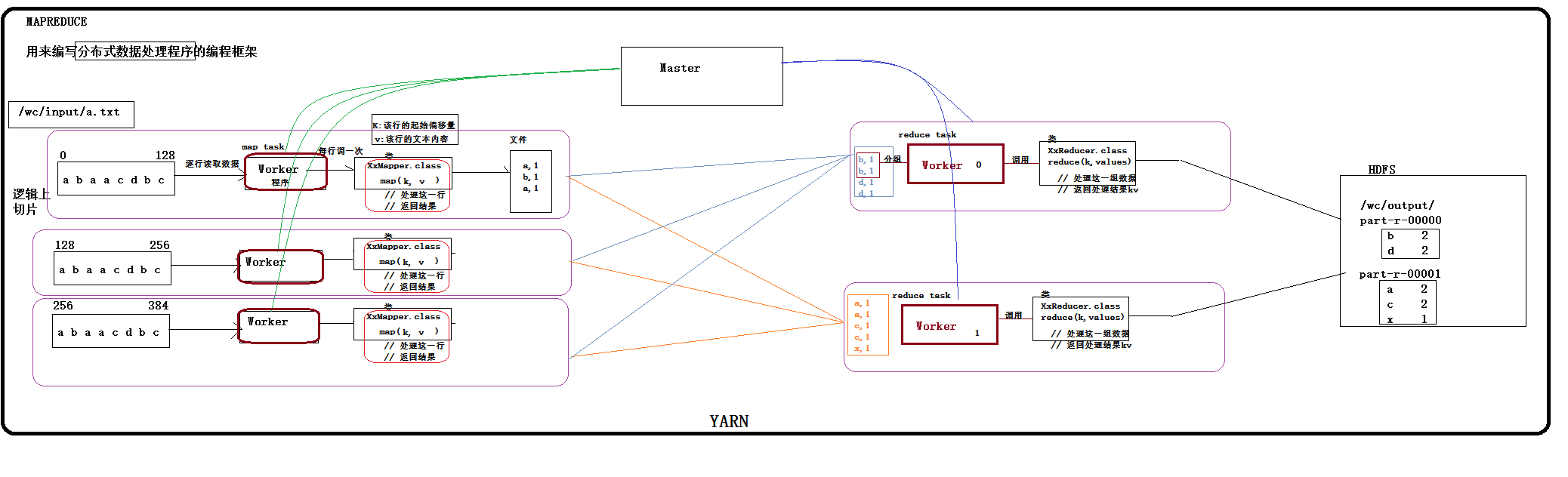
# 通过案例学习MAPREDUCE教程

# wordcount示例开发

## wordcount程序整体运行流程示意图



map阶段： 将每一行文本数据变成<单词,1>这样的kv数据

reduce阶段：将相同单词的一组kv数据进行聚合：累加所有的v

注意点：mapreduce程序中，

map阶段的进、出数据，

reduce阶段的进、出数据，

类型都应该是实现了HADOOP序列化框架的类型，如：

String对应Text

Integer对应IntWritable

Long对应LongWritable

## 编码实现

WordcountMapper类开发

WordcountReducer类开发

JobSubmitter客户端类开发

《详见代码》

## 运行mr程序

1. 将工程整体打成一个jar包并上传到linux机器上，
2. 准备好要处理的数据文件放到hdfs的指定目录中
3. 用命令启动jar包中的Jobsubmitter，让它去提交jar包给yarn来运行其中的mapreduce程序 ： hadoop jar wc.jar cn.edu360.mr.wordcount.JobSubmitter .....
4. 去hdfs的输出目录中查看结果

# yarn快速理解

## yarn的基本概念

yarn是一个分布式程序的**运行调度平台**

yarn中有**两大核心角色**：

1. **Resource Manager**

接受用户提交的分布式计算程序，并为其划分资源

管理、监控各个Node Manager上的资源情况，以便于均衡负载

1. **Node Manager**

管理它所在机器的运算资源（cpu + 内存）

负责接受Resource Manager分配的任务，创建容器、回收资源

## YARN的安装

node manager在物理上应该跟data node部署在一起

resource manager在物理上应该独立部署在一台专门的机器上

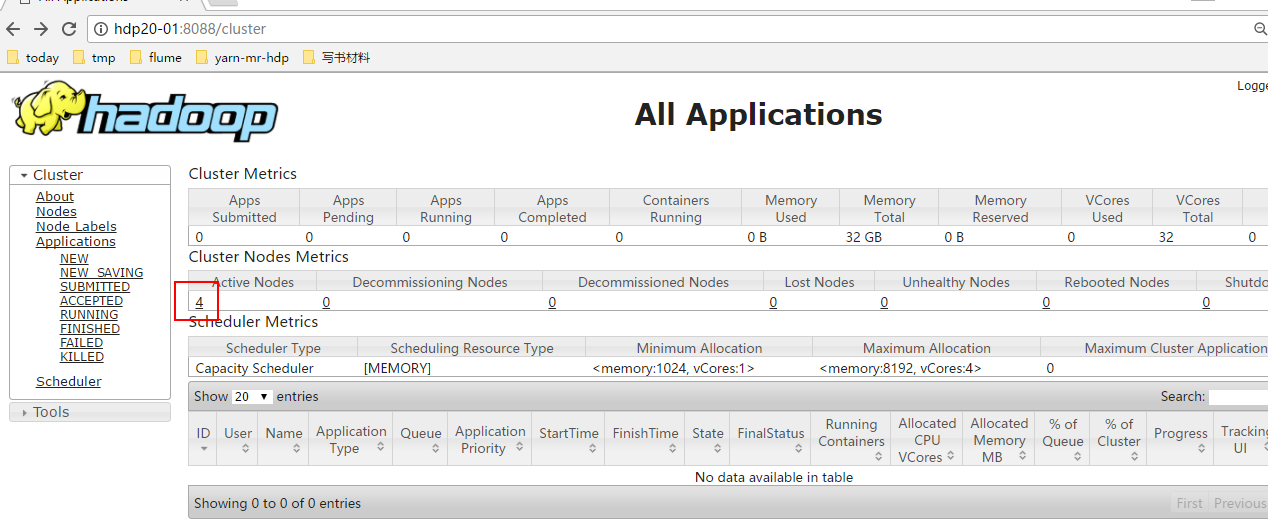
1、修改配置文件：

vi yarn-site.xml

|  |
| --- |
| <configuration>  <property>  <name>yarn.resourcemanager.hostname</name>  <value>hdp20-01</value>  </property>  <property>  <name>yarn.nodemanager.aux-services</name>  <value>mapreduce\_shuffle</value>  </property> |

1. scp这个yarn-site.xml到其他节点
2. 启动yarn集群：start-yarn.sh （注：该命令应该在resourcemanager所在的机器上执行）
3. 用jps检查yarn的进程，用web浏览器查看yarn的web控制台

http://hdp20-01:8088



# mr编程案例（一）

## mr编程案例1——求TOPN

1. 读取附件中的文件request.dat，

**需求1：**求出每一个url被访问的总次数，并将结果输出到一个结果文件中

思路：就是一个wordcount

map阶段： 解析数据，将url作为key，1作为value发出

reduce阶段：将一组数据的value累加

**需求2：**求出每个网站被访问次数最多的top3个url《分组TOPN》

思路：

map阶段——切字段，抽取域名作为key，url作为value，返回即可

reduce阶段——用迭代器，将一个域名的一组url迭代出来，挨个放入一个hashmap中进行计数，最后从这个hashmap中挑出次数最多的3个url作为结果返回

**需求3：**求访问次数最多的topn个网站（只能有1个reduce worker）《全局TOPN》

思路：

map阶段：解析数据，将域名作为key，1作为value

reduce阶段：

reduce方法中——对一个域名的一组1累加，然后将 <域名,总次数>放入一个成员变量Treemap中

cleanup方法中——从treemap中挑出次数最高的n个域名作为结果输出

要点1：每一个reduce worker程序，会在处理完自己的所有数据后，调用一次cleanup方法

要点2：如何向map和reduce传自定义参数

从JobSubmitter的main方法中，可以向map worker和reduce worker传递自定义参数（通过configuration对象来写入自定义参数）；然后，我们的map方法和reduce方法中，可以通过context.getConfiguration()来取自定义参数

## mr编程案例2——自定义类型

需求：统计一下文件中，每一个用户所耗费的总上行流量，总下行流量，总流量

|  |
| --- |
| 1363157982040 13502468823 5C-0A-5B-6A-0B-D4:CMCC-EASY 120.196.100.99 y0.ifengimg.com 综合门户 57 102 7335 110349 200  1363157986072 18320173382 84-25-DB-4F-10-1A:CMCC-EASY 120.196.100.99 input.shouji.sogou.com 搜索引擎 21 18 9531 2412 200  1363157990043 13925057413 00-1F-64-E1-E6-9A:CMCC 120.196.100.55 t3.baidu.com 搜索引擎 69 63 11058 48243 200 |

思路：

map阶段：将每一行按tab切分成各字段，提取其中的手机号作为输出key，流量信息封装到FlowBean对象中，作为输出的value

**要点：自定义类型如何实现hadoop的序列化接口**

*FlowBean这种自定义数据类型必须实现hadoop的序列化接口：Writable，实现其中的两个方法：*

*readFields(in) 反序列化方法*

*write(out) 序列化方法*

reduce阶段：遍历一组数据的所有value（flowbean），进行累加，然后以手机号作为key输出，以总流量信息bean作为value输出

## mr编程案例3——自定义Partitioner

统计每一个用户的总流量信息，并且按照其归属地，将统计结果输出在不同的文件中

1、思路：

想办法让map端worker在将数据分区时，按照我们需要的按归属地划分

实现方式：自定义一个Partitioner

1. 实现

先写一个自定义Paritioner

（代码见工程）

## mr编程案例4——全局排序

需求：统计request.dat中每个页面被访问的总次数，同时，要求输出结果文件中的数据按照次数大小倒序排序

关键技术点：

mapreduce程序内置了一个排序机制：

map worker 和reduce worker ，都会对数据按照key的大小来排序

所以最终的输出结果中，一定是按照key有顺序的结果

**思路：**

本案例中，就可以利用这个机制来实现需求：

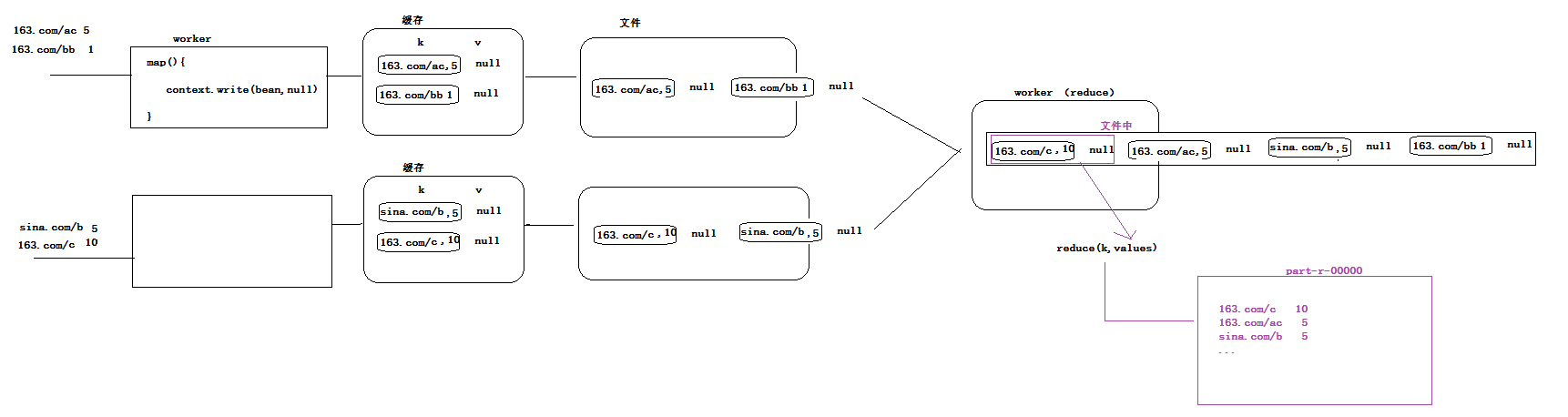
1. 先写一个mr程序，将每个页面的访问总次数统计出来
2. 再写第二个mr程序：

map阶段： 读取第一个mr产生的结果文件，将每一条数据解析成一个java对象UrlCountBean（封装着一个url和它的总次数），然后将这个对象作为key，null作为value返回

要点：这个java对象要实现WritableComparable接口，以让worker可以调用对象的compareTo方法来进行排序

reduce阶段：由于worker已经对收到的数据按照UrlCountBean的compareTo方法排了序，所以，在reduce方法中，只要将数据输出即可，最后的结果自然是按总次数大小的有序结果

看示意图：



## mr编程案例5——倒排索引创建

需求：有如下数据：

1. txt

|  |
| --- |
| hello tom  hello jim  hello kitty  hello rose |

1. txt

|  |
| --- |
| hello jerry  hello jim  hello kitty  hello jack |

1. txt

|  |
| --- |
| hello jerry  hello java  hello c++  hello c++ |

思路：

1、先写一个mr程序：统计出每个单词在每个文件中的总次数

hello-a.txt 4

hello-b.txt 4

hello-c.txt 4

java-c.txt 1

jerry-b.txt 1

jerry-c.txt 1

要点1：map方法中，如何获取所处理的这一行数据所在的文件名？

worker在调map方法时，会传入一个context，而context中包含了这个worker所读取的数据切片信息，而切片信息又包含这个切片所在的文件信息

那么，就可以在map中：

FileSplit split = context.getInputSplit();

String fileName = split.getpath().getName();

要点2：setup方法   
worker在正式处理数据之前，会先调用一次setup方法，所以，常利用这个机制来做一些初始化操作；

2、然后在写一个mr程序，读取上述结果数据：

map： 根据-切，以单词做key，后面一段作为value

reduce： 拼接values里面的每一段，以单词做key，拼接结果做value，输出即可

## mr编程案例6——自定义GroupingComparator

需求：一下数据，表示线段的左端点和右端点

|  |
| --- |
| 1,4  2,5  3,4  2,6  4,7  5,8  5,9  6,10  10,15  11,16  12,18  13,17 |

需求1：求所有交错的层数

map：将一条线段的范围内坐标点作为key，1作为value

reduce：累加1，就得到了坐标点上的重叠次数

需求2：求出重叠次数最高的坐标点及其重叠次数（全局TOPN）

map：读数据，将<标点,重叠次数>封装为一个PointTimes对象，作为key返回，null为value

*注意：PointTimes类要实现WritableComparable接口，以规定如何比较两个PointTimes对象的大小*

reduce端：

注意：要想办法让worker将所有的*PointTimes对象看成相同，以让worker把所有的数据看成一组，来调****一次****reduce方法。*

*实现方式就是自定义一个GroupingComparator(//具体见代码)*

然后，reduce方法中，输出这一组（其实是这个worker所收到的全部数据）的前n个key即可

## mr编程案例7——控制输入、输出格式

需求：还是对案例3中的流量数据进行汇总，然后求出汇总结果中的流量最大的TOPN条

步骤1:——

思路：统计逻辑跟之前的流量统计一致：

map：以手机号作为key，flowbean作为value

注：步骤1输出的结果文件通过指定SequenceFileOutputFormat来产生SequenceFile文件；SequenceFile文件是hadoop定义的一种文件，里面存放的是大量key-value的对象序列化字节（文件头部还存放了key和value所属的类型名）；

步骤2：

思路：读取步骤1的SequenceFile结果文件，需要指定inputformatclass为SequenceFileInputFormat组件

既然使用了这种输入组件，那么我们的map方法中直接就接收一对KEY-VALUE数据

如何实现topn呢？

通过把所有汇总数据发给同一个reduce端的worker，并且通过定义一个GroupingComparator来让这个worker把所有手机号的flowbean对象看成一组，调用一次reduce方法，我们就只需要在reduce方法中输出前n个即可

# mapreduce框架的运作机制

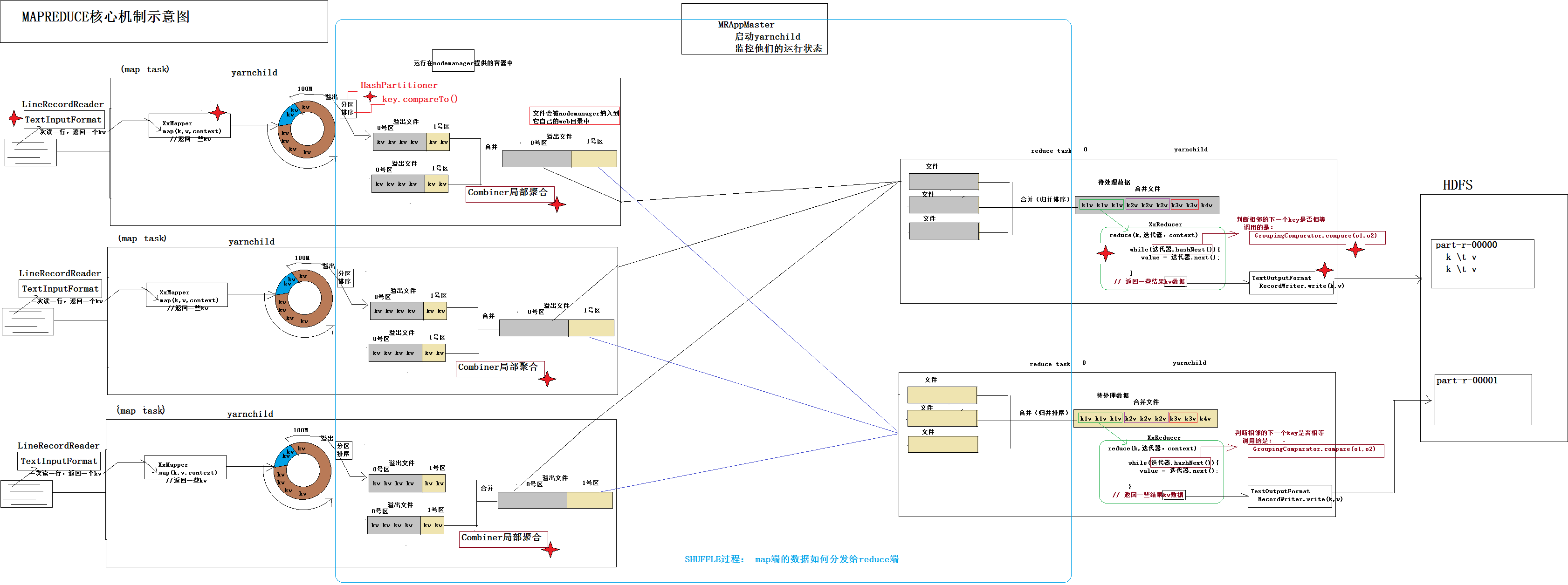
## 核心角色：

MRAppmaster

Worker(map task)

Worker(reduce task)

## 运行机制全图：



# mr编程案例（二）

## mr编程案例8——join算法

### 数据：

select a.\*,b.\* from a join b on a.uid=b.uid;

有订单数据：

|  |
| --- |
| order001,u001  order002,u001  order003,u005  order004,u002  order005,u003  order006,u004 |

有用户数据：

|  |
| --- |
| u001,senge,18,angelababy  u002,laozhao,48,ruhua  u003,xiaoxu,16,chunge  u004,laoyang,28,zengge  u005,nana,14,huangbo |

### 需求：

### 思路：

map端：

不管worker读到的是什么文件，我们的map方法中是可以通过context来区分的

对于order数据，map中切字段，封装为一个joinbean，打标记：t\_order

对于user数据，map中切字段，封装为一个joinbean，打标记：t\_user

然后，以uid作为key，以joinbean作为value返回

reduce端：

用迭代器迭代出一组相同uid的所有数据joinbean，然后判断

如果是标记字段为t\_order的，则加入一个arraylist<JoinBean>中

如果标记字段为t\_user的，则放入一个Joinbean对象中

然后，遍历arraylist，对里面的每一个JoinBean填充userBean中的user数据，然后输出这个joinBean即可

## mr编程案例9——数据倾斜场景

### 数据：

|  |
| --- |
| a a a a a a b b b a a a  a a a a c c b c a a a c  a b b c a a d d e e f f  f g a a a b a b h h g j |

### 需求：

需要做wordcount

但是，会有一个问题存在：

a特别多，

负责处理a这个单词数据的reduce worker就会很累（负载不均衡，过大）

思考：如何处理？会让整个数据处理过程中，数据倾斜的状况得到缓解

### 思路：

将key进行打散（具体方案：给key拼接一个随机字符串），以均衡reduce端worker的负载；

## mr编程案例10——json数据解析

### 数据：

|  |
| --- |
| {"movie":"1193","rate":"5","timeStamp":"978300760","uid":"1"}  {"movie":"661","rate":"3","timeStamp":"978302109","uid":"1"}  {"movie":"914","rate":"3","timeStamp":"978301968","uid":"1"}  {"movie":"1357","rate":"5","timeStamp":"978298709","uid":"2"}  {"movie":"3068","rate":"4","timeStamp":"978299000","uid":"2"}  {"movie":"1537","rate":"4","timeStamp":"978299620","uid":"2"}  {"movie":"647","rate":"3","timeStamp":"978299351","uid":"3"}  {"movie":"2194","rate":"4","timeStamp":"978299297","uid":"3"} |

求：

每个用户评分最高的N条记录

每个用户评分的平均值

最大方（评分给得高）的N个用户

最热门的N部电影

评价最高的N部电影

.......

### 需求：

求出每个用户的评分总和

### 思路：

典型的wordcount

## mr编程案例11——共同好友分析

### 数据：

冒号左边为用户id，冒号右边为用户的好友列表

|  |
| --- |
| A:B,C,D,F,E,O  B:A,C,E,K  C:F,A,D,I  D:A,E,F,L  E:B,C,D,M,L  F:A,B,C,D,E,O,M  G:A,C,D,E,F  H:A,C,D,E,O  I:A,O  J:B,O  K:A,C,D  L:D,E,F  M:E,F,G  O:A,H,I,J |

### 需求：

求出有共同好友的用户对，及他们的共同好友，如：

A与B，有共同好友C,E

A与C，有共同好友D

......

### 思路：

# mr客户端提交job到集群运行

## yarn的资源参数配置

yarn.scheduler.minimum-allocation-mb 默认值：1024 // yarn分配一个容器时最低内存

yarn.scheduler.maximum-allocation-mb 默认值：8192 // yarn分配一个容器时最大内存

yarn.scheduler.minimum-allocation-vcores 默认值：1 // yarn分配一个容器时最少cpu核数

yarn.scheduler.maximum-allocation-vcores 默认值：32 // yarn分配一个容器时最多cpu核数

// 1个nodemanager拥有的总内存资源

yarn.nodemanager.resource.memory-mb 默认值：8192

// 1个nodemanager拥有的总cpu资源（逻辑的，表示比例而已）

yarn.nodemanager.resource.cpu-vcores 默认值：8

# zookeeper 集群搭建

上传安装包，解压

修改conf/zoo.cfg

|  |
| --- |
| # The number of milliseconds of each tick  tickTime=2000  # The number of ticks that the initial  # synchronization phase can take  initLimit=10  # The number of ticks that can pass between  # sending a request and getting an acknowledgement  syncLimit=5  # the directory where the snapshot is stored.  # do not use /tmp for storage, /tmp here is just  # example sakes.  dataDir=/root/zkdata  # the port at which the clients will connect  clientPort=2181  # Set to "0" to disable auto purge feature  #autopurge.purgeInterval=1  server.1=hdp-01:2888:3888  server.2=hdp-02:2888:3888  server.3=hdp-03:2888:3888 |

配置文件修改完后，将安装包拷贝给hdp-02 和 hdp-03

接着，到hdp-01上，新建数据目录/root/zkdata，并在目录中生成一个文件myid，内容为1

接着，到hdp-02上，新建数据目录/root/zkdata，并在目录中生成一个文件myid，内容为2

接着，到hdp-03上，新建数据目录/root/zkdata，并在目录中生成一个文件myid，内容为3

启动zookeeper集群：