8天的课程安排

第一天：Hadoop基础、企业数据部的相关知识、Hadoop集群搭建

第二天：Hadoop分布式文件存储框架HDFS(shell，java)、HDFS原理（权威指南）

第三天：Hadoop分布式计算框架MapReduce，原理，案例

第四天：hive，Flume大数据收集框架、Sqoop数据导入导出

收获：

1. 对Hadoop整个生态环境有足够了解
2. 对Hadoop在企业数据部的位置和作用有足够了解
3. 对数据的一般工作流程职能划分有足够了解
4. Hadoop的集群安装部署、HDFS原理、MapReduce原理
5. Hive的理解，类似sql。

第五天：流式计算（三驾马车：Storm、Kafka、Redis），案例。

第六天：Storm技术增强（原理、调优）、Kafka负载均衡、文件存储原理、Redis回顾

第七天：案例：点击流日志的日志分析系统

第八天：案例：基于系统日志监控告警系统

收获：

1. 流式计算整体架构有足够了解，能够独立的开发Storm程序。
2. 对Storm、Kafka有足够的了解，具备实时计算系统架构的能力。

大数据的实际应用、需要的技能、未来的发展方向。

第一天的大纲

1. 数据部的职能划分、工作流程
2. 数据部的技术架构
3. Hadoop基础和搭建
4. Hadoop HDFS的操作（shell）

第二天的大纲

1. Hadoop Java api的操作（**主要了解元数据信息**）
2. Hadoop HDFS上传下载的详细流程
3. Hadoop secondary NameNode机制
4. 扩展：脚本，定时上传文件。

第三天大纲

1. 回顾并总结HDFS的内容
2. MapReduce的编程模型及原理初探
3. MapReduce的WordCount分析（任务设置、数据流程）
4. MapReduce数据分析案列

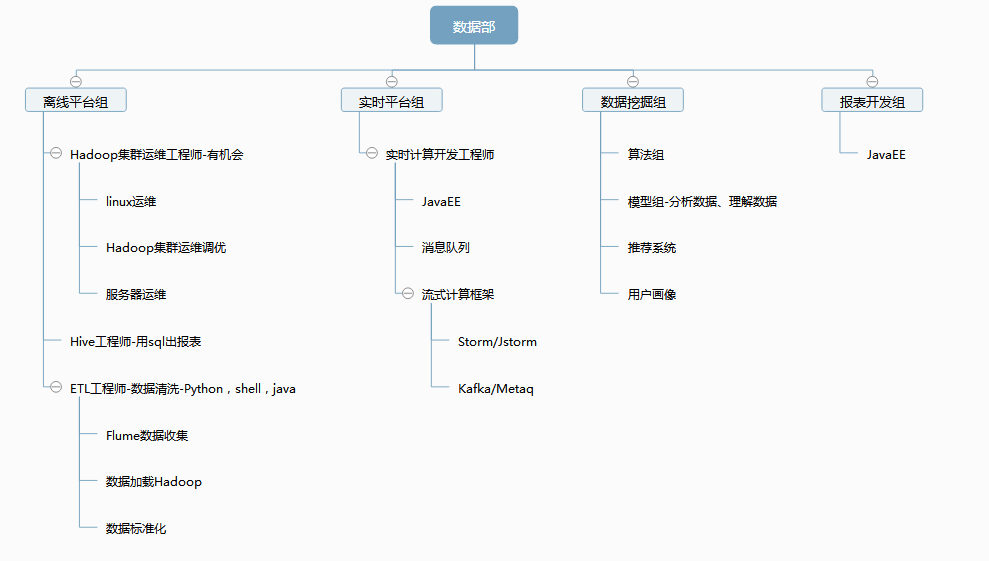
第四天大纲

1. MapReduce数据分析案例之Combiner设置
2. MapReduce数据分析案例之Partitioner设置
3. MapReduce总结
4. Hadoop总结
5. Hive安装及基础

# 背景知识

## 1.1、企业数据部的一般组织结构

企业数据部的一般组织结构，适用于大中型企业。



## 1.2、企业数据部的业务流程分析

业务流程：

电商业务人员：针对活动专题页(活动的效果)有业务需求

活动页的用户访问数、用户下单数、用户支付数、用户退单数

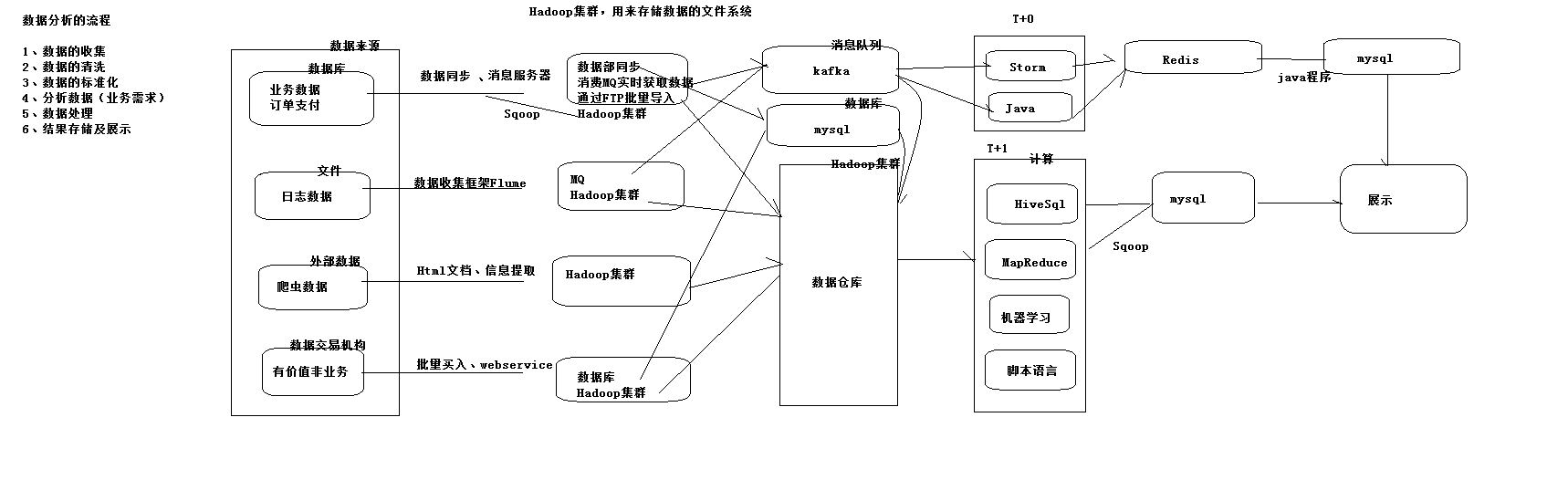
数据部部门：

数据分析人员（写sql），结果对外提供

数据展示：

报表平台组、发邮件excel

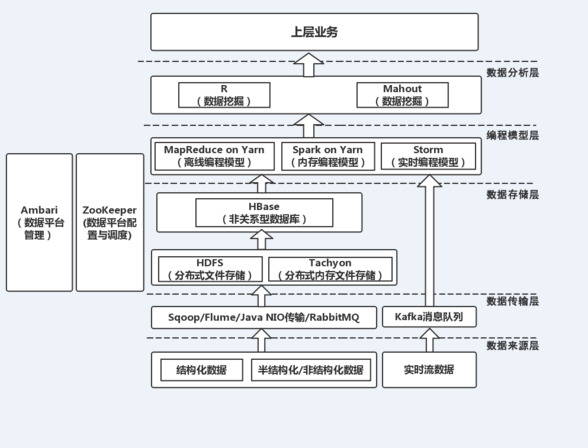
## 1.3、企业数据部的一般技术架构(重要)



数据分析的两个流程：

实时分析流程：业务数据、消息队列、Storm实时编程、Redis、数据展示（秒级计算）

离线分析流程：不同数据源获取数据、Hadoop集群、数据计算(Hive、Spark、MapReduce)、数据展示（T+1计算）



# 2、Hadoop基础知识

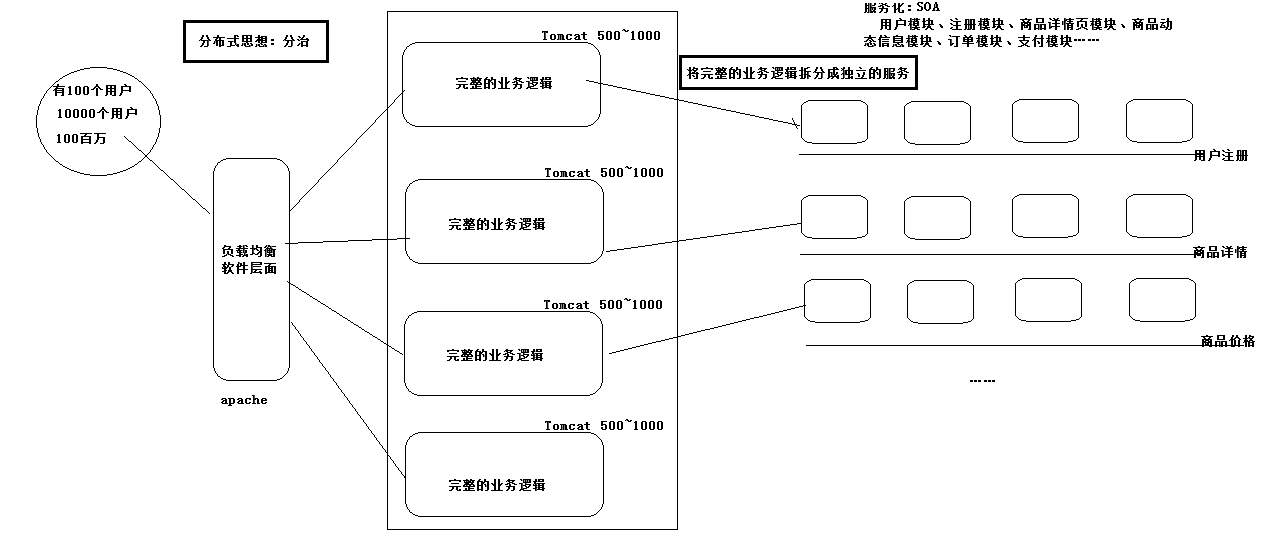
Hadoop包括两个部分：对海量数据进行存储和操作

* 大数据分布式文件存储框架(HDFS)
* 大数据分布式数据计算框架(MapReduce)

独立网站渐变到分布式网站的过程

数据库的读写分离、页面静态化、服务端的缓存、网络端的CDN、网络层面的负载均衡、软件层面的负载均衡。

业务逻辑的服务化及模块的集群服务



大数据的概念

产生数据的设备和系统越来越多

**丰富数据来源及种类**，从单一的业务数据库、企业数据到海量的外部数据

采集数据的手段变多

数据量变多了

数据处理方式和能力变多

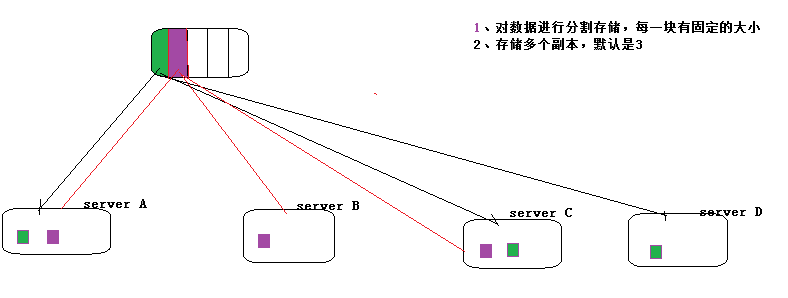
问题：企业里面的数据是否是真的多了？

## 2.1、Hadoop是什么

* Hadoop是一个由Apache基金会所开发的分布式系统基础架构
* Hadoop实现了一个分布式文件系统（Hadoop Distributed File System），简称HDFS。
* Hadoop实现了一个分布式计算系统，MapReduce。
* Hadoop实现了一个资源管理系统，yarn。

## 2.2、Hadoop的特点

* 高可靠性。Hadoop按位存储和处理数据的能力值得人们信赖。
* 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。
* 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。
* 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。
* 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。

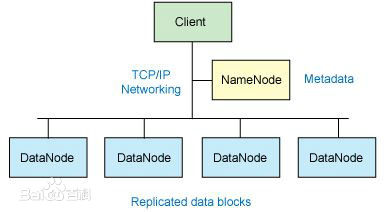


（HDFS的文件存储及可用性计算）

## 2.3、Hadoop HDFS核心组件

分布式的文件存储系统

* HDFS有两个核心角色：NameNode、DataNode



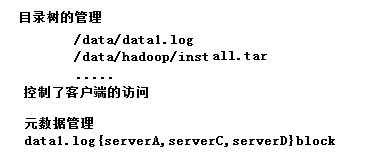
对外部客户机而言，HDFS就像一个传统的分级文件系统。可以创建、删除、移动或重命名文件，等等

这些节点包括 NameNode（仅一个），它在 HDFS 内部提供元数据服务；DataNode它为 HDFS 提供存储块

* NameNode的职责

它负责管理文件系统名称空间和控制外部客户机的访问

负责元数据信息管理

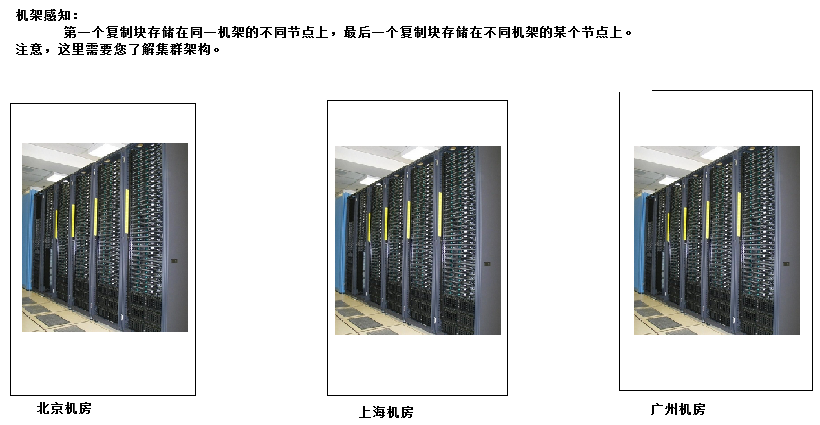


* DataNode的职责

数据存储

定期向NameNode汇报存储的文件信息

* 机架感知



## 2.4、Hadoop MapReduce核心组件

分布式计算框架

### 2.4.1 MapReduce是什么

MapReduce是一种编程模型，**用于大规模数据集(大于1TB)的并行运算**。

它极大地方便了编程人员在不会分布式并行编程的情况下，将自己的程序运行在分布式计算系统上.

**编程模型：**

在一定的前提条件下，依据研究的程序特点，再现程序的结构、功能、属性、关系、过程等本质特征的过程;

软件开发的模型：输入、计算、输出，(JDK FileInputStream process FileOutputStream)

**并行运算：**

并行运算是针对串行运行来讲的，串行运算是依次执行几个事情，事情有前后的时间依赖关系，并行运算是指同时可以运行多个指定，相当于是同时执行多个事情。从而减少处理的总体时长，提高处理问题的效率。

**分布式并行编程：**

多线程、线程安全(同步)、线程池的问题

多个服务器之间通信的问题，RPC(远程过程调用)技术

代码的复杂度很高

### 2.4.2 MapReduce能解决什么问题

* 在Google，MapReduce用在非常广泛的应用程序中，包括"分布grep，分布排序，web连接图反转，每台机器的词矢量，web访问日志分析，反向索引构建，文档聚类，机器学习，基于统计的机器翻译..."值得注意的是，MapReduce实现以后，它被用来重新生成Google的整个索引，并取代老的ad hoc程序去更新索引。
* MapReduce会生成大量的临时文件，为了提高效率，它利用Google文件系统来管理和访问这些文件。
* 在谷歌，超过一万个不同的项目已经采用MapReduce来实现,包括大规模的算法图形处理、文字处理、数据挖掘、机器学习、统计机器翻译以及众多其他领域。
* 其他实现

Nutch项目开发了一个实验性的MapReduce的实现，也即是后来大名鼎鼎的hadoop

Phoenix是斯坦福大学开发的基于多核/多处理器、共享内存的MapReduce实现。

### 2.4.3 Hadoop的作者Doug Cutting

Doug Cutting是Lucene、Nutch 、Hadoop等项目的发起人

## 2.5、大数据技术生态体系

Hadoop（hdfs、mapreduce、yarn） 最基础数据处理技术框架，擅长离线数据分析

Zookeeper 分布式协调服务基础组件

Hbase 分布式海量数据库（基于HDFS），离线分析和在线业务通吃

Hive sql 数据仓库工具，使用方便，功能丰富，基于MR延迟大

Sqoop数据导入导出工具，关系型数据库和Hadoop HDFS的数据桥梁

Flume数据采集框架

~~Storm 实时流式计算框架，流式处理领域头牌框架~~

Spark 基于内存的分布式运算框架，一站式处理 all in one，新秀，发展势头迅猛。**底层支持HDFS.**

## 2.6 Hadoop产生的历史

最早来自于google的三大论文（为什么google会需要这么一种技术）

后来经过doug cutting的山寨，出现了java版本的 hdfs mapreduce 和 hbase

以上三个组件整合起来成为apache的一个顶级项目 hadoop

到了v.0.20.2 雅虎搭建了2000节点的hadoop集群，参加了一次排序性能运算比赛，获得第一名，从此，hadoop声名鹊起，风靡全球

0.20.2 --🡪 1.2.1----------🡪|---🡪2.2.0(HDFS的namenode高可用)--🡪2.4.1 (YARN的高可用)--2.6🡪2.7。2

🡪0.23 ---🡪|

经过演化，hadoop的组件又多出一个yarn（mapreduce+ yarn + hdfs）

## 2.7 Hadoop Yarn和核心组件

ResourceManager 资源调度，管理集群中所有的物理机(内存、计算资源)

NodeManager 当前物理机上的管理者，用来创建具体执行任务进程。

## 2.8 Hadoop所有的核心组件汇总

Hadoop HDFS 分布式文件系统

NameNode 管理元数据(fsimage) 文件属性，节点属性，文件在哪些节点

DataNode 存储数据 文件块，block 128M 心跳（状态，资源信息）

SecondaryNameNode (合并元数据信息，fsimage--->NameNode)

Hadoop MapReduce 分布式计算系统

JobTracker 作业的管理和调度、~~资源分配~~

TaskTracker 作业的具体执行，发送心跳机制JobTracker汇报

MapTask Mapper.map() 执行用户自定的map函数

ReduceTask Reduce.reduce() 执行用户自定义的reduce函数

Hadoop Yarn 分布式资源管理系统

ResourceManager 资源调度，管理集群中所有的物理机(内存、计算资源)

NodeManager 当前物理机上的管理者，用来创建具体执行任务进程。

# 3、Hadoop基础知识

## 3.1、集群搭建的步骤

物理机、机架、联网 虚拟机，虚拟3台

初始化系统环境 linux CentOS 6.5

安装hadoop集群 Hadoop2.6

## 3.2、虚拟机安装

**如何创建一个虚拟机？**

1）服务名称修改

临时修改：hostname hadoop01 重启后失效

永久修改：vi /etc/sysconfig/network HOSTNAME=hadoop01 重启后生效

服务机组+部门+公司的域名

Hdp-node-01.bigdata.itcast.local

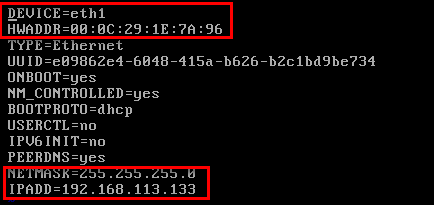
2）网卡配置

因为vm拷贝虚拟机时，会自动生成一个新的网卡，我们需要准备配置文件，并修改。

cd /etc/sysconfig/network-scripts/

cp ifcfg-eth0 ifcfg-eth1

vi ifcfg-eth1 修改device和mac

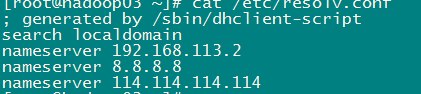


重启网卡

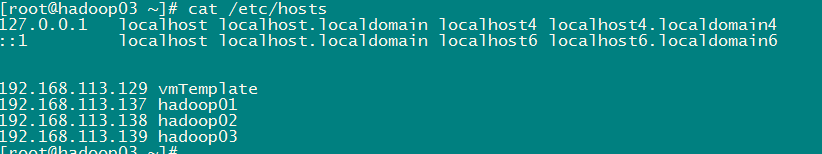
service network restart

DNS配置

cat /etc/resolv.conf



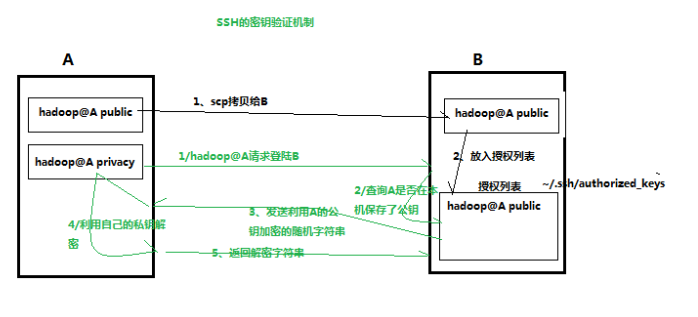
集群hosts ---用来内部通讯的



免密码登陆

ssh-keygen -t rsa

ssh-copy-id hostname



关闭防火墙和文件目录控制权限

临时关闭 service iptables stop && setenforce 0

永久关闭防火墙 chkconfig iptables off

安装环境的准备工作

<http://note.youdao.com/share/?id=649d48ee2d3a9bb3b00ad7fcddc02ac9&type=note>

编译Hadoop源码并部署Hadoop集群

<http://note.youdao.com/share/?id=08c9f62dd83a448ae43cc498814a25e9&type=note>

新装的虚拟机，需要配置网卡

<http://note.youdao.com/share/?id=d930ddc6c3434d123a380e92c8eb3963&type=note>

yum源配置

<http://note.youdao.com/share/?id=a95bef748556308c36c23f67caf7f8ec&type=note>

JDK安装

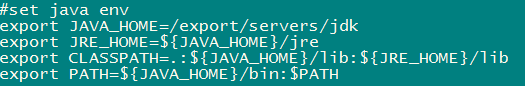
<http://note.youdao.com/share/?id=277cc3b22104ad9bd66d974976fbb427&type=note>

## 3.3、安装Hadoop

说明：在安装Hadoop之前需要安装JDK

JDK安装

<http://note.youdao.com/share/?id=277cc3b22104ad9bd66d974976fbb427&type=note>

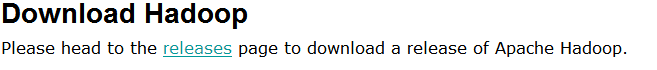


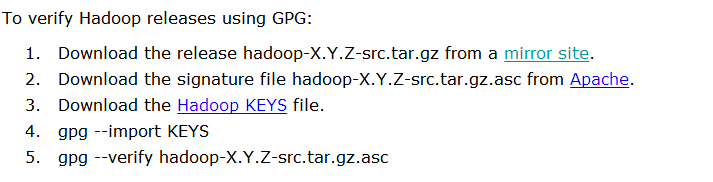
步骤：

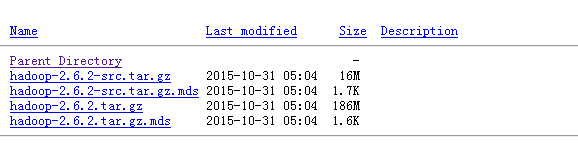
下载安装包、配置文件修改、将安装分发到其它机器、启动集群

### 3.2.1、下载安装包

<http://hadoop.apache.org/#Download+Hadoop>







问题：官网提供的安装包是针对32位操作系统的。 不支持64位。

实际使用中，需要自己编译安装包。

### 3.2.2、编译安装包

下载源码（SVN）、进行编译（maven）

* 下载源代码

svn checkout https://github.com/apache/hadoop.git/tags/release-2.6.1

* 进入项目根目录，进行编译：

mvn package -Pdist,native -DskipTests –Dtar

在hadoop项目的hadoop-dist/target就有了安装包

### 3.2.3、安装Hadoop

上传安装包、修改配置文件、分发安装包、启动集群

1)使用rz命令上传hadoop安装包

2)解压安装包到指定的目录

mkdir -p /export/servers

mkdir -p /export/software

mv hadoop-2.6.1.tar.gz /export/software/

cd /export/software/

tar -zxvf hadoop-2.6.1.tar.gz -C /export/servers/

cd /export/servers/

ln -s hadoop-2.6.1 hadoop

3)配置环境变量

#set java env

export JAVA\_HOME=/export/servers/jdk

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

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

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

#set hadoop env

export HADOOP\_HOME=/export/servers/hadoop

export PATH=${HADOOP\_HOME}/bin:${HADOOP\_HOME}/sbin:$PATH

4)分发程序

依次分发给hadoop02、hadoop03

scp /export/servers/ hadoop-2.6.1 hadoop02:/export/servers/

5)启动集群

Hadoop namenode –format

start-dfs.sh

### 3.2.4、修改配置文件并启动

Hadoop安装笔记

<http://note.youdao.com/share/?id=08c9f62dd83a448ae43cc498814a25e9&type=note>

hadoop的配置文件路径： {$hadooop}/etc/hadoop/

# 4、Hadoop HDFS SHELL

hadoop fs -put hadoop-2.6.1.tar.gz /data 上传文件

hadoop fs -get /data/hadoop-2.6.1.tar.gz . 下传文件

|  |  |  |
| --- | --- | --- |
| 选项名称 | 使用格式 | 含义 |
| -ls | -ls <路径> | 查看指定路径的当前目录结构 |
| -lsr | -lsr <路径> | 递归查看指定路径的目录结构 |
| -du | -du <路径> | 统计目录下个文件大小 |
| -dus | -dus <路径> | 汇总统计目录下文件(夹)大小 |
| -count | -count [-q] <路径> | 统计文件(夹)数量 |
| -mv | -mv <源路径> <目的路径> | 移动 |
| -cp | -cp <源路径> <目的路径> | 复制 |
| -rm | -rm [-skipTrash] <路径> | 删除文件/空白文件夹 |
| -rmr | -rmr [-skipTrash] <路径> | 递归删除 |
| -put | -put <多个linux上的文件> <hdfs路径> | 上传文件 |
| -copyFromLocal | -copyFromLocal <多个linux上的文件> <hdfs路径> | 从本地复制 |
| -moveFromLocal | -moveFromLocal <多个linux上的文件> <hdfs路径> | 从本地移动 |
| -getmerge | -getmerge <源路径> <linux路径> | 合并到本地 |
| -cat | -cat <hdfs路径> | 查看文件内容 |
| -text | -text <hdfs路径> | 查看文件内容 |
| -copyToLocal | -copyToLocal [-ignoreCrc] [-crc] [hdfs源路径] [linux目的路径] | 从本地复制 |
| -moveToLocal | -moveToLocal [-crc] <hdfs源路径> <linux目的路径> | 从本地移动 |
| -mkdir | -mkdir <hdfs路径> | 创建空白文件夹 |
| -setrep | -setrep [-R] [-w] <副本数> <路径> | 修改副本数量 |
| -touchz | -touchz <文件路径> | 创建空白文件 |
| -stat | -stat [format] <路径> | 显示文件统计信息 |
| -tail | -tail [-f] <文件> | 查看文件尾部信息 |
| -chmod | -chmod [-R] <权限模式> [路径] | 修改权限 |
| -chown | -chown [-R] [属主][:[属组]] 路径 | 修改属主 |
| -chgrp | -chgrp [-R] 属组名称 路径 | 修改属组 |
| -help | -help [命令选项] | 帮助 |

# 5、Hadoop Java API

如果非要在window上做客户端应用开发，需要设置以下环境：

* 在windows的某个目录下解压一个hadoop的安装包
* **将安装包下的lib和bin目录用对应windows版本平台编译的本地库替换**
* 在window系统中配置HADOOP\_HOME指向你解压的安装包
* 在windows系统的path变量中加入hadoop的bin目录

Windows开发的配置步骤详解（里面有针对win平台编译好的执行文件和库文件）

http://note.youdao.com/share/?id=a3c8bb7a1aa25bf3ab75f3c65d39f6f6&type=note

做Maven项目的开发，如 hadoop zookeeper ，需要寻找依赖（坐标）

从哪里找坐标：<http://search.maven.org/>

退出安全模式

hadoop dfsadmin -safemode leave

## 5.1、导入maven依赖

**<dependency>  
 <groupId>org.apache.hadoop</groupId>  
 <artifactId>hadoop-client</artifactId>  
 <version>2.6.1</version>  
</dependency>**

## 5.2、Hadoop的配置文件详解

在客户端连接Hadoop集群时，可以指定配置文件，用户根据自己的需求对配置选项修改。Hadoop集群在运行时，会优先使用用户的配置文件。

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

上传文件时两个备份和三备份的实验操作。

## 5.3、Hadoop用户权限的问题

在window下开发Hadoop程序，默认会使用当前电脑的用户账号，两种方式进行修改，一种是修改电脑的账户名称（不建议使用），另一种是通过Hadoop提供的api进行设置，“欺骗”hadoop底层的校验机制。

Configuration conf = **new** Configuration();  
 conf.set(**"dfs.replication"**, **"3"**);  
 **fileSystem** = FileSystem.*get*(**new**URI(**"hdfs://hadoop01:9000"**), conf, **"root"**);

## 5.4、API学习示例

简单api的学习

<http://note.youdao.com/share/?id=a2ccc08c2286ca3fec036aa3439b04c5&type=note>

底层api学习

http://note.youdao.com/share/?id=0cf1ac6c006fe4cb90bdd0559ff335dc&type=note

### 5.4.1、创建HDFS客户端

  HDFS是一个文件系统，如同windows的文件夹一样，与windows不同的是，HDFS支持数据分布式存储。

Configuration conf = **new** Configuration(); conf.set(**"fs.defaultFS"**, **"hdfs://hadoop01:9000"**); conf.set(**"dfs.replication"**, **"3"**);  
 **fileSystem** = FileSystem.*get*(conf);

以上代码在Windows及非hadoop集群用户组的电脑上，会出错。建议采用以下代码进行访问

Configuration conf = **new** Configuration();  
 conf.set(**"dfs.replication"**, **"3"**);  
 **fileSystem** = FileSystem.*get*(**new** URI(**"hdfs://hadoop01:9000"**), conf, **"root"**);

### 5.4.2、在HDFS上创建目录、上传文件、下传文件

**boolean** isDone = **fileSystem**.mkdirs(**new** Path(**"/software/hadoop"**));

Path srcPath = **new** Path(**"E:\\software\\install\\hadoop-2.6.1.tar.gz"**);  
 Path dstPath = **new** Path(**"/software/hadoop/hadoop-2.6.1.tar.gz"**);  
 **fileSystem**.copyFromLocalFile(srcPath, dstPath);

Path hadoopPath = **new** Path(**"/aaa/hadoop-2.6.1.tar.gz"**);  
 Path dstPath = **new** Path(**"G://"**);  
 **fileSystem**.copyToLocalFile(hadoopPath, dstPath);

以上第三部分的代码（下传文件）在windows文件下如果不使用winutils.exe文件可能会报错。建议采用下面的代码进行操作。

Path hadoopPath = **new** Path(**"/aaa/hadoop-2.6.1.tar.gz"**);

Path dstPath = **new** Path(**"G://"**);

**fileSystem**.copyToLocalFile(**false**, hadoopPath, dstPath, **true**);（避免以上问题）

四个参数的含义分别是：是否删除HDFS集群下的原始文件、HDFS文件地址、本地文件地址、是否使用本地文件系统进行操作。

四个参数中，起作用的就是是否使用本地文件系统进行操作这个参数，如果配置了winutils.exe，下传文件就会使用winutils.exe，如果没有配置建议使用该重载方法。

### 5.4.3、在HDFS上移动文件、重命名文件、删除文件

Path srcPath = **new** Path(**"/aaa/hadoop-2.6.1.tar.gz"**);  
 Path disPath = **new** Path(**"/itcast/hadoop/hadoop-2.6.1.tar.gz"**);  
 **fileSystem**.rename(srcPath,disPath);

**fileSystem**.delete(**new** Path(**"/software"**), **true**);

rename方法可以用来移动文件，也可以用来重命名文件。

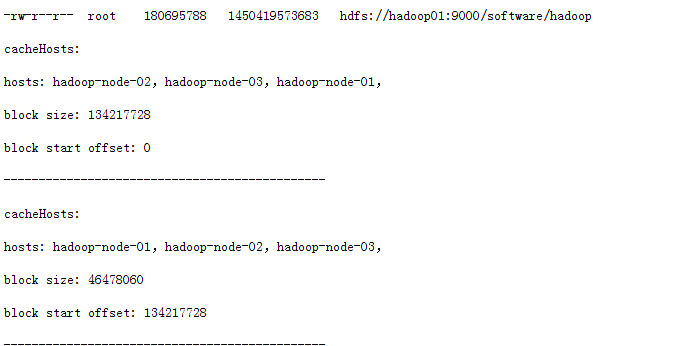
delete方法可以用来递归删除某个目录下的文件信息

### 5.4.4、查看HDFS上的文件信息

RemoteIterator<LocatedFileStatus> fileLists = **fileSystem**.listFiles(**new** Path(**"/"**), **true**);  
 **while** (fileLists.hasNext()) {  
 LocatedFileStatus locatedFileStatus = fileLists.next();  
 *// 按照以下格式打印HDFS上的文件信息  
 // drwxr-xr-x - root supergroup 0 2015-12-18 00:24 /itcast/hadoop* String fileType = **"-"**;  
 System.***out***.print(fileType);*//但因文件的类型* String authority = locatedFileStatus.getPermission().toString();  
 System.***out***.print(authority + **"\t"**);*//打印文件的权限* String user = locatedFileStatus.getOwner();  
 System.***out***.print(user + **"\t"**);*//打印文件所属的的用户* **long** size = locatedFileStatus.getLen();  
 System.***out***.print(size + **"\t"**);*//打印文件的大小* **long** date = locatedFileStatus.getModificationTime();  
 System.***out***.print(date + **"\t"**);*//打印文件的时间戳* String path = locatedFileStatus.getPath().toString();  
 System.***out***.print(path + **"\t"**);*//打印文件的路径* System.***out***.println();  
  
 **for** (BlockLocation blockLocation : locatedFileStatus.getBlockLocations()) {  
 System.***out***.print(**"cacheHosts: "**);  
 **for** (String hosts : blockLocation.getCachedHosts()) {  
 System.***out***.print(hosts + **"，"**);*//打印block的cachehosts* }  
 System.***out***.println();  
  
 System.***out***.print(**"hosts: "**);  
 **for** (String hosts : blockLocation.getHosts()) {  
 System.***out***.print(hosts + **"，"**);*//打印文件block所在的服务器* }  
 System.***out***.println();  
  
 System.***out***.print(**"block size: "**);  
 System.***out***.println(blockLocation.getLength());*//打印block的大小* System.***out***.print(**"block start offset: "**);  
 System.***out***.println(blockLocation.getOffset()); *//文件开始偏移量* System.***out***.println(**"----------------------------------------------"**);  
  
 }  
  
 }

为什么使用**RemoteIterator**<LocatedFileStatus>，这是一个远程的迭代器，由于HDFS服务器数据量比较多，不能一次性返回所有文件信息(可能撑爆内存)，所以一边迭代一边从远程服务器上获取。分治的思想。

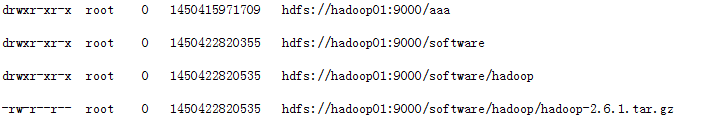
程序运行后，打印效果如下：



### 5.4.5、查看HDFS的所有文件信息(使用递归算法，线上不建议使用)

**public void** fileAll() **throws** Exception {  
 printContent(**new** Path(**"/"**));  
 }  
  
 **public void** printContent(Path path) **throws** Exception {  
 FileStatus[] fileStatuses = **fileSystem**.listStatus(path);  
 **for** (FileStatus fileStatus : fileStatuses) {  
 String fileType = **"d"**;  
 **if** (fileStatus.isFile()) {  
 fileType = **"-"**;  
 } **else if** (fileStatus.isSymlink()) {  
 fileType = **"l"**;  
 }  
 System.***out***.println(fileType + **"\t"** + fileStatus.getPath());  
 **if** (fileStatus.isDirectory()) {  
 printContent(fileStatus.getPath());  
 }  
 }  
 }

运行程序后，打印效果如下：



### 5.4.6、查看Hdfs默认配置文件

**public void** printConfig() {  
 Configuration conf = **new** Configuration();  
  
 Iterator<Map.Entry<String, String>> it = conf.iterator();  
  
 **while** (it.hasNext()) {  
  
 System.***out***.println(it.next());  
  
 }  
}

### 5.4.7、使用文件流的方式获取HDFS上的文件数据

FSDataInputStream fsDataInputStream = *fileSystem*.open(**new**Path(**"hdfs://hadoop01:9000/data/hadoop-2.6.1.tar.gz"**));

FileOutputStream fileOutputStream = **new**FileOutputStream(**new**File(**"G:/hadoop-2.6.1.tar.gz"**));

IOUtils.*copyBytes*(fsDataInputStream,fileOutputStream,4096);

### 5.4.8、使用文件流的方式随机读取一部分的数据（重要）

FSDataInputStream fsDataInputStream = *fileSystem*.open(**new**Path(**"hdfs://hadoop01:9000/data/hadoop-2.6.1.tar.gz"**));

**fsDataInputStream.seek(134217728);**

FileOutputStream fileOutputStream = **new**FileOutputStream(**new**File(**"G:/hadoop-2.6.1.tar.gz.block2"**));

IOUtils.*copyBytes*(fsDataInputStream,fileOutputStream,4096,**true**);

### 5.4.9、使用文件流的方式读取第一个文件块（重要）

FSDataInputStream fsDataInputStream = *fileSystem*.open(**new** Path(**"hdfs://hadoop01:9000/data/hadoop-2.6.1.tar.gz"**));  
 FileOutputStream fileOutputStream = **new** FileOutputStream(**new** File(**"G:/hadoop-2.6.1.tar.gz.block1"**));  
 **long** offset = 0;  
 **byte**[] b = **new byte**[4096];

//读取文件数据流，从offset开始，读到buffer中。 从buffer角标0的位置开始存放，存放4096个。  
 **while** (fsDataInputStream.read(offset, b, 0, 4096) != -1) {  
 **if** (offset >= 134217728) {  
 **return**;  
 }  
 fileOutputStream.write(b);  
 offset += 4096;  
 }  
 fileOutputStream.flush();  
 fsDataInputStream.close();  
 fileOutputStream.close();

### 5.4.10、逐行读取文本数据

FSDataInputStream fsDataInputStream = *fileSystem*.open(**new** Path(**"hdfs://hadoop01:9000/data/install.log"**));  
 BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(fsDataInputStream));  
 String line = **null**;  
 **while** ((line = reader.readLine()) != **null**) { *// 读取文件行内容* System.***out***.println(**"Record: "** + line);  
 }

### 5.4.11、将文件打印到控制台

FSDataInputStream fsDataInputStream = *fileSystem*.open(**new** Path(**"hdfs://hadoop01:9000/data/install.log"**));  
 IOUtils.*copyBytes*(fsDataInputStream, System.***out***, 1024);

### 5.4.12、打印block的元数据信息（重要信息）

DFSInputStream 的 fetchLocatedBlocksAndGetLastBlockLength 方法可以打印出元数据信息

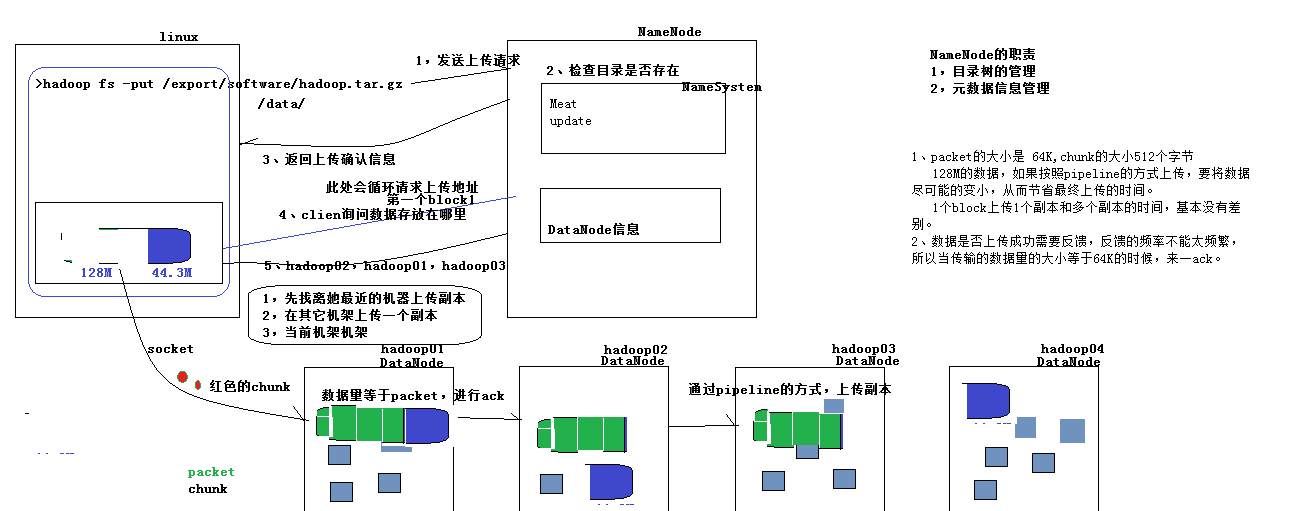
LocatedBlocks{fileLength=180695788 underConstruction=false blocks=[LocatedBlock{BP-1719864721-192.168.113.137-1450976314063:blk\_1073741825\_1001; getBlockSize()=134217728; corrupt=false; offset=0; locs=[192.168.113.138:50010, 192.168.113.137:50010]; storageIDs=[DS-02a4485b-8cfe-4afd-b61d-c9b43b53d28c, DS-aed3b6bb-1652-4c59-b35a-849a1fbc56cc];

storageTypes=[DISK, DISK]}, LocatedBlock{BP-1719864721-192.168.113.137-1450976314063:blk\_1073741826\_1002; getBlockSize()=46478060; corrupt=false; offset=134217728; locs=[192.168.113.137:50010, 192.168.113.139:50010]; storageIDs=[DS-02a4485b-8cfe-4afd-b61d-c9b43b53d28c, DS-aeccfa96-5d25-4458-8b21-99c4c1906780]; storageTypes=[DISK, DISK]}]

  lastLocatedBlock=LocatedBlock{BP-1719864721-192.168.113.137-1450976314063:blk\_1073741826\_1002; getBlockSize()=46478060; corrupt=false; offset=134217728; locs=[192.168.113.139:50010, 192.168.113.137:50010]; storageIDs=[DS-02a4485b-8cfe-4afd-b61d-c9b43b53d28c, DS-aeccfa96-5d25-4458-8b21-99c4c1906780]; storageTypes=[DISK, DISK]}

  isLastBlockComplete=true}

# 6、Hadoop HDFS文件上传



1、向namenode通信请求上传文件，

2、namenode检查目标文件是否已存在，父目录是否存在

3、namenode返回是否可以上传

4、client请求**第一个 block**该传输到哪些datanode服务器上

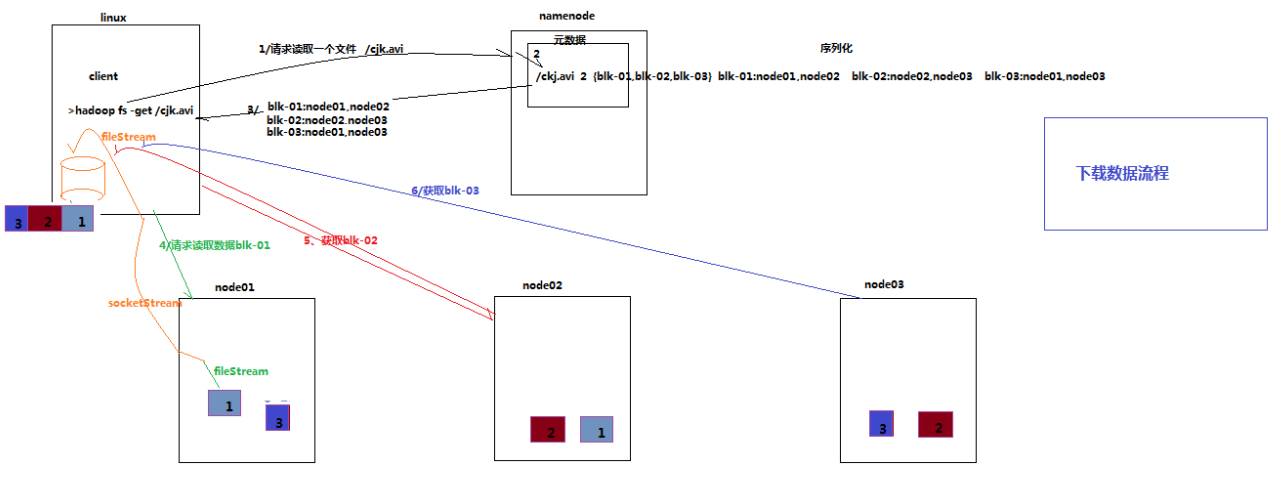
5、namenode返回3个datanode服务器ABC

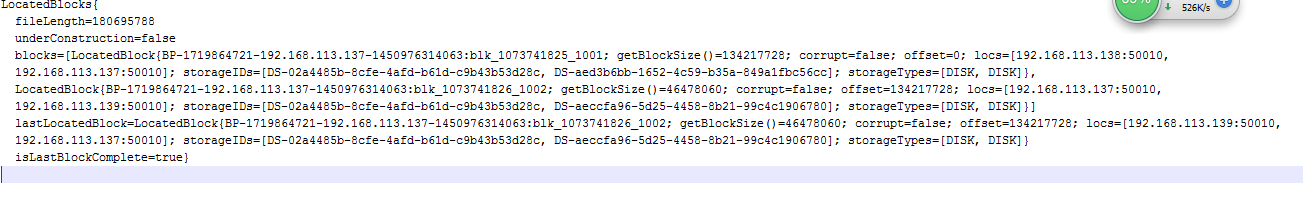
6、client 通过socket请求3台DataNode中的一台A上传数据（本质上是一个RPC调用，建立pipeline），A收到请求会继续调用B，然后B调用C，将真个pipeline建立完成，逐级返回客户端

7、client开始往A上传第一个block（先从磁盘读取数据放到一个本地内存缓存），以packet为单位，A收到一个packet就会传给B，B传给C；A每传一个packet会放入一个应答队列等待应答

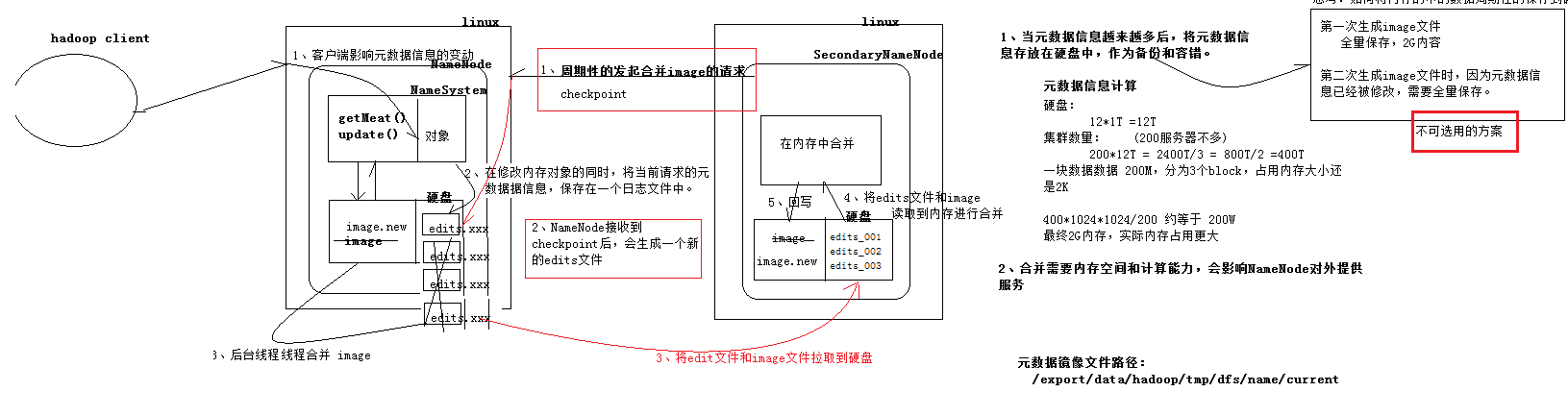
8、当一个block传输完成之后，client再次请求namenode上传第二个block的服务器。循环以上4到7的步骤。

# 7、Hadoop HDFS文件下传



1. 请求下载文件，该文件在namenode中有元数据信息
2. 检查文件是否存在，如果存在返回文件的block元数据信息
3. 
4. client根据block元数据信息向DataNode发送请求
5. 将所有的block合并成一个文件

# 8、Secondary NameNode工作机制



两个问题：**（在数据比较大的情况下）**

1. 数据流量越来越多的情况下，元数据信息也随之变大，如何对元数据信息进行容错和备份。
2. 如果在当前NameNode进行合并文件操作，会影响当前NameNode对外提供服务的能力。

补充说明：

A、内存中有一份完整的元数据

B、磁盘有一个“准完整”的元数据镜像

C、当客户端对hdfs中的文件进行新增或者修改操作，响应的记录首先被记入edits这种log日志中，当客户端操作成功后，相应的元数据会更新到内存中

每隔一段时间，会由secondary namenode将namenode上积累的所有edits和一个最新的fsimage下载到本地，并加载到内存进行merge（这个过程称为checkpoint）

D、checkpoint操作的触发条件配置参数：

|  |
| --- |
| dfs.namenode.checkpoint.check.period=60 #检查触发条件是否满足的频率，60秒  dfs.namenode.checkpoint.dir=file://${hadoop.tmp.dir}/dfs/namesecondary  #以上两个参数做checkpoint操作时，secondary namenode的本地工作目录  dfs.namenode.checkpoint.edits.dir=${dfs.namenode.checkpoint.dir}  dfs.namenode.checkpoint.max-retries=3 #最大重试次数  dfs.namenode.checkpoint.period=3600 #两次checkpoint之间的时间间隔3600秒  dfs.namenode.checkpoint.txns=1000000 #两次checkpoint之间最大的操作记录 |

E、namenode和secondary namenode的工作目录存储结构完全相同，所以，当namenode故障退出需要重新恢复时，可以从secondary namenode的工作目录中将fsimage拷贝到namenode的工作目录，以恢复namenode的元数据

F、可以通过hdfs的一个工具来查看edits和image的信息

bin/hdfs oev -i edits -o edits.xml

hdfs oiv -i fsimage\_0000000000000000087 -p XML -o fsimage.xml

# 9、扩展案例：使用HDFS shell上传文件

## 9.1需求说明

点击流日志每天都10T，需要上传数据仓库（Hadoop HDFS）上

## 9.2需求分析

一般上传文件都是在凌晨24点操作，由于很多种类的业务数据都要在晚上进行传输，为了减轻服务器的压力**，避开高峰期**。需要伪实时的上传，即当文件有10G的时候，就上传一个。

## 9.3技术分析

**HDFS SHELL**: hadoop fs –put xxxx.tar /data 还可以使用 Java Api

满足上传一个文件，不能满足定时、周期性传入。

**定时调度器**：

**Linux crontab**

crontab -e

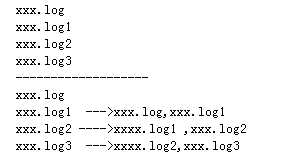
\*/5 \* \* \* \* $home/bin/command.sh //五分钟执行一次

系统会自动执行脚本，每分钟一次，执行时判断文件是否等于10G，如果等于10G就可以上传。

## 9.4实现流程

### 9.4.1日志产生程序

日志产生程序将日志生成后，产生一个一个的文件，使用滚动模式创建文件名。



日志生成的逻辑是 当xxxx.log 等于10G时，滚动生成新日志

log4j.logger.msg=info,msg

log4j.appender.msg=cn.maoxiangyi.MyRollingFileAppender

log4j.appender.msg.layout=org.apache.log4j.PatternLayout

log4j.appender.msg.layout.ConversionPattern=%m%n

log4j.appender.msg.datePattern='.'yyyy-MM-dd

log4j.appender.msg.Threshold=info

log4j.appender.msg.append=true

log4j.appender.msg.encoding=UTF-8

log4j.appender.msg.MaxBackupIndex=100

log4j.appender.msg.MaxFileSize=10GB

log4j.appender.msg.File=/ftphome/appClick/log/data.log

思考：

1. 如果日志文件后缀是1\2\3等数字，该文件满足需求可以上传的话。把该文件移动到准备上传的工作区间。
2. 工作区间有文件之后，可以使用hadoop put命令将文件上传。

阶段问题：

1. 待上传文件的工作区间的文件，在上传完成之后，是否需要删除掉。

### 9.4.2伪代码

使用ls命令读取指定路径下的所有文件信息，

Ls | while read line

//判断line这个文件名称是否包含数组

if Line 包含 数字 (

将文件移动到待上传的工作区间

)

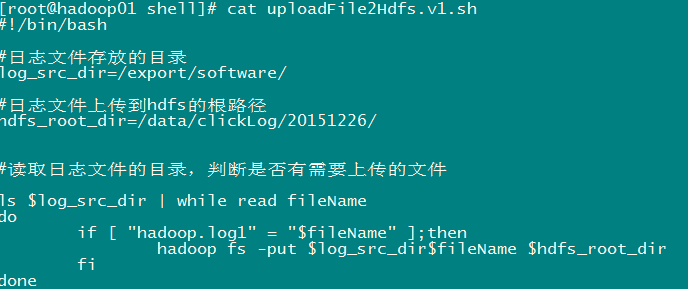
//批量上传工作区间的文件

hadoop fs –put xxx

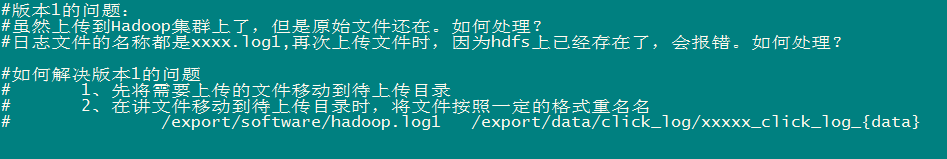
**脚本写完之后**，配置linux定时任务，每5分钟运行一次。

## 9.5代码实现

代码第一版本，实现基本的上传功能和定时调度功能



代码第二版本：增强版V2(基本能用，还是不够健全)





## 9.6效果展示及操作步骤

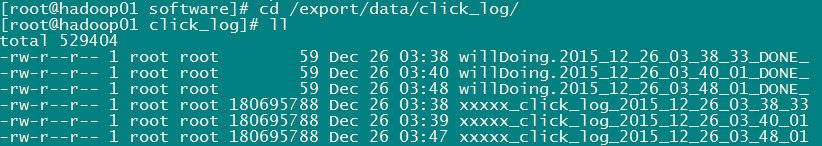
1、日志收集文件收集数据，并将数据保存起来，效果如下：



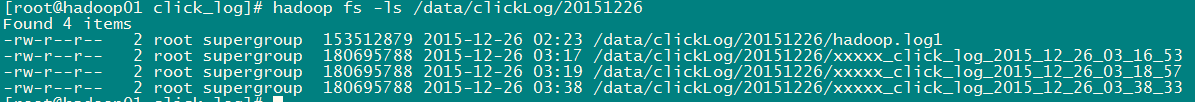
2、上传程序通过crontab定时调度



3、程序运行时产生的临时文件



4、Hadoo hdfs上的效果



# 10、MapReduce 背景知识

## 10.1，什么是单词计数

1、文件word.txt中含以下内容：

my name is lilei

she name is hanmeimei

her name is tom

2、需求：将文档中所有的单词出现的次数统计出来，并保存到文件中

格式：单词+次数，如my 1。

3、需要的最终结果：

my 1

name 3

is 3

lilei 1

she 1

hanmeimei 1

her 1

tom 1

## 10.2、实现单词计数的编程思路（单机版）

1、读取一个文件 FileReader 将文件读取到内存中

2、通过readline方法，得到一行数据

3、通过split(" "),将单词切割出来 String[]{my,name,is,lilei}

4、如果数据量较少的情况下能存能够存放文档的所有信息。

HashMap<String,Integer> String代表单词，Integer代表出现数量

由于Map，能够自动去重，在插入元素之前，判断这个元素是否存在。

如果存在，将Integer读取出来，+1，然后插入。

5、文件读取完成之后，得到一个HashMap的数据结构

HashMap<my,1>

HashMap<name,1>

6、将HashMap中的数据输出到统计的结果文件中，

FileWriter.write(key+" "+Integer)

## 10.3、在大数据环境下可能出现问题

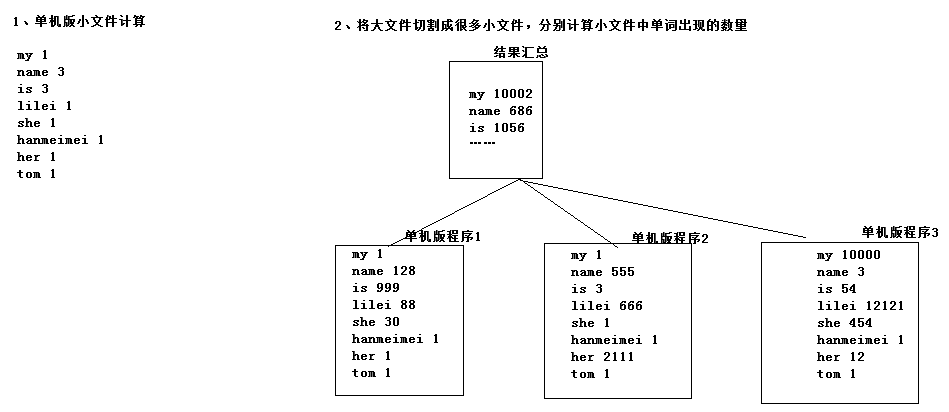
数据量较少的情况单机版可以执行，如果数据量较多的情况下如何处理？

数据量较多的情况，会出现两种问题：**计算慢**、内存不够用。

解决思路：

假设文档有100w行，大小一个1G, 能否将文档分成多个小文件？

如果可以的话，重复步骤2的编程思路，同时启动多个单机版程序，分别将每个小文件中单词出现的数量计算出来。最后交给一个程序进行汇总计算。



## 10.4、以上方案实施过程需要解决的问题

如果按照以上思路编写程序，需要考虑哪些问题？

1、谁来决定一个文件切多少份？

2、谁来决定起多少个单机版本程序？

3、单机版程序计算的结果，谁来负责汇总

4、汇总之后，将结果写回文件，谁来负责

附：

linux split 命令

功能说明：切割文件。

语　　法： split [--help][--version][-<行数>][-b <字节>][-C <字节>][-l <行数>][要切割的文件][输出文件名]

补充说明：split可将文件切成较小的文件，预设每1000行会切成一个小文件。

　　参　　数：

　　 -<行数>或-l<行数> 　指定每多少行就要切成一个小文件。

　　 -b<字节> 　指定每多少字就要切成一个小文件。支持单位:m,k

　　 -C<字节> 　与-b参数类似，但切割时尽量维持每行的完整性。

　　 --help 　显示帮助。

　　 --version 　显示版本信息。

[输出文件名]设置切割后文件的前置文件名，split会自动在前置文件名后再加上编号。

　　使用例子：split -b 100m filename

## 10.5、Hadoop中有针对以上问题的处理组件

JobTracker，任务管理

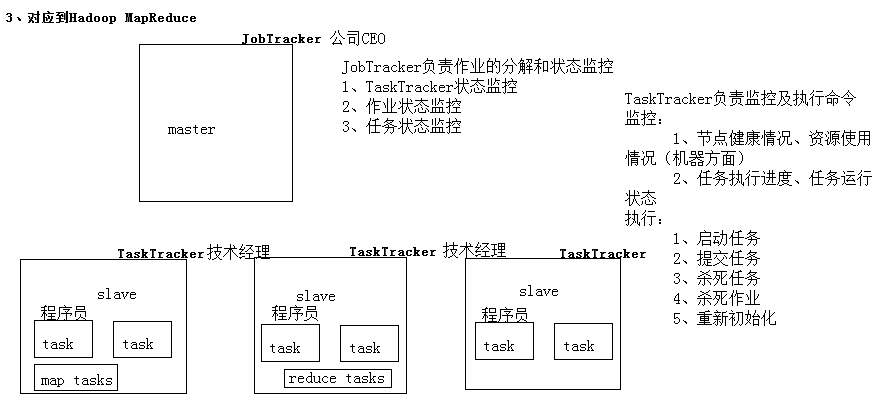
谁来决定一个文件切多少份、

谁来决定起多少个单机版本程序

TaskTracker，任务执行的角色

1、单机版程序计算的结果

2、将结果写回文件



# 11、MapReduce编程模型

## 11.1概述

* Hadoop MapReduce采用Master/Slave结构。
* Master：是整个集群的唯一的全局管理者，功能包括：作业管理、状态监控和任务调度等，即MapReduce中的JobTracker。
* Slave：负责任务的执行和任务状态的回报，即MapReduce中的TaskTracker。

## 11.2 JobTracker

JobTracker是一个后台服务进程，启动之后，会一直监听并接收来自各个TaskTracker发送的心跳信息，**包括资源使用情况和任务运行情况等信息**。

JobTracker的主要功能：

* 作业控制：在hadoop中每个应用程序被表示成一个作业，每个作业又被分成多个任务，JobTracker的作业控制模块则负责作业的分解和状态监控。
* **最重要的是**状态监控：主要包括TaskTracker状态监控、作业状态监控和任务状态监控。主要作用：容错和为任务调度提供决策依据。
* 资源管理。

## 11.3 TaskTracker

TaskTracker是JobTracker和Task之间的桥梁：一方面，从JobTracker接收并执行各种命令：运行任务、提交任务、杀死任务等；另一方面，将本地节点上各个任务的状态通过心跳周期性汇报给JobTracker。TaskTracker与JobTracker和Task之间采用了RPC协议进行通信。

TaskTracker的功能：

* 汇报心跳：Tracker周期性将所有节点上各种信息通过心跳机制汇报给JobTracker。

这些信息包括两部分：

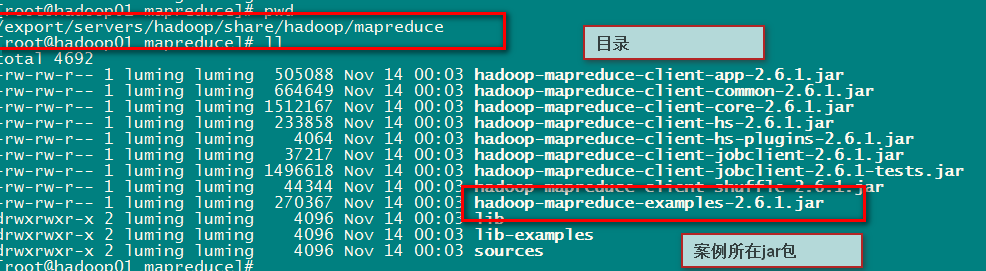
\*机器级别信息：节点健康情况、资源使用情况等。

\*任务级别信息：任务执行进度、任务运行状态等。

* 执行命令：JobTracker会给TaskTracker下达各种命令，主要包括：启动任务(LaunchTaskAction)、提交任务(CommitTaskAction)、杀死任务(KillTaskAction)、杀死作业(KillJobAction)和重新初始化(TaskTrackerReinitAction)。

## 11.4 wordcount任务运行

1**、hadoop集群自带案列**，存放在{安装目录}/share/hadoop/mapreduce/目录下



2、运行任务提交脚本

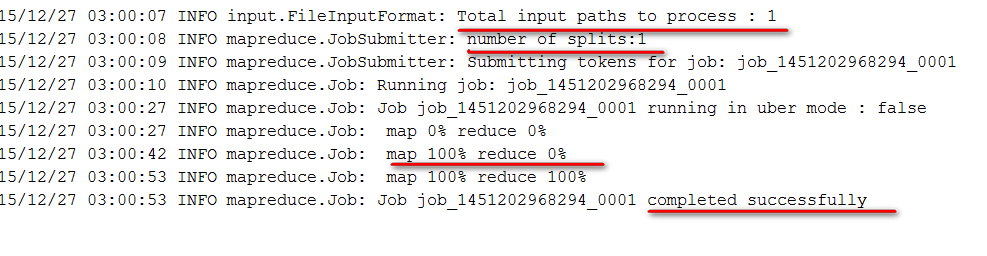
hadoop jar

/export/servers/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.1.jar wordcount /data/word /data/result/wordcountreslut

3、运行结果查看

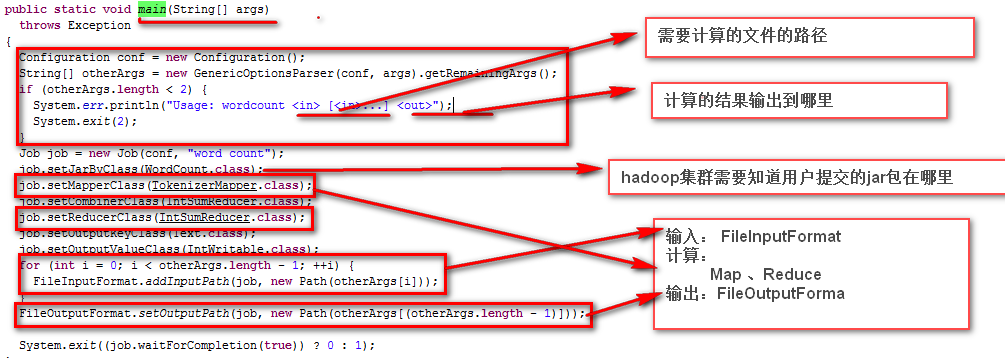
hadoop fs -cat /data/result/wordcountreslut/part-r-00000

4、运行日志分析

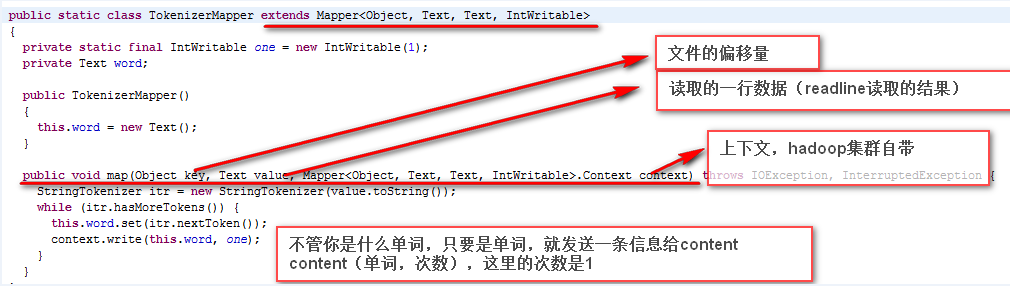


## 11.5 wordcount代码分析

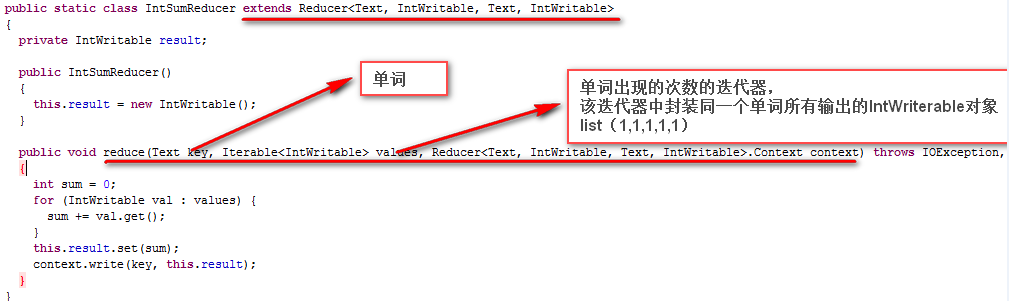
### 11.5.1驱动类说明



### 11.5.2、Map类说明

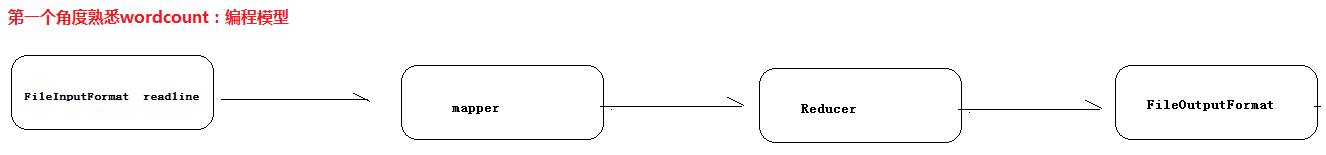


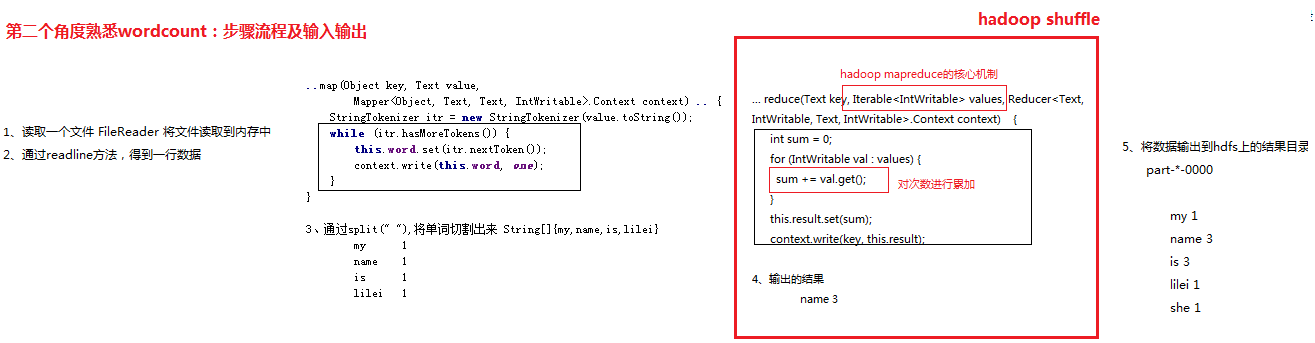
### 11.5.3、Reducer类说明

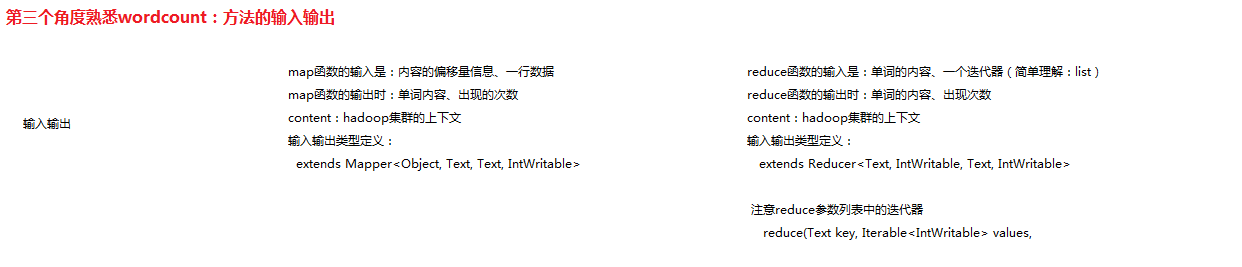


## 11.6、MapReduce编程模型(图解)

三个图一起看



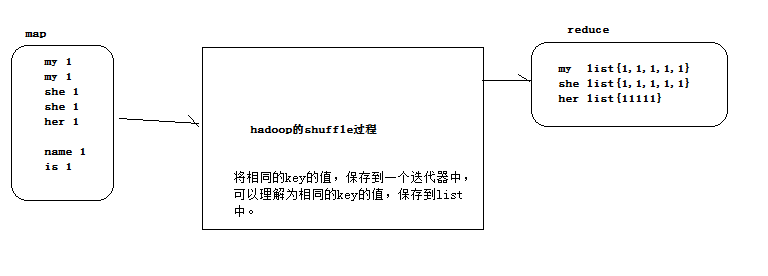




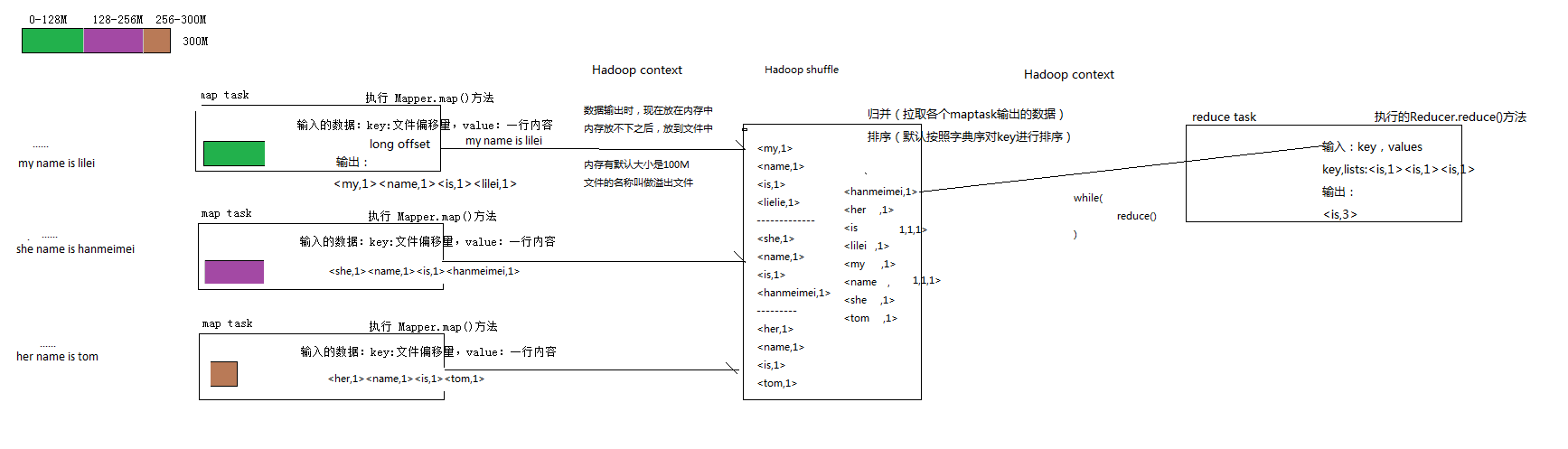
## 11.7、MapReduce编程模型(文字说明)

1. 用户程序会分成三个部分：Mapper，Reducer，Driver(提交mr程序的客户端，runjar)
2. Mapper的输入数据是KV对的形式，KV的类型可以设置
3. Mapper的输出数据是KV对的形式，KV的类型可以设置
4. Mapper中的业务逻辑写在map方法中
5. Mapper类map方法是（maptask进程）每进来一个KV对调用一次
6. Reducer的输入数据应该对应Mapper的输出数据，也是KV
7. Reducer的业务逻辑写在reduce方法中
8. Reducer类reduce方法是(reduceTask进程)对每一组相同key的key-values调用一次
9. 用户的Mapper和Reducer都要继承各自的父类
10. 整个程序需要一个Drvier来进行提交，提交的是一个描述了各种必要信息的job对象

## 11.8、MapReduce shuffle初探

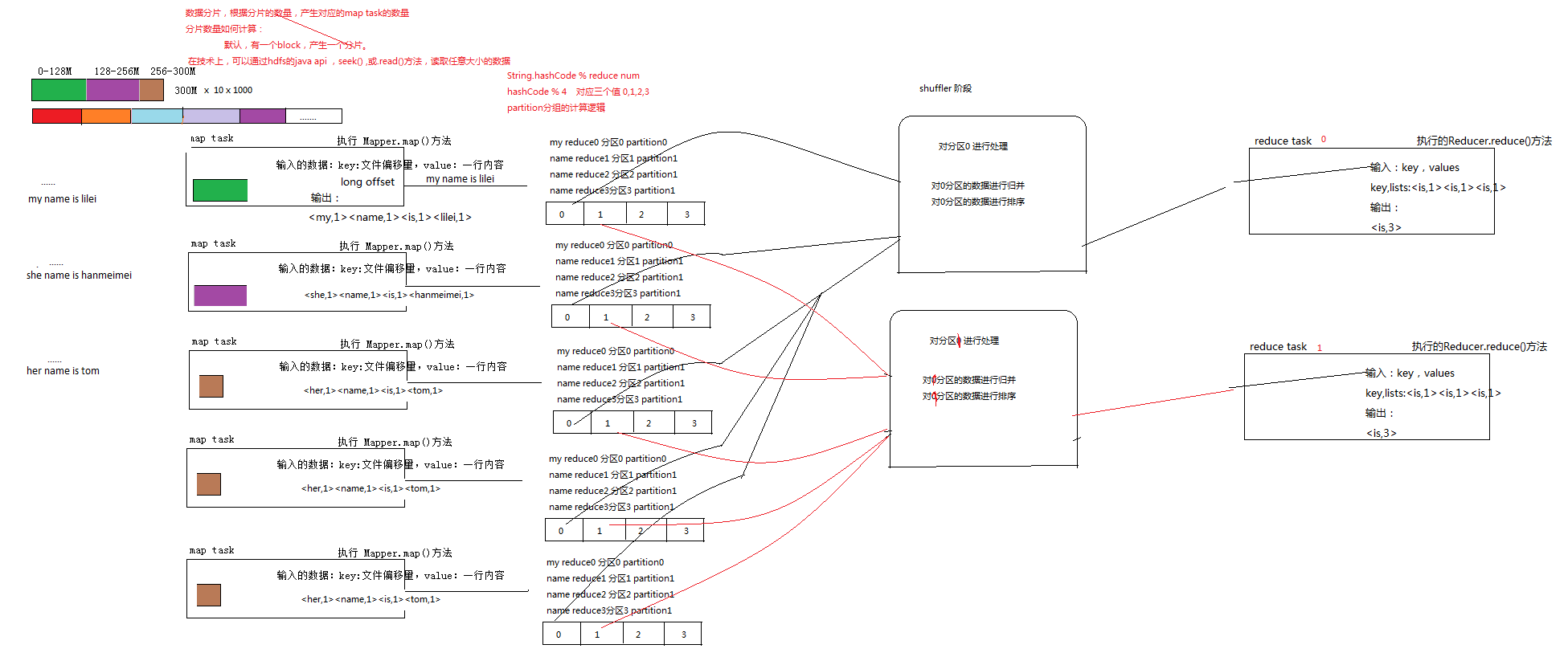


## 11.9、MapReduce shuffle归并和排序



1. map函数执行之后，将数据发送给Hadoop context组件
2. Hadoop context上下文组件会将数据保存到内存中，如果内存不够用，保存成文件。
3. Hadoop shuffle阶段会将，所有的taskmap输出的数据拉取到reduce端，进行排序操作。该操作在reduce函数执行之前。
4. shuffle阶段完成之后，会调用reduce方法，将数据按照（key，values）的方式传入。
5. reduce方法执行完毕之后，将数据传送给FileOutputFormat输出到HDFS

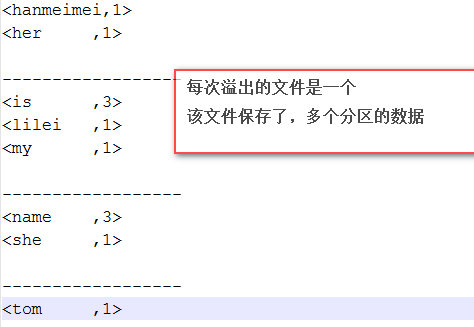
## 11.10、MapReduce partitioner



由于在某些场景下，map阶段输出的数据，并原始数据更多。单个reduce函数执行起来很慢，需要多个reduce函数并发执行。

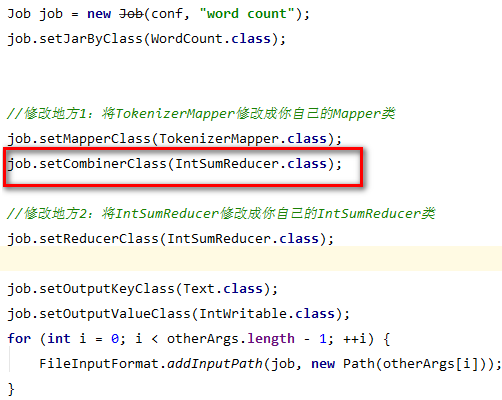
并发执行时，将要相同的key发到统一个reduce上，也就是对数据进行分组。

分组的逻辑是什么：使用key的hashcode % reduce个数，从map阶段开始就将数据放到分区文件中。该分区文件，在hadoop集群中叫做溢出文件，多个分区的数据都会保存在该文件中。



## 11.11、MapReduce combiner

Combiner 就是在maptask端执行的reduce任务，叫做本地reduce。



1. combiner是MR程序中Mapper和Reducer之外的一种组件
2. combiner组件的父类就是Reducer
3. combiner和reducer的区别在于运行的位置：

Combiner是在每一个maptask所在的节点运行

Reducer是接收全局所有Mapper的输出结果；

(4) combiner的意义就是对每一个maptask的输出进行局部汇总，以减小网络传输量

具体实现步骤：

1. 自定义一个combiner继承Reducer，重写reduce方法
2. 在job中设置： job.setCombinerClass(CustomCombiner.class)

(5) combiner能够应用的前提是不能影响最终的业务逻辑

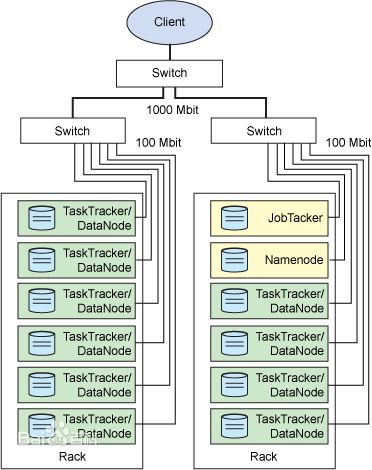
而且，combiner的输出kv应该跟reducer的输入kv类型要对应起来

## 11.12、MapReduce taskmap并发设置

分片数，默认等于hdfs的block的数量，hadoop框架启动多少个maptask是根据分片数量来决定的。

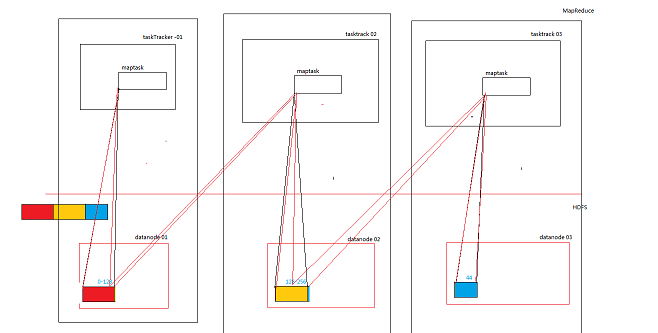
分片数，是一个逻辑上的概念，block是物理上的实实在在的数据。

一般情况下，maptask和datanode运行在一个物理节点，如下图。

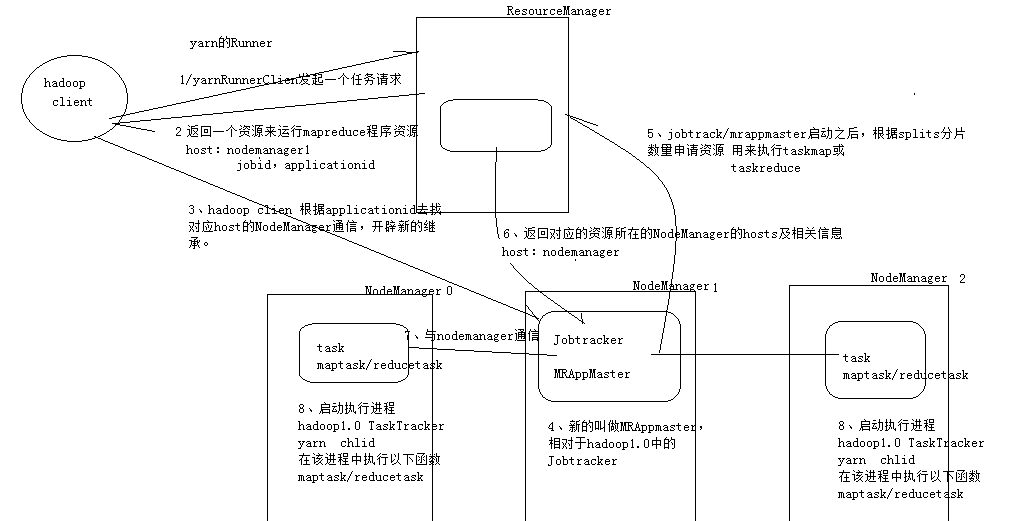


为什么要运行在一起？

避免数据的网络传输



## 11.13、MapReduce 运行在yarn上的流程



# 12、Hive快速入门

课程安排：

1、Hive的基本介绍

2、Hive的安装部署

3、Hive的使用操作

达到的目标：

1、了解企业中作数据分析的主要工具(Hive)

2、了解Hive基本操作

## 12.1 Hive什么

* 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，
* hive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。

Hive 将用户的HiveQL 语句通过解释器转换为MapReduce 作业提交到Hadoop 集群上，Hadoop 监控作业执行过程，然后返回作业执行结果给用户。

* hive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 并不提供实时的查询和基于行级的数据更新操，Hive 构建在基于静态批处理的Hadoop 上，适合离线计算。
* Hive 的最佳使用场合是大数据集的批处理作业，例如，网络日志分析

## 12.2 Hive安装的三种模式

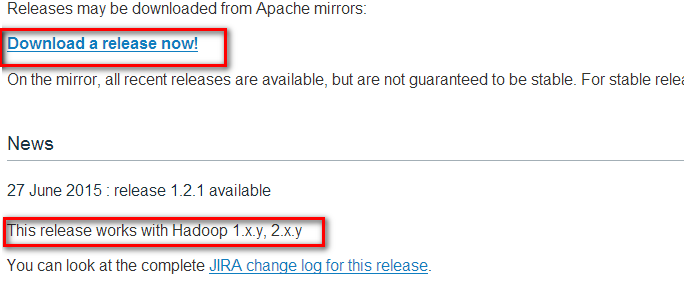
* 内嵌模式：元数据保持在内嵌的derby模式，只允许一个会话连接
* 本地独立模式：在本地安装Mysql，把元数据放到mySql内
* 远程模式：元数据放置在远程的Mysql数据库(企业中常用的)

## 12.3 安装Hive工具

下载安装包、修改配置文件、安装mysql、分发配置、启动测试

1、下载Hive安装包

<http://hive.apache.org/downloads.html>



2、将hive文件上传到HADOOP集群，并解压

    将文件上传到：/export/software

tar -zxvf apache-hive-1.2.1-bin.tar.gz -C /export/servers/

cd /export/servers/

ln -s apache-hive-1.2.1-bin hive

3、配置环境变量，编辑/etc/profile

#set hive env

export HIVE\_HOME=/export/servers/hive

export PATH=${HIVE\_HOME}/bin:$PATH

#让环境变量生效

source /etc/profile

4、修改hive配置文件

* 进入配置文件的目录

cd /export/servers/hive/conf/

* 修改hive-env.sh文件

cp hive-env.sh.template hive-env.sh

将以下内容写入到hive-env.sh文件中

export JAVA\_HOME=/export/servers/jdk

export HADOOP\_HOME=/export/servers/hadoop

export HIVE\_HOME=/export/servers/hive

* 修改log4j文件

cp hive-log4j.properties.template hive-log4j.properties

将EventCounter修改成org.apache.hadoop.log.metrics.EventCounter

#log4j.appender.EventCounter=org.apache.hadoop.hive.shims.HiveEventCounter

log4j.appender.EventCounter=org.apache.hadoop.log.metrics.EventCounter

* 配置远程登录模式

touch hive-site.xml

####### hive-site.xml和hive-default.xml

将以下信息写入到hive-site.xml文件中

<configuration>

        <property>

                <name>javax.jdo.option.ConnectionURL</name>

                <value**>jdbc:mysql://hadoop02:3306**/hivedb?createDatabaseIfNotExist=true</value>

        </property>

        <property>

                <name>javax.jdo.option.ConnectionDriverName</name>

                <value>com.mysql.jdbc.Driver</value>

        </property>

        <property>

                <name>javax.jdo.option.ConnectionUserName</name>

                <value>root</value>

        </property>

        <property>

                <name>javax.jdo.option.ConnectionPassword</name>

                <value>root</value>

        </property>

</configuration>

**需要额外：拷贝一个jdbc的驱动包到 hive的lib目录下。**

5、安装mysql并配置hive数据库及权限

* 安装mysql数据库及客户端

yum install mysql-server

yum install mysql

**service mysqld start**

* 配置hive元数据库

mysql -u root -p

create database hivedb;

* 对hive元数据库进行赋权，开放远程连接，开放localhost连接

grant all privileges on \*.\* to root@"%" identified by "root" with grant option;

grant all privileges on \*.\* to root@"localhost" identified by "root" with grant option;

6、运行hive命令即可启动hive

hive

jar冲突的错误：

附录1：如果报错Terminal initialization failed; falling back to unsupported

将/export/servers/hive/lib 里面的jline2.12替换了hadoop 中/export/servers/hadoop/hadoop-2.6.1/share/hadoop/yarn/lib/jline-0.09\*.jar

# 13、离线课程总计

