《大数据处理实践》
实验一 大数据处理平台基本原理、环境搭建及参数性能试验

华中科技大学科学与技术学院
大数据课程组

莫益军、石宣化、姚德中、郑龙

2020 年 9 月

目录

1. 实验目的

大数据处理平台 Hadoop 是 Apache 基于 Google 的三篇论文《分布式文件系统》、《分布式计算模型》和《Bigtable》的开源系统框架。Hadoop 基本框架包含 HDFS（分布式文件系统）、YARN（资源任务管理）和 MapReduce 编程模型基础类，具体涉及到大数据文件在分布式存取过程中的数据切片、数据组织、副本管理、一致性保证、复制策略和读写访问等，其中各功能模块参数对分布式文件系统的 I/O 性能及分布书大数据处理性能影响较大。

本实验基于 Docker 容器模拟 4 台机器搭建 hadoop 平台，从黑盒测试和使用角度了解大数据处理系统的模块结构和运行原理，通过对比本机文件系统结构，加深对 HDFS 中系统框架、组件功能、数据一致性、故障容错和副本策略等的理解，以达到以下目的：

1、 通过运行脚本和进程状态了解分布式文件系统和分布式处理系统的系统结构和功能模块，并能对系统故障进行故障定位排除；

2、 了解 Hadoop 分布式文件系统的结构及其在本地文件系统的映射结构，加深对 Hadoop 集群、数据节点、数据版本一致性、数据块、副本、元数据等的理解；

3、 了解副本数量对实际存储空间的影响及副本在数据节点上复制的顺序；

4、 了解 HDFS 数据块大小对不同大小的数据文件在 HDFS 系统存取性能的影响；

5、 了解 HDFS 故障容错机制及其对数据一致性的影响；

6、 了解副本复制的机架感知策略配置方法；

7、 了解 Hadoop 核心组件参数对 HDFS 存取性能的影响；

8、 自行设计 HDFS 黑盒试验加深对 HDFS 原理的影响。

除上述分布式系统和 HDFS 的目的之外，在系统环境搭建过程中顺带了解：

1、 分布式系统运行的必备条件，如免密登陆和可信远程拷贝；

2、 对分布式运维部署和 Devop 等概念有初步认识。

2.实验评分构成

本实验以环境搭建、黑盒测试和观察理解为主。实验评分以满分 100 分计算完成 1-2 的基本实验要求为 60 分，每增加两个实验目的增加 10 分。

3. Hadoop 原理及组成

Hadoop 分布式处理平台包括分布式文件系统和任务资源调度系统两大核心系统。

3.1 分布式文件系统

分布式文件系统（HDFS）解决分布式文件存储问题，包括命名节点（Name Node）、数据节点（Data Node）和客户端三大核心组件。

命名节点：该组件负责 HDFS 中元数据管理和存储，包括 HDFS 数据版本、分块、位置和索引等信息。HDFS 数据上传、拷贝和删除时，数据节点告知命名节点更新相应的元数据存储结构，类似单机操作系统中的文件分配表（FAT）。HDFS 系统中命名节点是集中式的，为保证系统可靠性，命名节点有主备两个节

点，可以运行一台机器或主备两台机器上。

数据节点 :该组件负责 HDFS 中具体数据文件的切片、存储与副本存储。HDFS 数据上传、拷贝和删除时，首先对数据进行切片，然后根据 HDFS 中配置将切片复制到多个数据节点上成为副本，并保证副本之间的一致性。数据节点运行在多个分布式节点上。

客户端 :HDFS 客户端实现命名行输入指令的解析，以方便用户和应用对 HDFS 文件系统的文件操作，包括文件目录创建、删除、拷贝和文件的上传、创建、拷贝和删除，其操作与 Linux 系统中的命令类似。

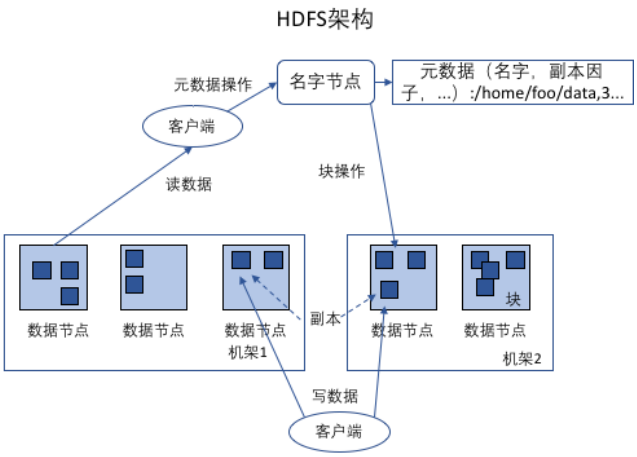

图 1 Hadoop 中 HDFS 系统相关组件

Hadoop 中的 HDFS 系统相关的组件的结构如图 1 所示。

3.2 任务资源调度系统

任务资源调度系统（YARN）解决分布式作业和任务调度问题，包括资源管理器（Resource Manager）、节点管理器（Node Manager）和客户端三大核心组件。

资源管理器：负责汇总节点管理器状和资源分配请求，接收客户端的作业执

行请求，调度分布式资源执行具体应用作业。

节点管理器：节点管理管理器负责本节点的任务调度和资源容器管理。客户端上产生的应用执行请求是一个作业请求，拆分为多个任务请求，任务在容器中运行，包括 Map 任务、Reduce 任务和 MPI 任务。应用控制器负责作业的管理和任务调度和状态监控。

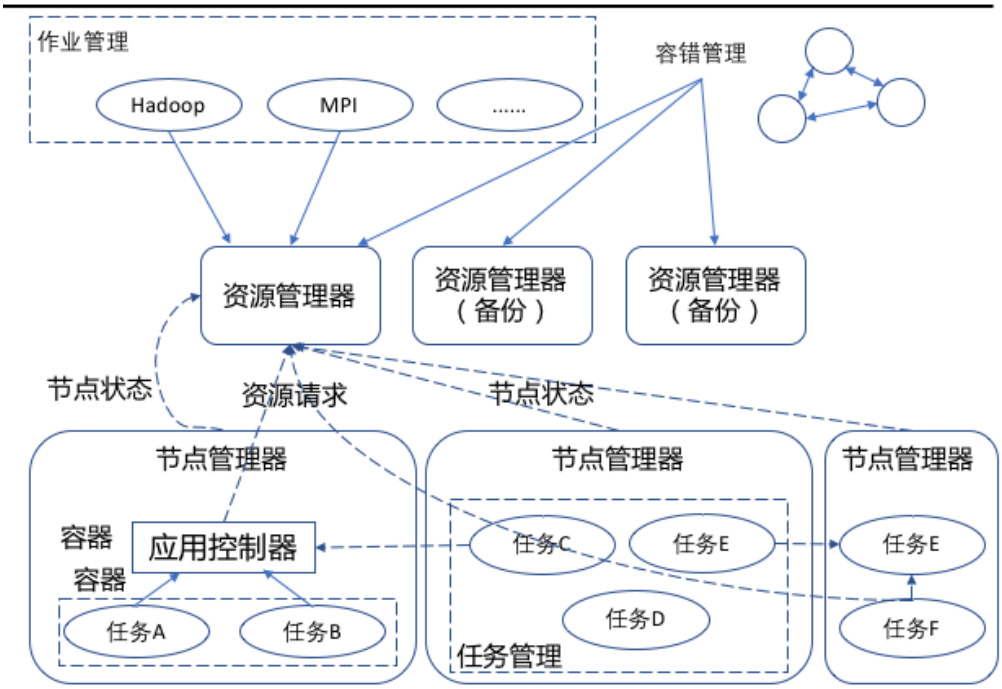

图 2 Hadoop 中任务资源管理相关组件

Hadoop 中的任务资源管理结构如图 2 所示。

3.3 Hadoop 系统的任务进程

Hadoop 中 HDFS 组件和资源管理组件以独立进程形式运行，具体进程与组件关系如图 3 所示。

图中示例由 1 个主节点和 3 个从节点组成，主节点运行完整的 HDSF 组件和任务资源组件。完整的 HDFS 系统组件包括主从 NameNode 和 DataNode，完整

的任务资源系统组件包括 ResourceManager 和 NodeManger。从节点仅运行 DataNode 和 NodeManager 组件。

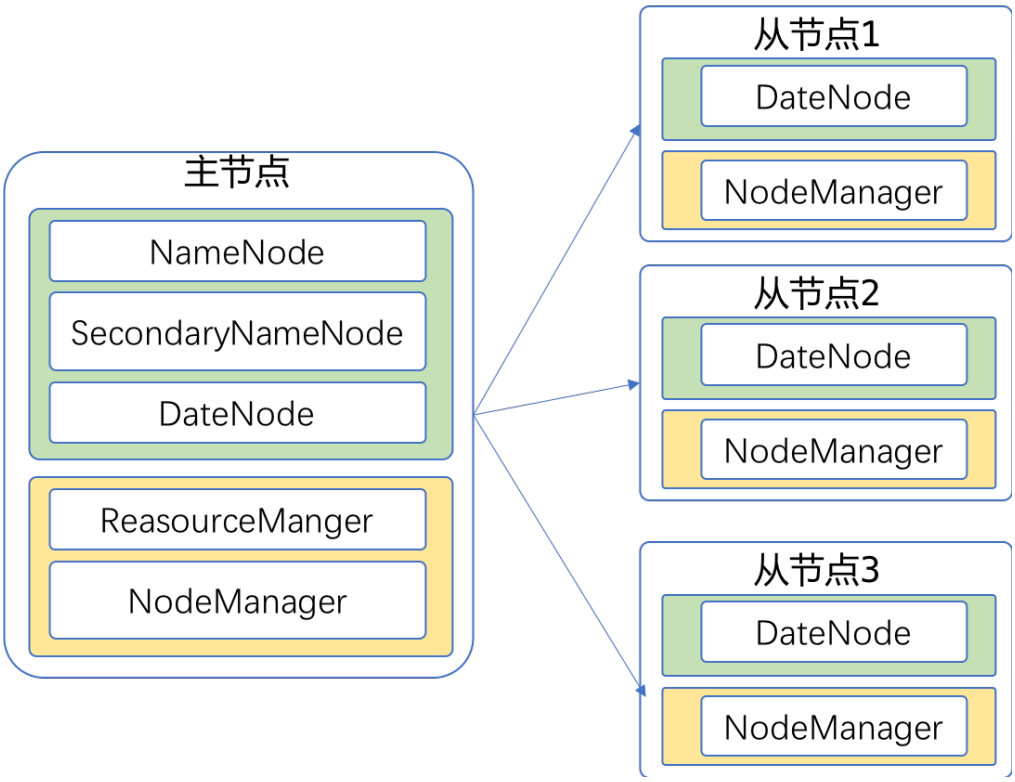

图 3 进程组件关系图

4. 实验环境搭建流程

本次实验环境在单主机上运行多个 Docker 实例来实现大数据处理平台。本章对相应的环境和流程进行描述。

4.1 概念

宿主机：本实验中，宿主机是指用来安装 docker 容器环境的工作主机，可以是 Windows、Linux 和 MAC OS 等任意物理主机环境，也可以 VirtualBox 或 VMWare 搭建的虚拟主机环境。

客户端：本实验中，客户端特指连接 HDFS 和 Hadoop 集群的客户端，既可

以是宿主机上的 SSH 客户端，或者是通过 Web 浏览器访问 Hadoop 集群的客户端，也可以是 Docker 容器中各节点的命令行客户端。

Docker 容器 :本实验中，宿主机通过 Docker 容器模拟物理主机，运行 Hadoop 实例。Docker 容器中运行操作系统的基本环境和 Hadoop 应用。每个 Docker 实例相当于裁剪的独立主机，可独立运行，拥有各自独立的 IP 和文件系统。实验过程中可以独立备份恢复镜像，方便配置或环境部署过程中的错误恢复。

本地文件系统：本实验中，本地文件系统特指 Docker 容器中承载 Hadoop 集群和存储 HDFS 文件系统的主机文件系统。

分布式文件系统：本实验中，分布式文件系统是由命名节点维护，通过 Hadoop 分布式节点存储在多个数据节点上的文件系统，在客户端上可以对分布式文件系统进行文件目录创建、拷贝和删除操作。分布式文件系统中的文件和目录以数据切片文件存储在本地文件系统中。

副本：本实验中，副本是指对客户端某一数据文件在 HDFS 上存储的多份，一个数据文件被拆分成多个数据块，每个数据块在 HDFS 上的多个节点上都有多个副本备份。

数据节点：本实验中，数据节点是指文件系统相关进程运行和数据块存储的节点，

任务节点：本实验中，任务节点是指分布式处理任务运行的节点。任务节点和数据节点可以不是同一个节点。

4.2 流程

了解上述基本概念后，本次实验的流程涉及到宿主机、Docker 虚拟主机的安

装与配置，对 Docker 虚拟主机来说，涉及命名节点和数据节点、资源管理节点和任务节点的配置。本实验涉及的流程步骤如图 4 所示。

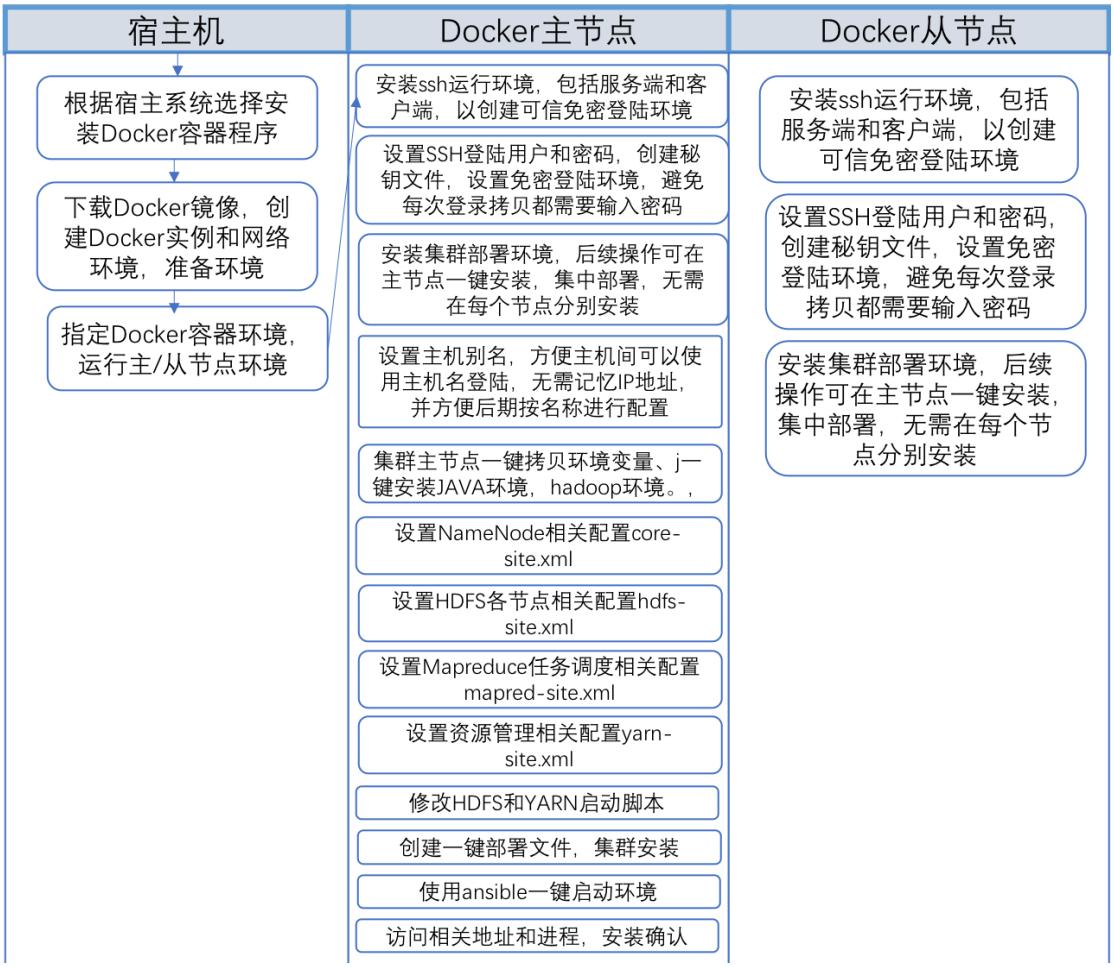

图 4 Hadoop 平台安装流程及节点关系

5 实验内容及步骤

5.1 实验内容构成

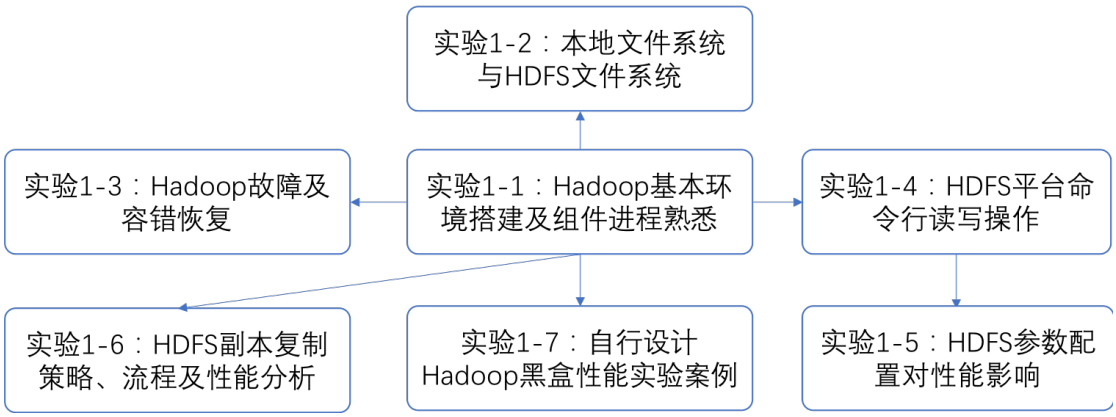

图 5 实验一内容

5.2 实验步骤

5.2.1 实验 1-1

实验名称：Hadoop 基本环境搭建及组件进程熟悉

实验操作步骤：

1、 实验准备

根据实验机器性能和操作系统, 选择相应的 Docker 应用程序版本进行安装。

2、 拉取 CentOS 镜像版本

在宿主机命令行终端执行以下命令

docker pull daocloud.io/library/centos:latest

3、 创建 Docker 容器的网络环境

在宿主机命令行终端执行以下命令

docker network create --subnet=10.0.0.0/16 netgroup

4、 创建 Docker 容器的网络环境

在宿主机命令行终端执行以下命令

docker network create --subnet=10.0.0.0/16 netgroup

注：创建固定网络，最好与宿主机不属于同一网络，可根据自己偏好设置网络前缀。

5、 拉取并运行 centos 镜像

在宿主机命令行终端执行以下命令，创建运行主节点

docker run -d -p 18088:18088 -p 9870:9870 --privileged -ti -v /sys/fs/cgroup:/sys/fs/cgroup --name cluster-master -h cluster-master --net netgroup --ip 10.0.0.2 daocloud.io/library/centos /usr/sbin/init

注：

a) 端口 18088 是 YARN 中资源管理器的 WEB 服务端口，可在 yarn-site.xml 修改

b) 端口 9870 是 HDFS 中节点管理器的 WEB 服务端口，可在 core-site.xml 修改

c) –ip 指定 IP 地址，--name cluster-master -h cluster-master 指定主机名，--net netgroup 指定网络号

d) 详细命令可以查询 docker 运行帮助手册。

在宿主机命令行终端执行以下命令，创建运行多个从节点，参考以下命令：

docker run -d --privileged -ti -v /sys/fs/cgroup:/sys/fs/cgroup --name cluster-slave1 -h cluster-slave1 --net netgroup --ip 10.0.0.3 daocloud.io/library/centos /usr/sbin/init

Docker 创建运行成功后，会返回 docker 镜像实例 ID

6、 启动控制台并进入容器

在宿主机命令行终端执行以下命令：

docker exec -it cluster-master /bin/bash

也可以使用 docker 应用程序中的 dashboard 进入

7、 在所有镜像上安装并启动 ssh 服务，为免密登陆做准备

在镜像主机命令行终端执行以下命令

yum -y install openssh openssh-server openssh-clients

systemctl start sshd

注：，参与 Hadoop 的主从节点全部安装运行。

8、 在所有镜像上设置免密登陆

在所有镜像主机命令行终端执行以下命令

vi /etc/ssh/ssh_config，启动编辑器

查找到配置文件中的

StrictHostKeyChecking ask

将其修改为

StrictHostKeyChecking no

重新启动 sshd 服务

systemctl restart sshd

因拉取的 centos 最小镜像无 passwd 工具，需安装

yum -y install passwd

设置 root 用户的密码，执行

passwd

设置成自己偏好的密码。

创建用于免密登陆的秘钥文件，建立集群节点的可信关系，执行以下指令

每个节点都配置好 ssh 环境后，可以通过 ssh 和 scp 在主节点进行拷贝配置，无需分别登陆各镜像节点的终端进行操作。具体执行以下操作：

```
ssh-keygen -t rsa
ssh root@cluster-slave1 'mkdir ~/.ssh'
scp ~/.ssh/authorized_keys root@cluster-slave1:~/.ssh
ssh root@cluster-slave2 'mkdir ~/.ssh'
scp ~/.ssh/authorized_keys root@cluster-slave2:~/.ssh
ssh root@cluster-slave3 'mkdir ~/.ssh'
scp ~/.ssh/authorized_keys root@cluster-slave3:~/.ssh
```

9、 安装 ansible 工具，准备环境，以便在主节点上进行一键安装和集中部署

主节点镜像终端输入命令行，安装 ansible 包

```
yum -y install epel-release
yum -y install ansible
```

修改 ansible 集群配置文件，在主节点镜像终端中执行

```
vi /etc/ansible/hosts
```

修改配置文件参考如下：

```
[cluster]
cluster-master
cluster-slave1
cluster-slave2
```

```
cluster-slave3

[master]
cluster-master

[slaves]
cluster-slave1
cluster-slave2
cluster-slave3
```

修改主机文件，以便通过主机名称进行连接和登陆。
将/etc/host 文件修改为如下

```
127.0.0.1     localhost
10.0.0.2    cluster-master
10.0.0.3    cluster-slave1
10.0.0.4    cluster-slave2
10.0.0.5    cluster-slave3
```

为了避免重启动后/etc/host 文件被重写，可以修改.bashrc 配置文件如下：

```
~/.bashrc
:>/etc/hosts
cat >>/etc/hosts<<EOF
127.0.0.1     localhost
10.0.0.2    cluster-master
10.0.0.3    cluster-slave1
10.0.0.4    cluster-slave2
10.0.0.5    cluster-slave3
```

执行以下命令
source ~/.bashrc 以生成相应环境
将上述配置分发给所有集群中的节点
ansible cluster -m copy -a "src=~/.bashrc dest=~/"

在集群中一键安装 java 运行环境
ansible cluster -m yum -a "name=java-1.8.0-openjdk,java-1.8.0-openjdk-devel state=latest"

下载 hadoop 安装包，解压至相关目录，为保证下载速度，可以选择国内镜像，如清华镜像
cd /opt
wget https://mirrors.tuna.tsinghua.edu.cn/apache/hadoop/common/hadoop-3.2.1/hadoop-3.2.1.tar.gz
tar -xzvf hadoop-3.2.1.tar.gz

创建 hadoop 解压目录的软链接
ln -s hadoop-3.2.1 hadoop

设置 JAVA 和 HADOOP 的环境变量，修改.bashrc，增加以下内容
export HADOOP_HOME=/opt/hadoop
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.262.b10-0.el8_2.x86_64
#注意版本和目录
export PATH=$JAVA_HOME/bin:$PATH

执行 source ~/.bashrc 生效配置

10、 主节点配置 hadoop 核心的四个配置文件 core-site.xml、hdfs-site.xml、mapred-site.xml、yarn-site.xml

配置文件目录参考：/opt/hadoop/etc/hadoop

按如下配置修改文件，后续试验会修改也会继续修改这 4 个配置文件

core-site.xml

```
<configuration>
    <property>
        <name>hadoop.tmp.dir</name>
        <value>/home/hadoop/tmp</value>
        <description>A base for other temporary directories.</description>
    </property>
    <!-- file system properties -->
    <property>
        <name>fs.default.name</name>
        <value>hdfs://cluster-master:9000</value>
    </property>
    <property>
    <name>fs.trash.interval</name>
        <value>4320</value>
    </property>
</configuration>
```

hdfs-site.xml

```
<property>
```

```
    <name>dfs.namenode.name.dir</name>
    <value>/home/hadoop/tmp/dfs/name</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>/home/hadoop/data</value>
  </property>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
  <property>
    <name>dfs.webhdfs.enabled</name>
    <value>true</value>
  </property>
  <property>
    <name>dfs.permissions.superusergroup</name>
    <value>staff</value>
  </property>
  <property>
    <name>dfs.permissions.enabled</name>
    <value>false</value>
  </property>
```

mapred-site.xml

```
  <configuration>
<property>
   <name>mapreduce.framework.name</name>
   <value>yarn</value>
</property>
<property>
        <name>mapred.job.tracker</name>
        <value>cluster-master:9001</value>
</property>
<property>
   <name>mapreduce.jobtracker.http.address</name>
   <value>cluster-master:50030</value>
</property>
<property>
   <name>mapreduce.jobhisotry.address</name>
   <value>cluster-master:10020</value>
</property>
<property>
```

```
    <name>mapreduce.jobhistory.webapp.address</name>
    <value>cluster-master:19888</value>
</property>
<property>
    <name>mapreduce.jobhistory.done-dir</name>
    <value>/jobhistory/done</value>
</property>
<property>
    <name>mapreduce.intermediate-done-dir</name>
    <value>/jobhisotry/done_intermediate</value>
</property>
<property>
    <name>mapreduce.job.ubertask.enable</name>
    <value>true</value>
</property>
</configuration>
```

yarn-site.xml

```
<configuration>
      <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>cluster-master</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
  <property>
    <name>yarn.resourcemanager.address</name>
    <value>cluster-master:18040</value>
  </property>
<property>
    <name>yarn.resourcemanager.scheduler.address</name>
    <value>cluster-master:18030</value>
  </property>
  <property>
    <name>yarn.resourcemanager.resource-tracker.address</name>
    <value>cluster-master:18025</value>
  </property> <property>
```

```
    <name>yarn.resourcemanager.admin.address</name>
    <value>cluster-master:18141</value>
 </property>
<property>
    <name>yarn.resourcemanager.webapp.address</name>
    <value>cluster-master:18088</value>
 </property>
<property>
    <name>yarn.log-aggregation-enable</name>
    <value>true</value>
 </property>
<property>
    <name>yarn.log-aggregation.retain-seconds</name>
    <value>86400</value>
 </property>
<property>
    <name>yarn.log-aggregation.retain-check-interval-seconds</name>
    <value>86400</value>
 </property>
<property>
    <name>yarn.nodemanager.remote-app-log-dir</name>
    <value>/tmp/logs</value>
 </property>
<property>
    <name>yarn.nodemanager.remote-app-log-dir-suffix</name>
    <value>logs</value>
 </property>
</configuration>
```

修改 Hadoop 相关启停脚本, 将 HDFS 分布式文件系统的 start-dfs.sh 和 stop-dfs.sh 启停文件增加以下内容

```
#!/usr/bin/env bash
HDFS_DATANODE_USER=root
HADOOP_SECURE_DN_USER=hdfs
HDFS_NAMENODE_USER=root
HDFS_SECONDARYNAMENODE_USER=root
```

将 Hadoop 资源任务相关的 start-yarn.sh 和 stop-yarn.sh 启停文件增加以下内容

```
#!/usr/bin/env bash
YARN_RESOURCEMANAGER_USER=root
HADOOP_SECURE_DN_USER=yarn
YARN_NODEMANAGER_USER=root
```

11、 使用 ansible 进行集中安装，一键部署

进入/opt 目录：

cd /opt

tar -cvf hadoop-dis.tar hadoop-3.2.1

创建 hadoop-dis.yaml 分发文件

```
---
- hosts: cluster
  tasks:
    - name: copy .bashrc to slaves
      copy: src=~/.bashrc dest=~/
      notify:
        - exec source
    - name: copy hadoop-dis.tar to slaves
      unarchive: src=/opt/hadoop-dis.tar dest=/opt

  handlers:
    - name: exec source
      shell: source ~/.bashrc
```

开始集中部署

ansible-playbook hadoop-dis.yaml

至此相关程序和配置完成

12、 对 hadoop 集群中进行 HDFS 文件系统的格式化和程序启动

hadoop namenode -format

cd $HADOOP_HOME/sbin

完整启动 hadoop 平台

start-all.sh

如果仅启动 HDFS，只需执行 start-dfs.sh

13、 检查 Hadoop 的 Java 进程，看是否正常运行

jps

若包含以下 java 进程则表示安装启动成功

主节点

```
ResourceManager
DataNode
NameNode
NodeManager
SecondaryNameNode
```

从节点

```
DataNode
NodeManager
```

14、 访问相关网址，确认系统是否安装完成

http://10.0.0.2:18088

http://10.0.0.2:9870

5.2.2 实验 1-2

实验名称：本地文件系统与 HDFS 文件系统操作及组织结构

1、 HDFS 文件系统操作命令

主节点镜像的终端尝试目录创建、目录列表和文件上产等命令

#创建目录

hadoop fs -mkdir /input

#目录列表

hadoop fs -ls /

#上传文件

hadoop fs -put LICENSE.txt /input

选择一个 1G 左右的文件上传到 HDFS 系统，至少为 200MB 以上的文件。

2、 通过 Hadoop 节点管理器访问 HDFS 的目录文件，观察上传的本地文件在系统中的存储大小。

3、 完成上述操作后，观察并分析本地文件系统的目录和 HDFS 文件系统目录的关系。具体在主节点镜像主机上观察以下目录

/home/hadoop/data/current

尤其观察类似以下目录，理解 HDFS 文件切片和元数据

/home/hadoop/data/current/BP-1356942046-10.0.0.2-1598150555243/current/finalized/subdir0/subdir0

同时观察以下目录，理解 namenode 和元数据

/home/hadoop/tmp/dfs/name/current

对比上述和下面目录的

/home/hadoop/tmp/dfs/namesecondary/current

4、 在上述文件上传过程中，同时监测主节点和从节点的上述目录的动态变化

5、 结合 HDFS 原理分析上述观察结果。

5.2.3 实验 1-3

名称：Hadoop 故障及容错恢复

NameNode 主备实验

1、停止 SecondaryNameNode

2、在 HDFS 上进行文件上传和删除操作

3、对比 NameNode 和 SecondaryNameNode 目录的变化

4、监测 HDFS 目录动态变化

5、停止 NameNode

6、监测 HDFS 目录动态变化

7、启动 SecondaryNameNode

8、监测 HDFS 目录动态变化

DataNode 故障恢复实验

1、在主节点删除实验 1-2 中上传第 1 步中上传的大文件（1GB 左右）

2、删除过程中，快速随机制造故障，关停从节点

3、监测主节点和从节点和 HDFS 目录的动态变化

4、恢复启动之前关停节点，监测主节点和从节点和 HDFS 目录的动态变化

5、等待一段时间后，监测主节点和从节点和 HDFS 目录的动态变化

6、分析总结分布式文件系统的一致性和故障容错恢复机制

5.2.4 实验 1-4

名称：Hadoop 参数配置对参数影响

1、修改 hdfs-site.xml 中参数，观察读写性能

2、将 dfs.block.size 分别设置成为 512KB、16M、32M、64M、128M、256M 的大小，拷贝删除文件

3、使用 top 统计在 docker 节点的 CPU、内存变化

4、统计完成任务的时间

5、修改 dfs.namenode.handler.count 、dfs.datanode.handler.count dfs.datanode.max.xcievers 等参数，重复上述 2-4 中的步骤

6、修改 dfs.replication 的数值，观察 HDFS 文件目录变化

5.2.5 实验 1-5

名称：HDFS 副本复制策略、流程及性能分析

1、在上述实验 1-2 到实验 1-4 的参数调整过程中

2、定时观察主节点和 3 个从节点的空间占用大小变化

3、分析 HDFS 副本的复制策略

4、设置机架感知

```
<property>
    <name>net.topology.node.switch.mapping.impl</name>
    <value>org.apache.hadoop.net.TableMapping</value>
</property>
<property>
    <name>net.topology.table.file.name</name>
    <value>/opt/hadoop/rack_topology.data</value>
</property>
```

5、设置机架分布

```
# cat rack_topology.data
10.0.0.2 /rack1
10.0.0.3 /rack3
10.0.0.4 /rack1
10.0.0.5 /rack2
```

6、重启 dfs

7、上传 1GB 左右文件，监测各节点目录结构和存储空间变化

5.2.5 实验 1-6

名称：自行设计 Hadoop 黑盒性能实验案例

根据上述实验 1-1 至实验 1-5，自行设计黑盒实验以验证

例：

1、运行 wordcount MR 任务

Hadoop jar /opt/hadoop/share/hadoop/mapreduce/sources/hadoop-mapreduce-examples-3.2.1-sources.jar
org.apache.hadoop.examples.WordCount demo.txt /output

注：demo.txt 可以复制大于 200MB

2、观察其性能和完成时间

3、修改 hdfs-site.xml 中 dfs.block.size 和 mapred-site.xml 中的 mapred.min.split.size 和 mapred.max.split.size

4、观察其性能和完成时间

5. 相关资源

6. 实验问题及解决方案

问题 1: slave 节点未成功运行 datanode

解决方案：

hadoop 3.0 以后需要修改/opt/hadoop/etc/hadoop/workers，3.0 以前是/etc/hadoop/slaves，将其修改为以下内容

```
cluster-master
cluster-slave1
cluster-slave2
cluster-slave3
```

问题 2: Mac 系统无法访问 docker 内的地址，

解决方案：

设置 socks 代理，具体方法如下：

$ cd ~/Library/Group\ Containers/group.com.docker/

$ mv settings.json settings.json.backup

$ cat settings.json.backup | jq '.["socksProxyPort"]=8888' > settings.json

设置成相应端口

系统偏好设置 -> 网络 -> 高级 -> 代理

设置 SOCKS 代理：localhost:8888 并保存、应用

问题 3：yarn 启动后无法访问 http://localhost:8088/cluster

解决方案：

检查 yarn-site.xml

```
<configuration>
  <!-- Site specific YARN configuration properties -->
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
```