MapReduce8 优化

MapReduce跑得慢的原因

MapReduce程序效率的瓶颈在于两点：计算机性能和 I/O操作

计算机性能

CPU、内存、磁盘健康、网络

I/O操作

- 数据倾斜
- Map和Reduce设置不合理
- Map运行时间太长，导致Reduce等待过久
- 小文件过多
- 大量的不可分块的超大文件
- spill次数过多
- Merge次数过多等

MapReduce优化方案

数据输入优化

大量小文件在数据输入的时候进行处理

```
1. 合并小文件，在执行MR任务之前先将小文件合并，大量的小文件会产生大量的Map任务，增加Map任务装载次数，而任务的装载比较耗时，从而导致MR运行较慢
2. 修改InputFormat的combineTextInputFormat作为输入，解决输入端大量小文件场景
```

Map阶段优化

减少溢写（spill）次数

```
通过调整io.sort.mb 以及sortsill percent参数值，增大触发split的内存上限，减少split的次数，从而减少磁盘I/O
```

减少合并（Merge）次数

```
通过调整io.sort.factor参数，增大Merge的文件数目，减少Merge的次数，从而缩短MR处理时间
```

性能优化

```
在Map之后，不影响业务逻辑的情况下，先进性conbine处理，减少io
```

Reduce阶段

Map和Reduce不合理

合理设置map和reduce数量
两个都不能设置太少，不能设置太多
太少会道中Task等待，延长时间
太多会导致Map、Reduce任务间竞争资源，造成处理超时错误

Map运行时间太长，导致Reduce等待过久

调整slowstart.completedmaps参数，使map运行到一定程度后，Reduce开始工作，减少Reduce的等待时间

规避使用Reduce

因为Reduce在用于连接数据集的时候会产生大量的网络消耗

合理设置Reduce端的bufffer

合理设置 Redluce端的Buffer：默认情况下，数据达到一个阈值的时候， Buffer中的数据就会写入磁盘，然后 Reduce会从磁盘中获得所有的数据。也就是说， Buffer和 Reduce是没有直接关联的，中间多次写 磁盘->读磁盘的过程，既然有这个弊端，那么就可以通过参数来配置，使得 Buffer中的一部分数据可以直接输送 到Reduce，从而减少IO开销：mapred.job.reduce.input.buffer.percent，默认为0.0.当值大于0的 时候， 会保留指定比例的内存读 Buffer中的数据直接拿给Reduce使用。这樺一来，设置 Buffer需要内存，读取数据需要内存， Reduce计算也要内存，所以要根据作业的运行情况进行调整。

I/O传输

1. 采用数据压缩的方式，减少IO的时间，安装Snappy和LZO压缩编辑器
2. 使用Sequencefile输出

数据倾斜问题

数据倾斜现象

- 数据频率倾斜——某个区域的数据量要远远大于其他区域
- 数据大小倾斜——部分记录的大小远远大于平均值

解决数据倾斜的办法

- 抽样和范围分区

  可以通过对原始数据进行抽样得到的结果来预设分区边界值的方法

- 自定义分区

> 基于输出键的背景知识进行分区，例如：如果Map输出键的单词来源于一本书，且其中某几个专业词汇比较多，那么久可以自定义分区将这些专业词汇发送给一部分实例，而其他的都发送给剩余的Reduce实例。

- Combine

  > 使用Combiner减少数据倾斜，在可能的情况下目的就是聚合并精简数据的方法

- 采用Map Join

  > 尽量避免Reduce Join

常用的调优参数

资源相关的参数

以下参数是在用户自己的MR应用程序中配置就可以生效。mapred-default.xml

配置参数	参数说明
mapreduce.map.memory.mb	一个MapTask可使用的资源上限（单位:MB），默认为1024。如果MapTask实际使用的资源量超过该值，则会被强制杀死。
mapreduce.reduce.memory.mb	一个ReduceTask可使用的资源上限（单位:MB），默认为1024。如果ReduceTask实际使用的资源量超过该值，则会被强制杀死。
mapreduce.map.cpu.vcores	每个MapTask可使用的最多cpu core数目，默认值: 1
mapreduce.reduce.cpu.vcores	每个ReduceTask可使用的最多cpu core数目，默认值: 1
mapreduce.reduce.shuffle.parallelcopies	每个Reduce去Map中取数据的并行数。默认值是5

配置参数	参数说明
mapreduce.reduce.shuffle.merge.percent	Buffer中的数据达到多少比例开始写入磁盘。默认值0.66
mapreduce.reduce.shuffle.input.buffer.percent	Buffer大小占Reduce可用内存的比例。默认值0.7
mapreduce.reduce.input.buffer.percent	指定多少比例的内存用来存放Buffer中的数据，默认值是0.0

应该YARN启动之前就在服务器配置文件中才能生效 yarn-default.xml

配置参数	参数说明
yarn.scheduler.minimum-allocation-mb	给应用程序Container分配的最小内存，默认值：1024
yarn.scheduler.maximum-allocation-mb	给应用程序Container分配的最大内存，默认值：8192
yarn.scheduler.minimum-allocation-vcores	每个Container申请的最小CPU核数，默认值：1
yarn.scheduler.maximum-allocation-vcores	每个Container申请的最大CPU核数，默认值：32
yarn.nodemanager.resource.memory-mb	给Containers分配的最大物理内存，默认值：8192

Shffle性能优化的关键参数，应在YARN启动之前就配置好 mapred-defalut.xml

配置参数	参数说明
mapreduce.task.io.sort.mb	Shuffle的环形缓冲区大小，默认100m
mapreduce.map.sort.spill.percent	环形缓冲区溢出的阈值，默认80%

容错相关参数（MapReduce性能优化）

参数配置	参数说明
mapreduce.map.maxattempts	每个Map Task最大重试次数，一旦重试参数超过该值，则认为Map Task运行失败，默认值：4。
mapreduce.reduce.maxattempts	每个Reduce Task最大重试次数，一旦重试参数超过该值，则认为Map Task运行失败，默认值：4。
mapreduce.task.timeout	Task超时时间，经常需要设置的一个参数，该参数表达的意思为：如果一个Task在一定时间内没有任何进入，即不会读取新的数据，也没有输出数据，则认为该Task处于Block状态，可能是卡住了，也许永远会卡住，为了防止因为用户程序永远Block住不退出，则强制设置了一个该超时时间（单位毫秒），默认是600000。如果你的程序对每条输入数据的处理时间过长（比如会访问数据库，通过网络拉取数据等），建议将该参数调大，该参数过小常出现的错误提示是“AttemptID:attempt_14267829456721_123456_m_000224_0 Timed out after 300 secsContainer killed by the ApplicationMaster

HDFS小文件优化方法

HDFS小文件弊端

HDFS上每个文件都要在 Namenode上建立一个索引，这个索引的大小约为150byte，这样当小文件比较多的时候，就会产生很多的索引文件，一方面会大量占用 NameNode的内存空间，另一方面就是索引文件过大使得索引速度变慢

小文件解决方案

小文件的优化无非以下几种方式：

- 在数据采集的时候，就将小文件或小批数据合成大文件再上传HDFS。
- 在业务处理之前，在HDFS上使用MapReduce程序对小文件进行合并。
- 在MapReduce处理时，可采用CombineTextInputFormat提高效率。

Hadoop Archive

是一个高效地将小文件放入HDFS块中的文件存档工具，它能够将多个小文件打包成一个HAR文件，这样就减少了 Namenode的内存使用。

Sequence File

一系列的二进制 key/value组成，如果key为文件名， value为文件内容，则可以将大批小文件合并成一个大文件

CombineFileInputFormat

CombineFileInputFormat是一种新的 InputFormat，用于将多个文件合并成个单独的 Split，另外，它会考虑数据的存储位置。