

Linux 下 Hadoop 分布式配置和使用

詹坤林 2010 年 5 月

目 录

介绍	2
0 集群网络环境介绍	2
1 /etc/hosts文件配置	2
2 SSH无密码验证配置	3
2.1 选择一：配置Namenode无密码登录所有Datanode	3
2.1 选择二：配置所有节点之间SSH无密码验证.....	4
3 JDK安装和Java环境变量配置	5
3.1 安装 JDK 1.6	5
3.2 Java环境变量配置	5
4 Hadoop集群配置	5
5 Hadoop集群启动	8
6 Hadoop使用	10
6.1 客户机与HDFS进行交互.....	10
6.1.1 客户机配置.....	10
6.1.2 列出HDFS根目录/下的文件.....	11
6.1.3 列出当前用户主目录下的文件.....	11
6.1.4 HDFS用户管理.....	11
6.1.5 复制本地数据到HDFS中.....	12
6.1.6 数据副本说明.....	12
6.1.7 hadoop-site.xml参数说明.....	13
6.1.8 HDFS中的路径.....	13
6.1.8 Hadoop相关命令.....	14
6.2 客户机提交作业到集群.....	14



Linux公社（LinuxIDC.com）于2006年9月25日注册并开通网站，Linux现在已经成为一种广受关注和支持的一种操作系统，IDC是互联网数据中心，LinuxIDC就是关于Linux的数据中心。

LinuxIDC.com提供包括Ubuntu，Fedora，SUSE技术，以及最新IT资讯等Linux专业类网站。

并被收录到Google 网页目录-计算机 > 软件 > 操作系统 > Linux 目录下。

Linux公社（LinuxIDC.com）设置了有一定影响力的Linux专题栏目。

包括：

[Ubuntu专题](#)

[Fedora专题](#)

[RedHat专题](#)

[SUSE专题](#)

[红旗Linux专题](#)

[Android专题](#)

[Linux公社简介](#) - [广告服务](#) - [网站地图](#) - [帮助信息](#) - [联系我们](#)

本站（LinuxIDC）所刊载文章不代表同意其说法或描述，仅为提供更多信息，也不构成任何建议。

本站带宽由[\[6688.CC\]](#)友情提供

Copyright © 2006-2011 [Linux公社](#) All rights reserved

6.2.1 客户机配置.....	14
6.2.2 一个测试例子WordCount.....	15
6.2.3 编写Hadoop应用程序并在集群上运行.....	16
6.2.4 三种模式下编译运行Hadoop应用程序.....	16
6.2.5 提交多个作业到集群.....	18
附 程序	19

介绍

这是本人在完全分布式环境下在 Cent-OS 中配置 Hadoop-0.19.1 时的总结文档，但该文档也适合其他版本的 Linux 系统和目前各版本的 Hadoop(Hadoop-0.20 之后的版本配置文件 `hadoop-site.xml` 被拆分成了三个 `core-site.xml`, `hdfs-site.xml` 和 `mapred-site.xml`，这里会说明 0.20 后的版本中如何配置这三个文件)。

Hadoop 配置建议所有配置文件中使用主机名进行配置，并且机器上应在防火墙中开启相应端口，并设置 SSHD 服务为开机启动，此外 java 环境变量可以在 `/etc/profile` 中配置。

0 集群网络环境介绍

集群包含三个节点：1 个 namenode，2 个 datanode，节点之间局域网连接，可以相互 ping 通。节点 IP 地址和主机名分布如下：

10.10.97.132	gc03vm12	namenode
10.10.97.142	gc04vm12	datanode01
10.10.97.144	gc04vm14	datanode02

所有节点均是 Cent-OS 系统，**防火墙均禁用，sshd 服务均开启并设置为开机启动。**

所有节点上均创建了一个 hadoop 用户，用户主目录是 `/home/hadoop`。

所有节点上均创建了一个目录 `/usr/local/hadoop`，并且拥有者是 hadoop 用户。因为该目录用于安装 hadoop，用户对其必须有 `rwX` 权限。（一般做法是 root 用户在 `/usr/local` 下创建 hadoop 目录，并修改该目录拥有者为 nutch(`chown -R nutch:nutch /usr/local/hadoop`)）。

1 /etc/hosts 文件配置

(1)namenode 节点上编辑 `/etc/hosts` 文件

将所有节点的名字和 IP 地址写入其中，写入如下内容，注意注释掉 127.0.0.1 行，保证内容如下：

10.10.97.132	gc03vm12
--------------	----------

```
10.10.97.142 gc04vm12
10.10.97.144 gc04vm14
# 127.0.0.1 centos54 localhost.localdomain localhost
```

(2) 将 Namenode 上的/etc/hosts 文件复制到所有数据节点上，操作步骤如下：

root 用户登录 namenode；

执行命令：scp /etc/hosts root@datanode ip:/etc/hosts

2 SSH 无密码验证配置

Hadoop 需要使用 SSH 协议，namenode 将使用 SSH 协议启动 namenode 和 datanode 进程，**datanode 向 namenode 传递心跳信息可能也是使用 SSH 协议，这是我认为是，还没有做深入了解，datanode 之间可能也需要使用 SSH 协议。假若是，则需要配置使得所有节点之间可以相互 SSH 无密码登陆验证。**下面给出了两种配置方式，用户可以选择第一种，若实验中出现问题可选择第二种进行尝试。

2.1 选择一：配置 Namenode 无密码登录所有 Datanode

(0) 原理

Namenode 作为客户端，要实现无密码公钥认证，连接到服务端 datanode 上时，需要在 namenode 上生成一个密钥对，包括一个公钥和一个私钥，而后将公钥复制到 datanode 上。当 namenode 通过 ssh 连接 datanode 时，datanode 就会生成一个随机数并用 namenode 的公钥对随机数进行加密，并发送给 namenode。namenode 收到加密数之后再用私钥进行解密，并将解密数回传给 datanode，datanode 确认解密数无误之后就允许 namenode 进行连接了。这就是一个公钥认证过程，其间不需要用户手工输入密码。重要过程是将客户端 namenode 公钥复制到 datanode 上。

所有机器上生成密码对，所有节点上执行以下命令：

```
ssh-keygen -t rsa
```

Generating public/private rsa key pair.

Enter file in which to save the key (/home/ hadoop /.ssh/id_rsa): 默认路径

Enter passphrase (empty for no passphrase): 回车，空密码

Enter same passphrase again:

Your identification has been saved in /home/ hadoop /.ssh/id_rsa.

Your public key has been saved in /home/ hadoop /.ssh/id_rsa.pub.

这将在 /home/hadoop/.ssh/ 目录下生成一个私钥 id_rsa 和一个公钥 id_rsa.pub。

在 namenode 节点上做如下配置

```
cp id_rsa.pub authorized_keys namenode 的公钥
```

使用 SSH 协议将 namenode 的公钥信息 authorized_keys 复制到所有 DataNode 的.ssh 目录下。

```
scp authorized_keys data 节点 ip 地址:/home/hadoop/.ssh
```

这样配置过后，namenode 可以无密码登录所有 datanode，可以通过命令

“ssh 节点 ip 地址”来验证。

配置完毕，在 namenode 上执行“ssh 本机，所有数据节点”命令，因为 ssh 执行一次之后将不会再询问。

2.1 选择二：配置所有节点之间 SSH 无密码验证

(0) 原理

节点 A 要实现无密码公钥认证连接到节点 B 上时，节点 A 是客户端，节点 B 是服务端，需要在客户端 A 上生成一个密钥对，包括一个公钥和一个私钥，而后将公钥复制到服务端 B 上。当客户端 A 通过 ssh 连接服务端 B 时，服务端 B 就会生成一个随机数并用客户端 A 的公钥对随机数进行加密，并发送给客户端 A。客户端 A 收到加密数之后再用私钥进行解密，并将解密数回传给 B，B 确认解密数无误之后就允许 A 进行连接了。这就是一个公钥认证过程，其间不需要用户手工输入密码。重要过程是将客户端 A 公钥复制到 B 上。

因此如果要实现所有节点之间无密码公钥认证，则需要将所有节点的公钥都复制到所有节点上。

(1) 所有机器上生成密码对

(a) 所有节点用 hadoop 用户登陆，并执行以下命令，生成 rsa 密钥对：

```
ssh-keygen -t rsa
```

这将在 /home/hadoop/.ssh/ 目录下生成一个私钥 id_rsa 和一个公钥 id_rsa.pub。

(b) 将所有 datanode 节点的公钥 id_rsa.pub 传送到 namenode 上：

```
cp id_rsa.pub datanode01.id_rsa.pub
```

```
scp datanode01.id_rsa.pub namenode 节点 ip 地址:/home/hadoop/.ssh
```

.....

```
cp id_rsa.pub datanoden.id_rsa.pub
```

```
scp datanoden.id_rsa.pub namenode 节点 ip 地址:/home/hadoop/.ssh
```

(c) namenode 节点上综合所有公钥(包括自身)并传送到所有节点上

```
cp id_rsa.pub authorized_keys 这是 namenode 自己的公钥
```

```
cat datanode01.id_rsa.pub >> authorized_keys
```

.....

```
cat datanode0n.id_rsa.pub >> authorized_keys
```

然后使用 SSH 协议将所有公钥信息 authorized_keys 复制到所有 DataNode 的 .ssh 目录下

```
scp authorized_keys data 节点 ip 地址:/home/hadoop/.ssh
```

这样配置过后，所有节点之间可以相互 SSH 无密码登陆，可以通过命令“ssh 节点 ip 地址”来验证。

配置完毕，在 namenode 上执行“ssh 本机，所有数据节点”命令，因为 ssh 执行一次之后将不会再询问。



Linux公社（LinuxIDC.com）于2006年9月25日注册并开通网站，Linux现在已经成为一种广受关注和支持的一种操作系统，IDC是互联网数据中心，LinuxIDC就是关于Linux的数据中心。

LinuxIDC.com提供包括Ubuntu，Fedora，SUSE技术，以及最新IT资讯等Linux专业类网站。

并被收录到Google 网页目录-计算机 > 软件 > 操作系统 > Linux 目录下。

Linux公社（LinuxIDC.com）设置了有一定影响力的Linux专题栏目。

包括：

[Ubuntu专题](#)

[Fedora专题](#)

[RedHat专题](#)

[SUSE专题](#)

[红旗Linux专题](#)

[Android专题](#)

[Linux公社简介](#) - [广告服务](#) - [网站地图](#) - [帮助信息](#) - [联系我们](#)

本站（LinuxIDC）所刊载文章不代表同意其说法或描述，仅为提供更多信息，也不构成任何建议。

本站带宽由[\[6688.CC\]](#)友情提供

Copyright © 2006-2011 [Linux公社](#) All rights reserved

3 JDK 安装和 Java 环境变量配置

3.1 安装 JDK 1.6

root 用户登陆，在 Namenode 节点上新建文件夹/usr/program，下载 JDK 安装包 jdk-6u13-linux-i586.bin，复制到目录/usr/ program 下，在命令行进入该目录，执行命令“./ jdk-6u13-linux-i586.bin”，命令运行完毕，将在目录下生成文件夹 jdk1.6.0_13，安装完毕。

安装完成后，修改/usr/program 目录拥有着为 nutch 用户，

```
Chown -R nutch:nutch /usr/program
```

/usr/ program 目录需要复制到所有数据节点上。

3.2 Java 环境变量配置

root 用户登陆，命令行中执行命令” vi /etc/profile”，并加入以下内容，配置环境变量(注意/etc/profile 这个文件很重要，后面 Hadoop 的配置还会用到)。

```
# set java environment
```

```
export JAVA_HOME=/usr/program/jdk1.6.0_13/
```

```
export JRE_HOME=/usr/program/jdk1.6.0_13/jre
```

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

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

保存并退出，执行以下命令使配置生效

```
chmod +x /etc/profile
```

```
source /etc/profile
```

配置完毕，在命令行中使用命令” java -version” 可以判断是否成功。在 hadoop 用户下测试 java -version，一样成功。

将 Namenode 上的/etc/profile 复制到所有数据节点上。操作步骤如下：

root 用户登录 namenode；

执行命令：scp /etc/profile root@datanode ip:/etc/profile

4 Hadoop 集群配置

在 namenode 上执行：

Hadoop 用户登录。

下载 `hadoop-0.19.1`，将其解压到 `/usr/local/hadoop` 目录下，解压后目录形式是 `/usr/local/hadoop/hadoop-0.19.1`。使用如下命令：

```
tar zxvf hadoop-0.19.1.tar.gz
```

(1) 配置 Hadoop 的配置文件

(a) 配置 `hadoop-env.sh`

```
$ vi nutch-1.0/conf/hadoop-env.sh
```

```
# set java environment
```

```
export JAVA_HOME=/usr/program/jdk1.6.0_13/
```

(b) 配置 `conf/hadoop-site.xml`

Hadoop 配置参数的含义请参考 `conf/Hadoop-default.xml`。

Hadoop-0.20 之后的版本请分别配置 `core-site.xml`，`hdfs-site.xml` 和 `mapred-site.xml` 三个配置文件，配置方法即将下面 `hadoop-site.xml` 文件中的三块参数分别复制到三个文件当中。

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!-- Put site-specific property overrides in this file. -->

<configuration>

<!--core-site.xml-->
<property>
  <name>fs.default.name</name>
  <value>hdfs://gc03vm12:9000</value>
  <description>HDFS 的 URI，文件系统://namenode 标识:端口号</description>
</property>

<property>
  <name>hadoop.tmp.dir</name>
  <value>/usr/local/hadoop/hadooptmp</value>
  <description>namenode 上本地的 hadoop 临时文件夹</description>
</property>

<!--hdfs-site.xml-->
<property>
  <name>dfs.name.dir</name>
  <value>/usr/local/hadoop/hdfs/name</value>
  <description>namenode 上存储 hdfs 名字空间元数据 </description>
</property>

<property>
  <name>dfs.data.dir</name>
  <value>/usr/local/hadoop/hdfs/data</value>
  <description>datanode 上数据块的物理存储位置</description>
```



```

</property>

<property>
  <name>dfs.replication</name>
  <value>2</value>
  <description>副本个数，不配置默认是 3, 应小于 datanode 机器数量</description>
</property>

<!--mapred-site.xml-->
<property>
  <name>mapred.job.tracker</name>
  <value>gc03vm12:9001</value>
  <description>jobtracker 标识:端口号，不是 URI</description>
</property>

<property>
  <name>mapred.local.dir</name>
  <value>/usr/local/hadoop/mapred/local</value>
  <description>tasktracker 上执行 mapreduce 程序时的本地目录</description>
</property>

<property>
  <name>mapred.system.dir</name>
  <value>/tmp/hadoop/mapred/system</value>
  <description>这个是 hdfs 中的目录，存储执行 mr 程序时的共享文件</description>
</property>

</configuration>

```

(c) 配置 masters 文件, 加入 namenode 的主机名

```
gc03vm12
```

(d) 配置 slaves 文件, 加入所有 datanode 的主机名

```
gc04vm12
gc04vm14
```

(2) 复制配置好的各文件到所有数据节点上。

root 用户下:

```

scp /etc/hosts      数据节点 ip 地址:/etc/hosts
scp /etc/profile    数据节点 ip 地址:/etc/profile
scp /usr/program     数据节点 ip 地址:/usr/program

```

nutch 用户下:

```
scp /usr/local/hadoop 数据节点 ip 地址: /usr/local/
```

5 Hadoop 集群启动

Namenode 执行：

格式化 namenode，格式化后在 namenode 生成了 `hdfs/name` 文件夹

```
bin/hadoop namenode -format
```

启动 hadoop 所有进程，

```
bin/start-all.sh (或者先后执行 start-dfs.sh 和 start-mapreduce.sh)。
```

可以通过以下启动日志看出，首先启动 namenode，然后启动 datanode1, datanode2, 然后启动 secondarynamenode。再启动 jobtracker，然后启动 tasktracker1, 最后启动 tasktracker2。

```
starting namenode, logging to /home/zkl/hadoopinstall/hadoop-0.20.1/bin/../logs/hadoop-zkl-namenode-zkl-ubuntu.out
210.77.9.199: starting datanode, logging to /home/zkl/hadoopinstall/hadoop-0.20.1/bin/../logs/hadoop-zkl-datanode-zkl-ubuntu.out
210.77.9.216: starting datanode, logging to /home/zkl/hadoopinstall/hadoop-0.20.1/bin/../logs/hadoop-zkl-datanode-zkl-ubuntu.out
210.77.9.204: starting secondarynamenode, logging to /home/zkl/hadoopinstall/hadoop-0.20.1/bin/../logs/hadoop-zkl-secondarynamenode-zk
starting jobtracker, logging to /home/zkl/hadoopinstall/hadoop-0.20.1/bin/../logs/hadoop-zkl-jobtracker-zkl-ubuntu.out
210.77.9.199: starting tasktracker, logging to /home/zkl/hadoopinstall/hadoop-0.20.1/bin/../logs/hadoop-zkl-tasktracker-zkl-ubuntu.out
210.77.9.216: starting tasktracker, logging to /home/zkl/hadoopinstall/hadoop-0.20.1/bin/../logs/hadoop-zkl-tasktracker-zkl-ubuntu.out
```

启动 hadoop 成功后，在 namenode 中生成了 `hadooptmp` 文件夹，在 datanode 中生成了 `hdfs` 文件夹和 `mapred` 文件夹。

namenode 上用 java 自带的小工具 `jps` 查看进程

```
# jps
8383 JobTracker
8733 Jps
8312 SecondaryNameNode
8174 NameNode
```

每个 datanode 上查看进程

```
# jps
7636 DataNode
7962 Jps
7749 TaskTracker
```

在 namenode 上查看集群状态

```
bin/hadoop dfsadmin -report
```

```
Configured Capacity: 16030539776 (14.93 GB)
Present Capacity: 7813902336 (7.28 GB)
DFS Remaining: 7748620288 (7.22 GB)
DFS Used: 65282048 (62.26 MB)
DFS Used%: 0.84%
```

```
-----
Datanodes available: 2 (2 total, 0 dead)
```

```
Name: 10.10.97.142:50010
```

Decommission Status : Normal
 Configured Capacity: 8015269888 (7.46 GB)
 DFS Used: 32641024 (31.13 MB)
 Non DFS Used: 4364853248 (4.07 GB)
 DFS Remaining: 3617775616(3.37 GB)
 DFS Used%: 0.41%
 DFS Remaining%: 45.14%
 Last contact: Thu May 13 06:17:57 CST 2010

Name: 10.10.97.144:50010
 Decommission Status : Normal
 Configured Capacity: 8015269888 (7.46 GB)
 DFS Used: 32641024 (31.13 MB)
 Non DFS Used: 3851784192 (3.59 GB)
 DFS Remaining: 4130844672(3.85 GB)
 DFS Used%: 0.41%
 DFS Remaining%: 51.54%
 Last contact: Thu May 13 06:17:59 CST 2010

Hadoop 的 web 方式查看: [http:// namenode ip 地址:50070](http://namenodeip地址:50070)

NameNode 'gc03vm12:9000'

Started: Thu May 13 02:39:45 CST 2010
 Version: 0.19.1, r745977
 Compiled: Fri Feb 20 00:16:34 UTC 2009 by ndaley
 Upgrades: There are no upgrades in progress.

[Browse the filesystem](#)
[NameNode Logs](#)

Cluster Summary

337 files and directories, 360 blocks = 697 total. Heap Size is 13.36 MB / 966.69 MB (1%)

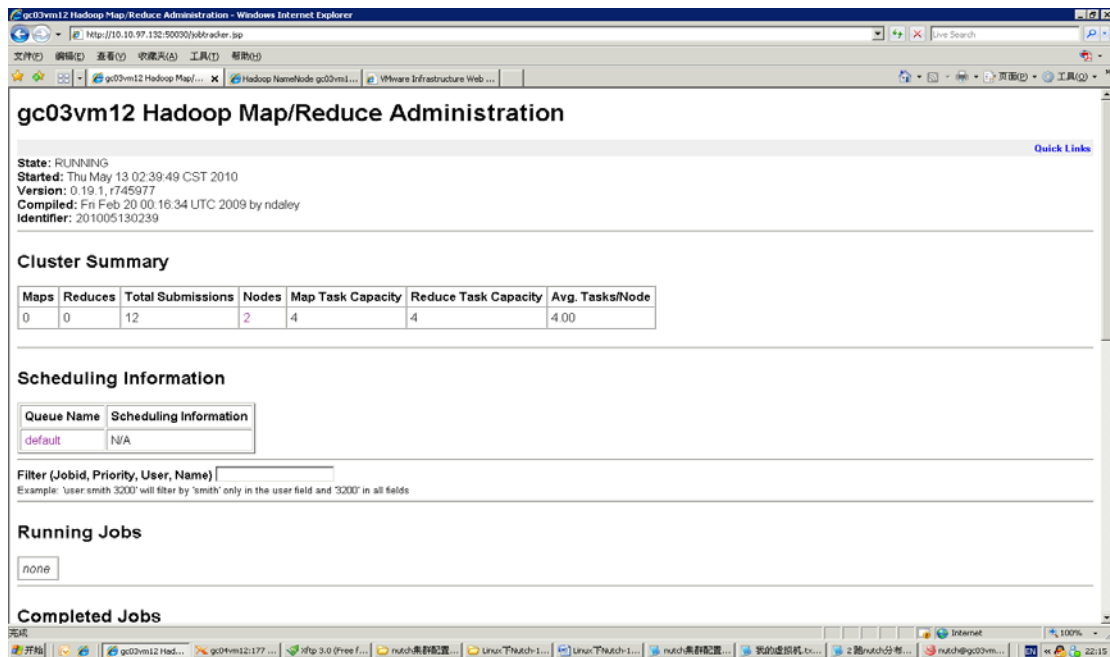
Configured Capacity : 14.93 GB
 DFS Used : 109.84 MB
 Non DFS Used : 7.28 GB
 DFS Remaining : 7.54 GB
 DFS Used% : 0.72 %
 DFS Remaining% : 50.53 %
[Live Nodes](#) : 2
[Dead Nodes](#) : 0

Live Datanodes : 2

Node	Last Contact	Admin State	Configured Capacity (GB)	Used (GB)	Non DFS Used (GB)	Remaining (GB)	Used (%)	Used (%)	Remaining (%)	Blocks
gc04vm12	0	In Service	7.46	0.05	3.64	3.77	0.72		50.53	360
gc04vm14	0	In Service	7.46	0.05	3.64	3.77	0.72		50.53	360

Dead Datanodes : 0

Hadoop 查看工作情况: [http:// namenode ip 地址:50030](http://namenodeip地址:50030)



6 Hadoop 使用

Hadoop 有两个重要的模块：MapReduce 和 HDFS，HDFS 是一个分布式文件系统，用于存储数据，MapReduce 是一个编程框架，Hadoop 中运行的程序均是 MapReduce 作业，一个作业分为若干个 Map 任务和 Reduce 任务。

6.1 客户机与 HDFS 进行交互

6.1.1 客户机配置

可以查看 HDFS 中的数据，向 HDFS 中写入数据。

(1) 选择一台机器，该机器可以是 Hadoop 集群中的节点，也可以是集群之外的机器。下面说明在集群之外的客户机上如何操作与 HDFS 交互，集群之内的节点省去配置过程。

(2) 集群之外的机器请保证和 Hadoop 集群是网络连通的，并且安装了 Hadoop(解压安装包即可)并在 conf/hadoop-site.xml 中做了相关配置，至少配置如下：

```
<configuration>

<!--core-site.xml-->
  <property>
    <name>fs.default.name</name>
    <value>hdfs://gc04vm12:9000</value>
    <description> </description>
  </property>
```

```

<!--mapred-site.xml-->
<property>
  <name>mapred.job.tracker</name>
  <value>gc03vm12:9001</value>
  <description>jobtracker 标识:端口号, 不是 URI</description>
</property>

</configuration>

```

(3) 按照以上步骤配置完成后, 即可在客户机的命令行中执行命令, 查看 HDFS 文件系统。

6.1.2 列出 HDFS 根目录/下的文件

```

[root@gc03vm12 nutch-1.0]# bin/hadoop dfs -ls /
Found 3 items
drwxr-xr-x   - hadoop supergroup          0 2010-05-21 00:42 /tmp
drwxr-xr-x   - hadoop supergroup            0 2010-05-21 00:53 /user
drwxr-xr-x   - hadoop supergroup            0 2010-05-21 00:55 /usr

```

第一列是目录权限, 第二列的 **hadoop** 是目录拥有者, 第三列是组名, 第 4 列是目录大小(单位是 B), 第 5 列是目录的绝对路径。这里表示/目录下有三个目录。这里的用户 **hadoop** 是安装 hadoop 的用户, 是超级用户, 相当于 Linux 操作系统的 root 用户, 组 **supergroup** 相当于 root 用户组。

6.1.3 列出当前用户主目录下的文件

Hadoop 默认当前 HDFS 中的用户就是当前登录客户机的用户。

```

[root@gc03vm12 nutch-1.0]# bin/hadoop dfs -ls
ls: Cannot access .: No such file or directory.

```

提示不能访问时因为 hdfs 中没有 /user/root 目录。

注意: 这里的当前用户主目录是客户机中当前登录用户的主目录, 即 HDFS 中的 “/user/用户名” 目录, 由于当前是 root 用户登录, 命令 “bin/hadoop dfs -ls” 访问的是 HDFS 中的 “/user/root 目录”。此时若 /user/root 目录不存在, 则会出现上面的提示。

由于 HDFS 中不存在 root 用户, 所以客户机当前登录用户无法向 HDFS 中写入数据, 因为对 HDFS 中的所有目录没有写权限, 只有 r 读权限。要使当前用户能够向 HDFS 中写入数据, 必须在 HDFS 中创建 root 用户并且创建相应目录, 赋予相关权限。

总之, HDFS 的用户权限和 Linux 一样重要。

6.1.4 HDFS 用户管理

创建 HDFS 用户需要使用 hadoop 用户登录客户机器, 并且执行 hadoop 相关

命令。由于 **Hadoop 默认当前 HDFS 中的用户就是当前登录客户机的用户**，所以当前 HDFS 用户即为 Hadoop 超级用户 hadoop。

Hadoop 似乎没有提供创建用户的命令，但要在 HDFS 中创建用户和用户组可以这样做。

(i) Hadoop 超级用户 hadoop 在 hdfs 中创建目录 /user/root，

即 `bin/hadoop dfs -mkdir /user/root`

(ii) 更改 /user/root 目录所属用户和组，

即 `bin/hadoop dfs -chown -R root:root /user/root`，命令执行完毕 Hadoop 将默认创建有用户 root，用户组 root。

注意：若此处没有指定组，则默认将 root 用户分配到 supergroup 组，

`bin/hadoop dfs -chown -R root /user/root`

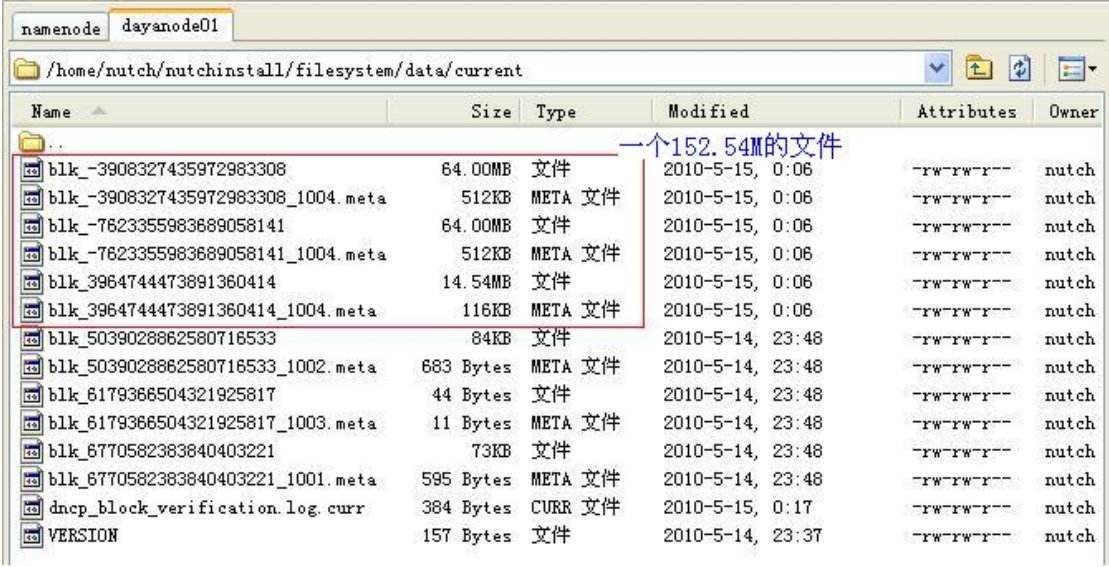
(ii) 这样就相当于在 hdfs 中创建了用户 root，组 root。并且当前客户机的 root 用户对 hdfs 中的 /user/root 目录进行 rwx。

6.1.5 复制本地数据到 HDFS 中

```
[root@gc03vm12 nutch-1.0]# bin/hadoop dfs -copyFromLocal /local/x
/user/root/
```

执行以上命令即能将本地数据上传到 HDFS 中，上传的文件将会被分块，并且数据块将物理存储在集群数据节点的 `hadoop-site.xml` 文件中的 `dfs.data.dir` 参数指定的目录下，用户可以登录数据节点查看相应数据块。

HDFS 中一个文件对应若干数据块，如果文件小于块大小（默认 64M），则将会存储到一个块中，块大小即文件大小。若文件很大，则分为多个块存储。



Name	Size	Type	Modified	Attributes	Owner
blk_-3908327435972983308	64.00MB	文件	2010-5-15, 0:06	-rw-rw-r--	nutch
blk_-3908327435972983308_1004.meta	512KB	META 文件	2010-5-15, 0:06	-rw-rw-r--	nutch
blk_-7623355983689058141	64.00MB	文件	2010-5-15, 0:06	-rw-rw-r--	nutch
blk_-7623355983689058141_1004.meta	512KB	META 文件	2010-5-15, 0:06	-rw-rw-r--	nutch
blk_3964744473891360414	14.54MB	文件	2010-5-15, 0:06	-rw-rw-r--	nutch
blk_3964744473891360414_1004.meta	116KB	META 文件	2010-5-15, 0:06	-rw-rw-r--	nutch
blk_5039028862580716533	84KB	文件	2010-5-14, 23:48	-rw-rw-r--	nutch
blk_5039028862580716533_1002.meta	683 Bytes	META 文件	2010-5-14, 23:48	-rw-rw-r--	nutch
blk_6179366504321925817	44 Bytes	文件	2010-5-14, 23:48	-rw-rw-r--	nutch
blk_6179366504321925817_1003.meta	11 Bytes	META 文件	2010-5-14, 23:48	-rw-rw-r--	nutch
blk_6770582383840403221	73KB	文件	2010-5-14, 23:48	-rw-rw-r--	nutch
blk_6770582383840403221_1001.meta	595 Bytes	META 文件	2010-5-14, 23:48	-rw-rw-r--	nutch
dncp_block_verification.log.curr	384 Bytes	CURR 文件	2010-5-15, 0:17	-rw-rw-r--	nutch
VERSION	157 Bytes	文件	2010-5-14, 23:37	-rw-rw-r--	nutch

6.1.6 数据副本说明

Hadoop-site.xml 文件中的 `dfs.replication` 参数指定了数据块的副本数量。一个文件被分为若干数据块，其所有数据块副本的名字和元数据都是一样的，例如下图显示了上传一个目录（包含两个小文件）到 HDFS 后数据节点中数据块情

况:

Name	Size	Type	Modified	Attributes	Owner
..					
blk_6842982285481439513	4 Bytes	文件	2010-5-22, 3:59	-rw-rw-r--	hadoop
blk_6842982285481439513_1002.meta	11 Bytes	META 文件	2010-5-22, 3:59	-rw-rw-r--	hadoop
blk_2086040824686293736	120 Bytes	文件	2010-5-22, 4:01	-rw-rw-r--	hadoop
blk_2086040824686293736_1003.meta	11 Bytes	META 文件	2010-5-22, 4:01	-rw-rw-r--	hadoop
dncp_block_verification.log.curr	290 Bytes	CURR 文件	2010-5-22, 4:07	-rw-rw-r--	hadoop
VERSION	157 Bytes	文件	2010-5-22, 3:58	-rw-rw-r--	hadoop

图 节点一上的副本

Name	Size	Type	Modified	Attributes	Owner
..					
blk_6842982285481439513	4 Bytes	文件	2010-5-22, 4:05	-rw-rw-r--	hadoop
blk_6842982285481439513_1002.meta	11 Bytes	META 文件	2010-5-22, 4:05	-rw-rw-r--	hadoop
blk_2086040824686293736	120 Bytes	文件	2010-5-22, 4:08	-rw-rw-r--	hadoop
blk_2086040824686293736_1003.meta	11 Bytes	META 文件	2010-5-22, 4:08	-rw-rw-r--	hadoop
dncp_block_verification.log.curr	193 Bytes	CURR 文件	2010-5-22, 4:14	-rw-rw-r--	hadoop
VERSION	156 Bytes	文件	2010-5-22, 4:04	-rw-rw-r--	hadoop

图 节点二上的副本

6.1.7 hadoop-site.xml 参数说明

(1) hadoop.tmp.dir

该参数默认值是“/tmp/hadoop-**{当前登录用户名}**”。

它是本地路径，当第一次启动 Hadoop 集群进程时在 namenode 节点的本地创建该目录，其作用是存储相关临时文件。

(2) mapred.system.dir

该参数默认值是\${hadoop.tmp.dir}/mapred/system，它所表示的目录是 hdfs 中的路径，是相对于 dfs.default.name 的路径，即它在 hdfs 中的绝对路径是\${dfs.default.name}/{mapred.system.dir}。

该参数指定的目录的作用是当作业运行时，存储作业相关文件，供 tasktracker 节点共享。

一般 hdfs 系统中/目录下可以看到该参数指定的目录，如

```
[nutch@gc04vml2 nutch-1.0]$ bin/hadoop dfs -lsr /
drwxr-xr-x  - nutch supergroup    /tmp
drwxr-xr-x  - nutch supergroup    /tmp/hadoop
drwxr-xr-x  - nutch supergroup    /tmp/hadoop/mapred
drwx-wx-wx  - nutch supergroup    /tmp/hadoop/mapred/system (即此)
```

其他参数参见 hadoop-default.xml 中的说明。

6.1.8 HDFS 中的路径

首先请查阅资料，把握 URI 的概念。在 HDFS 中，例如下面这些形式均是 URI（注意不是 URL，URI 概念比 URL 更广）。例如 file:///，hdfs://x/y/z，/x/y/z，Z。



Linux公社（LinuxIDC.com）于2006年9月25日注册并开通网站，Linux现在已经成为一种广受关注和支持的一种操作系统，IDC是互联网数据中心，LinuxIDC就是关于Linux的数据中心。

LinuxIDC.com提供包括Ubuntu，Fedora，SUSE技术，以及最新IT资讯等Linux专业类网站。

并被收录到Google 网页目录-计算机 > 软件 > 操作系统 > Linux 目录下。

Linux公社（LinuxIDC.com）设置了有一定影响力的Linux专题栏目。

包括：

[Ubuntu专题](#)

[Fedora专题](#)

[RedHat专题](#)

[SUSE专题](#)

[红旗Linux专题](#)

[Android专题](#)

[Linux公社简介](#) - [广告服务](#) - [网站地图](#) - [帮助信息](#) - [联系我们](#)

本站（LinuxIDC）所刊载文章不代表同意其说法或描述，仅为提供更多信息，也不构成任何建议。

本站带宽由[\[6688.CC\]](#)友情提供

Copyright © 2006-2011 [Linux公社](#) All rights reserved

HDFS 路径应该可以分为三种：绝对 URI 路径，即

`hdfs://namenode:端口/xxxx/xxxx`

这种形式；HDFS 绝对路径，例如 `/user` 或者 `///user`，**注意使用/或者///表示根目录，而不能使用//**；HDFS 相对路径，例如 `x`，此路径往往是相对于当前用户主目录 `/user/用户名` 而言，例如 `x` 对应的 HDFS 绝对路径是 `/user/hadoop/x`。

Hadoop-site.xml 中参数 `mapred.system.dir` 的值所指的路径是 HDFS 的绝对路径，它和 `fs.default.name` 参数一起构成了绝对 URI，例如参数值 `/tmp/hadoop/mapred/system` 最终将对应的是绝对 URI：

`hdfs://gc04vm12:9000/tmp/hadoop/mapred/system`

无论使用何种路径，均能查看 HDFS 中的数据，例如：

```
[nutch@gc04vm12 nutch-1.0]$ bin/hadoop dfs -ls hdfs://gc04vm12:9000/user/nutch/data
Found 5 items
drwxr-xr-x - nutch supergroup          0 2010-05-21 01:05 /user/nutch/data/crawldb
drwxr-xr-x - nutch supergroup          0 2010-05-21 01:08 /user/nutch/data/index
drwxr-xr-x - nutch supergroup          0 2010-05-21 01:07 /user/nutch/data/indexes
drwxr-xr-x - nutch supergroup          0 2010-05-21 01:05 /user/nutch/data/linkdb
drwxr-xr-x - nutch supergroup          0 2010-05-21 01:03 /user/nutch/data/segments

[nutch@gc01vm13 nutch-1.0]$ bin/hadoop dfs -ls data 这样访问的是当前用户的主目录下的data目录
Found 5 items
drwxr-xr-x - nutch supergroup          0 2010-05-18 23:43 /user/nutch/data/crawldb
drwxr-xr-x - nutch supergroup          0 2010-05-18 23:45 /user/nutch/data/index
drwxr-xr-x - nutch supergroup          0 2010-05-18 23:44 /user/nutch/data/indexes
drwxr-xr-x - nutch supergroup          0 2010-05-18 23:44 /user/nutch/data/linkdb
drwxr-xr-x - nutch supergroup          0 2010-05-18 23:43 /user/nutch/data/segments

[nutch@gc01vm13 nutch-1.0]$ bin/hadoop dfs -ls /
Found 2 items
drwxr-xr-x - nutch supergroup          0 2010-05-18 23:45 /home
drwxr-xr-x - nutch supergroup          0 2010-05-18 23:45 /user
```

6.1.8 Hadoop 相关命令

Hadoop 提供一系列的命令，在 bin 中，例如 `bin/hadoop fs -x`；`bin/hadoop namenode -x` 等等。其中有些命令只能在 namenode 上执行。

bin 下还有一些控制脚本，例如 `start-all.sh`、`start-mapred.sh`、`start-dfs.sh` 等等。数据节点上运行 `start-all.sh` 将会只启动本节点上的进程，如 `datanode`、`tasktracker`。

6.2 客户机提交作业到集群

6.2.1 客户机配置

可以在客户机上向 Hadoop 集群提交作业。

(1) 选择一台机器，该机器可以是 Hadoop 集群中的节点，也可以是集群之外的机器。下面说明在集群之外的客户机上如何向 hadoop 提交作业，集群之内的节点省去配置过程。

(2) 集群之外的机器请保证和 Hadoop 集群是网络连通的，并且安装了 Hadoop(解压安装包即可)并在 conf/hadoop-site.xml 中做了相关配置，至少配置如下：

```
<configuration>

<!--core-site.xml-->
  <property>
    <name>fs.default.name</name>
    <value>hdfs://gc04vm12:9000</value>
    <description> </description>
  </property>

<!--mapred-site.xml-->
<property>
  <name>mapred.job.tracker</name>
  <value>gc03vm12:9001</value>
  <description>jobtracker 标识:端口号，不是 URI</description>
</property>

</configuration>
```

(3) 按照以上步骤配置完成后，即可在客户机的命令行中执行命令，向 hadoop 提交作业。

6.2.2 一个测试例子 WordCount

计算输入文本中词语数量的程序 WordCount 在 Hadoop 主目录下的 java 程序包 hadoop-0.19.1-examples.jar 中，执行步骤如下：

(1) 上传数据到 HDFS 中

```
bin/hadoop fs -mkdir mytest
```

```
bin/hadoop fs -copyFromLocal /home/hadoop/mytest/input1
```

```
bin/hadoop fs -copyFromLocal /home/hadoop/mytest/input2
```

(2) 执行命令，提交作业

```
bin/hadoop jar hadoop-0.19.1-examples.jar wordcount mytest/* output
```

命令执行完毕，在页面<http://namenodeip:50030/>中能够看到作业执行情况。

(3) 程序输出

程序将统计 mytest 目录下的所有文本文件中词语的数量，并将结果输出到 hdfs 的 output 目录下的 part-00000 文件中。这里的 output 目录是程序生成的

目录，程序运行前不可存在。执行以下命令可以查看结果。

```
bin/hadoop fs -cat output/part-00000
```

6.2.3 编写 Hadoop 应用程序并在集群上运行

这里介绍一个向 HDFS 中写入数据的例子(注意不是 MR 程序)来说明编写 Hadoop 应用程序并放到集群上运行的步骤。

(1) 客户端编写应用程序并编译运行，进行测试。

编写程序一般需要引入 hadoop 相关 jar 包或者直接使用 hadoop 整个程序包，相关代码见附录。

(2) 打包应用程序

在 eclipse 中打包成 jar 文件存储到相应目录下，例如 /hadoop/jarseclipse/dfsOperator.jar。

(3) 上传数据到 HDFS

```
bin/hadoop fs -copyFromLocal local dst
```

本实例中不需要上传数据，一般的程序都涉及输入数据。

(4) 执行应用程序

```
bin/hadoop jar x.jar jar包中主类名 [输入参数] [输出参数]
```

这里使用的命令是：

```
bin/hadoop jar ~/jarseclipse/dfsOperator.jar DFSOperator
```

6.2.4 三种模式下编译运行 Hadoop 应用程序

集群是完全分布式环境，Hadoop 的 MR 程序将以作业的形式提交到集群中运行。我们在客户端编写 Hadoop 应用程序时一般是在伪分布式模式或单击模式下进行编译，然后将编译无误的程序打成包提交到 Hadoop 集群中，当然我们仍可直接让程序在 Hadoop 集群中编译。

(1) 让 Hadoop 应用程序在直接在集群中编译

将 hadoop 整个包导入 eclipse 中，配置 hadoop-site.xml 文件如下：

```
<configuration>

<!--core-site.xml-->
  <property>
    <name>fs.default.name</name>
    <value>hdfs://gc04vm12:9000</value>
    <description> </description>
```

```

</property>

<!--mapred-site.xml-->
<property>
  <name>mapred.job.tracker</name>
  <value>gc03vm12:9001</value>
  <description>jobtracker 标识:端口号, 不是 URI</description>
</property>

</configuration>

```

编写应用程序，编译运行，此时程序将直接在 Hadoop 集群中运行。此种方法在开发中不建议使用，以防止破坏集群环境。

(2) 单机模式下编译 Hadoop 应用程序

将 hadoop 整个包导入 eclipse 中，hadoop-site.xml 文件不做任何配置，保留默认的空配置。

单机模式下运行 Hadoop 应用程序时，程序使用的是本地文件系统。

(3) 伪分布式模式下编译 Hadoop 应用程序

在单机上配置 Hadoop 伪分布式模式，配置文件 hadoop-site.xml 如下：

```

<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!-- Put site-specific property overrides in this file. -->

<configuration>

<!--core-site.xml-->
<property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:9000</value>
  <description>HDFS 的 URI, 文件系统://namenode 标识:端口号</description>
</property>

<!--hdfs-site.xml-->
<property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>副本个数, 不配置默认是 3, 应小于 datanode 机器数量</description>
</property>

<!--mapred-site.xml-->

```

```
<property>
  <name>mapred.job.tracker</name>
  <value>localhost:9001</value>
  <description>jobtracker 标识:端口号, 不是 URI</description>
</property>

</configuration>
```

使用 bin/start-all.sh 启动伪分布式集群。

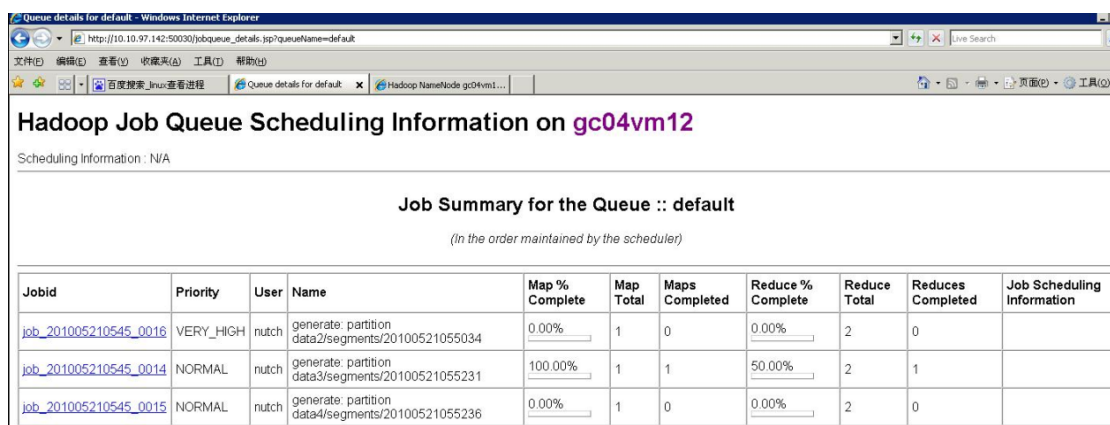
将 hadoop 整个包导入 eclipse 中, 并将 hadoop-site.xml 文件配置为以上内容。(或者直接将上面的 hadoop 目录导入 eclipse 中)。

编写应用程序并运行, 此时程序将在伪分布式模式下运行, 使用的是 HDFS。

6.2.5 提交多个作业到集群

提交一个后, 可以继续提交, 这样集群中将有多个作业, Hadoop 有一个作业队列, 可以在以下网址中查看。

http://10.10.97.142:50030/jobqueue_details.jsp?queueName=default



Hadoop Job Queue Scheduling Information on gc04vm12

Scheduling Information : N/A

Job Summary for the Queue :: default
(In the order maintained by the scheduler)

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information
job_201005210545_0016	VERY_HIGH	nutch	generate: partition data2/segments/20100521055034	0.00%	1	0	0.00%	2	0	
job_201005210545_0014	NORMAL	nutch	generate: partition data3/segments/20100521055231	100.00%	1	1	50.00%	2	1	
job_201005210545_0015	NORMAL	nutch	generate: partition data4/segments/20100521055236	0.00%	1	0	0.00%	2	0	

涉及多个作业时, Hadoop 将对作业进行调度, 默认调度方式是基于优先级的 FIFO 方式。

更改作业优先级命令

作业优先级有五种: VERY_HIGH HIGH NORMAL LOW VERY_LOW

例如:

bin/hadoop job -set-priority job_201005210042_0074 VERY_HIGH

附 程序

Apache所有开源项目下载地址是: <http://archive.apache.org/dist/>

```
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FSDataInputStream;

import org.apache.hadoop.fs.FSDataOutputStream;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IOUtils;

public class DFSOperator {

    /*
     * Write to HDFS
     */

    public static void main(String args[]) {

        Configuration conf=new Configuration();//it will read conf/*.xml, such as
        hadoop-site.xml

        try{

            System.out.println("write to hdfs");

            FileSystem fs=FileSystem.get(conf);

            //Path f=new Path("hdfs:///javawrhdfsfile");//hdfs root directory,it must be
            three "/" or one "/"

            //Path f=new Path("hdfs:/user/nutch/javawrhdfsfile");// "hdfs:" or "hdfs:/" is
            the root directory of hdfs, like "/" in linux

            //Path f=new Path("hdfs:/javawrhdfsfile");//absolute path,f.isAbsolute() is
            "true"
```

```
//Path f=new Path("//javawrhdfsfile"); //it will have error

//Path f=new Path("///javawrhdfsfile");

Path f=new Path("/javawrhdfsfile");

//Path f=new Path("hdfs://10.10.97.142:9000/javawrhdfsfile"); //绝对 URI 前缀请
与 dfs.defaule.name 一致，使用 ip 时因为/etc/hosts 改了，会出现问题

//Path f=new Path("hdfs://gc04vm12:9000/javawrhdfsfile");

//Path f=new Path("javawrhdfsfile");//relative path,it will be in the user home
dir,f.isAbsolute() is "false"

FSDataOutputStream out=fs.create(f,true);

for(int i=0;i<5;i++){

    out.writeChars("zhankunlin"+i+"\n");

}

out.close();

System.out.println("write finished,it produced a file: "+f.toString());//

System.out.println("read the file: "+f.toString());

FSDataInputStream in=fs.open(f);

IOUtils.copyBytes(in, System.out, 4096,false);

System.out.println("read finished!");

IOUtils.closeStream(out);

}catch(IOException e){

    e.printStackTrace();

}

}

}
```