019/04/12 21:46:15	90 ms		683.0 B / 13	2 ms	372.0 B / 53	