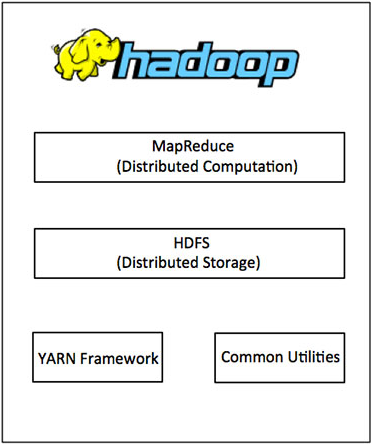
# 简介

Hadoop是由Apache基金会所开发的一个可靠的、可扩展的及可容错的分布式系统基础架构，其提供分布式大数据存储及计算功能。

Hadoop包括以下四个模块：

1. Hadoop Common（公共模块）：用于支撑其他模块；
2. Hadoop Distributed File System (HDFS，hadoop分布式文件系统): 提供高吞吐量（high throughput）来访问应用数据（海量数据存储能力）；
3. Hadoop YARN：hadoop作业调度和集群资源管理的框架；
4. Hadoop MapReduce：可提供海量数据计算的能力



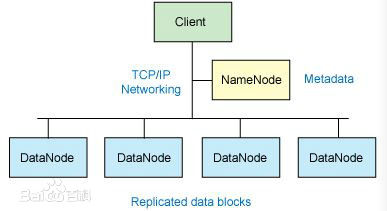
## Hadoop分布式文件系统（HDFS）描述

Hadoop分布式文件系统（HDFS）是基于谷歌文件系统（GFS），并提供了一个设计在普通硬件上运行的分布式文件系统。它与现有的分布式文件系统有许多相似之处。来自其他分布式文件系统的差别是显著。它高度容错并设计成部署在低成本的硬件。提供了高吞吐量的应用数据访问，并且适用于具有大数据集的应用程序。

## MapReduce描述

MapReduce是面向大数据并行处理的计算模型、框架和平台。

## 核心架构



对外部客户机而言，HDFS就像一个传统的分级文件系统。可以创建、删除、移动或重命名文件等等。但是HDFS的架构是基于一组特定的节点构建的（如上图所示），这些节点包括 NameNode（仅一个），它在 HDFS 内部提供元数据服务；DataNode节点为 HDFS 提供存储块。由于仅存在一个 NameNode，因此这是 HDFS 的一个缺点（单点失败）。

存储在 HDFS 中的文件被分成块，然后将这些块复制到多个计算机中（DataNode）。这与传统的 RAID 架构大不相同。块的大小（通常为 64MB）和复制的块数量在创建文件时由客户机决定。NameNode 可以控制所有文件操作。HDFS 内部的所有通信都基于标准的 TCP/IP 协议。

### NameNode节点

NameNode 是一个通常在 HDFS 实例中的单独机器上运行的软件。它负责管理文件系统名称空间和控制外部客户机的访问。NameNode 决定是否将文件映射到 DataNode 上的复制块上。对于最常见的 3 个复制块，第一个复制块存储在同一机架的不同节点上，最后一个复制块存储在不同机架的某个节点上。注意，这里需要您了解集群架构。实际的 I/O事务并没有经过 NameNode，只有表示 DataNode 和块的文件映射的元数据经过 NameNode。当外部客户机发送请求要求创建文件时，NameNode 会以块标识和该块的第一个副本的 DataNode IP 地址作为响应。这个 NameNode 还会通知其他将要接收该块的副本的 DataNode。它执行以下任务：

* 管理文件系统命名空间。
* 规范客户端对文件的访问。
* 它也执行文件系统操作，如重命名，关闭和打开的文件和目录。

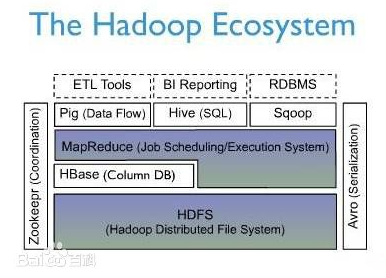
### DataNode节点

DataNode 也是一个通常在 HDFS实例中的单独机器上运行的软件。Hadoop 集群包含一个 NameNode 和大量 DataNode。DataNode 通常以机架的形式组织，机架通过一个交换机将所有系统连接起来。DataNode 响应来自 HDFS 客户机的读写请求。它们还响应来自 NameNode 的创建、删除和复制块的命令。ameNode 依赖来自每个 DataNode 的定期心跳（heartbeat）消息。每条消息都包含一个块报告，NameNode 可以根据这个报告验证块映射和其他文件系统元数据。如果 DataNode 不能发送心跳消息，NameNode 将采取修复措施，重新复制在该节点上丢失的块。根据客户的请求，数据节点上的文件系统执行的读写操作；还根据NameNode节点的指令执行操作，如块的创建，删除和复制。

## hadoop优点

1. **高可靠性**。Hadoop按位存储和处理数据的能力值得人们信赖。
2. **高扩展性**。Hadoop是在可用的计算机集簇间分配数据并完成计算任务的，这些集簇可以方便地扩展到数以千计的节点中。
3. **高效性**。Hadoop能够在节点之间动态地移动数据，并保证各个节点的动态平衡，因此处理速度非常快。
4. **高容错性**。Hadoop能够自动保存数据的多个副本，并且能够自动将失败的任务重新分配。
5. **低成本**。与一体机、商用数据仓库以及QlikView、Yonghong Z-Suite等数据集市相比，hadoop是开源的，项目的软件成本因此会大大降低。

## hadoop子项目



### Ambari

### Avro

### Cassandra

### Chukwa

### Hbase

HBase – Hadoop Database，是一个高可靠性、高性能、面向列、可伸缩的分布式存储系统，利用HBase技术可在廉价PC Server上搭建起大规模结构化存储集群。

HBase是一个分布式的、面向列的开源数据库，HBase不同于一般的关系数据库，它是一个适合于非结构化数据存储的数据库。另一个不同的是HBase基于列的而不是基于行的模式。

HBase利用Hadoop HDFS作为其文件存储系统，利用Hadoop MapReduce来处理HBase中的海量数据。HBase位于结构化[存储](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8" \t "_blank)层，Hadoop HDFS为HBase提供了高可靠性的底层存储支持，Hadoop MapReduce为HBase提供了高性能的计算能力，Zookeeper为HBase提供了稳定服务和failover机制。此外，Pig和Hive还为HBase提供了高层语言支持，使得在HBase上进行数据统计处理变的非常简单。 Sqoop则为HBase提供了方便的RDBMS数据导入功能，使得[传统数据库](https://baike.baidu.com/item/%E4%BC%A0%E7%BB%9F%E6%95%B0%E6%8D%AE%E5%BA%93" \t "_blank)数据向HBase中迁移变的非常方便。

### Hive

hive是基于Hadoop的一个数据仓库工具，可以将结构化的数据文件映射为一张数据库表，并提供简单的sql查询功能，可以将sql语句转换为MapReduce任务进行运行。 其优点是学习成本低，可以通过类SQL语句快速实现简单的MapReduce统计，不必开发专门的MapReduce应用，十分适合数据仓库的统计分析。

Hive是建立在 Hadoop 上的数据仓库基础构架。它提供了一系列的工具，可以用来进行数据提取转化加载（ETL），这是一种可以存储、查询和分析存储在 Hadoop 中的大规模数据的机制。Hive 定义了简单的类 SQL 查询语言，称为 HQL，它允许熟悉 SQL 的用户查询数据。同时，这个语言也允许熟悉 MapReduce 开发者的开发自定义的 mapper 和 reducer 来处理内建的 mapper 和 reducer 无法完成的复杂的分析工作。hive数据存储特点：

首先，Hive 没有专门的数据存储格式，也没有为数据建立索引，用户可以非常自由的组织 Hive 中的表，只需要在创建表的时候告诉 Hive 数据中的列分隔符和行分隔符，Hive 就可以解析数据。

其次，Hive 中所有的数据都存储在 HDFS 中，Hive 中包含以下数据模型：表(Table)，外部表(External Table)，分区(Partition)，桶(Bucket)。

### Mahout

### Pig

### Spark

Apache Spark是一个基于内存计算的分布式计算框架，旨在简化运行于计算机集群上的并行程序的编写。该框架对资源调度，任务的提交、执行和跟踪，节点间的通信以及数据并行处理的内在底层操作都进行了抽象。它提供了一个更高级别的API用于处理分布式数据。从这方面说，它与Apache Hadoop等分布式处理框架类似。但在底层架构上，Spark与它们有所不同。

### Tez

### ZooKeeper

sdfd

# 生产环境搭建

本生产环境只设置两个节点，一个NameNode节点（192.168.0.119）及两个DataNode节点（192.168.0.120及192.168.0.121），系统版本为CentOS Linux release 7.3.1611。

以后可以增加DataNode节点。

## 修改主机名

在root用户下执行hostname或hostnamectl命令可以查看主机名，随后使用hostnamectl set-hostname xxx命令（永久修改）更改主机名，执行完后不需要重启系统，重新打开连接即可。对应机器ip的主机名如下：

192.168.0.119：namenode

192.168.0.120：datanode1

192.168.0.121：datanode2

集群中的所有机器需要能够通过主机名连接到相应的机器（如ssh hostname），必须手动更新/etc/hosts，将集群所有机器的ip及相应的主机对应上，使用vi /etc/hosts命令编辑即可。对每台机器所需添加的内容都一致。本案例需要添加以下两行。

192.168.0.119 namenode

192.168.0.120 datanode1

192.168.0.121 datanode2

## 创建一个单独的用户

在开始时，建议创建一个单独的用户hadoop（可以起别的用户名），以从Unix文件系统隔离Hadoop文件系统。建议所有节点的用户名与密码均相同。【所有节点】操作命令如下：

[root@localhost ~]# useradd hadoop

[root@localhost ~]# passwd hadoop

Changing password for user hadoop.

New password:

BAD PASSWORD: The password is shorter than 8 characters

Retype new password:

passwd: all authentication tokens updated successfully.

修改密码时，需要输入两次。

之后，使用su - hadoop命令跳转新建的用户，用户新建的用户继续完成环境搭建。

## SSH免密码登录

配置好SSH免密码登录之后，机器之间无须用户输入密码即可相互访问。下面介绍安装与配置过程：

1. 安装SSH：只需使用相应的用户（root或hadoop）执行一次ssh跳转到其他机器即可生成。
2. 生成公私钥：cd ~/.ssh，然后执行ssh-keygen -t rsa（连续按几次回车键即可），在目录下将生成id\_rsa.pub与id\_rsa两个密钥文件。其中id\_rsa.pub为公钥。
3. 将公钥id\_rsa.pub拷贝到其他机器上：scp -r id\_rsa.pub <用户名>@IP地址:~/.ssh/authorized\_keys
4. 更改authorized\_keys文件权限：chmod 600 ~/.ssh/authorized\_keys（这一步可以放到最后对所有机器执行一遍）

注：

* 每台服务器都必须执行一遍；
* 第三、第四步可以改为先将每台机器的公钥放到一个authorized\_keys中，然后更改权限，最后拷贝到每一台机器上。
* 如果是使用root用户操作，则仅对root用户可以免密码跳转到别的机器；其他用户（如hadoop）需要再执行一次操作才可以面密码登录。

## 安装Jdk

1. 下载最新版本的jdk，下载地址：http://www.oracle.com/technetwork/java/javase/downloads/index.html
2. 使用root用户操作，为了使Java提供给所有用户，将它放到目录 “/usr/local/”。解压：tar -zxvf jdk-9.0.1\_linux-x64\_bin.tar.gz
3. 配置环境变量（vi /etc/profile）

export JAVA\_HOME=/usr/local/jdk-9.0.1

export JRE\_HOME=${JAVA\_HOME}/jre

export CLASSPATH=.:${JAVA\_HOME}/lib:${JRE\_HOME}/lib

export PATH=${JAVA\_HOME}/bin:$PATH

1. 重新加载：source /etc/profile
2. 测试：java –version（不同用户验证结果一致）

## 安装hadoop

hadoop安装，全程使用hadoop用户执行。hadoop支持以下三种模式，下面主要从完全分布式模式讲解：

* 本地/独立模式：默认情况下之后，它会被配置在一个独立的模式，无需进行其他配置即可运行，用于运行Java程序，适合于开发期间运行MapReduce程序。测试如下：

第一条命令：

cd /home/hadoop/hadoop-2.9.0

第二条命令：

./bin/hadoop jar ./share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.0.jar

执行完，可以看到该程序所实现的一些功能，并可测试这些功能。

* 模拟分布式模式：这是在单台机器的分布式模拟。Hadoop守护每个进程，如 hdfs, yarn, MapReduce等，都将作为一个独立的java程序运行。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。
* 完全分布式模式：这种模式是完全分布式的最小两台或多台计算机的集群。

### hadoop分布式集群安装

1. 登录hadoop官网http://hadoop.apache.org/releases.html下载（其他地址https://archive.apache.org/dist/hadoop/common/，下载稳定本版stable）
2. 解压到/home/hadoop目录下：tar -zxvf hadoop-2.9.0.tar.gz
3. 配置环境变量（vi ~/.bashrc），并执行source ~/.bashrc使其生效（也可以添加到/etc/profile文件中）

export HADOOP\_INSTALL=/home/hadoop/hadoop-2.9.0

export HADOOP\_PREFIX=/home/hadoop/hadoop-2.9.0

export HADOOP\_HOME=/home/hadoop/hadoop-2.9.0

export HADOOP\_CONF\_DIR=$HADOOP\_HOME/etc/hadoop

export YARN\_HOME=$HADOOP\_INSTALL

export YARN\_CONF\_DIR=$YARN\_HOME/etc/hadoop

export HADOOP\_HDFS\_HOME=$HADOOP\_INSTALL

export HDFS\_CONF\_DIR=$HADOOP\_HOME/etc/hadoop

export PATH=$PATH:$HADOOP\_INSTALL/bin:$HADOOP\_INSTALL/sbin

export HADOOP\_MAPRED\_HOME=$HADOOP\_INSTALL

export HADOOP\_COMMON\_HOME=$HADOOP\_INSTALL

export HADOOP\_LOG\_DIR=/home/hadoop/bigdata/logs

export YARN\_LOG\_DIR=/home/hadoop/bigdata/logs

1. 在hadoop-env.sh中，再显示地重新声明一遍JAVA\_HOME，即写明具体的存放目录。
2. 配置hadoop

进入Hadoop配置目录：/home/hadoop/hadoop-2.9.0/etc/hadoop，对slaves、core-site.xml、hdfs-site.xml、mapred-site.xml、yarn-site.xml等 5个文件进行配置。

* slaves文件

将作为 DataNode 的主机名写入该文件，每行一个数据节点。本案例中有两个数据节点datanode1与datanode2，在文件中加入以下两行。

datanode1

datanode2

* 配置core-site.xml

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://namenode:9000</value>

</property>

<property>

<name>hadoop.tmp.dir</name>

<value>file:/home/hadoop/bigdata/tmp</value>

</property>

<property>

<name>io.file.buffer.size</name>

<value>131072</value>

</property>

</configuration>

|  |  |
| --- | --- |
| **配置项** | **描述** |
| fs.defaultFS | 缺省的hdfs访问URL入口设定，ip或主机名（namenode节点主机名）及端口号。 |
| hadoop.tmp.dir | 临时数据存放目录设定，加file:表示本地目录，事先手动建立目录。 |
| io.file.buffer.size | 流文件的缓冲区，设置太大会消耗内存131072B（128KB） |

* 配置hdfs-site.xml

创建/home/hadoop/bigdata/namenode和/home/hadoop/bigdata/datanode目录，并配置路径。

<configuration>

<property>

<name>dfs.namenode.name.dir</name>

<value>file:/home/hadoop/bigdata/namenode</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>file:/home/hadoop/bigdata/datanode</value>

</property>

<property>

<name>dfs.replication</name>

<value>2</value>

</property>

<property>

<name>dfs.blocksize</name>

<value>256m</value>

</property>

<property>

<name>dfs.namenode.secondary.http-address</name>

<value>datanode1:50090</value>

</property>

<property>

<name>dfs.namenode.handler.count</name>

<value>20</value>

</property>

<property>

<name>dfs.datanode.handler.count</name>

<value>40</value>

</property>

</configuration>

|  |  |
| --- | --- |
| **配置项** | **描述** |
| dfs.namenode.name.dir | namenode数据的存放地点。也就是namenode元数据存放的地方，记录了hdfs系统中文件的元数据 |
| dfs.datanode.data.dir | datanode数据的存放地点。也就是block块存放的目录了。 |
| dfs.replication | hdfs的副本数设置，副本数目不能大于datanode数目，默认是3。 |
| dfs.replication.min（非必需） | 块副本的最少数量，默认是1 |
| dfs.replication.max（非必需） | 块副本的最多数量，默认是512 |
| dfs.namenode.http-address | Namenode节点的WEB UI管理端口。格式：0.0.0.0:50070 |
| **dfs.namenode.handler.count** | 设置更多的namenode线程，处理从 datanode发出的大量RPC请求，默认是10，建议设置大点。 |
| **dfs.datanode.handler.count** | datanode节点处理线程，默认是10，建议设置大点。 |
| dfs.namenode.secondary.http-address（必须配置，将其设置在一个数据节点上，不要配置在同一台机器上） | 备机namenode节点的WEB UI访问地址与端口，因为hadoop容易出现单点故障，hadoop做了namenode备份节点，一旦提供服务的namenode出现故障，由备份节点提供服务。参考值：0.0.0.0:50090，配置另外一台机器的ip |
| dfs.webhdfs.enabled（非必需） | 访问namenode的hdfs使用50070端口，访问datanode的webhdfs使用50075端口。访问文件、文件夹信息使用namenode的IP和50070端口，访问文件内容或者进行打开、上传、修改、下载等操作使用datanode的IP和50075端口。要想不区分端口，直接使用namenode的IP和端口进行所有的webhdfs操作，就需要在所有的datanode上都设置hefs-site.xml中的dfs.webhdfs.enabled为true。 |
| dfs.hosts  （用/分割ip或主机名，考虑到安全性可以设置） | 控制所包括的数据节点。 |
| dfs.hosts.exclude  （用户, 分割ip或主机名，考虑到安全性可以设置） | 控制不包括的数据节点。 |
| **dfs.blocksize** | HDFS块大小，k(kilo), m(mega), g(giga), t(tera), p(peta), e(exa) to specify the size (such as 128k, 512m, 1g, etc.) |

* 配置mapred-site.xml

mapreduce计算模型配置。

<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

<property>

<name>mapreduce.map.memory.mb</name>

<value>256</value>

</property>

<property>

<name>mapreduce.map.java.opts</name>

<value>-Xmx170M</value>

</property>

<property>

<name>mapreduce.reduce.memory.mb</name>

<value>512</value>

</property>

<property>

<name>mapreduce.reduce.java.opts</name>

<value>-Xmx320M</value>

</property>

<property>

<name>mapreduce.task.io.sort.mb</name>

<value>150</value>

</property>

<property>

<name>mapreduce.task.io.sort.factor</name>

<value>100</value>

</property>

<property>

<name>mapreduce.reduce.shuffle.parallelcopies</name>

<value>50</value>

</property>

</configuration>

|  |  |
| --- | --- |
| **配置项** | **描述** |
| mapreduce.framework.name | 运行mapreduce的执行与调度框架，配置在yarn上运行。 |
| mapreduce.map.memory.mb | 每个Map Task需要的内存量，默认1024MB |
| mapreduce.map.java.opts | Map的jvm内存，大概是0.6至0.75的mapreduce.map.memory.mb大小 |
| mapreduce.reduce.memory.mb | 每个Reduce Task需要的内存量，默认1024MB |
| mapreduce.reduce.java.opts | reduce的jvm内存，大概是0.6至0.75的mapreduce.reduce.memory.mb大小 |
| mapreduce.task.io.sort.mb | 任务内部文件排序缓冲区大小，默认100MB |
| mapreduce.task.io.sort.factor | 打开文件句柄的数量，默认10个。 |
| mapreduce.reduce.shuffle.parallelcopies | Reduce Task启动的并发拷贝数据的线程数目，默认5个。 |

* 配置yarn-site.xml

<configuration>

<property>

<name>yarn.acl.enable</name>

<value>false</value>

</property>

<property>

<name>yarn.admin.acl</name>

<value>\*</value>

</property>

<property>

<name>yarn.log-aggregation-enable</name>

<value>false</value>

</property>

<property>

<name>yarn.nodemanager.log.retain-seconds</name>

<value>3600</value>

</property>

<property>

<name>yarn.resourcemanager.address</name>

<value>namenode:8032</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address</name>

<value>namenode:8030</value>

</property>

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

<value>namenode:8031</value>

</property>

<property>

<name>yarn.resourcemanager.admin.address</name>

<value>namenode:8033</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.address</name>

<value>namenode:8088</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.https.address</name>

<value>namenode:8090</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.class</name>

<value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler</value>

</property>

<property>

<name>yarn.scheduler.minimum-allocation-mb</name>

<value>512</value>

</property>

<property>

<name>yarn.scheduler.maximum-allocation-mb</name>

<value>1024</value>

</property>

<property>

<name>yarn.scheduler.minimum-allocation-vcores</name>

<value>1</value>

</property>

<property>

<name>yarn.scheduler.maximum-allocation-vcores</name>

<value>10</value>

</property>

<property>

<name>yarn.nodemanager.resource.memory-mb</name>

<value>1024</value>

</property>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

</configuration>

|  |  |
| --- | --- |
| **配置项** | **描述** |
| yarn.acl.enable | 通过设置配置属性yarn.acl.enable为true来启动应用程序的ACL。查看访问的ACL决定谁可以通过RPC接口查看一些所有应用程序的相关细节。默认是false |
| yarn.admin.acl | 谁能够执行ACL操作，集群中的管理员，用逗号隔开，默认是\*，表示谁都可以 |
| yarn.log-aggregation-enable | 是否开启日志聚类运算。日志聚合开启后在任务完成后会保存到HDFS上。 |
| yarn.nodemanager.remote-app-log-dir | 当应用程序运行结束后，日志被转移到的HDFS目录（启用日志聚集功能时有效），修改为保存的日志文件夹。  默认值：/tmp/logs |
| yarn.nodemanager.remote-app-log-dir-suffix | 远程日志目录子目录名称（启用日志聚集功能时有效）。  默认值：logs 日志将被转移到目录${yarn.nodemanager.remote-app-log-dir}/${user}/${thisParam}下 |
| yarn.log-aggregation.retain-seconds | 聚合后的日志在HDFS上保存多长时间，单位为s。  默认值：-1（不启用日志聚合），例如设置为86400，24小时 |
| yarn.log-aggregation.retain-check-interval-seconds | 删除任务在HDFS上执行的间隔，执行时候将满足条件的日志删除（超过yarn.log-aggregation.retain-seconds参数设置的时间的日志），如果是0或者负数，则为yarn.log-aggregation.retain-seconds参数设置值的1/10。 |
| yarn.nodemanager.log.retain-seconds | NodeManager上日志最多存放时间（不启用日志聚集功能时有效）。默认值：10800（3小时） |
| yarn.resourcemanager.hostname |  |
| yarn.resourcemanager.address | ResourceManager 对客户端暴露的地址。客户端通过该地址向RM提交应用程序，杀死应用程序等。  默认值：${yarn.resourcemanager.hostname}:8032 |
| yarn.resourcemanager.scheduler.address | ResourceManager 对ApplicationMaster暴露的访问地址。ApplicationMaster通过该地址向RM申请资源、释放资源等。默认值：${yarn.resourcemanager.hostname}:8030 |
| yarn.resourcemanager.resource-tracker.address | ResourceManager 对NodeManager暴露的地址.。NodeManager通过该地址向RM汇报心跳，领取任务等。  默认值：${yarn.resourcemanager.hostname}:8031 |
| yarn.resourcemanager.admin.address | ResourceManager 对管理员暴露的访问地址。管理员通过该地址向RM发送管理命令等。  默认值：${yarn.resourcemanager.hostname}:8033 |
| yarn.resourcemanager.webapp.address | ResourceManager对外web ui地址。用户可通过该地址在浏览器中查看集群各类信息。  默认值：${yarn.resourcemanager.hostname}:8088 |
| yarn.resourcemanager.webapp.https.address | web https 地址，认值：${yarn.resourcemanager.hostname}:8090 |
| yarn.resourcemanager.scheduler.class | 启用的资源调度器主类。目前可用的有FIFO、Capacity Scheduler和Fair Scheduler。  默认值：  org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler  在Yarn中有三种调度器可以选择：FIFOScheduler ，CapacityScheduler，FairS cheduler。   1. FIFOScheduler把应用按提交的顺序排成一个队列，这是一个先进先出队列，在进行资源分配的时候，先给队列中最头上的应用进行分配资源，待最头上的应用需求满足后再给下一个分配，以此类推。FIFO Scheduler是最简单也是最容易理解的调度器，也不需要任何配置，但它并不适用于共享集群。大的应用可能会占用所有集群资源，这就导致其它应用被阻塞。在共享集群中，更适合采用CapacityScheduler或FairScheduler，这两个调度器都允许大任务和小任务在提交的同时获得一定的系统资源。 2. 对于Capacity调度器，有一个专门的队列用来运行小任务，但是为小任务专门设置一个队列会预先占用一定的集群资源，这就导致大任务的执行时间会落后于使用FIFO调度器时的时间。 3. 在Fair调度器中，我们不需要预先占用一定的系统资源，Fair调度器会为所有运行的job动态的调整系统资源。 |
| yarn.scheduler.minimum-allocation-mb  yarn.scheduler.maximum-allocation-mb | 单个可申请的最小/最大内存资源量。比如设置为1024和3072，则运行MapRedce作业时，每个Task最少可申请1024MB内存，最多可申请3072MB内存。单位是MB。默认是1024和8192。 |
| yarn.scheduler.minimum-allocation-vcores  yarn.scheduler.maximum-allocation-vcores | 单个可申请的最小/最大虚拟CPU个数。比如设置为1和4，则运行MapRedce作业时，每个Task最少可申请1个虚拟CPU，最多可申请4个虚拟CPU。默认是1和4 |
| yarn.resourcemanager.nodes.include-path /  yarn.resourcemanager.nodes.exclude-path |  |
| yarn.nodemanager.resource.memory-mb | NodeManager总的可用物理内存。注意，该参数是不可修改的，一旦设置，整个运行过程中不可动态修改。另外，该参数的默认值是8192MB，即使你的机器内存不够8192MB，YARN也会按照这些内存来使用 |
| yarn.nodemanager.vmem-pmem-ratio | 每使用1MB物理内存，最多可用的虚拟内存数。默认值：2.1，yarn.nodemanager.resource.memory-mb\*2.1 |
| yarn.nodemanager.local-dirs | 中间结果存放位置，类似于1.0中的mapred.local.dir。注意，这个参数通常会配置多个目录，已分摊磁盘IO负载。  默认值：${hadoop.tmp.dir}/nm-local-dir |
| yarn.nodemanager.log-dirs | 日志存放地址（可配置多个目录）。  默认值：${yarn.log.dir}/userlogs |
| yarn.nodemanager.aux-services | 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”” |

* 启动与测试（在主节点上操作）

第一步：格式化HDFS文件系统

$HADOOP\_HOME/bin/hdfs namenode -format

第二步：启动集群：$HADOOP\_HOME/sbin/start-all.sh

或者

$HADOOP\_PREFIX/sbin/start-dfs.sh

$HADOOP\_PREFIX/sbin/start-yarn.sh

第三步：访问

hdfs访问：<http://master-ip:50070>

yarn访问：http://master-ip:8088

第四步：使用jps查看进程。

* 停止集群（在主节点上操作）

$HADOOP\_HOME/sbin/stop-all.sh

或者

$HADOOP\_PREFIX/sbin/stop-dfs.sh

$HADOOP\_PREFIX/sbin/stop-yarn.sh

# hdfs

## HDFS命令操作

### 格式化配置HDFS文件系统

打开NameNode(HDFS服务器)，然后执行以下命令：

$HADOOP\_HOME/bin/hadoop namenode -format

### HDFS的文件列表

使用'ls' 可以找出文件列表中的目录，文件状态。可以传递一个目录或文件名作为参数的语法。

$HADOOP\_HOME/bin/hadoop fs -ls <args>

ls命令后面还可以加其他的参数，如[-ls [-C] [-d] [-h] [-q] [-R] [-t] [-S] [-r] [-u] [<path> ...]]。下面加-R递归列出所有文件及目录。

格式：hadoop fs -ls -R <path>

### 查看一个文件或目录大小

格式：hadoop fs -du <path>（以字节为单位）

du命令后面还可以加其他参数，指定显示单位等。如[-du [-s] [-h] [-x] <path> ...]。如以KB为单位显示：hadoop fs -du -h <path>

### 创建一个目录

格式：hadoop fs -mkdir <args>，如hadoop fs -mkdir /checkpoint。

注意：多级目录，必须确保上一次的目录已创建。

### 创建一个空文件

格式：hadoop fs -touchz URI [URI ...]

### 将输入插入到HDFS

传输并使用本地系统put命令，Hadoop文件系统中存储的数据文件。格式：hadoop fs -put <文件参数> <目录参数>

如：hadoop fs -put /home/file.txt /user/input

[-put [-f] [-p] [-l] [-d] <localsrc> ... <dst>]

### 将本地文件追加到HDFS上的一个文件

使用“appendToFile”将一个或者多个本地文件追加到目的文件。格式：hadoop fs -appendToFile <一个或者多个本地文件> <HDFS文件>

如：hadoop fs -appendToFile b.txt /hadoop/txt/a.txt

hadoop fs -appendToFile c.txt d.txt /hadoop/txt/a.txt

hadoop fs -appendToFile c.txt d.txt hdfs://192.168.0.119:9000/hadoop/txt/a.txt

### 查看文件里的数据信息

使用cat命令来查看来自HDFS的数据。格式：hadoop fs -cat <文件>，如下所示：

hadoop fs -cat /user/input/file.txt

### 计算paths下的目录数，文件数和字节数

格式：hadoop fs -count [-q] [-h] [-v] <paths>

### 改变文件所属的组

格式：hadoop fs -chgrp [-R] GROUPNAME URI [URI ...]，使用-R 将使改变在目录结构下递归进行。

hadoop fs -chgrp supergroup hdfs://192.168.0.119:9000/hadoop/txt/a.txt

### 改变文件的所有者

格式：hadoop fs -chown [-R] [OWNER][:[GROUP]] URI [URI ]，使用-R 将使改变在目录结构下递归进行。命令的使用者必须是超级用户。

hadoop fs -chown hadoop:hadoop hdfs://192.168.0.119:9000/hadoop/txt/a.txt

### 改变文件访问权限

格式：hadoop fs -chmod [-R] <MODE[,MODE]... | OCTALMODE> URI [URI ...]，使用-R 将使改变在目录结构下递归进行。权限设置可以参考 linux下文件系统的chmod的用法，基本类似。

### 查找满足表达式的文件和文件夹

格式：hadoop fs -find <path> ... <expression>，没有配置path的话，默认的就是全部目录/；如果表达式没有配置，则默认为-print。

打印：

hadoop fs -find /hadoop/txt/

hadoop fs -find /hadoop/txt/ -print

hadoop fs -find /hadoop/txt/ -print0 打印在一行上

### 从HDFS取回文件到本地文件系统

格式：hadoop fs -get <文件或目录> <本地文件系统目录>，

如下所示：

hadoop fs -get /user/input/ /home/hadoop\_tp/(将input目录取回本地文件系统，并存放在/home/hadoop\_tp/下)

[-get [-f] [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]

### getfacl

[-getfacl [-R] <path>]

### getfattr

[-getfattr [-R] {-n name | -d} [-e en] <path>]

### getmerge

格式：hadoop fs -getmerge <src> <localdst>，将源文件地址的内容合并到目标文件上，将覆盖目标文件的内容。

[-getmerge [-nl] [-skip-empty-file] <src> <localdst>]

### 移动文件或文件夹

格式：hadoop fs -mv <原文件或目录> <移动后的文件或目录>

### 复制文件或文件夹

格式：hadoop fs -cp <原文件或目录> <复制后的文件或目录>

[-cp [-f] [-p | -p[topax]] [-d] <src> ... <dst>]

-f 参数选项：当文件存在时，进行覆盖。

-p 参数选项：将权限、所属组、时间戳、ACL以及XAttr等也进行拷贝。

### copyFromLocal

格式：hadoop fs -copyFromLocal <localsrc> URI，类似于put命令，和put不同的是，拷贝的源地址必须是本地文件地址。-f参数当拷贝的目标文件存在时，进行覆盖。

[-copyFromLocal [-f] [-p] [-l] [-d] <localsrc> ... <dst>]

### copyToLocal

格式：hadoop fs -copyToLocal [-ignorecrc] [-crc] URI <localdst>，类似于get指令。和get不同的是，拷贝的目的地址必须是本地文件地址。

[-copyToLocal [-f] [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]

### 删除文件或目录

格式：hadoop fs -rm <文件>

删除目录需要加-R参数，如hadoop fs -rm -R <文件夹>

### 递归删除文件或目录

格式：hadoop fs -rm -r <文件或目录>

### 清空回收站

用法：hadoop fs -expunge

### moveFromLocal

[-moveFromLocal <localsrc> ... <dst>]

### moveToLocal

[-moveToLocal <src> <localdst>]

### setfacl

hadoop fs -setfacl [-R] [-b |-k -m |-x <acl\_spec> <path>] |[--set <acl\_spec> <path>]

### 将HDFS中文件以文本形式输出（包括zip包，jar包等形式）

hadoop fs -text <src>

### 将文件按照length进行截取

hadoop fs -truncate [-w] <length> <paths>

### 返回命令的help信息

hadoop fs -usage command

### 帮助文档

hadoop fs -help

### 查看文件校验和

格式：hadoop fs -checksum URI

hadoop fs -checksum /hadoop/txt/a.txt

# 遇到的问题

## Hadoop datanode 无法启动，报错：java.io.IOException: All specified directories are failed to load

解决办法：

可能是由于在关机的时候，按照正常顺序退出集群造成的，导致dfs中的id不同。

进入dfs/data，修改VERSION文件即可，将version里面的内容修改成和master一直的。

或者直接删除dfs，格式化hdfs即可（./hdfs namenode -format）

## hadoop创建文件夹与授权

(创建文件夹)

hadoop fs -mkdir /checkpoint/ssyj

hadoop dfs –chmod 777 /checkpoint（修改文件夹权限）