法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

- 课程详情请咨询
  - 微信公众号：小象学院
  - 新浪微博：小象AI学院

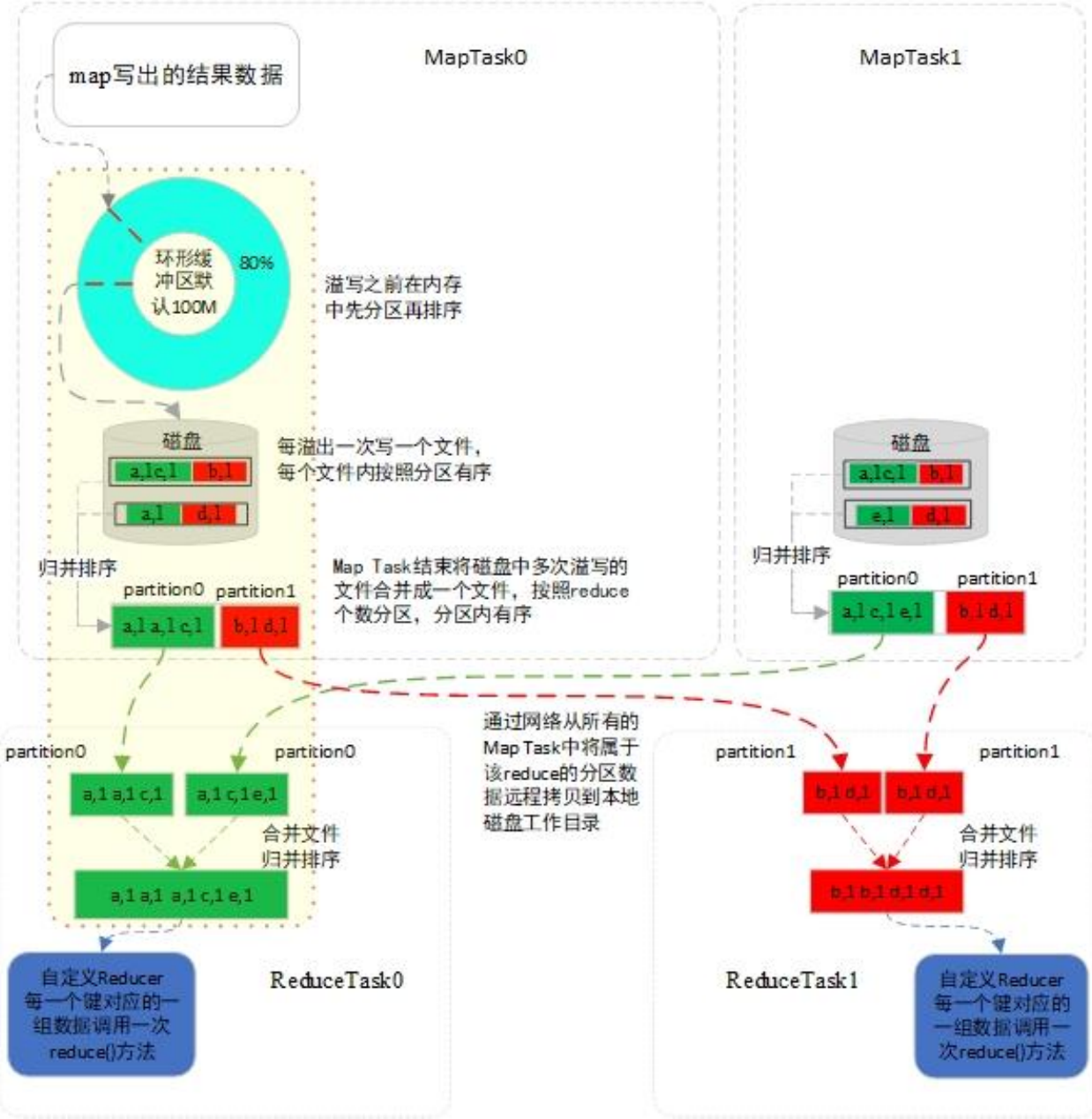
大纲

- Shuffle过程与Combiner优化
- YARN内置调度器
- MapReduce编程进阶
- MapReduce优化技巧

大纲

Shuffle过程与Combiner优化

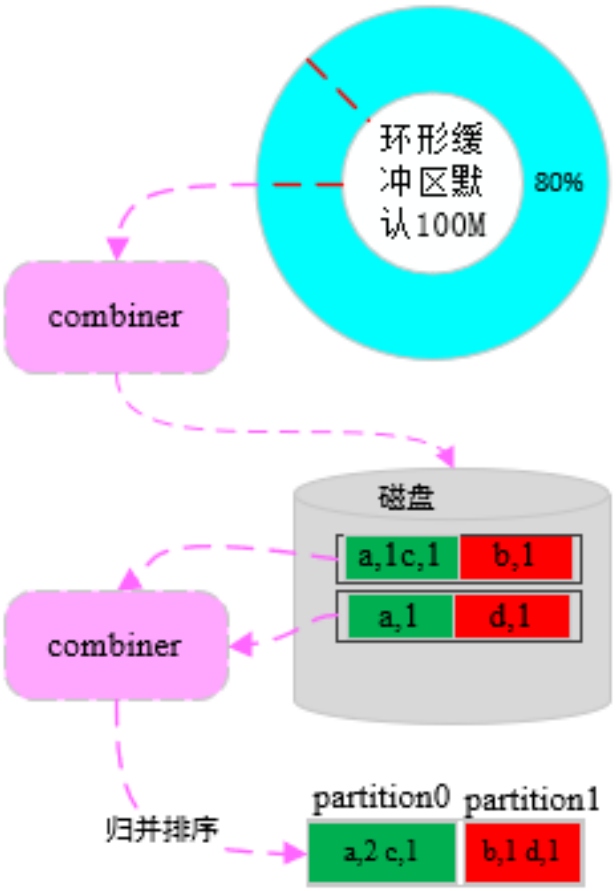
Shuffle过程


map写出的结果数据
MapTask0
MapTask1
环形缓冲区默认100M
80%
溢写之前在内存中先分区再排序
磁盘
a,1 c,1
b,1
a,1
d,1
每溢出一次写一个文件，每个文件内按照分区有序
归并排序
partition0
partition1
a,1 a,1 c,1
b,1 d,1
Map Task结束将磁盘中多次溢写的文件合并成一个文件，按照reduce个数分区，分区内有序
磁盘
a,1 c,1
b,1
e,1
d,1
归并排序
partition0
partition1
a,1 c,1 e,1
b,1 d,1
通过网络从所有的Map Task中将属于该reduce的分区数据远程拷贝到本地磁盘工作目录
partition0
partition0
a,1 a,1 c,1
a,1 c,1 e,1
合并文件
归并排序
a,1 a,1 a,1 c,1 e,1
partition1
partition1
b,1 d,1
b,1 d,1
合并文件
归并排序
b,1 b,1 d,1 d,1
自定义Reducer每一个键对应的一组数据调用一次reduce()方法
ReduceTask0
ReduceTask1
自定义Reducer每一个键对应的一组数据调用一次reduce()方法

Shuffle过程总结

- 每个Map Task把输出结果写到内存中的环形缓冲区
- 当内存环形缓冲区写入的数据量达到一定阈值时，后台线程会把数据溢写到磁盘
  - 根据Partitioner，把数据写入到不同的partition
  - 对于每个partition的数据进行排序
- 随着Map Task的不断运行，磁盘上的溢出文件越来越多
  - 将这些溢出文件合并
  - 对于一个partition下的不同分片，使用归并排序，同一分区内数据有序
- Reduce Task通过网络远程拷贝MapTask的结果文件中的属于它的分区数据
  - 合并所有已拷贝过来的数据文件
  - 采用归并排序算法，对文件数据内容整理排序，将相同key的数据分为一组，不同key之间有序
  - 最终生成一个key对应一组值的数据集，一个key对应的一组数据会调用一次reduce方法

Combiner优化



溢写之前先分区再排序，在分区内执行combiner预聚合

每溢出一次写一个文件，每个文件内按照分区有序

Map Task结束将磁盘中多次溢写的文件合并成一个文件，分区内有序，再次在分区内执行combiner预聚合

Combiner总结

- Combiner调用的地方
  - MapTask的环形缓冲区向磁盘溢写文件之前调用Combiner
  - Map阶段在合并本地多个文件写入一个大文件之前调用Combiner
- 使用Combiner的好处
  - 减少Map Task输出数据量，由于临时结果写入到本地磁盘，所以能够减少磁盘IO
  - 减少Reduce-Map网络传输数据量，由于reduce需要远程通过网络从Map拷贝数据，提高拷贝速度
- 应用场景
  - 针对结果可以叠加的场景
  - SUM(YES) Average（NO）
- 设置方法（local reducer）
  - job.setCombinerClass(WordCountReducer.class)

大纲

YARN内置调度器

数据本地性

- 数据本地性含义（data locality）
  - 如果任务运行在与它需要处理的数据在同一个节点，则称该任务具有“数据本地性”
- 数据本地性级别（性能由高到低排列）
  - 同节点（node-local）
  - 同机架（rack-local）
  - 跨机架（off-switch）
- 数据本地性优点
  - 避免通过网络远程读取数据进而提高数据读取效率

推测执行

- 作业完成时间取决于最慢的任务完成时间
  - 一个作业由若干个Map任务和Reduce任务构成
  - 因硬件老化、软件Bug等，某些任务可能运行非常慢
- 推测执行机制
  - 发现拖后腿的任务，比如某个任务运行速度远慢于任务平均速度
  - 为拖后腿任务启动一个备份任务，同时运行
  - 谁先运行完，则采用谁的结果
- 不适用推测执行的场景
  - 任务存在严重的负载倾斜
  - 特殊任务，比如任务向数据库中写数据

YARN调度器-FIFO

- 将所有应用程序放入到一个队列中
  - 先进入队里排在前面的程序先获得资源
- 局限性
  - 资源利用率低，无法交叉运行作业
  - 不够灵活，比如紧急的作业无法插队，耗时长作业拖慢耗时短作业

YARN调度器-多队列分开调度

- 所有资源按照比例划分到不同的队列
- 每个队列可以实现单独的调度策略
- 优点
  - 按照不同的资源使用情况将资源划分到不同队列
  - 能够让更多的应用程序获得资源
  - 使用灵活，资源利用率高
- 调度器
  - CapacityScheduler调度器
  - FairScheduler调度器

YARN调度器-CapacityScheduler

- 由Yahoo开源，共享集群调度器
- 以队列方式组织作业
- 每个队列内部采用FIFO调度策略
- 每个队列分配一定比例资源
- 可限制每个用户使用资源量

CapacityScheduler配置方法

- 在yarn-site.xml配置文件中设置使用CapacityScheduler调度器

```
<property>
    <name>yarn.resourcemanager.scheduler.class</name>
    <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler</value>
</property>
```

- 在hadoop配置文件目录下创建capacity-scheduler.xml文件，添加各队列资源分配情况

```
<configuration>
  <property>
    <name>yarn.scheduler.capacity.root.queues</name>
    <value>default,data_bi</value>
  </property>
  <property>
    <name>yarn.scheduler.capacity.root.default.capacity</name>
    <value>60</value>
  </property>
  <property>
    <name>yarn.scheduler.capacity.root.default.maximum-capacity</name>
    <value>80</value>
  </property>
  <property>
    <name>yarn.scheduler.capacity.root.bi.capacity</name>
    <value>40</value>
  </property>
</configuration>
```

CapacityScheduler配置方法

- capacity-scheduler.xml参数说明
  - capacity：队列占用的集群资源容量百分比，所有队列的容量之和应小于100
  - maximum-capacity：由于存在资源共享，因此一个队列使用的资源量可能超过其容量，而最多使用资源量可通过该参数限制
- 配置完成无需重启YARN，使用管理命令刷新调度配置
  - bin/yarn rmadmin -refreshQueues

YARN调度器-FairScheduler

- 由Facebook开源的，共享集群调度器
- 以队列方式组织作业
- 基于最小资源量和公平共享量进行调度
- 支持资源抢占
- 内部队列中可使用的策略
  - FIFO
  - fair（默认），基于内存使用量调度分配资源
- 任务延时调度
  - 提高数据本地性
  - 提高系统整体吞吐率

YARN调度器-FairScheduler

- 公平调度器的目的
  - 允许多用户共享集群资源。
  - 允许短时的临时作业与长时作业共享集群资源
  - 根据比例来管理集群资源，确保集群资源的有效利用

FairScheduler配置方法

- 在yarn-site.xml配置文件中设置调度器类型，指定公平调度器配置文件路径

```
<property>
    <name>yarn.resourcemanager.scheduler.class</name>
    <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler</value>
</property>
<property>
    <name>yarn.scheduler.fair.allocation.file</name>
    <value>/usr/local/hadoop/etc/hadoop/fair-scheduler.xml</value>
</property>
<property>
    <name>yarn.scheduler.fair.user-as-default-queue</name>
    <value>true</value>
</property>
<property>
    <name>yarn.scheduler.fair.preemption</name>
    <value>true</value>
</property>
```

FairScheduler配置方法

yarn-site.xml参数说明

- yarn.resourcemanager.scheduler.class配置yarn使用的调度器类型
- yarn.scheduler.fair.allocation.file配置公平调度器自定义配置文件路径，该文件每隔10秒就会被加载一次，这样就可以在集群运行过程中改变队列的配置
- yarn.scheduler.fair.user-as-default-queue当应用程序未指定队列名时，是否指定用户名作为应用程序所在的队列名。如果设置为false或者未设置，所有未知队列的应用程序将被提交到default队列中，默认值为true
- yarn.scheduler.fair.preemption如果一个队列占用的资源量少于最小资源量限制，是否启用资源抢占，默认false。抢占机制可以使其他队列的作业容器终止，从而使占用的资源让出，将资源分配给占用资源量少于最小资源量限制的队列

FairScheduler配置方法

- 在fair-scheduler.xml配置文件中设置调度器相关配置项

```
<allocations>
  <queue name="data_bi">
    <minResources>8000 mb,4 vcores</minResources>
    <maxResources>10000 mb,6 vcores</maxResources>
    <maxRunningApps>2</maxRunningApps>
    <weight>1.0</weight>
  </queue>
</allocations>
```

FairScheduler配置方法

fair-scheduler.xml参数说明

- queue name：配置队列名
- minResources ：分配给该队列的最小资源量，设置格式为“X mb, Y vcores”，当调度策略属性schedulingPolicy的属性值是fair时，其cores值会被忽略，仅按照申请的内存大小来调度。
- maxResources：分配给该队列的最大资源量。设置格式为“X mb, Y vcores”，当调度策略属性schedulingPolicy的属性值是fair时，其cores值会被忽略，仅按照申请的内存大小来调度。
- maxRunningApps：最多同时运行的应用程序数目。通过限制该数目，可防止超量Map Task同时运行时产生的中间输出结果撑爆磁盘。
- weight：标记了资源池的权重，当资源池中有任务等待，并且集群中有空闲资源时候，每个资源池可以根据权重获得不同比例的集群空闲资源，默认值是1

大纲

MapReduce编程进阶

广告数据分析背景

- 日志数据格式
  - 地域编码（area_id），字符串类型
  - 用户编号（user_id），字符串类型
  - 浏览类型（view_type），整数类型，1表示曝光 2表示点击
  - 日期（date）：字符串类型，格式如：20171228
  - 字段之间分隔符：Tab键
- 名词解释
  - 广告曝光量：广告被浏览的次数，简称PV（Page View）
  - 广告点击量：广告被点击的次数，常用click表示
  - 广告点击率：广告点击量/广告曝光量 （ clicks / views ）,常用click_ratio表示

广告数据统计需求（一）

一批TB或者PB量级的历史广告数据，需要完成如下功能

- 统计粒度：按天统计
- 统计指标：计算曝光量（PV）
- 按照曝光量升序排列和倒序排列

广告数据统计需求分析（一）

整个需求实现分为两个作业

- 作业1：统计
  - 按天统计曝光量
- 作业2：排序
  - 按照曝光量进行全排序
  - 依赖于前一个作业的输出结果
  - 升序依赖MR框架Shuffle阶段对key进行升序排列的特性
  - 降序需要重写key的比较器

Count

20171224	16
20171225	25
20171226	15
20171227	20
20171228	10

Sort

20171225	25
20171227	20
20171224	16
20171226	15
20171228	10

HDFS中的数据文件　　按天统计曝光量　　按照曝光量排序

广告数据统计需求（二）

对前一天产生的广告数据进行统计，需要完成如下功能

- 统计粒度：按天统计
- 统计频率：每天统计前一天的数据
- 统计指标：曝光量pv，点击量click，点击率click_ratio
- 统计维度：地域area_id

广告数据统计需求分析（二）

- 统计的指标有pv、click和click_ratio， click_ratio可以通过在同一个Reduce任务中使用pv、click的最终计算结果求得，
- 统计指标是多个，需要自定义对象封装多个指标，对象需要实现Writable序列化接口
- 按照地域和日期两个维度统计，使用组合键，保证同一天同一个地域的数据被分发到同一个Reduce Task中

广告数据统计需求（三）

找出不同性别的不同年龄段用户对某个产品的最高打分,需要完成如下功能

- 日志数据格式
  - 姓名（name），字符串类型
  - 年龄（age），整数类型
  - 性别（gender），字符串类型
  - 打分（core），整数类型，0到100的整数值
- 统计指标
  - 指标：最高分
  - 维度：姓名，性别，年龄

广告数据统计需求分析（三）

- 要保证同一年龄段的数据分发到同一个reduce task中
  - 默认HashPartitioner不满足要求
  - 自定义Partitioner
- 借助Reduce阶段的Shuffle对相同key的数据整合成key对应一组数据的特点，保证同一性别的数据被一个reduce方法处理
- 默认Reduce Task数为1，需要根据分区数调整Reduce Task数

优化技巧

- Map Task重试次数，默认4次
  - 调整参数：mapreduce.map.maxattempts
- Reduce Task重试次数，默认4次
  - 调整参数： mapreduce.reduce.maxattempts
- 是否打开Map阶段的推测执行机制，默认true
  - 调整参数： mapreduce.map.speculative
- 是否打开Reduce阶段的推测执行机制，默认true
  - 调整参数： mapreduce.reduce.speculative
- 调整一个分片的最小数据量，进而调整Map Task的任务数
  - 调整参数： mapreduce.input.fileinputformat.split.minsize

疑问

- 小象问答官网
  - http://wenda.chinahadoop.cn

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：小象学院
- 新浪微博：小象AI学院