大数据离线阶段

02

一、 课程计划

目录

二、 Apache Hadoop

1. Hadoop 介绍

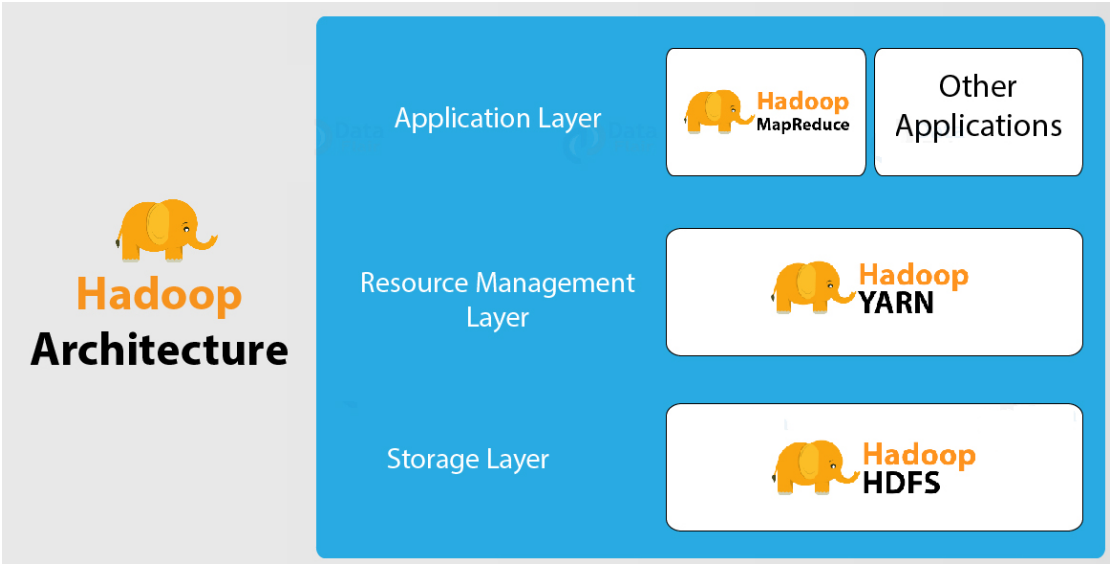
Hadoop 是 Apache 旗下的一个用 java 语言实现开源软件框架，是一个开发和运行处理大规模数据的软件平台。允许使用简单的编程模型在大量计算机集群上对大型数据集进行分布式处理。

狭义上说，Hadoop 指 Apache 这款开源框架，它的核心组件有：

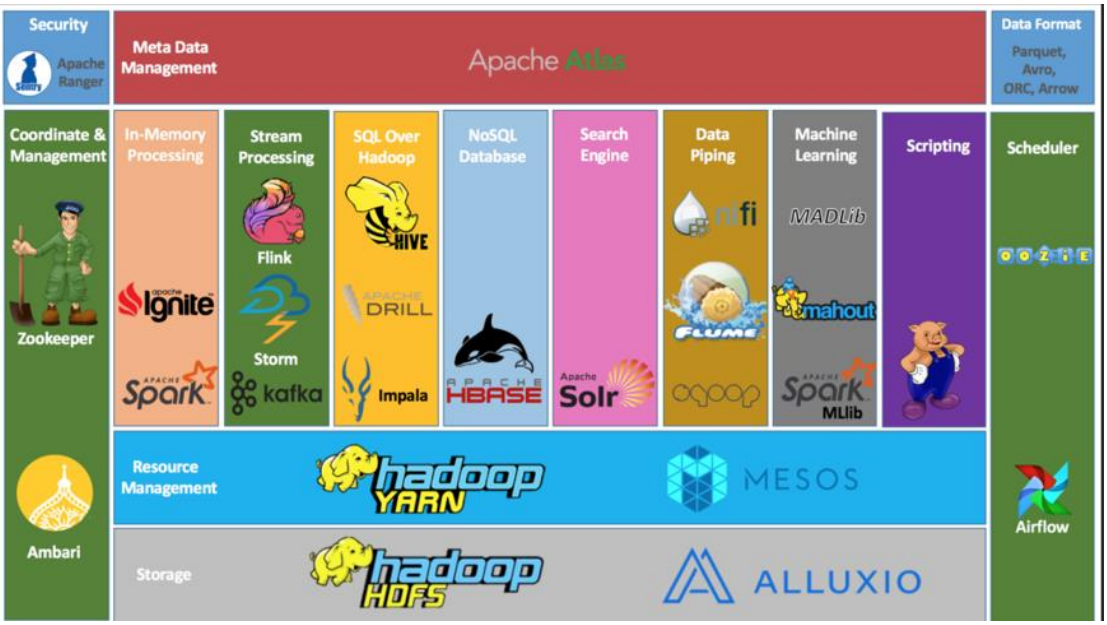
HDFS（分布式文件系统）：解决海量数据存储

YARN（作业调度和集群资源管理的框架）：解决资源任务调度

MAPREDUCE（分布式运算编程框架）：解决海量数据计算



广义上来说，Hadoop 通常是指一个更广泛的概念——Hadoop 生态圈。



当下的 Hadoop 已经成长为一个庞大的体系，随着生态系统的成长，新出现的项目越来越多，其中不乏一些非 Apache 主管的项目，这些项目对 HADOOP 是很好的补充或者更高层的抽象。

2. Hadoop 发展简史

Hadoop 是 Apache Lucene 创始人 Doug Cutting 创建的。最早起源于 Nutch，它是 Lucene 的子项目。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 Google 发表了一篇论文为该问题提供了可行的解决方案。论文中描述的是谷歌的产品架构，该架构称为：谷歌分布式文件系统（GFS），可以解决他们在网页爬取和索引过程中产生的超大文件的存储需求。

2004 年 Google 发表论文向全世界介绍了谷歌版的 MapReduce 系统。

同时期，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。

2006 年 Google 发表了论文是关于 BigTable 的，这促使了后来的 Hbase 的发展。

因此，Hadoop 及其生态圈的发展离不开 Google 的贡献。

3. Hadoop 特性优点

扩容能力（Scalable）：Hadoop 是在可用的计算机集群间分配数据并完成计算任务的，这些集群可用方便的扩展到数以千计的节点中。

成本低（Economical）：Hadoop 通过普通廉价的机器组成服务器集群来分发以及处理数据，以至于成本很低。

高效率（Efficient）：通过并发数据，Hadoop 可以在节点之间动态并行的移动数据，使得速度非常快。

可靠性（Rellable）：能自动维护数据的多份复制，并且在任务失败后能自动地重新部署（redeploy）计算任务。所以 Hadoop 的按位存储和处理数据的能力值得人们信赖。

4. Hadoop 国内外应用

不管是国内还是国外，Hadoop 最受青睐的行业是互联网领域，可以说互联网公司是 hadoop 的主要使用力量。

国外来说，Yahoo、Facebook、IBM 等公司都大量使用 hadoop 集群来支撑业务。比如：

Yahoo 的 Hadoop 应用在支持广告系统、用户行为分析、支持 Web 搜索等。

Facebook 主要使用 Hadoop 存储内部日志与多维数据，并以此作为报告、分析和机器学习的数据源。

国内来说，BAT 领头的互联网公司是当仁不让的 Hadoop 使用者、维护者。比如 Ali 云梯（14 年国内最大 Hadoop 集群）、百度的日志分析平台、推荐引擎系统等。

国内其他非互联网领域也有不少 hadoop 的应用，比如：

金融行业： 个人征信分析

证券行业： 投资模型分析

交通行业： 车辆、路况监控分析

电信行业： 用户上网行为分析

总之：hadoop 并不会跟某种具体的行业或者某个具体的业务挂钩，它只是一种用来做海量数据分析处理的工具。

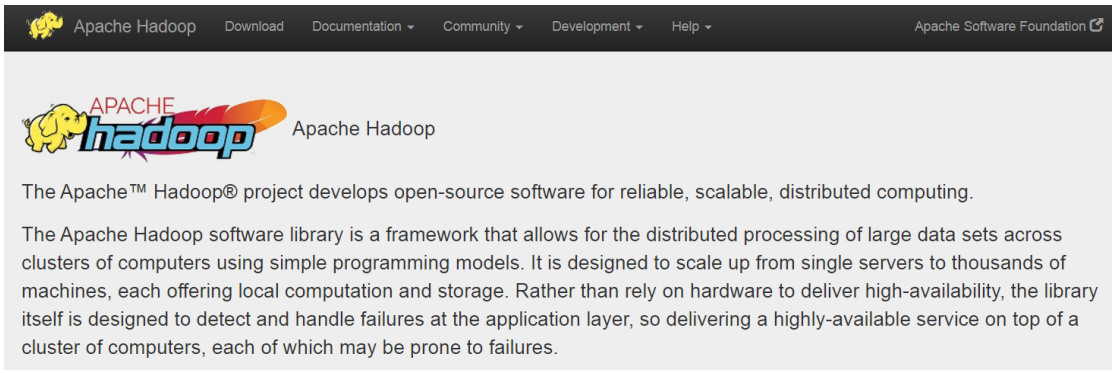
三、 Hadoop 集群搭建

1. 发行版本

Hadoop 发行版本分为开源社区版和商业版。

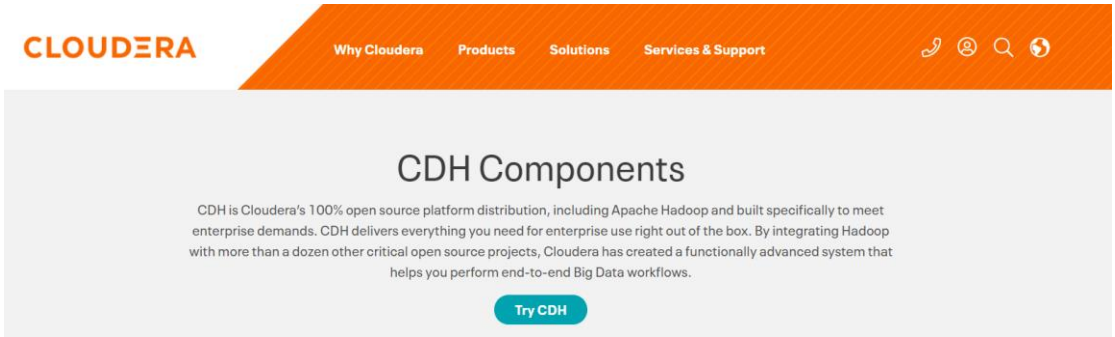
社区版是指由 Apache 软件基金会维护的版本，是官方维护的版本体系。

https://hadoop.apache.org/



商业版 Hadoop 是指由第三方商业公司在社区版 Hadoop 基础上进行了一些修改、整合以及各个服务组件兼容性测试而发行的版本，比较著名的有 cloudera 的 CDH、mapR、hortonWorks 等。

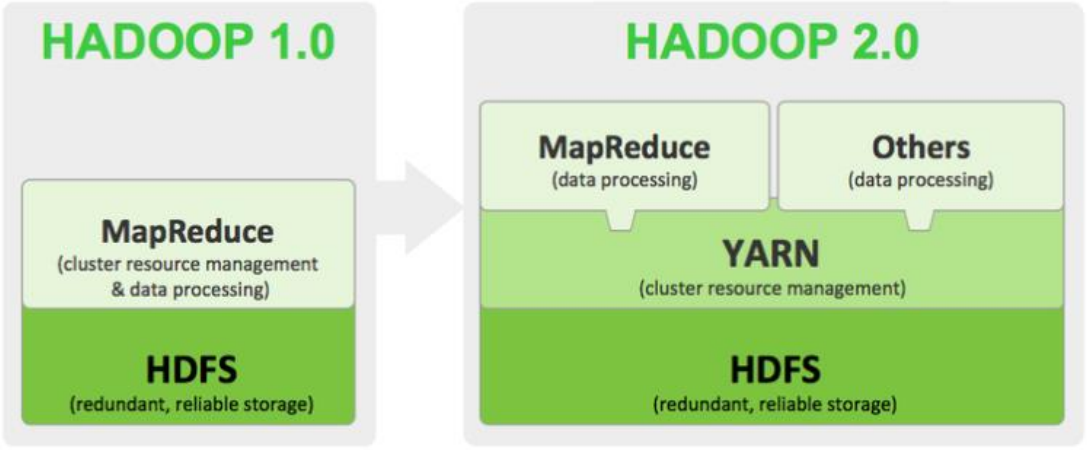
https://www.cloudera.com/products/open-source/apache-hadoop/key-cdh-components.html



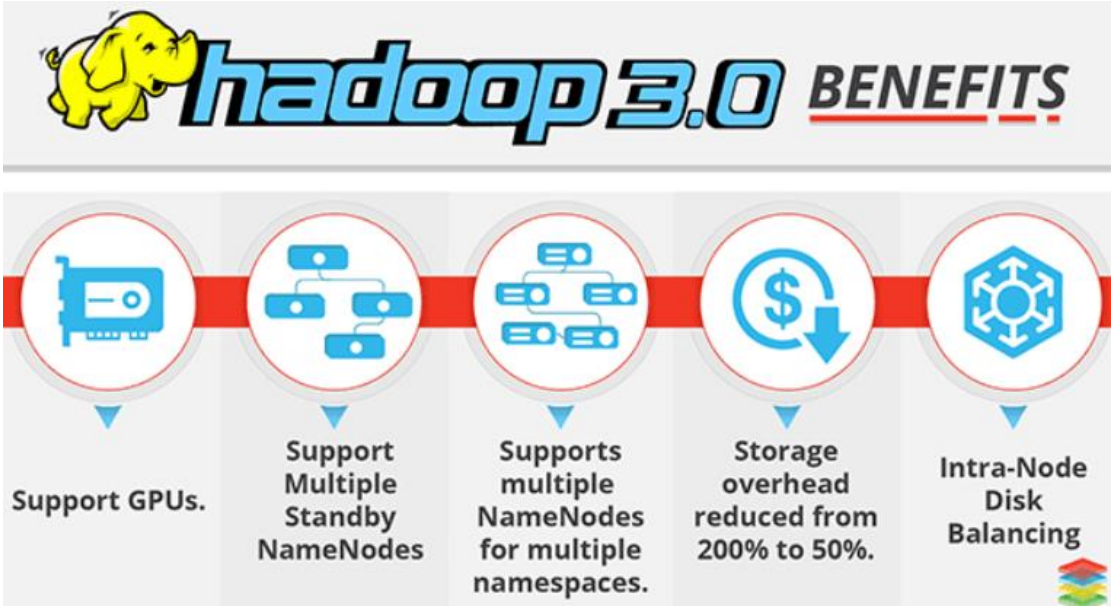
Hadoop 的版本很特殊，是由多条分支并行的发展着。大的来看分为 3 个大的系列版本：1.x、2.x、3.x。

Hadoop1.0 由一个分布式文件系统 HDFS 和一个离线计算框架 MapReduce 组成。架构落后，已经淘汰。

Hadoop 2.0 则包含一个分布式文件系统 HDFS，一个资源管理系统 YARN 和一个离线计算框架 MapReduce。相比于 Hadoop1.0，Hadoop 2.0 功能更加强大，且具有更好的扩展性、性能，并支持多种计算框架。



Hadoop 3.0 相比之前的 Hadoop 2.0 有一系列的功能增强。目前已经趋于稳定，可能生态圈的某些组件还没有升级、整合完善。



我们课程中使用的是：Apache Hadoop 3.3.0。

2. 集群简介

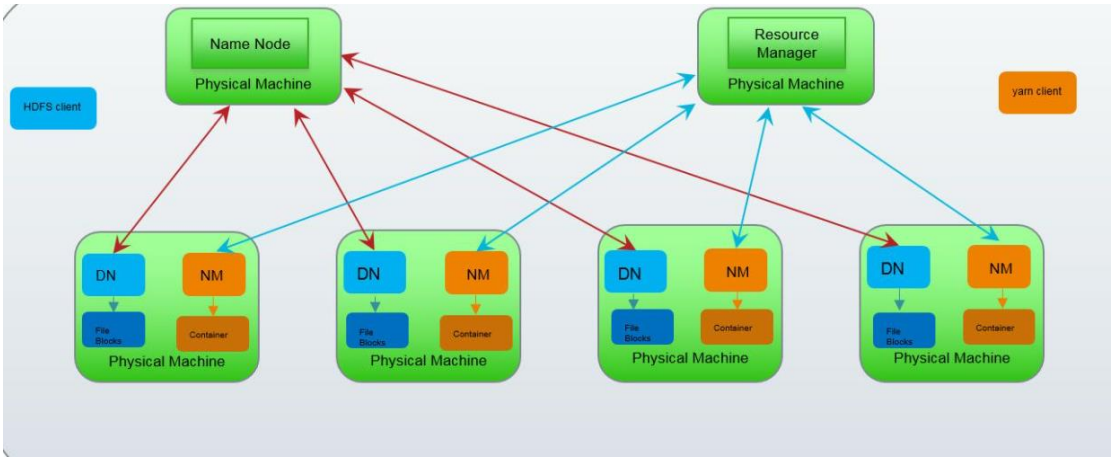
HADOOP 集群具体来说包含两个集群：HDFS 集群和 YARN 集群，两者逻辑上分离，但物理上常在一起。

HDFS 集群负责海量数据的存储，集群中的角色主要有：

NameNode、DataNode、SecondaryNameNode

YARN 集群负责海量数据运算时的资源调度，集群中的角色主要有：

ResourceManager、NodeManager



那 mapreduce 是什么呢？它其实是一个分布式运算编程框架，是应用程序开发包，由用户按照编程规范进行程序开发，后打包运行在 HDFS 集群上，并且受到 YARN 集群的资源调度管理。

Hadoop 部署方式分三种，Standalone mode（独立模式）、Pseudo-Distributed mode（伪分布式模式）、Cluster mode（群集模式），其中前两种都是在单机部署。

独立模式又称为单机模式，仅 1 个机器运行 1 个 java 进程，主要用于调试。

伪分布模式也是在 1 个机器上运行 HDFS 的 NameNode 和 DataNode、YARN 的 ResourceManger 和 NodeManager，但分别启动单独的 java 进程，主要用于调试。

集群模式主要用于生产环境部署。会使用 N 台主机组成一个 Hadoop 集群。这种部署模式下，主节点和从节点会分开部署在不同的机器上。

我们以 3 节点为例进行搭建，角色分配如下：

node1	NameNode	DataNode	ResourceManager
node2	DataNode	NodeManager	SecondaryNameNode
node3	DataNode	NodeManager	

3. 服务器基础环境准备

1.0 配置好各虚拟机的网络（采用 NAT 联网模式）

1.1 修改各个虚拟机主机名

vi /etc/hostname

node1.itcast.cn

1.2 修改主机名和 IP 的映射关系

vi /etc/hosts

192.168.227.151 node1.itcast.cn node1
192.168.227.152 node2.itcast.cn node2
192.168.227.153 node3.itcast.cn node3

1.3 关闭防火墙

#查看防火墙状态
systemctl status firewalld.service
#关闭防火墙
systemctl stop firewalld.service
#关闭防火墙开机启动
systemctl disable firewalld.service

1.4.配置 ssh 免登陆(配置 node1-->node1,node2,node3)
#node1 生成 ssh 免登陆密钥

ssh-keygen -t rsa （一直回车）
执行完这个命令后，会生成两个文件 id_rsa（私钥）、id_rsa.pub（公钥）
将公钥拷贝到要免密登陆的目标机器上
ssh-copy-id node1
ssh-copy-id node2
ssh-copy-id node3

1.5 同步集群时间

yum install ntpdate
ntpdate cn.pool.ntp.org

4. JDK 环境安装

```
2.1 上传 jdk
    jdk-8u65-linux-x64.tar.gz

2.2 解压 jdk
    tar -zxvf jdk-8u65-linux-x64.tar.gz -C /export/server

2.3 将 java 添加到环境变量中
    vim /etc/profile
    #在文件最后添加
    export JAVA_HOME=/export/server/jdk1.8.0_65
    export PATH=$PATH:$JAVA_HOME/bin
    export
CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar

    #刷新配置
    source /etc/profile
```

5. Hadoop 重新编译

Hadoop 官方一般都给出了对应版本安装包，一般情况下是不需要自己进行编译的,但是由于官方编译好的hadoop的安装包没有提供带C程序访问的接口，所以在使用本地库（**本地库可以用来做压缩，以及支持 C 程序等等**）的时候就会出问题，因此生产环境中，一般会重新编译。

此外，作为开源软件，针对源码进行修改某些属性，之后也需要重编译。

可以使用课程提供编译好的安装包。

6. Hadoop 安装包目录结构

解压 hadoop-3.3.0-Centos7-64-with-snappy.tar.gz，目录结构如下：

bin：Hadoop 最基本的管理脚本和使用脚本的目录，这些脚本是 sbin 目录下管理脚本的基础实现，用户可以直接使用这些脚本管理和使用 Hadoop。

etc：Hadoop 配置文件所在的目录，包括 core-site,xml、hdfs-site.xml、mapred-site.xml 等从 Hadoop1.0 继承而来的配置文件和 yarn-site.xml 等 Hadoop2.0 新增的配置文件。

include：对外提供的编程库头文件（具体动态库和静态库在 lib 目录中），这些头文件均是用 C++定义的，通常用于 C++程序访问 HDFS 或者编写 MapReduce 程序。

lib：该目录包含了 Hadoop 对外提供的编程动态库和静态库，与 include 目录中的头文件结合使用。

libexec：各个服务对用的 shell 配置文件所在的目录，可用于配置日志输出、启动参数（比如 JVM 参数）等基本信息。

sbin：Hadoop 管理脚本所在的目录，主要包含 HDFS 和 YARN 中各类服务的启动/关闭脚本。

share：Hadoop 各个模块编译后的 jar 包所在的目录，官方自带示例。

7. Hadoop 配置文件修改

Hadoop 安装主要就是配置文件的修改，一般在主节点进行修改，完毕后 scp 下发给其他各个从节点机器。

7.1. hadoop-env.sh

文件中设置的是 Hadoop 运行时需要的环境变量。JAVA_HOME 是必须设置的，即使我们当前的系统中设置了 JAVA_HOME，它也是不认识的，因为 Hadoop 即使是在本机上执行，它也是把当前的执行环境当成远程服务器。

```
export JAVA_HOME=/export/server/jdk1.8.0_65


#文件最后添加
export HDFS_NAMENODE_USER=root
export HDFS_DATANODE_USER=root
export HDFS_SECONDARYNAMENODE_USER=root
export YARN_RESOURCEMANAGER_USER=root
export YARN_NODEMANAGER_USER=root
```

7.2. core-site.xml

hadoop 的核心配置文件，有默认的配置项 core-default.xml。

core-default.xml 与 core-site.xml 的功能是一样的，如果在 core-site.xml 里没有配置的属性，则会自动会获取 core-default.xml 里的相同属性的值。

```
<property>
    <name>fs.defaultFS</name>
    <value>hdfs://node1:8020</value>
</property>


<property>
```

```
    <name>hadoop.tmp.dir</name>
    <value>/export/data/hadoop-3.3.0</value>
</property>

<!-- 设置 HDFS web UI 用户身份 -->
<property>
    <name>hadoop.http.staticuser.user</name>
    <value>root</value>
</property>

<!-- 整合 hive -->
<property>
    <name>hadoop.proxyuser.root.hosts</name>
    <value>*</value>
</property>

<property>
    <name>hadoop.proxyuser.root.groups</name>
    <value>*</value>
</property>
```

7.3. hdfs-site.xml

HDFS 的核心配置文件，有默认的配置项 hdfs-default.xml。

hdfs-default.xml 与 hdfs-site.xml 的功能是一样的，如果在 hdfs-site.xml 里没有配置的属性，则会自动会获取 hdfs-default.xml 里的相同属性的值。

```
<!-- 指定 secondarynamenode 运行位置 -->
<property>
```

```
    <name>dfs.namenode.secondary.http-address</name>
    <value>node2:50090</value>
</property>
```

7.4. mapred-site.xml

MapReduce 的核心配置文件，有默认的配置项 mapred-default.xml。

mapred-default.xml 与 mapred-site.xml 的功能是一样的，如果在 mapred-site.xml 里没有配置的属性，则会自动会获取 mapred-default.xml 里的相同属性的值。

```
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>

<property>
  <name>yarn.app.mapreduce.am.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>

<property>
  <name>mapreduce.map.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>

<property>
  <name>mapreduce.reduce.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>
```

7.5. yarn-site.xml

YARN 的核心配置文件，有默认的配置项 yarn-default.xml。

yarn-default.xml 与 yarn-site.xml 的功能是一样的，如果在 yarn-site.xml 里没有配置的属性，则会自动会获取 yarn-default.xml 里的相同属性的值。

```
<!-- 指定 YARN 的主角色（ResourceManager）的地址 -->
<property>
    <name>yarn.resourcemanager.hostname</name>
    <value>node1</value>
</property>

<!-- NodeManager 上运行的附属服务。需配置成 mapreduce_shuffle,才可运行 MapReduce 程序默认值：”” -->
<property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
</property>

<!-- 是否将对容器实施物理内存限制 -->
<property>
    <name>yarn.nodemanager.pmem-check-enabled</name>
    <value>false</value>
</property>

<!-- 是否将对容器实施虚拟内存限制。 -->
<property>
    <name>yarn.nodemanager.vmem-check-enabled</name>
    <value>false</value>
</property>
```

```
<!-- 开启日志聚集 -->
<property>
  <name>yarn.log-aggregation-enable</name>
  <value>true</value>
</property>

<!-- 设置 yarn 历史服务器地址 -->
<property>
    <name>yarn.log.server.url</name>
    <value>http://node1:19888/jobhistory/logs</value>
</property>

<!-- 保存的时间 7 天 -->
<property>
  <name>yarn.log-aggregation.retain-seconds</name>
  <value>604800</value>
</property>
```

7.6. workers

workers 文件里面记录的是集群主机名。主要作用是配合一键启动脚本如 start-dfs.sh、stop-yarn.sh 用来进行集群启动。这时候 workers 文件里面的主机标记的就是从节点角色所在的机器。

```
vi workers

node1.itcast.cn
node2.itcast.cn
node3.itcast.cn
```

8. scp 同步安装包

```
cd /export/server


scp -r hadoop-3.3.0 root@node2:$PWD
scp -r hadoop-3.3.0 root@node3:$PWD
```

在 node1 上进行了配置文件的修改，使用 scp 命令将修改好之后的安装包同步给集群中的其他节点。

9. Hadoop 环境变量

3 台机器都需要配置环境变量文件。

```
vim /etc/proflie

export HADOOP_HOME=/export/server/hadoop-3.3.0
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

source /etc/profile
```

四、 Hadoop 集群启动、初体验

1. 启动方式

要启动 Hadoop 集群，需要启动 HDFS 和 YARN 两个集群。

注意：**首次启动 HDFS 时，必须对其进行格式化操作**。本质上是一些清理和准备工作，因为此时的 HDFS 在物理上还是不存在的。

```
hadoop namenode -format
```

1.1. 单节点逐个启动

在主节点上使用以下命令启动 HDFS NameNode：

```
$HADOOP_HOME/bin/hdfs --daemon start namenode
```

在每个从节点上使用以下命令启动 HDFS DataNode：

```
$HADOOP_HOME/bin/hdfs --daemon start datanode
```

在 node2 上使用以下命令启动 HDFS SecondaryNameNode：

```
$HADOOP_HOME/bin/hdfs --daemon start secondarynamenode
```

在主节点上使用以下命令启动 YARN ResourceManager：

```
$HADOOP_HOME/bin/yarn --daemon start resourcemanager
```

在每个从节点上使用以下命令启动 YARN nodemanager：

```
$HADOOP_HOME/bin/yarn --daemon start nodemanager
```

如果想要停止某个节点上某个角色，只需要把命令中的 `start` 改为 `stop` 即可。

1.2. 脚本一键启动

如果配置了 etc/hadoop/workers 和 ssh 免密登录，则可以使用程序脚本启动所有 Hadoop 两个集群的相关进程，在主节点所设定的机器上执行。

hdfs：$HADOOP_PREFIX/sbin/start-dfs.sh

yarn: $HADOOP_PREFIX/sbin/start-yarn.sh

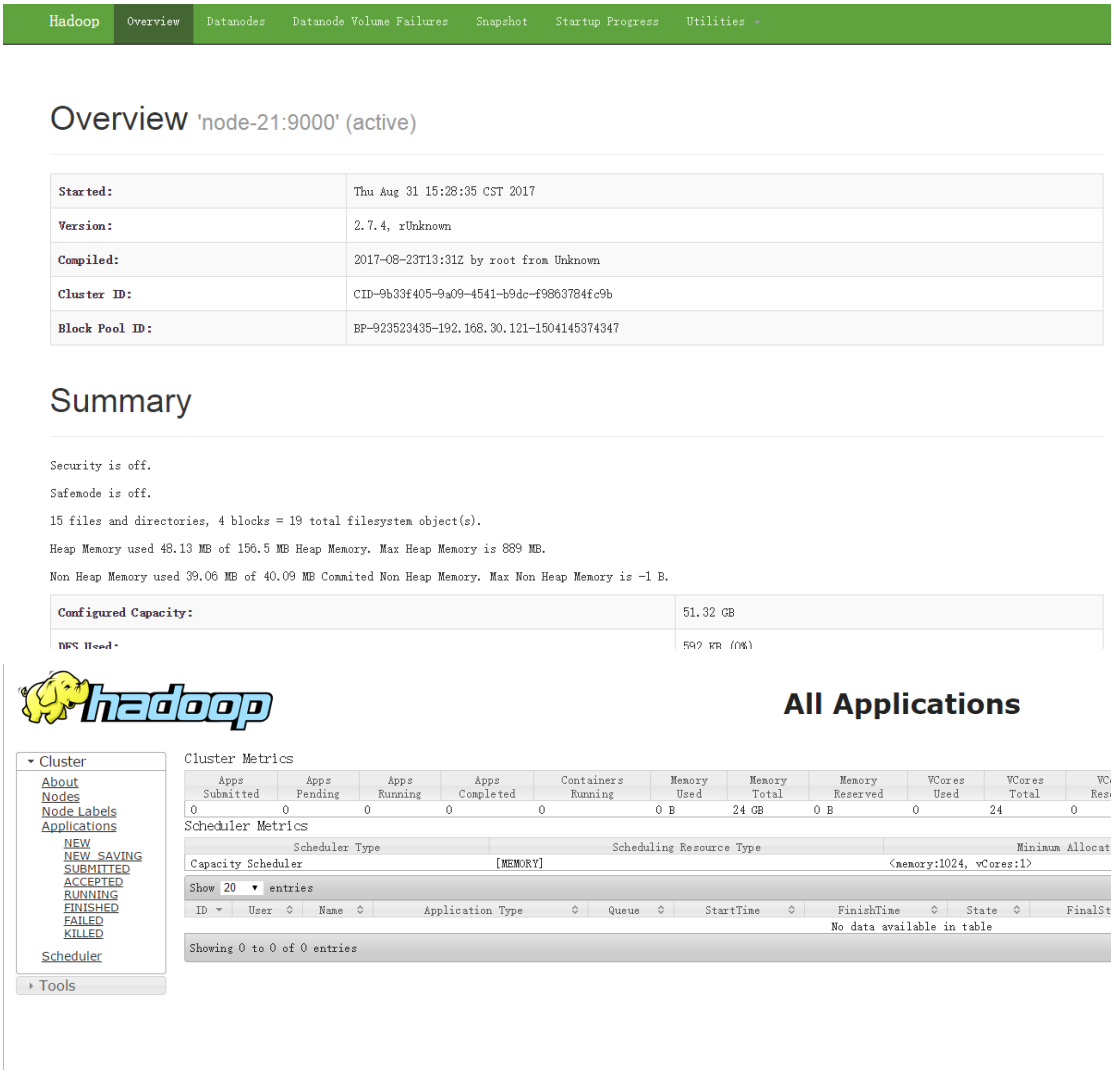
停止集群：stop-dfs.sh、stop-yarn.sh

2. 集群 web-ui

一旦 Hadoop 集群启动并运行，可以通过 web-ui 进行集群查看，如下所述：

NameNode http://nn_host:port/ 默认 9870.

ResourceManager http://rm_host:port/ 默认 8088.



3. Hadoop 初体验

3.1. HDFS 使用

从 Linux 本地上传一个文本文件到 hdfs 的/test/input 目录下

```
hadoop fs -mkdir -p /wordcount/input
hadoop fs -put /root/somewords.txt  /test/input
```

3.2. 运行 mapreduce 程序

在 Hadoop 安装包的 share/hadoop/mapreduce 下有官方自带的 mapreduce 程序。我们可以使用如下的命令进行运行测试。

示例程序 jar:

```
hadoop-mapreduce-examples-3.3.0.jar
```

计算圆周率:

```
hadoop jar hadoop-mapreduce-examples-3.3.0.jar pi 20 50
```

关于圆周率的估算,感兴趣的可以查询资料 Monte Carlo 方法来计算 Pi 值。

五、 MapReduce jobHistory

JobHistory 用来记录已经 finished 的 mapreduce 运行日志，日志信息存放于 HDFS 目录中，默认情况下没有开启此功能，需要在 mapred-site.xml 中配置并手动启动。

1. 修改 mapred-site.xml

```
cd /export/servers/hadoop-3.3.0/etc/hadoop
vim mapred-site.xml
```

MR JobHistory Server 管理的日志的存放位置

```
<property>
    <name>mapreduce.jobhistory.address</name>
    <value>node1:10020</value>
</property>
```

查看历史服务器已经运行完的 Mapreduce 作业记录的 web 地址，需要启动该服务才行

```
<property>
    <name>mapreduce.jobhistory.webapp.address</name>
    <value>node1:19888</value>
</property>
```

2. 分发配置到其他机器

```
cd /export/servers/hadoop-3.3.0/etc/hadoop
scp -r mapred-site.xml node2:$PWD
scp –r mapred-site.xml node3:$PWD
```

3. 启动 jobHistoryServer 服务进程

```
mapred --daemon start historyserver
```

如果关闭的话 用下述命令

```
mapred --daemon stop historyserver
```

4. 页面访问 jobhistoryserver

http://node1:19888/jobhistory

六、 HDFS 的垃圾桶机制

1. 垃圾桶机制解析

每一个文件系统都会有垃圾桶机制，便于将删除的数据回收到垃圾桶里面去，避免某些误操作删除一些重要文件。回收到垃圾桶里里面的资料数据，都可以进行恢复。

2. 垃圾桶机制配置

HDFS 的垃圾回收的默认配置属性为 0，也就是说，如果你不小心误删除了某样东西，那么这个操作是不可恢复的。

修改 core-site.xml ：

那么可以按照生产上的需求设置回收站的保存时间，这个时间以分钟为单位，例如 1440=24h=1 天。

```
<property>
        <name>fs.trash.interval</name>
        <value>1440</value>
</property>
```

然后重启 hdfs 集群

3. 垃圾桶机制验证

如果启用垃圾箱配置，dfs 命令删除的文件不会立即从 HDFS 中删除。相反，HDFS 将其移动到垃圾目录（每个用户在/user/<username>/.Trash 下都有自己的垃圾目录）。只要文件保留在垃圾箱中，文件可以快速恢复。

使用 skipTrash 选项删除文件，该选项不会将文件发送到垃圾箱。它将从 HDFS 中完全删除。