L1、安装环境

1. Hadoop简介
2. 什么是大数据

定义：大数据（big data），指无法在一定时间范围内用常规软件工具进行捕捉、管理和处理的数据集合，是需要新处理模式才能具有更强的决策力、洞察发现力和流程优化能力的海量、高增长率和多样化的信息资产。[

特点：

* 体量巨大
* 数据类型多样，以非结构化数据为主
* 价值密度低
* 产生和要求处理速度快

优势：

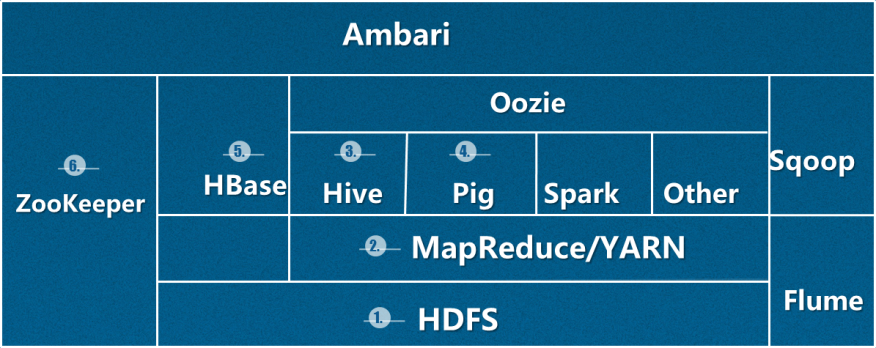
* 低成本
* 高扩展性
* 高效性
* 高容错性

1. 常见的大数据处理系统有哪些

Hadoop、Spark、Stom

1. Hadoop的核心是什么

Hadoop体系结构



Hbase:随机数据读写，实时访问

MapReduce/YARN:基本数据的处理

HDFS:存储管理

1. Hadoop目录结构

bin：执行文件目录

etc：Hadoop配置文件都在此目录

include：包含C语言接口开发所需头文件

lib：包含C语言接口开发所需链接库文件

libexec：运行sbin目录中的脚本会调用该目录下的脚本

logs：日志目录，在运行过Hadoop后会生成该目录

sbin：仅超级用户能够执行的脚本，包括启动和停止dfs、yarn等

share：包括doc和hadoop两个目录

doc目录包含大量的Hadoop帮助文档

hadoop目录包含了运行Hadoop所需的所有jar文件，在开发中用到的jar文件也可在该目录找到

1. 搭建运行环境

1、创建虚拟机

下载VMware-workstation-full-12.5.7.20721，序列号5A02H-AU243-TZJ49-GTC7K-3C61N

激活后创建虚拟机，选择Linux的镜像，这里使用CentOS-7-x86\_64-Minimal-1611.iso。

创建三个虚拟机，分别是master、slave1、slave2

使用root登录

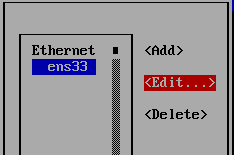
2、连接网络

报错could not retrieve mirrorlist http

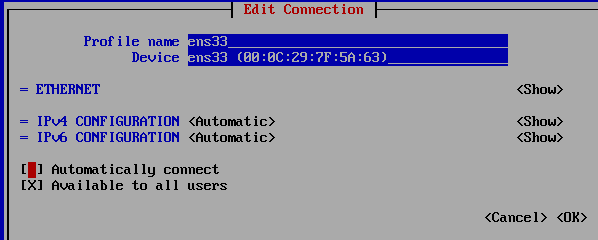
输入nmtui



Enter



Enter



按空格

OK



Enter

 windows使用无线网，因而在虚拟机关闭有线

Backr



设置hostname

选择OK

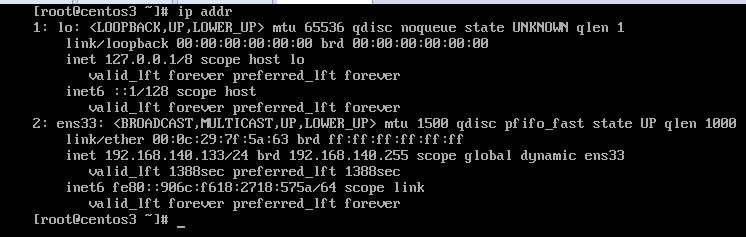
重启网络

systemctl restart network 重启网卡

3、网络配置：不能省略

1. 查看IP

用ip addr查看ip地址，



ens33表示第一块网卡

inet 用来表示网卡的IP地址，此网卡的 IP地址: 192.168.140.133

这里三台虚拟机ip分别为192.168.140.131、192.168.140.132、192.168.140.133

1. 暂时设置

在三台机器上分别设置三个ip和hostname的映射关系

vi /etc/hosts

添加

192.168.140.131 master

192.168.140.132 slave1

192.168.140.133 slave2

:wq

使其生效

在master的终端上输入hostname master，使其生效，并用hostname查看是否生效

在另外两台机器上以此类推

1. 开机后自动设置

vi /etc/sysconfig/network

修改 HOSTNAME=master

:wq

以此类推

1. 关闭防火墙

关闭防火墙systemctl stop firewalld.service

禁止开机启动防火墙systemctl disable firewalld.service

查看防火墙状态firewall-cmd --state

1. 关闭selinux

setenforce 0

getenforce 如果出现Permissive 就表示关闭

4、使用ssh登录

安装ssh-server。输入命令yum install openssh-server -y

用ssh登录

ssh root@192.168.140.131 或者ssh root@master

如果不能登录报错WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!

编辑windows下的.ssh目录中的known\_hosts把192.168.140.131和下面的秘钥删除

* 或者使用Xshell可以保存用户名和密码
* ssh秘钥登录

在客户端主机中生成公/私钥对:ssh-keygen -t rsa

将客户端主机中的公钥分发给服务端主机:ssh-copy-id root@服务端主机地址

在客户端主机上免密码登录至服务端:ssh root@服务端主机地址

或者使用xshell秘钥登录

5、使用scp传送(覆盖)文件(夹)

语法 scp -r 文件或目录 linux用户名@ip地址:linux目录

scp -r jdk-8u144-linux-x64.tar.gz [root@192.168.140.131:/root](mailto:root@192.168.140.131:/root)

或者使用WinSCP

6、安装jdk，配置环境变量

mkdir /usr/java/

mv jdk-8u144-linux-x64.tar.gz /usr/java/

cd /usr/java/

tar -zvxf jdk-8u144-linux-x64.tar.gz

mv jdk1.8.0\_144 jdk

补充：

Ubuntu vi 上下左右变ABCD问题解决方法

错误问题：vi上下左右键显示为ABCD的问题

解决方法：

只要依次执行以下两个命令即可完美解决Ubuntu下vi编辑器方向键变字母的问题。

　　一.执行命令 sudo apt-get remove vim-common

　　二.执行命令 sudo apt-get install vim

vi /etc/profile

export JAVA\_HOME=/usr/java/jdk

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

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

:wq 保存并推出

补充 vi中，i为编辑，:q为推出，:q!为推出并还原，:qw为推出并保存

source /etc/profile 使其生效

java -version 检验是否配置成功

1. 搭建Hadoop环境
2. Hadoop三种运行方式：

* 单机模式：下载Hadoop在系统中，默认情况下之后，它会被配置在一个独立的模式，用于运行Java程序。
* 伪分布式：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。这种模式对开发非常有用。
* 完全分布式：这种模式是完全分布式的最小两台或多台计算机的集群。

这里介绍完全分布式

1. 实现master、 slave1、slave2之间的ssh互相免登陆

在客户端主机中生成公/私钥对:ssh-keygen -t rsa

将客户端主机中的公钥分发给服务端主机:ssh-copy-id 服务器用户名@服务端主机地址或hostname

从windows上将hadoop-2.8.1.tar.gz上传到master服务器上。

1. 配置环境变量

解压hadoop-2.8.1.tar.gz到用户root目录cd /root~tar -zxvf hadoop-2.8.1.tar.gzmv hadoop-2.8.1 hadoop

配置环境变量：/etc/profileexport HADOOP\_HOME=/root/hadoop export PATH=$HADOOP\_HOME/bin:$PATH

source /etc/profile

hadoop version检验是否配置成功

1. 修改8个配置文件

打印环境变量 echo $JAVA\_HOME

cd /root/hadoop/etc/hadoop/

1. vi hadoop-env.sh

export JAVA\_HOME=/usr/java/jdk

1. vi yarn-env.sh

export JAVA\_HOME=/usr/java/jdk

1. vi master

添加master

1. vi slaves

添加 slave1 和slave2

1. core-site.xml hadoop全局配置

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://master:9000</value>

</property>

<property>

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

<value>file:/root/hadoop/tmp</value>

</property>

</configuration>

1. hdfs-site.xml 分布式文件系统HDFS配置

|  |  |
| --- | --- |
| **属性** | **说 明** |
| dfs.namenode.secondary.http-address | Secondary NameNode服务器HTTP地址和端口 |
| dfs.namenode.name.dir | NameNode存储名字空间及汇报日志的位置 |
| dfs.datanode.data.dir | DataNode存放数据块的目录列表 |
| dfs.replication | 冗余备份数量，一份数据可设置多个拷贝 |
| dfs.webhdfs.enabled | 在NameNode和DataNode中启用WebHDFS |

<configuration>

<property>

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

<value>master:50090</value>

</property>

<property>

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

<value>file:/root/hadoop/dfs/name</value>

</property>

<property>

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

<value> file:/root/hadoop/dfs/data</value>

</property>

<property>

<name>dfs.replication</name>

<value>2</value>

</property>

<property>

<name>dfs.webhdfs.enabled</name>

<value>true</value>

</property>

</configuration>

1. mapred-site.xml

<configuration>

<property>

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

<value>yarn</value>

</property>

<property>

<name>mapreduce.jobhistory.address</name>

<value>master:10020</value>

</property>

<property>

<name>mapreduce.jobhistory.webapp.address</name>

<value> master:19888</value>

</property>

</configuration>

1. yarn-site.xml

如果在mapred-site.xml 配置了使用YARN框架，那么YARN框架使用此文件中的配置

配置如下基本属性以指定相关服务地址

（1） yarn.resourcemanager.address

参数解释：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

（2） yarn.resourcemanager.scheduler.address

参数解释：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

（3） yarn.resourcemanager.resource-tracker.address

参数解释：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

（4） yarn.resourcemanager.admin.address

参数解释：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

（5） yarn.resourcemanager.webapp.address

参数解释：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

1. 最后将前面七个配置文件从master复制到slave1和slave2

scp -r /root/hadoop/ root@slave1:/root/

scp -r /root/hadoop/ root@slave2:/root/

注意，不能写成

scp -r /root/hadoop/ root@slave1:/root/hadoop

1. 启动hadoop集群
2. 初始化

格式化NodeName

cd ~/hadoop/bin

hdfs namenode -format

出现exited with status 0 表示退出成功

出现exited with status 1 表示退出失败

1. 启动HDFS

cd sbin

./start-all.sh

1. 查看Hadoop集群运行状态

输入jps查看进程，如果成功则出现

NameNode

SecondaryNameNode

Jps

ResourceManager

查看Hadoop集群运行状态

输入hdfs dfsadmin -report

Web端查看

http://192.168.140.131:50070

L2、HDFS实现分布式存储

1. HDFS简介

1、什么是HDFS

HDFS是Hadoop的核心模块之一

Hadoop的所有模块均构筑于HDFS之上

HDFS为分布式计算提供底层存储支持

2、优势

存储超大文件

标准流式访问：一次写入，多次读取

运行在廉价的商用机器集群上

3、不足

不能满足低延迟的数据访问

无法高效存储大量小文件

暂时不支持多用户写入及随意修改文件

4、数据块

* 每个磁盘都有默认的数据块，这是磁盘进行数据读写的最小单位，一般为512字节。
* 构建于单个磁盘之上的文件系统通过磁盘块来管理文件系统中的块。
* 文件系统块的大小是磁盘块的整数倍，一般为几千字节

5、HDFS的数据块(block)

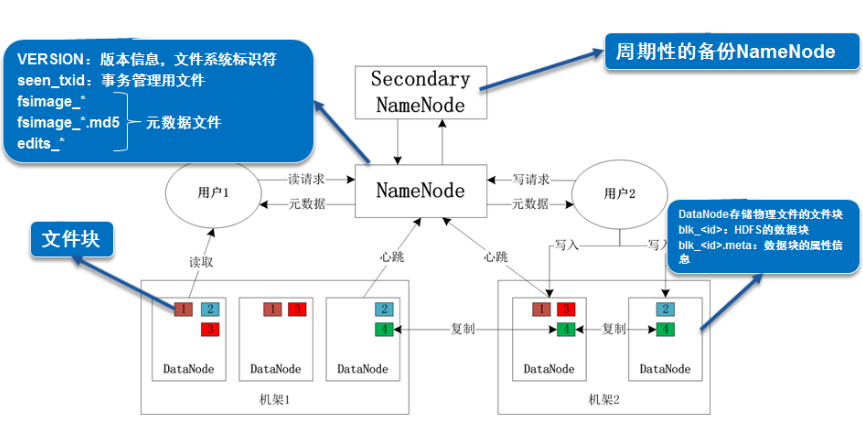
* HDFS也有块(block)，默认64M，与单一磁盘上的文件系统相似。
* HDFS上的文件被划分为块大小的多个分块(chunk)，作为独立的存储单元。
* HDFS与其它文件系统不同的是，HDFS中小于一个块大小的文件不会占据整个块的空间。
* 为什么HDFS中的块比磁盘块大？其目的是为了最小化寻址开销。
* 对分布式文件系统中使用抽象块的好处是，一个文件的大小可以大于网络中任意一个磁盘的容量，简化了存储子系统的设计，块还非常适用于数据备份而提供数据容错能力和提高可用性，将每个块复制到少数几个独立的机器上（默认3个），可确保在块、磁盘或机器发生故障后数据不会丢失。

1. HDFS基本概念

NameNode(管理者)、

DataNode(工作者)节点

Secondary NameNode(周期性备份NameNode)节点



三、HDFS属于Master/Slave主从架构

NameNode节点：Master

负责管理文件块映射

负责处理客户端请求

负责管理HDFS命名空间

负责配置副本策略

DataNode节点：Slave

负责存储文件块

负责处理文件块的读写操作，一次写入、多次读取(不修改)

四、基于Master/Slave的节点管理

1、HDFS添加DataNode节点

修改NameNode节点中的slaves文件加入新主机

2、HDFS负载均衡

命令:sbin/start-balancer.sh

3、HDFS安全机制

将NameNode上存储的元数据文件转移到其它文件系统中

使用Secondary NameNode同步备份

五、查看HDFS的Web界面

http://NameNodeIP:50070

查看NameNode基本信息

查看所有DataNode基本信息

浏览文件系统

五、Hadoop FS Shell命令行

支持多种文件系统的访问：Local和HDFS

fs.defaultFS 的值决定了默认访问的文件系统

hadoop fsck / -files -blocks 列出文件系统中各个文件由哪些块构成

hadoop fs -<args>

在master、slave1、slave2三台服务器上共享

<args>是fs的子命令，包括：

创建目录：mkdir

列表文件：ls

查看文件：cat

转移文件：put、get、mv、cp

删除文件：rm、rmr

管理命令：test、du、expunge

语法：

1、创建目录

在文件系统中创建一个或多个目录

hadoop fs -mkdir /user

hadoop fs -mkdir /user/hadoop

hadoop fs -mkdir /user/hadoop/dir1 /user/hadoop/dir1

2、列出目录和文件

hadoop fs -ls /

hadoop fs -ls /user

3、查看文件

查看HDFS文件内容

hadoop fs -cat /user/text

查看本地文件内容(在slave服务器无法访问master服务器的文件)

hadoop fs -cat file:///root/text

4、从本地系统中复制一个或多个文件到HDFS

hadoop fs -put <localFileUrl>... <hdfsDir>

hadoop fs -put /root/text /user

hadoop fs -put /root/text /root/text2 /user

5、从HDFS复制到本地

hadoop fs -get <hdfsFileUrl> <localDir|localFileUrl>

hadoop fs -get /user/text /root/

hadoop fs -get /user/text $HOME/filename

6、移动文件

hadoop fs -mv <hdfsFileUrl> <hafsDir>

hadoop fs -mv /user/text /user1/

7、复制HDFS文件

hadoop fs -cp <hdfsFileUrl>.. <hdfsDir>

hadoop fs -cp <hdfsFileUrl>.. <hdfsFile2Url>

hadoop fs -cp <localFileUrl>.. <localDir>

hadoop fs -cp /user/text /user1/

hadoop fs -cp file:///root/text file:///root/hadoop

8、删除文件(夹)

hadoop fs -rm <hdfsFileUrl> 只删除文件

hadoop fs -rm -r <hdfsDir> 递归删除文件或文件夹

9、检查文件

hadoop fs -test -<arg> <hdfsFileUrl>

arg为

e:检查是否存在，如果存在返回0，否则返回1

z:是否为0字节，如果是返回0

d:是否是文件夹，如果是返回0，否则返回1

echo $? 查看返回值

10、查看文件或目录大小

hadoop fs -du <hdfsFileUrl>

11、回收站

hadoop fs -expunge 清空回收站

六、Hadoop Java Api编程

1、概述

Hadoop FileSystem API主要类

org.apache.hadoop.fs.FileSystem

org.apache.hadoop.conf.Configuration

org.apache.hadoop.fs.FSDataInputStream

org.apache.hadoop.fs.FSDataOutputStream

org.apache.hadoop.fs.Path

org.apache.hadoop.fs.FileStatus

使用FileSystem API编程步骤

获取Configuration对象

获取文件系统的实例FileSystem对象

使用FileSystem对象操作文件

FileSystem主要方法

|  |  |
| --- | --- |
| **方法** | **说明** |
| static FileSystem get(Configuration conf) | 获得FileSystem实例 |
| FSDataInputStream open(Path f) | 打开文件获得输入流 |
| FSDataInputStream create(Path f) | 创建文件且自动创建目录 |
| void copyFromLocalFile(Path srcPath,Path dstPath) | 将文件从本地系统复制到HDFS |
| boolean delete(Path f,boolean recursive) | 删除文件 |
| boolean rename(Path fromPath,Path toPath) | 重命名文件 |
| FileStatus getFileStatus(Path f) | 获得文件的FileStatus实例 |
| boolean exists(Path f) | 查询文件是否存在 |

2、搭建环境

创建spring boot项目，添加依赖

<dependency>

<groupId>jdk.tools</groupId>

<artifactId>jdk.tools</artifactId>

<version>1.8</version>

<scope>system</scope>

<systemPath>${JAVA\_HOME}/lib/tools.jar</systemPath>

</dependency>

<dependency>

<groupId>org.apache.hadoop</groupId>

<artifactId>hadoop-common</artifactId>

<version>2.8.1</version>

</dependency>

<dependency>

<groupId>org.apache.hadoop</groupId>

<artifactId>hadoop-hdfs</artifactId>

<version>2.8.1</version>

</dependency>

从master服务器上下载core-site.xml，拷贝到src/main/resources下

浏览https://wiki.apache.org/hadoop/WindowsProblems，从github上克隆hadoop

git clone https://github.com/steveloughran/winutils.git

在java的主类中添加static块

**static**{

System.*setProperty*("hadoop.home.dir", "F:\\programs\\winutils\\hadoop-2.8.0-RC3");

System.*setProperty*("HADOOP\_USER\_NAME", "root"); //root为Master服务器上的用户

URL.*setURLStreamHandlerFactory*(**new** FsUrlStreamHandlerFactory());

}

在src/test/java中测试

3、读取文件

@Test

**public** **void** readFile() **throws** IOException {

//查看/user/text文件到

Configuration conf = **new** Configuration();

FileSystem fs = FileSystem.*get*(conf);

FSDataInputStream in = fs.open(**new** Path("hdfs://master:9000/user/text"));

//第四个参数为是否关闭输入流和输出流

IOUtils.*copyBytes*(in, System.***out***, 4096, **true**);

}

4、创建文件

@Test

**public** **void** createFile() **throws** Exception {

Configuration conf = **new** Configuration();

FileSystem fs = FileSystem.*get*(conf);

//如果没有puglic目录，就自动创建该目录

FSDataOutputStream out = fs.create(**new** Path("hdfs://master:9000/public/file"));

InputStream in = **new** ByteArrayInputStream("hello hadoop".getBytes("UTF-8"));

IOUtils.*copyBytes*(in, out, 4096, **true**);

}

5、拷贝本地文件到hdfs

@Test

**public** **void** copyFromLocalFile() **throws** Exception {

Configuration conf = **new** Configuration();

FileSystem fs = FileSystem.*get*(conf);

fs.copyFromLocalFile(**new** Path("D:/hadoop\_dir/test.txt"), **new** Path("hdfs://master:9000/public/test1"));

}

6、从hdfs上拷贝到本地

@Test

**public** **void** copyToLocalFile() **throws** Exception {

Configuration conf = **new** Configuration();

FileSystem fs = FileSystem.*get*(conf);

fs.copyToLocalFile(**new** Path("hdfs://master:9000/public/"), **new** Path("D:/hadoop\_dir/public/"));

}

7、删除文件

@Test

**public** **void** delete() **throws** Exception {

Configuration conf = **new** Configuration();

FileSystem fs = FileSystem.*get*(conf);

//第二个参数为是否递归

fs.delete(**new** Path("hdfs://master:9000/public"), **true**);

}

8、重命名文件

@Test

**public** **void** rename() **throws** Exception {

Configuration conf = **new** Configuration();

FileSystem fs = FileSystem.*get*(conf);

fs.rename(**new** Path("hdfs://master:9000/public/file"), **new** Path("hdfs://master:9000/public/file2"));

}

9、创建目录

@Test

**public** **void** createDir() **throws** IOException {

Configuration conf = **new** Configuration();

FileSystem fs = FileSystem.*get*(conf);

//可以递归创建目录

**boolean** success = fs.mkdirs(**new** Path("hdfs://master:9000/public2/user"));

System.***out***.println(success);

}

10、移动文件或目录

@Test

**public** **void** move() **throws** Exception {

Configuration conf = **new** Configuration();

FileSystem fs = FileSystem.*get*(conf);

//目标目录必须存在

fs.rename(**new** Path("hdfs://master:9000/public/file2"), **new** Path("hdfs://master:9000/public2/file"));

}

11、查询文件是否存在

@Test

**public** **void** exists() **throws** Exception {

Configuration conf = **new** Configuration();

FileSystem fs = FileSystem.*get*(conf);

**boolean** exists = fs.exists(**new** Path("hdfs://master:9000/public/file"));

System.***out***.println(exists);

}

12、列出目录下的所有文件

@Test

**public** **void** listDir() **throws** Exception {

Configuration conf = **new** Configuration();

FileSystem fs = FileSystem.*get*(conf);

FileStatus[] fileStates = fs.listStatus(**new** Path("hdfs://master:9000/public"));

**for** (FileStatus f : fileStates) {

System.***out***.println((f.isDirectory() ? "d" : "f") + "," + f.getPath());

}

}

L3、MapReduce实现分布式运算

一、概述

并行计算：将大型而复杂的任务分解为许多小型任务同时计算，以提高计算机的处理能力。

MapReduce是实现了并行计算的编程模型，为用户提供接口，屏蔽并行计算的细节。

map：映射，负责将任务分解为多个子任务。

reduce：归纳，负责把分解后多任务的处理结果进行汇总。

二、四个基本概念：

Jop:MapReduce程序编写完成后需要配置成一个MapReduce作业。

Task：作业中的任务，分为两类：MapTask和ReduceTask

JopTracker：负责创建MapTask和ReduceTask

TaskTracker：负责执行MapTask和ReduceTask

三、架构设计：

MapReduce V1

Master：JopTracker，初始化作业、创建调度子任务

Slave：TaskTracker，执行任务

MapReduce V2：YARN

四、MapReduce执行数据流

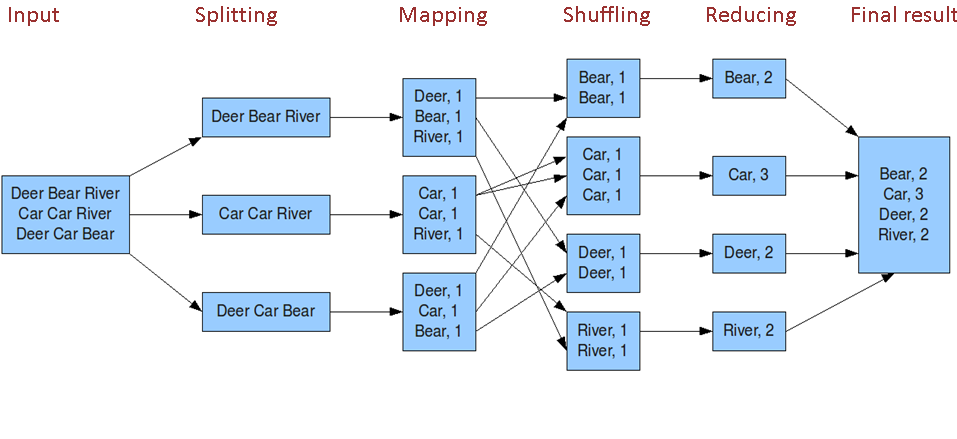
1、input

2、split

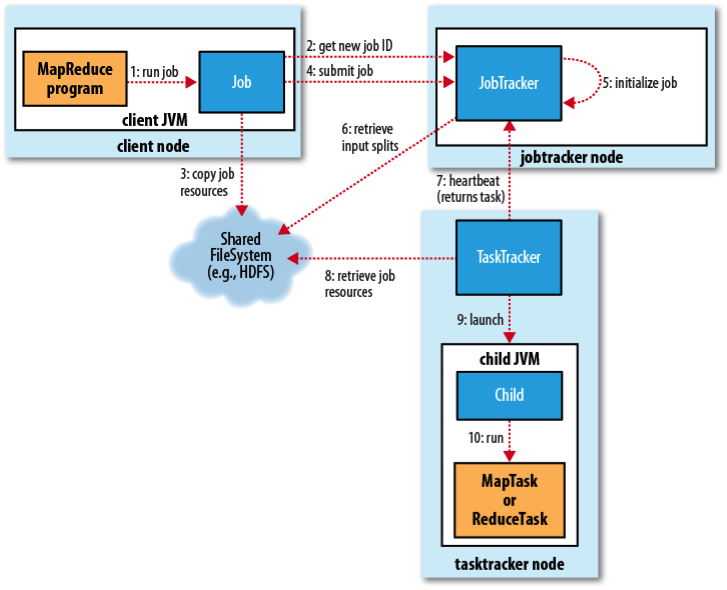
3、map

4、shuffle

5、reduce



五、MapReduce作业的工作原理



六、MapReduce编程模型

Mapper的实现

Reducer的实现

Job的配置

1、org.apache.hadoop.io.Writable接口

* Hadoop重新实现序列化机制
* Hadoop数据类型都实现该接口

2、Hadoop的数据类型

IntWritable：整型数

LongWritable：长整型数。

FloatWritable：浮点数。

DoubleWritable：双字节数。

BooleanWritable：布尔型。

ByteWritable：单字节，对应Java中byte类型。

BytesWritable：字节数组，对应Java中byte[]。

Text：使用UTF8格式存储的文本，对String类型的重写。

ObjectWritable：是一种对多类型的封装，可以用于Java的基本类型，String等，如同Java的Object类。

NullWritable：是一个占位符，序列化长度为零，当<key,value>中的key或value为空时使用。

ArrayWritable：针对数组的数据类型。

MapWritable：对java.util.Map的实现。

3、实现步骤

* 实现Mapper

继承org.apache.hadoop.mapreduce.Mapper

* 实现Reducer

继承org.apache.hadoop.mapreduce.Reducer

* 创建并配置Job

实例化org.apache.hadoop.mapreduce.Job

七、配置环境

在resources下添加mapred-site.xml

在pom.xml中添加

<dependency>

<groupId>org.apache.hadoop</groupId>

<artifactId>hadoop-mapreduce-client-core</artifactId>

<version>2.8.1</version>

</dependency>

<dependency>

<groupId>org.apache.hadoop</groupId>

<artifactId>hadoop-mapreduce-client-common</artifactId>

<version>2.8.1</version>

</dependency>

<dependency>

<groupId>org.apache.hadoop</groupId>

<artifactId>hadoop-mapreduce-client-jobclient</artifactId>

<version>2.8.1</version>

</dependency>

<dependency>

<groupId>org.apache.hadoop</groupId>

<artifactId>hadoop-mapreduce-client-shuffle</artifactId>

<version>2.8.1</version>

</dependency>

出现

java.net.ConnectException: Call From phil-PC/192.168.77.1 to master:8020 failed on connection exception: java.net.ConnectException: Connection refused:

在windows上使用ipconfig查看，有两个VMware网卡，从设备管理器上禁用192.168.77.1的网卡

Call From phil-PC/192.168.140.1 to master:8020 failed on connection exception

找到8020的端口的配置文件