# Hadoop笔记

### Hadoop的简介

#### Hadoop与云计算

##### Hadoop的简介

Hadoop适合处理大规模数据

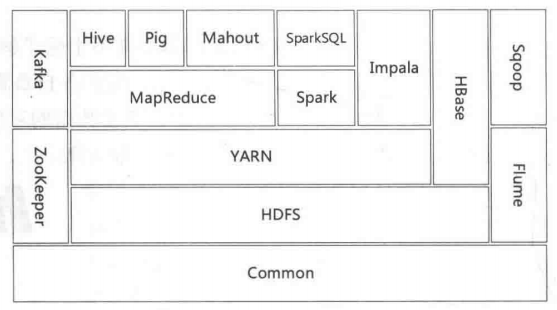
Hadoop被部署到一个集群上

Hadoop主要用于分布式存储和分布式计算

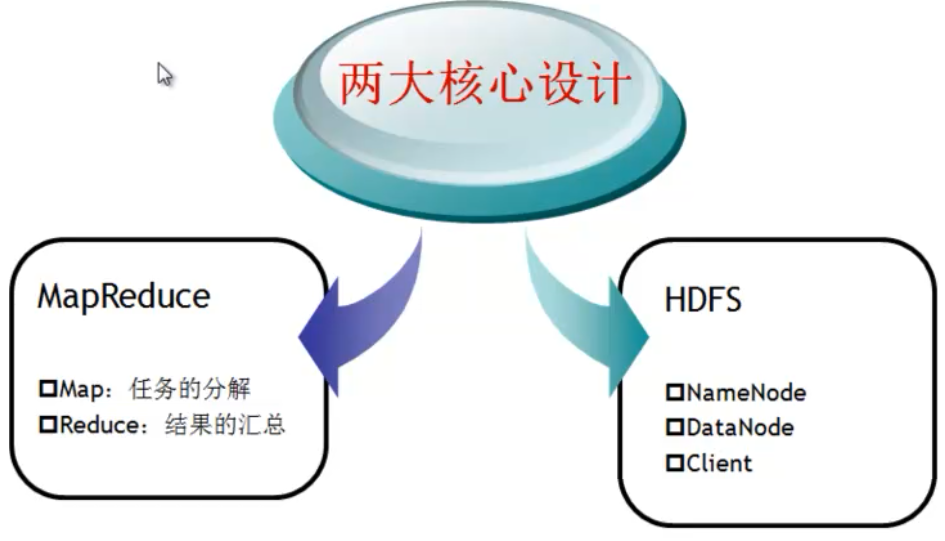
Hadoop的特点：无共享，高可用，高扩展

##### Hadoop的生态圈

Hadoop本身只有Common、HDFS、YARN和MapReduce模块，但是扩展之后，如图所示：



Hadoop的两大核心



##### Hadoop与云计算

* + - 1. 云计算

1. 按需提供服务
2. 宽带网络访问
3. 资源池化—虚拟化技术
4. 高伸缩性，自动适用业务变化
5. 可量化，动态针对服务收费
6. 大规模，服务廉价
   * + 1. 云计算类型

基础设施即服务：相当于对用户提供裸机，常用的虚拟化产品如：

Vmware vShpere,Hyper-V,KVM,Xen,Docker等

平台即服务：相当于对用户提供系统。

软件即服务：相当于对用户提供软件使用权

1.1.3.3 Hadoop相当于云计算中的基础设置即服务层的其中一种实现。

#### 1.2 Hadoop与大数据

##### 1.2.1 大数据

数据量（volume）多，数据类型多样化（variety），速度（velocity）

数据类型：结构化（表），半结构化（json，xml），准结构化（可结构化处理），非结构化（图片，视频，文档）

##### 1.2.2 Hadoop与大数据

**HDFS**，Google的GFS的开源实现。

HDFS的设计理念：流式数据访问模式，存储超大文件，运行于廉价硬件集群上。

**MapReduce**，Google的MapReduce的开源实现。

MapReduce的分布式计算模型。

**HBase**，Google的Bigtable的开源实现。结构化存储的分布式数据库。

HBase弥补了HDFS的流式读写，其擅长实时随机读写。

##### 1.2.3 其他大数据处理平台

**Storm**，针对流式数据，现实生活中很多数据不是一个文件，而是一个源源不断的数据流，需要实时分析。Storm也可以运行在YARN之上。

**Spark**，基于内存计算，且在并行计算有很大优势。减少了HDFS的开销。提供Cache，减少了I/O开销。使用多线程池，减少了任务启动开销。

广泛的数据集操作类型：Spark SQL支持SQL的结构化查询，Spark Streaming是其流计算框架，Mllib集成了主流机器学习算法，GraphX则是Spark的图计算框架。

#### Hadoop应用领域

数据挖掘

离散数据分析和数据仓库 MPP，Hadoop，HBase

大规模在线实时应用HBase

数据仓库

商业智能

其中，MPP是一种实时海量数据分析架构，Hadoop是一个数据存储处理的项目群，MapReduce是一个离线海量数据分析架构

国内的大部分网盘都是使用Hadoop实现的。

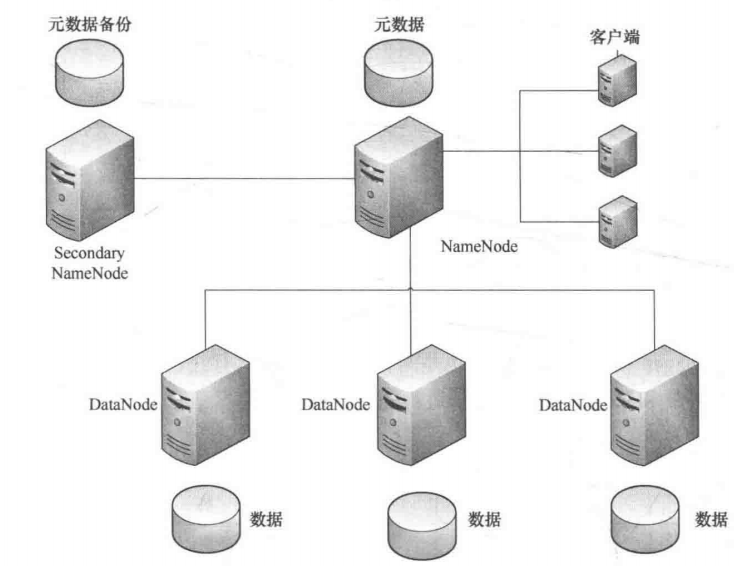
### Hadoop的安装

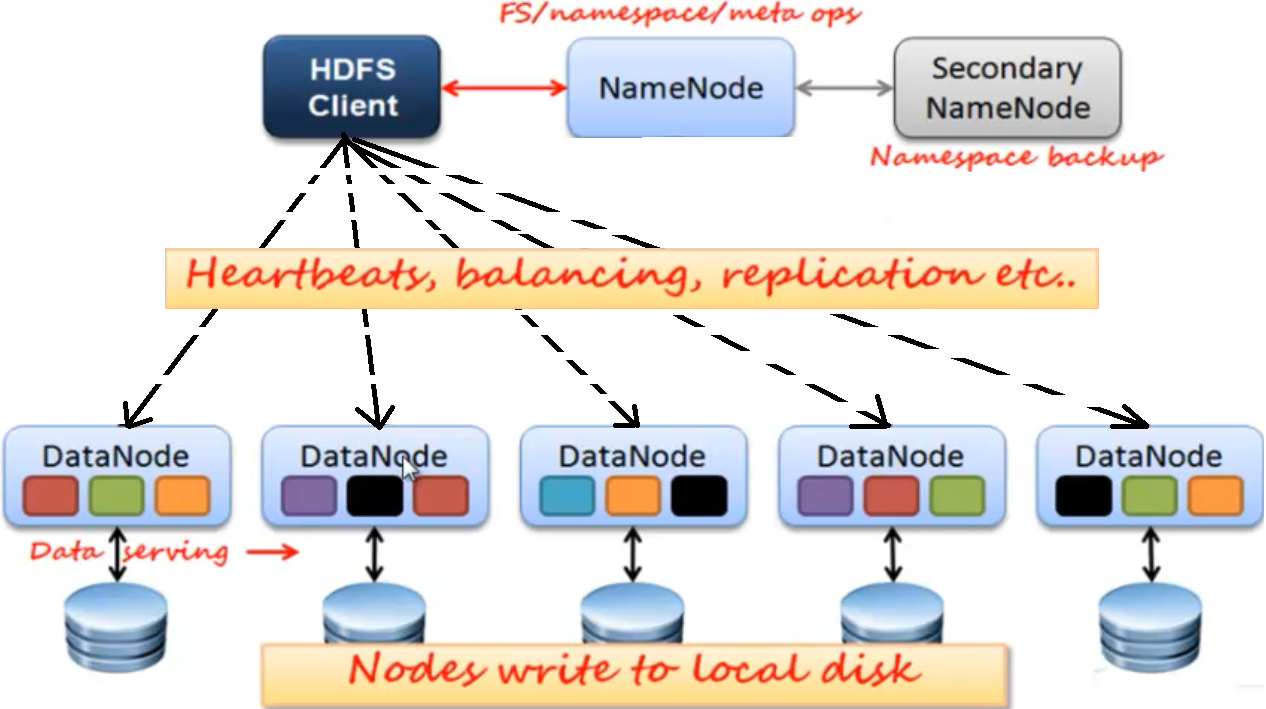
#### 2.1 Hadoop的架构

##### 2.1.1 HDFS的架构

构成HDFS集群的主要有两类节点，以主从（master/slave）模式运行，即一个NameNode和多个DataNode。而SecondaryNameNode为NameNode的备份。其实所讲的NameNode，DataNode，SecondaryNameNode是之其守护进程，当然也指运行指定进程的节点。

客户端则是用户与整个文件系统交互的客户端。





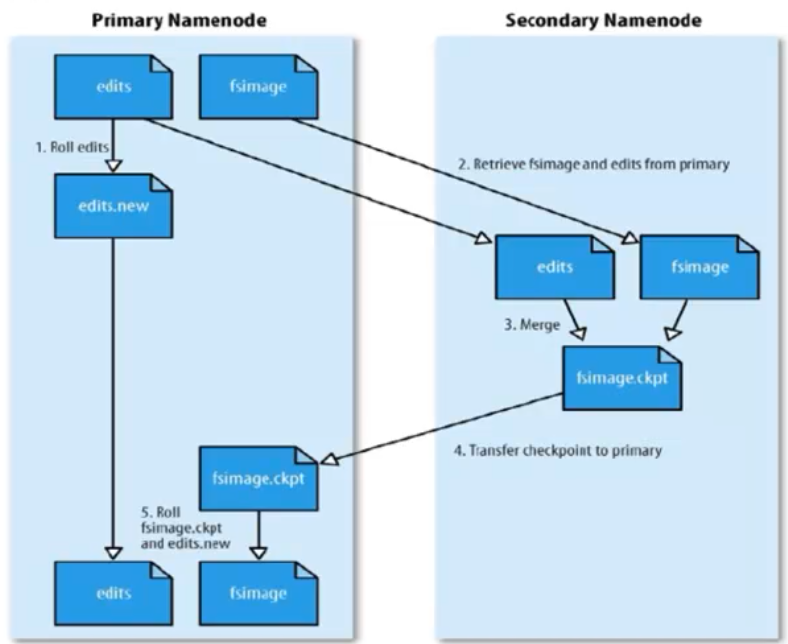
上图中，显示了有一个元数据，SecondNameNode不是一个元数据的备份，但是也可以做备份，客户端访问元数据，获得数据的所在位置，之后根据这个位置访问数据，在数据的DataNode上由数据块存储，并且数据块是有备份的，从上图可以看到在不同的DataNode上有相同颜色的数据块。

NameNode主要做读写服务，SecondNameNode主要做合并，删除等操作。SecondNameNode会将合并后的edit.log，fsImage文件推送到NameNode。

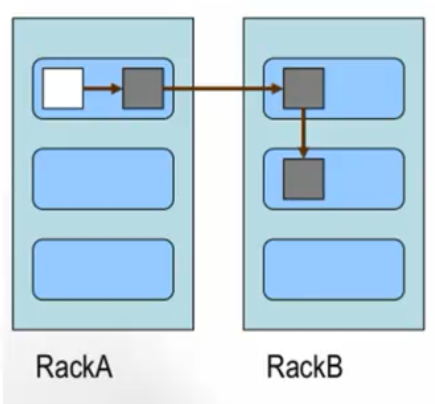
3600s合并一次，或者

Log文件设置大小，默认64M，如果满了，没到3600s也会合并一次。

SNN合并流程：

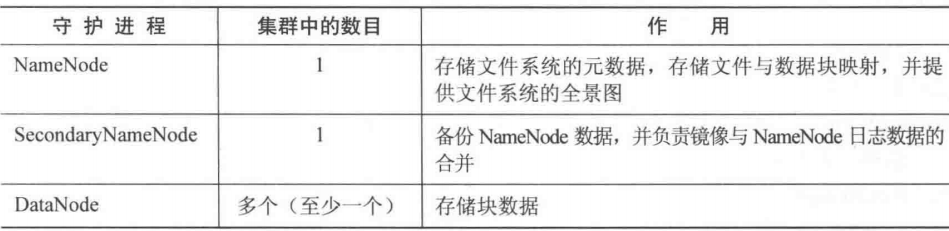


数据块副本存放策略：



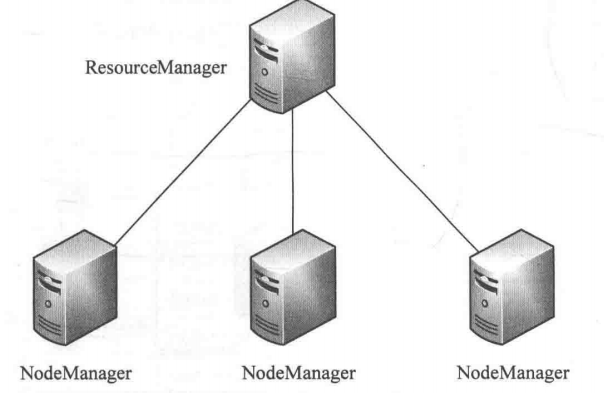
机架感应配置。第一个副本在本机架，第二个副本在不同机架，第三个副本和第二个副本同一个机架，更多的副本随机机架。

各个守护进程的作用以及在集群中的数目：

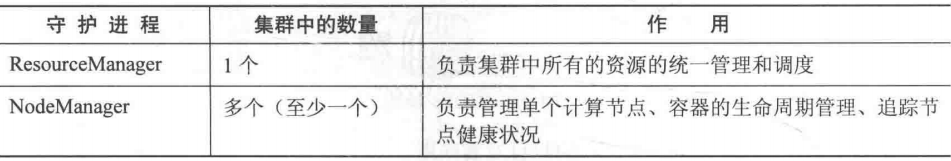


##### 2.1.2 YARN架构

构成YARN的集群也是两类节点：ResourceManager和NodeManager。也是（master/slave）架构。



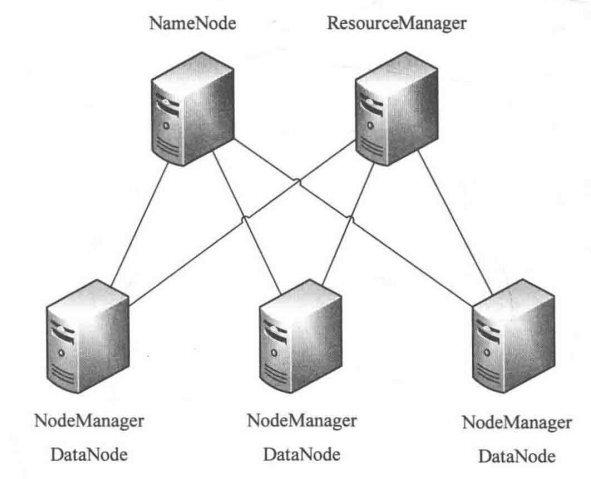
各个守护进程的作用：



##### 2.1.4 总结

综上所述，其实HDFS和YARN集群是由守护进程组成。其中NameNode进程和ResourceManager进程在一个节点上运行，同时，DataNode和NodeManager进程在同一个节点上运行。

注意：NameNode进程和ResourceManager进程不一定在同一个节点，并且在应用上建议分开。而DataNode和NodeManager进程需要配对部署到同一个节点上。



#### 2.2 Hadoop的安装

##### 2.2.1 安装Hadoop

***Hadoop的安装步骤***：

###### 2.2.1.1 安装运行环境（所有节点）

通过虚拟机安装CentOS

修改MAC地址：MACADDR=xx:xx:xx:xx

###### 2.2.1.2 修改主机名和用户名（所有节点）

a. 修改用户名和密码：useradd hadoop；passwd hadoop

b. 修改主机名：vi /etc/sysconfig/network

NETWORKING=yes

HOSTNAME=master

其他节点依次修改为HOSTNAME= slave1，slave2 ……

c. 修改所有节点hosts文件: vi /etc/hosts,可以让节点间使用主机名访问

#填写IP地址与主机名

192.168.10.3 master

#填写slave的IP地址和主机名

192.168.10.4 slave1

192.168.10.5 slave2

……

###### 2.2.1.3 配置静态IP地址（所有节点）

配置静态IP地址：

vi /etc/sysconfig/network-scripts/ifcfg-enn33

BOOTPROTO=static

NM\_CONTROLLED=yes

ONBOOT=yes

IPADDR=192.168.10.3

GATEWAY=192.168.10.1

NETMASK=255.255.255.0

###### 2.2.1.4 配置SSH无密码连接

a.关闭防火墙，service iptables stop(6.x） （所有节点）

或者永久关闭防火墙，chkconfig iptables off(6.x）

查看防火墙状态，service firewalld status

关闭防火墙，service firewalld stop

开启防火墙，service firewalld start

重启防火墙，service firewalld restart

开启开机启动，systemctl enable firewalld.service

关闭开机启动，systemctl disable firewalld.service

b.检查ssh是否安装，yum install ssh;yum install rsync（所有节点）

c.启动SSH服务，service sshd restart

d.检查是否安装成功，rpm -qa|grep openssh;rpm -qa|grep rsync

e.生成ssh公钥，ssh-keygen -t rsa(主节点)，（hadoop用户）

f.将公钥发送到从节点，（hadoop用户）

ssh-copy-id -i ~/.ssh/id\_rsa.pub hadoop@slave1

ssh-copy-id -i ~/.ssh/id\_rsa.pub hadoop@slave2

如果是root用户：

ssh-copy-id -i ~/.ssh/id\_rsa.pub root@slave1

ssh-copy-id -i ~/.ssh/id\_rsa.pub root@slave2

g.验证安装，完全分布（ssh slave1），伪分布（ssh master）

如果失败可能是权限问题，

chmod700 /home/hadoop/.ssh

chmod644 /home/hadoop/.ssh/authorized\_keys

###### 2.2.1.5 安装JDK

配置环境变量：

JAVA\_HOME=/opt/jdk1.8.0\_144

CLASSPATH=$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar

PATH=$PATH:$JAVA\_HOME/bin

Export JAVA\_HOME CLASSPATH PATH

###### 2.2.1.6 配置Hadoop

1. 下载：

<https://www.cloudera.com/documentation/enterprise/release-notes/topics/cm_vd_cdh_package_tarball_512.html#cm_vd_cdh_package_tarball_512>

2. 安装与配置

a) 配置环境变量 vi /etc/profile：

HADOOP\_HOME=/opt/hadoop

PATH=$PATH:$HADOOP\_HOME/bin

Source /etc/profile

**b**) 修改hadoop配置文件

Hadoop的配置文件：



安装hadoop只涉及前6个文件(hadoop-2.6.x/etc/hadoop/)

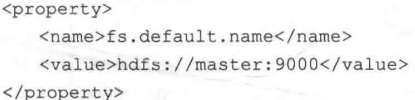
c) 修改 hadoop-env.sh

配置JAVA\_HOME：export JAVA\_HOME=/opt/jdk1.8.0\_144

配置HADOOP\_HOME：export HADOOP\_HOME=/opt/hadoop

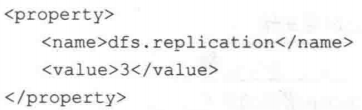
d) 修改core-site.xml

配置HDFS服务的主机名和端口号，同时也指明了NameNode节点

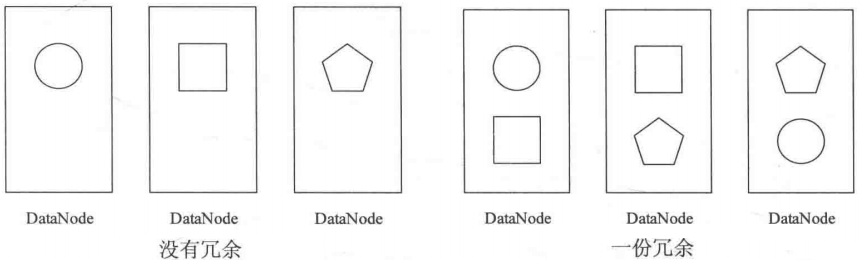


e) 修改hdfs-site.xml

配置冗余副本数



其中dfs.replication为1，没有冗余，如下图是没有冗余和冗余为2的情况



存放name元数据目录

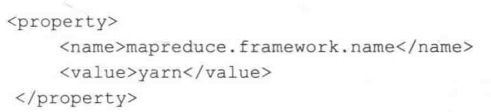


存放data的数据目录



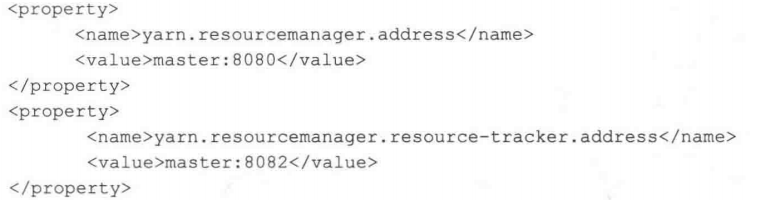
f) 修改mapred-site.xml

指明MapReduce计算框架基于YARN

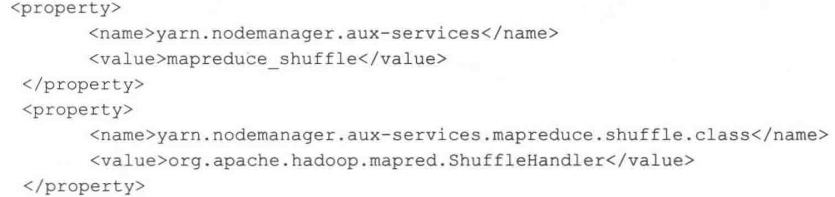


g) 修改yarn-site.xml

指明ResourceManager服务的主机名和端口号

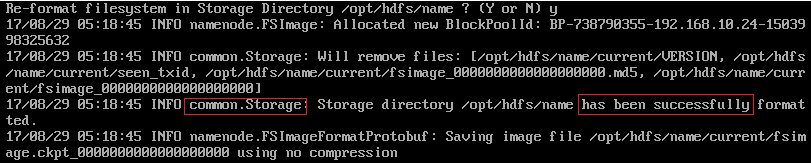


指明mapreduce\_shuffle的类



h) 服务器之间复制文件

scp -r /opt/hadoop hadoop@slave:/opt/



i) **格式化HDFS**

格式化：hadoop namenode -format

j) **启动Hadoop并验证安装**

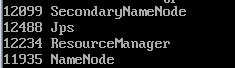
启动hadoop

/opt/hadoop/sbin/start-all.sh

使用jps验证是否成功

会显示本机节点。

Master：



Salve：



##### 2.2.2 安装Hive

###### 2.2.2.1 安装元数据库—mysql

1. 安装mysql yum install MySQL-\*.rpm

2. 启动mysq

###### 2.2.2.2 修改Hive配置文件

1. 安装Hive ，解压hive.tar.gz文件

2. 配置Hive环境变量 vi conf/hive-env.xml

Export JAVA\_HOME = /opt/jdk1.8.0\_14

Export HADOOP\_HOME=/opt/hadoop

3. 配置系统环境变量

HIVE\_HOME= /opt/hive

PATH=$PATH:$HIVE\_HOME/bin

Export HIVE\_HOME PATH

4. 配置元数据库

a) 将mysql驱动包，移动到hive/lib目录下

b) vi hive-site.xml

a) 配置数据库连接

<property>  
< name>javax.jdo.option.ConnectionURL</name>  
< value>jdbc:mysql://localhost:3306/hive?createDatabaseIfNoExist=true</value>  
< /property>

b) 配置数据库驱动

<property>  
< name>javax.jdo.option.ConnectionDriverName</name>  
< value>com.mysql.jdbc.Driver</value>  
< /property>

c) 配置数据库用户名和密码

<property>  
< name>javax.jdo.option.ConnectorUserName</name>  
< value>hive</value>  
< /property>

<property>  
< name>javax.jdo.option.ConnectionPassword</name>  
< value>hive</value>  
< /property>

d) 设定数据目录

<property>  
< name>hive.metastore.warehouse.dir</name>  
< value>/usr/hive/warehouse</value>  
< /property>

e) 设定临时文件目录

<property>  
< name>hive.exec.scratdir</name>  
< value>/usr/hive/tmp</value>  
< /property>

f) 配置日志目录

<property>  
< name>hive.querylog.location</name>  
< value>/usr/hive/log</value>  
< /property>

##### 2.2.3 安装HBase

##### 2.2.4 安装Sq第三章 Hadoop之HDFS

#### 3.1 命令行实现增删改查 hadoop fs 或 hadoop dfs

##### 3.1.1 新建文件夹

hadoop fs -mkdir /user

hadoop dfs -mkdir /user/root

##### 3.1.2 查看文件夹权限

查看所有文件夹

hadoop fs -ls / (根目录）

hadoop fs -ls /user (显示/user目录下的文件)

hadoop fs -ls -d /user/root （显示root文件夹的权限）

##### 3.1.3 上传文件

hadoop fs -put main.c /user/root

hadoop fs -ls -R /user/root (显示root下的文件)

##### 3.1.4 查看文件内容

hadoop fs -cat /user/root/main.c

##### 3.1.5 复制移动与删除

复制

hadoop fs -cp /user/root/main.c /user

hadoop fs -ls /user

移动

hadoop fs -mv /user/main.c /hadoop/

hadoop fs -ls /hadoop

hadoop fs -ls /user

删除

hadoop fs -rm -r /user/root/main.c

hadoop fs -ls /user/root

hadoop fs -rm -r /user

#### 3.2 使用JAVA API实现增删改查

##### 3.2.1 HDFS的读流程

客户端调用API的open方法访问NameNode获取DataNode的位置信息，然后调用API的read方法并发读取DataNode，最后调用API的close方法关闭访问DataNode的线程。。

##### 3.2.2 HDFS的写流程

客户端调用API的create方法访问NameNode，创建一个文件的基本信息，并处理可以分成几个数据块。将数据块写到DataNode，之后DataNode会产生一个线程往其他DataNode上写副本。返回一些信息。关闭操作DataNode的线程，返回给NameNode完成信息。

##### 3.2.3 HDFS的文件权限

HDFS的文件权限和Linux的文件权限是类似的。

##### 3.2.4 Java API实现增删改查

package cn.edu.scnu.main;  
  
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.FSDataInputStream;  
import org.apache.hadoop.fs.FileStatus;  
import org.apache.hadoop.fs.FileSystem;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.io.IOUtils;  
  
import java.io.ByteArrayOutputStream;  
import java.io.IOException;  
import java.io.OutputStream;  
import java.net.URI;  
  
public class HdfsTest {  
  
 private URI hadoop\_url;  
 private Configuration conf;  
  
 public HdfsTest(URI hadoop\_url, Configuration conf) {  
 this.hadoop\_url = hadoop\_url;  
 this.conf = conf;  
 }  
  
 */\*\*  
 \* 创建目录  
 \*/* public void createDir(String folder) throws IOException, InterruptedException {  
 Path path = new Path(folder);  
 FileSystem fileSystem = FileSystem.*get*(hadoop\_url,conf,"root");  
 if(!fileSystem.exists(path)){  
 fileSystem.mkdirs(path);  
 System.*out*.println("Create:"+folder);  
 }  
 fileSystem.close();  
 }  
  
 */\*\*  
 \* 上传文件  
 \*/* public void copyFile(String local,String remote) throws IOException, InterruptedException {  
 FileSystem fileSystem = FileSystem.*get*(hadoop\_url,conf,"root");  
 fileSystem.copyFromLocalFile(new Path(local),new Path(remote));  
 System.*out*.println("copy file from "+local+" to "+remote);  
 fileSystem.close();  
 }  
  
 */\*\*  
 \* 遍历文件  
 \*/* public void ls(String folder) throws IOException, InterruptedException {  
 FileSystem fileSystem = FileSystem.*get*(hadoop\_url,conf,"root");  
 FileStatus[] lists = fileSystem.listStatus(new Path(folder));  
 for(FileStatus f :lists){  
 System.*out*.println("name:"+f.getPath()+",folder:"+f.isDirectory()+",size:"+f.getLen());  
 }  
 fileSystem.close();  
 }  
  
  
 */\*\*  
 \* 查看文件内容  
 \*/* public void cat(String remoteFile) throws IOException, InterruptedException {  
 FileSystem fileSystem = FileSystem.*get*(hadoop\_url,conf,"root");  
 FSDataInputStream inputStream = null;  
 System.*out*.println("content: "+remoteFile);  
  
 OutputStream outputStream = new ByteArrayOutputStream();  
 String str = null;  
  
 try{  
 inputStream = fileSystem.open(new Path(remoteFile));  
 IOUtils.*copyBytes*(inputStream,outputStream,4096,false);  
 str = outputStream.toString();  
 }finally {  
 IOUtils.*closeStream*(inputStream);  
 fileSystem.close();  
 }  
 System.*out*.println(str);  
 }  
  
 */\*\*  
 \* 下载文件  
 \*/* public void download(String remote,String local) throws IOException, InterruptedException {  
 FileSystem fileSystem = FileSystem.*get*(hadoop\_url,conf,"root");  
 fileSystem.copyToLocalFile(false,new Path(remote),new Path(local),true);  
 System.*out*.println("download from "+remote+" to "+local);  
 fileSystem.close();  
 }  
  
  
 */\*\*  
 \* 重命名文件  
 \*/* public void rename(String src,String des) throws IOException, InterruptedException {  
 FileSystem fileSystem = FileSystem.*get*(hadoop\_url,conf,"root");  
 fileSystem.rename(new Path(src),new Path(des));  
 System.*out*.println("rename: "+src+" to "+des);  
 fileSystem.close();  
 }  
  
  
 */\*\*  
 \* 删除文件或目录  
 \*/* public void delete(String folder) throws IOException, InterruptedException {  
 FileSystem fileSystem = FileSystem.*get*(hadoop\_url,conf,"root");  
 fileSystem.deleteOnExit(new Path(folder));  
 System.*out*.println("delete:" + folder);  
 }  
}

***测试文件***

package cn.edu.scnu.test;  
  
import cn.edu.scnu.main.HdfsTest;  
import org.apache.hadoop.conf.Configuration;  
  
import java.io.IOException;  
import java.net.URI;  
  
public class main {  
 public static void main(String[] args) throws IOException, InterruptedException {  
 URI uri = URI.*create*("hdfs://192.168.10.16:9000");  
 Configuration conf=new Configuration();  
 HdfsTest hdfsTest = new HdfsTest(uri,conf);  
 //创建目录  
 hdfsTest.createDir("/home/lay");  
  
 //上传文件  
 hdfsTest.copyFile("src/files/test.txt","/home/lay/test.txt");  
  
 //查看文件内容  
  
 hdfsTest.cat("/home/lay/test.txt");  
  
 //下载文件  
 hdfsTest.download("/home/lay/test.txt","src/files/main.txt");  
  
 //遍历文件  
 hdfsTest.ls("/");  
  
 //重命名文件  
 hdfsTest.rename("/home/lay/test.txt","/home/lay/main.txt");  
  
 //删除文件  
 hdfsTest.delete("/home/lay");  
  
 }  
}

###### 1. maven依赖

#### 3.3 HDFS的数据存储单元

—文件被切分成固定大小的数据块

1. 默认数据块大小为64M。可配置
2. 若文件大小不到64M，则单独存成一个block（数据块）

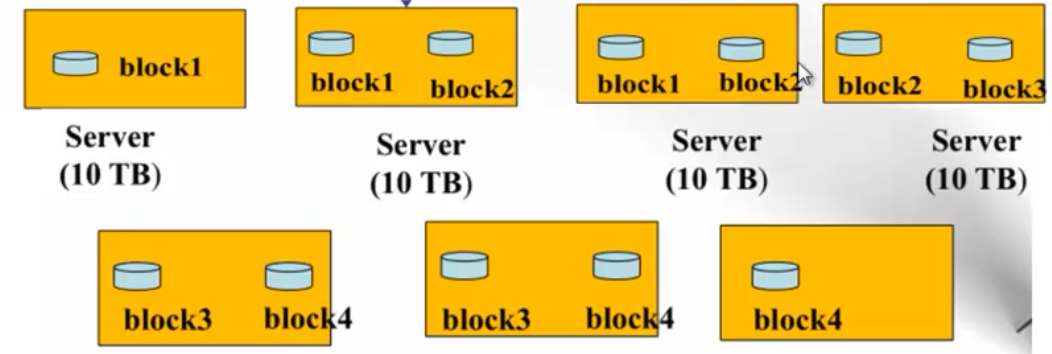
—一个文件的存储方式

1. 按大小切分成若干个block，存储到不同的节点上。
2. 默认轻快下每个block都有三个副本。

—Block大小和副本数通过Client端上传文件时设置，文件上传成功后副本数可以变更，Block Size不可变更。

#### 3.4 HDFS的设计思想

假设一个文件分成4个数据块存储，block1，block2，block3，block4



### 第四章 Hadoop之YARN（MRv2）

### 第五章 Hadoop之MapReduce

#### 5.1 MapReduce原理

一个MapReduce作业（job），通常会把输入的数据集切分成若干独立的数据块，由map任务（task）以完全并行的方式处理他们。处理之后会进行排序后输出给Reduce任务。

MapReduce框架的计算节点和存储节点是运行在相同的节点上的。这就是所谓的“移动计算比移动数据更经济”。

MapReduce框架包括一个主节点（ResourceManager）和多个子节点（NodeManager）和MRAppMaster（每个任务一个）共同组成。

### 第六章 Hadoop之Hive

#### 6.1 Hive的原理

Hive是基于Hadoop的一个数据仓库工具，可以将结构化数据文件映射成一张数据表，并提供简单的sql查询功能。

其原理就是将sql语句转换成MapReduce任务去运行。通过类SQL（HQL）语句实现简单的MapReduce统计，而不必开发专门的MapReduce应用，适合统计分析。

### 第七章 Hadoop之Sqoop

### 第八章 Hadoop之HBase

### 第九章 Hadoop之应用开发