笔记内容

033课时MR初步认识

Hadoop MapReduce是一个编程模型，它包含两个过程Map&Reduce。

034课时MR编写mapper类

配置Eclipse关于hadoop2的开发环境：

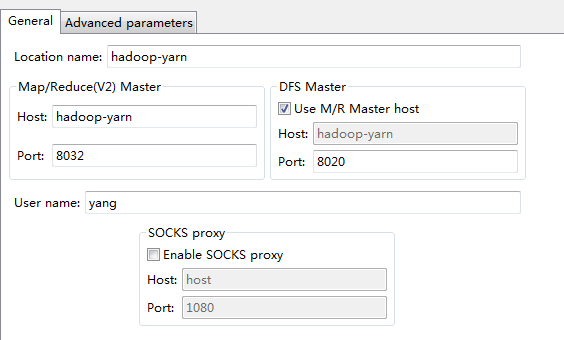
首先现在hadoop-eclipse-plugin工具包，位于github上面的开源项目，复制hadoop-eclipse-plugin-2.6.0.jar到Eclipse的plugin目录下面，重新启动Eclipse软件，在Perference中多了一个MapReduce的选项，在其中我们需要配置hadoop2的安装目录。

这里我们将hadoop2安装包解压到指定的文件夹，然后配置对应的hadoop主目录即可。这样创建hadoop项目的时候就会自动加载hadoop2安装包中的所有jar包。

关于eclipse配置信息：

其中DFSMaster是HDFS的fs.defaultFS配置为hadoop-yarn:8020(hostname:port)该信息在core-site.xml配置文件下面。

同时MapReduce配置为hadoop-yarn:8032,还有关于其他我们在配置文件中的选项，如dfs.replications dfs.permissions等信息。配置完成之后，确认配置好了本地主机的hosts文件，然后重新启动Eclipse即可通过Eclipse连接HDFS文件系统。



035课时MR编写Reducer类

036课时MR编写main函数

037课时MR独立模式运行

在单独的主机上运行即可

hadoop jar xxx.jar <MainClassName> <inputpathname> <outputpathname>

038课时 MR伪分布式运行

上传数据文件到HDFS文件系统中

hadoop dfs -put <localfile> <destfile>

在命令行窗口运行如下命令

hadoop jar HadoopDemo.jar <inputpathdir> <outputpathdir>

039课时 MR集群模式调整配置与Shell脚本处理

1. 配置虚拟机默认以文本方式启动，这样比较少的消耗系统资源

/etc/default/grub文件

# If you change this file, run 'update-grub' afterwards to update

# /boot/grub/grub.cfg.

# For full documentation of the options in this file, see:

# info -f grub -n 'Simple configuration'

GRUB\_DEFAULT=0

GRUB\_HIDDEN\_TIMEOUT=0

GRUB\_HIDDEN\_TIMEOUT\_QUIET=true

GRUB\_TIMEOUT=10

GRUB\_DISTRIBUTOR=`lsb\_release -i -s 2> /dev/null || echo Debian`

GRUB\_CMDLINE\_LINUX\_DEFAULT="**quiet splash**" 改成text

GRUB\_CMDLINE\_LINUX=""

**sudo update-grub # do not forget to execute the command**

sudo reboot now

这样就会进入命令行模式启动

1. 使用脚本批量处理上面的功能：

|  |
| --- |
| #!/bin/bash  for((index = 1; index<9; index++))  {  scp /etc/default/grub root@hadoop-yarn$index:/etc/default/  ssh -t hadoop-yarn sudo update-grub  } |

1. 修改副本数量和块大小
2. 默认配置文件

core-site.xml 默认的配置文件在share/hadoop/common/hadoop-common-2.6.4.jar

hdfs-site.xml: share/hadoop/hdfs/hadoop-hdfs-2.6.4.jar

mapred-site.xml: share/hadoop/mapred/hadoop-mapreduce-client-core-2.6.4.jar

yarn-site.xml:share/hadoop/yarn/ hadoop-yarn-common-2.6.4.jar

1. 修改配置文件

dfs.blocksize 20k 针对小文件

dfs.replication 1

发送修改后的文件到所有节点，并且重新启动集群（删除节点下面的临时目录之后再启动）

|  |
| --- |
| #!/bin/bash  for((index=1; index<8;index++))  {  scp $HADOOP\_HOME/etc/hadoop/hdfs-site.xml yang@hadoop-yarn$index:$HADOOP\_HOME/etc/hadoop  ssh yang@hadoop-yarn$index rm -rvf /opt/hadoop/tmp  } |

再重新启动集群

040课时 MR集群模式调整配置最小块设置与Web访问

我们配置的hadoop dfs.blocksize 的值是20k，但是上传文件的时候，就会出现错误，因为在HDFS配置文件中设置了最小的块大小，我们设置的blocksize必须大于等于dfs.namenode.fs-limits.min-block-size值

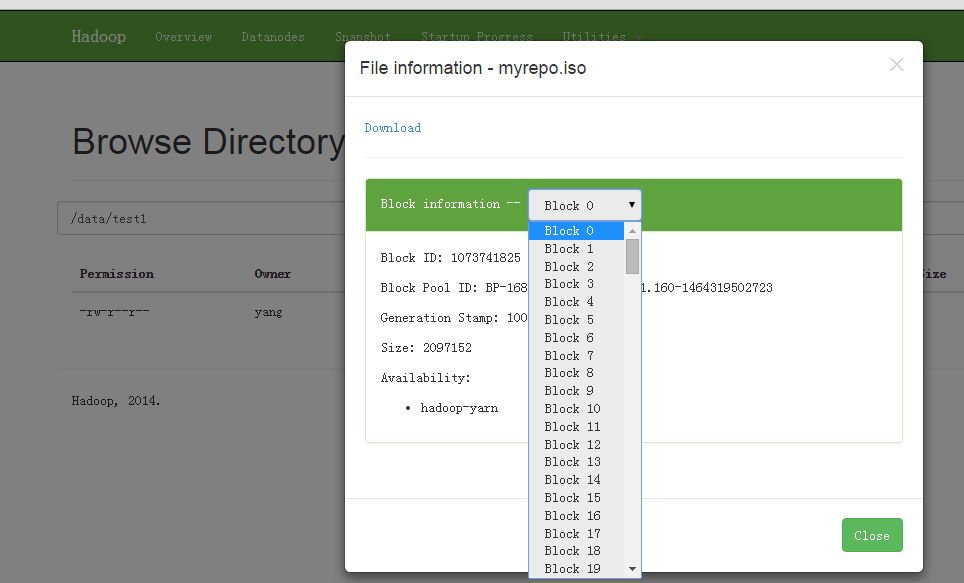
重新启动Hadoop集群

然后在将文件上传到HDFS文件系统

通过Web的方式可已查看一个文件中每一个块分布在哪一个数据节点

041课时 MR集群模式调整配置考察数据分布

当我们把replication修改为2的时候，并且集群中存在多个主机节点，那么我们可以通过Web页面查看指定的文件被分割的每一个block保存在哪一台数据节点上面。为了便于我们测试的，我们将dfs.blocksize设置为1024\*1024\*2 = 2M 并且我们观察到dfs.namenode.fs-limits.min-block-size的值是1024\*1024，dfs.blocksize必须大于等于这个值，最好是这个值的倍数。



042课时 MR集群模式深入定制切割和副本数以及块带下定制定制

Split = filesize / block size

对于gz文件是不能够进行切割的，只能把文件看做一个整体输入map()函数。

|  |
| --- |
| InputSplit split = context.getInputSplit();  **long** length = split.getLength(); |

在单独的数据节点上，我们查看用户日志下面执行任务的记录情况。

对于dfs.replication大于1的时候，运行MapReduce程序的时候，只会在一个replication上面运行，具体在哪一个节点上运行，取决于Hadoop框架的决策：

Block0：s4 s6

Block1: s3 s3

Block2: s1 s4

Block3: s2 s4

s1 s2 s3: no log

s4 s5 s6: log

虽然是将计算移动到数据的部分，但是Hadoop框架也会根据实际计算节点资源的情况进行优化调整，没有数据的节点也可能会有计算（比如将数据节点的数据移动到s5节点进行计算）

043课时 MR集群模式深入定制切割与切割法则考察

Hadoop端口详情

|  |  |  |
| --- | --- | --- |
| HDFS | ip:50070 | Name node |
| ip:50075 | Data node |
| ip:50090 | Secondary name node |
| MapReduce(Yarn) | ip:8088 | Resource manager |
| ip:8042 | Node manager |

|  |
| --- |
| Blocksize： 1024\*1024 \* 2 = 2M  在程序中设置最大的切割大小  configuration.setLong(FileInputFormat.*SPLIT\_MAXSIZE*, 1024\*1024\*4);  但是当切割块大小设置成为大于blocksize,如果超过就默认为blocksize  可以小于min blocksize  可以介于max & min之间 |

044课时 MR Win7 下配置networks文件映射主机名称和端口号利于网络访问

配置windows host配置文件在C:\Windows\System32\drivers\etc

045课时 MR原理——物理导航-Eclipse安装-后台启动-命令化

./eclipse &

sudo ln -s eclipse /usr/local/bin/eclipse

046课时 MR原理——hadoop插件安装-配置-验证

配置自动补全 Alt+/ Keys Content Assist

后台运行程序：

1. 启动的时候 command + &
2. Ctrl +z 将已经启动的程序放到后台暂停

bg %<job id> 后台运行程序， 使用jobs指令查看作业号

fg %<job id> 前台运行程序的切换

在Eclipse安装插件 Hadoop2x-eclipse-plugin

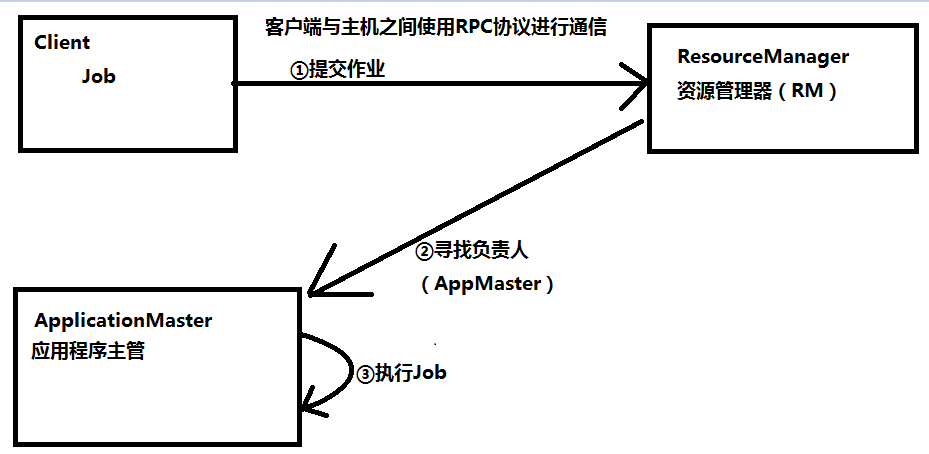
将对应的release中的jar放到eclipse/plugins目录下

重新启动Eclipse，在Preference下多了一个 hadoop map/reduce

指定hadoop的安装目录

配置hadoop的主机地址 hadoop-yarn 8020

047课时 MR原理——大象装进水箱的相似点

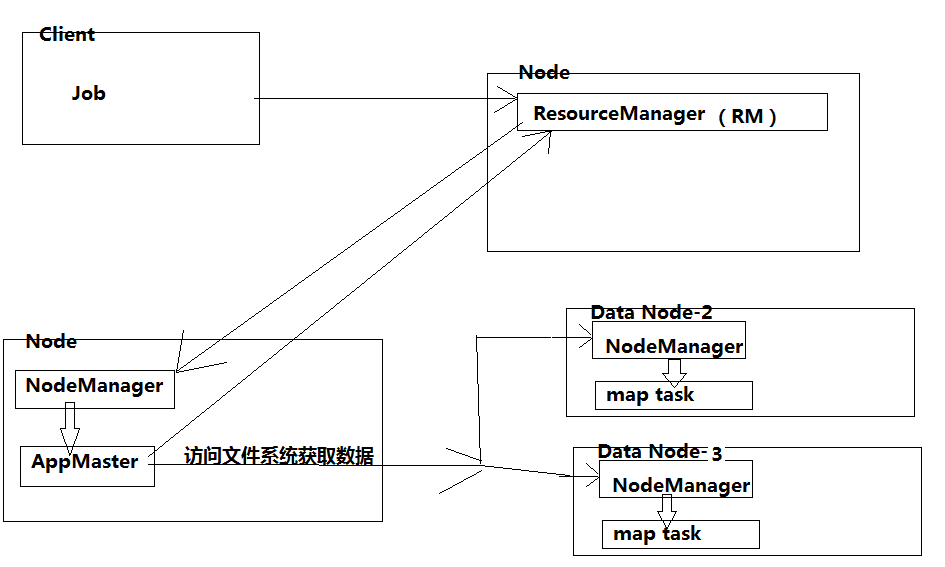


1.客户端提交作业给Yarn的ResourceManager（RM）

2.ResourceMaster寻找APPMaster

3.APPmaster执行Application

048课时 MR原理——进入流程图



Job == Application， 在客户端称作Job，在服务器端被称为Application

049课时 MR原理——核心类Job与ResourceManager解读

Job：之前我们使用的是new Job(); 推荐使用Job.getIntance(Configuration conf)

ResourceManager: 集群中所有的资源都属于我管理，"I am the ResourceManager. All your resources belong to us..."

050课时 MR原理——核心类NodeManager与MRAdapter解读

NodeManager

ContainerManagerImpl

ContainsLAuncher

ContainLauncher

MrAppMaster

The Map-Reduce Application Master. The state machine is encapsulated in the implementation of Job interface. All state changes happens via Job interface. Each event results in a Finite State Transition in Job. MR AppMaster is the composition of loosely coupled services. The services interact with each other via events. The components resembles the Actors model. The component acts on received event and send out the events to other components. This keeps it highly concurrent with no or minimal synchronization needs. The events are dispatched by a central Dispatch mechanism. All components register to the Dispatcher. The information is shared across different components using AppContext.

状态机，封装了Job接口的实现，所有状态的变化都会通过Job接口时期发生变化

每一个Event都会导致最终状态的变换

状态机变换是基于时间的，并且组件之间收发Event，Event是载体

事件有核心的分发机制进行分发。

051课时 MR原理代码实战——Job提交推演核心类Yarn MapTask ReduceTask解读

YarnChild

MR Child 主要进程的启动

MapTaskImpl extends TaskImpl

对于Map任务的封装

ReduceTaskImpl extends TaskImpl

对于Reduce任务的封装

052课时 MR原理代码实战——核心类RM与NM、MRAM的体系结构分析

ResourceManager, NodeManager, MRAppMaster三者作为服务运行

ResourceManager

extends CompositeService extends AbstractService implements Service

implements Recoverable

NodeManager

extends CompositeService

implements EventHandler<NodeManagerEvent>

MRAppMaster

extends CompositeService

|  |  |
| --- | --- |
| Service | Interface Definition init() start() stop() |
| AbstractService | Listener<ServiceStateChangeListener> |
| CompositeService | List<Service> serviceList  addService(xxxService) |
| ResourceManager  NodeManager  MRAppMaster | Service具体实现类 |

053课时 MR原理代码实战——ClassNotFound错误信息展现于分析

略

054课时 MR原理代码实战——Ant使用

Eclipse升级Ant到1.9 不然会出现java1.8 ClassNotFound 也就是低版本的Ant不支持Java1.8，升级方法，找到Eclipse/plugin/\*Ant\*目录，将里面的文件替换成为新的Ant1.9解压包里的文件（先删除对应的文件夹，在复制）。

使用Ant工具自动编译项目：

|  |
| --- |
| <project name="hadoopdemo2" basedir="." default="prepare">  <target name="prepare">  <delete dir="${basedir}/build/classes" />  <mkdir dir="${basedir}/build/classes"/>  </target>    <path id="path1">  <fileset dir="${basedir}/lib" >  <include name="\*.jar"/>  </fileset>  </path>    <target name="compile" depends="prepare">  <javac srcdir="${basedir}/src" destdir="${basedir}/build/classes" classpathref="path1" includeantruntime="true" />  </target>    <target name="package" depends="compile">  <jar destfile="${basedir}/My.jar" basedir="${basedir}/build/classes"/>  </target>  </project> |

055课时 MR原理代码实战——Job提交推演

YarnChild

MR Task 任务的主要进程，负责启动MapTask & ReduceTask

MapTaskImpl

对于Map任务的封装

ReduceTaskImpl

对于Reduce任务简单的封装

056课时 MR原理代码实战——Job初始化分析

1. Job提交过程

submit() 方法内部创建submitter并调用submitJobInternal()

submit过程如下：请求RM获取appid，用作MR job id；检查output的有效性；计算input split; 复制资源（JAR conf input split）到HDFS中，存放在以Job ID命名的目录下。

2. Hadoop框架

HDFS：分布式文件系统，存储海量数据

NameNode：名称节点org.apache.hadoop.hdfs.server.namenode.NameNode

DataNode：数据节点org.apache.hadoop.hdfs.server.datanode.DataNode

SecondaryNameNode：辅助名称节点org.apache.hadoop.hdfs.server.namenode

MapReduce：map+reduce

Mapper: map()

Reducer:reduce()

Yarn：执行框架，调度执行作业

ResourceManager org.apache.hadoop.yarn.server.resourcemanager.ResourceManager

NodeManager org.apache.hadoop.yarn.server.nodemanager.NodeManager

3.Job 的初始化

RM收到submitApp()后将请求转给Yarn scheduler

scheduler分配container

RM在该container启动APP Master，并且交由NameNode管理

App Master创建多个记录对象跟踪Job进度，它将接受task的进度或完成报告。

检索input split，对每一个split创建map()任务和一定数量的reduce任务(setReduceNum())

AppMaster 判断如何运行task，如果是小的job，AppMaster会在同一个JVM中运行。User task就是指这一点，因为开启新容器分配和运行程序更耗费资源。小的Job标准是map<10,只有reduce=1，而且input size < block size,这些值是可以进行修改的。

mapreduce.job.ubertask.maxmaps

mapreduce.job.ubertask.maxreduces

mapreduce.job.ubertask.enable

最后 AppMaster调用OutputCommintter的setupJob()方法，默认是FileOutputCommimter，主要是创建output目录和临时工作目录。

057课时 MR原理代码实战——ResourceManager远程调试

推荐使用的Hadoop启动方式

start-dfs.sh(stop-dfs.sh)

hadoop namenode

hadoop datanode

hadoop secondarynamenode

start-yarn.sh(stop-yarn.sh)

yarn resourcemanager

yarn nodemanager

使用Eclipse执行在Server集群，进行远程调试，跟踪远程服务器上的资源

1.启动hadoop集群

2.设置

客户端必须有资源管理器的代码:Eclipse代码中

2.1 运行Java程序的时候，直接指定参数

java –xxx agentlib:jdwp=transport=dt\_socket,server=y,suspend=y,address=8000

2.2先指定环境变量，再启动JVM，

export YARN\_RESOURCEMANAGER\_OPTS=

”-agentlib:jdwp=transport=dt\_socket,server=y,suspend=y,address=8000”

2.3 ${HADOOP\_HOME}\bin\yarn

$YARN\_RESOURCEMANAGER\_OPTS设置环境变量

创建一个脚本文件 touch enable\_yarn\_remotedebug.sh

内容：

*export YARN\_RESOURCEMANAGER\_OPTS=*

*-agentlib:jdwp=transport=dt\_socket,server=y,suspend=y,address=8000*

source enable\_yarn\_remotedebug.sh

disable\_yarn\_remotedebug.sh

*export YARN\_RESOURCEMANAGER\_OPTS=*

启动资源管理器 start-yarn.sh:

这样会启动多个NodeManager，但是ResourceManager暂停在箭筒8000端口中

到客户端设置断点，连接到远程额RM程序，进行调试

Eclipse Debug Configuration

Socket Attach （选择连接类型）

Host: ip+port hadoop-yarn:8000

058课时 Yarn框架底层事件分发机制

Yarn：基于事件

Job：

1. application基于状态机模型

NEW NEW\_SAVING ACCEPTED RUNNING FINISHED KILLED

2.状态之间的变换通过事件触发

AsyncDispatcher:异步事件分发器，事件队列和事件分发器，是底层事件的总管道。





需要有专门的Thread，从eventQueue中获取Event，然后在有dispatcher进行处理：

|  |
| --- |
| Runnable createThread() {  return new Runnable() {  @Override  public void run() {  while (!stopped && !Thread.currentThread().isInterrupted()) {  drained = eventQueue.isEmpty();  // blockNewEvents is only set when dispatcher is draining to stop,  // adding this check is to avoid the overhead of acquiring the lock  // and calling notify every time in the normal run of the loop.  if (blockNewEvents) {  synchronized (waitForDrained) {  if (drained) {  waitForDrained.notify();  }  }  }  Event event;  try {  **event = eventQueue.take();**  } catch(InterruptedException ie) {  if (!stopped) {  LOG.warn("AsyncDispatcher thread interrupted", ie);  }  return;  }  if (event != null) {  **dispatch(event);**  }  }  }  };  } |



向队列中放置event

在RecourceManager中进行Event的二次分发

RMAppAttemptEvent

**ApplicationAttemptEventDispatcher** implement EventHandler 将事件分发给RMApp进行处理，RMAppImpl实现集体的功能。二次分发，在ApplicationAttemptEventDispatcher分发是无状态的，但是在RMAppImpl是有状态的进行处理。

事件处理实体 事件分发器 处理事件 事件类型

RMAppAttempt -> ApplicationAttemptEventDispatcher--> RMAppAttemptEvent-> RMAppAttemptEventType

RMAppImpl-> ApplicationEventDispatcher -> RMAppEvent -> RMAppEventType

059课时 Yarn框架底层事件分发调试准备

通过远程调试

1.编写enable\_yarn\_remotedebug.sh disable\_yarn\_remotedebug.sh

在其中设置YARN\_RESOURCEMANAGER\_OPTS

*-agentlib:jdwp=transport=dt\_socket,server=y,suspend=y,address=8000*

并且使其生效 source \*\*.sh

2.启动yarn框架

Start-yarn.sh

3.配置Eclipse远程调试，连接到ResourceManager8000端口（没有成功）

060课时 Yarn框架底层事件分发调试Async分发调试

Hadoop Java核心类

HDFS：分布式文件系统，存储海量数据

NameNode：名称节点org.apache.hadoop.hdfs.server.namenode.NameNode

DataNode：数据节点org.apache.hadoop.hdfs.server.datanode.DataNode

SecondaryNameNode：辅助名称节点org.apache.hadoop.hdfs.server.namenode

Yarn：执行框架，调度执行作业

ResourceManager org.apache.hadoop.yarn.server.resourcemanager.ResourceManager

NodeManager org.apache.hadoop.yarn.server.nodemanager.NodeManager

Job在ResourceManager上面的初始化

1.RM 收到submitApp()请求后转给YarnScheduler

2.Schedule分配给container

3.在Container上面启动AppMaster， 交给NodeManager去管理

4.AppMaster会创建记录，跟踪job进度，并且接受task的进度报告和完成报告

5.检索inputsplit

6. 为每一个split创建一个map任务和一定梳理的reduce任务

7. AppMaster 判断如何运行task，如果是小的job，AppMaster会在同一个JVM中运行。User task就是指这一点，因为开启新容器分配和运行程序更耗费资源。小的Job标准是map<10,只有reduce=1，而且input size < block size,这些值是可以进行修改的。

mapreduce.job.ubertask.maxmaps

mapreduce.job.ubertask.maxreduces

mapreduce.job.ubertask.enable

8.最后 AppMaster调用OutputCommintter的setupJob()方法，默认是FileOutputCommimter，主要是创建output目录和临时工作目录。

061课时 Yarn框架底层实践分发调试RMAppImpl事件处理调试-状态机模型变换

NEW

NEW\_SAVING

SUBMITTED

ACCEPTED

RUNNING

FINISHED

FAILED

KILLED

062课时 Yarn框架底层事件分发调试——状态机事件触发原理

略

063课时 Hadoop底层IPC和RPC通信原理解析

IPC：进程间通信技术

RPC(remote procedure call远程过程调用)

Hadoop底层通信使用的是Google Proto Buffer实现的

基于都是基于Socket编程（Socket ServerSocket）

传统的套接字是基于阻塞的同步的通信，在新的JDK中有异步通信(NIO). Hadoop使用的是NIO技术：NIO（Channel Selector）

Client Server

ProtobufRpcEngine

064课时 Hadoop运行参数与Driver——数据环境处理与编程式

自动化脚本处理过程处理 删除

065课时Hadoop运行参数与Driver——配置文件方式

1.配置文件方式设置Hadoop的参数

2.编程方式

066课时Hadoop运行参数与Driver——参数指定方式与Driver

hadoop jar -D mapreduce.job.reduces=2 xxx.jar <input> <output>

使用Tool方式进行编写

067课时 Hadoop MapReduce类型——组合函数

略

068课时 Hadoop MapReduce类型——分区函数以及自定义与注意事项

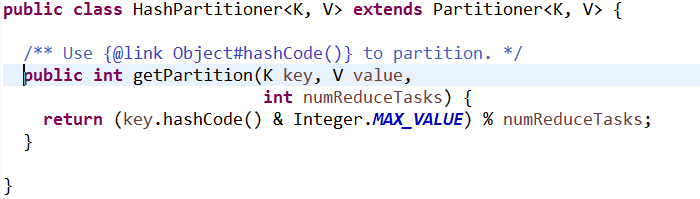
相同的key一定要传给同一个reduce节点

Partitioner

分区函数：将map的output传递给reduce中的一个

传递法则：相同的key值传递给同一个reduce

在Hadoop中提供了Partitioner的默认实现：HashPartitioner默认使用该方式进行Partition。



可以根据自己的需要实现自己的MyPartitioner