



FusionInsight HD 性能调优指导手册 - MapReduce

FusionInsight HD 性能调优指导手册 - MapReduce

文档版本 01

发布日期 2016-12-16

华为技术有限公司



版权所有 © 华为技术有限公司 2016。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址：深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址：<http://www.huawei.com>

客户服务邮箱：support@huawei.com

客户服务电话：4008302118

目录

1 概述	1
1.1 模块架构模型	2
1.2 性能衡量指标	3
1.2.1 衡量指标	4
1.2.2 指标观测方法	4
2 集群服务部署规划	5
2.1 JobHistory 的部署	6
2.2 MapReduce 任务提交资源使用配置	6
2.3 MapReduce 任务 CPU 使用配置	7
2.4 队列与多租户	8
3 典型业务的调优	11
4 二次开发业务应用指导	12
5 MapReduce 优化参数配置	13
5.1 确定 Job 基线	14
5.2 Shuffle 调优	15
5.2.1 Map 阶段调优	15
5.2.2 Combiner	16
5.2.3 Copy 阶段的优化	17
5.2.4 Merge 阶段的调优	17
5.3 推测执行	18
5.4 容器可重用性提高任务的完成效率	19
5.5 Slow Start	20
6 性能瓶颈监控及调优	21
6.1 监控手段	22
6.1.1 资源监控	22
6.1.2 性能监控	22
6.2 常见性能问题及解决方案	22
6.2.1 ResourceManager 的处理能力	22
6.2.2 网络瓶颈	22
7 常见 MR 标准测试场景样例	24

1 概述

[1.1 模块架构模型](#)

[1.2 性能衡量指标](#)

1.1 模块架构模型

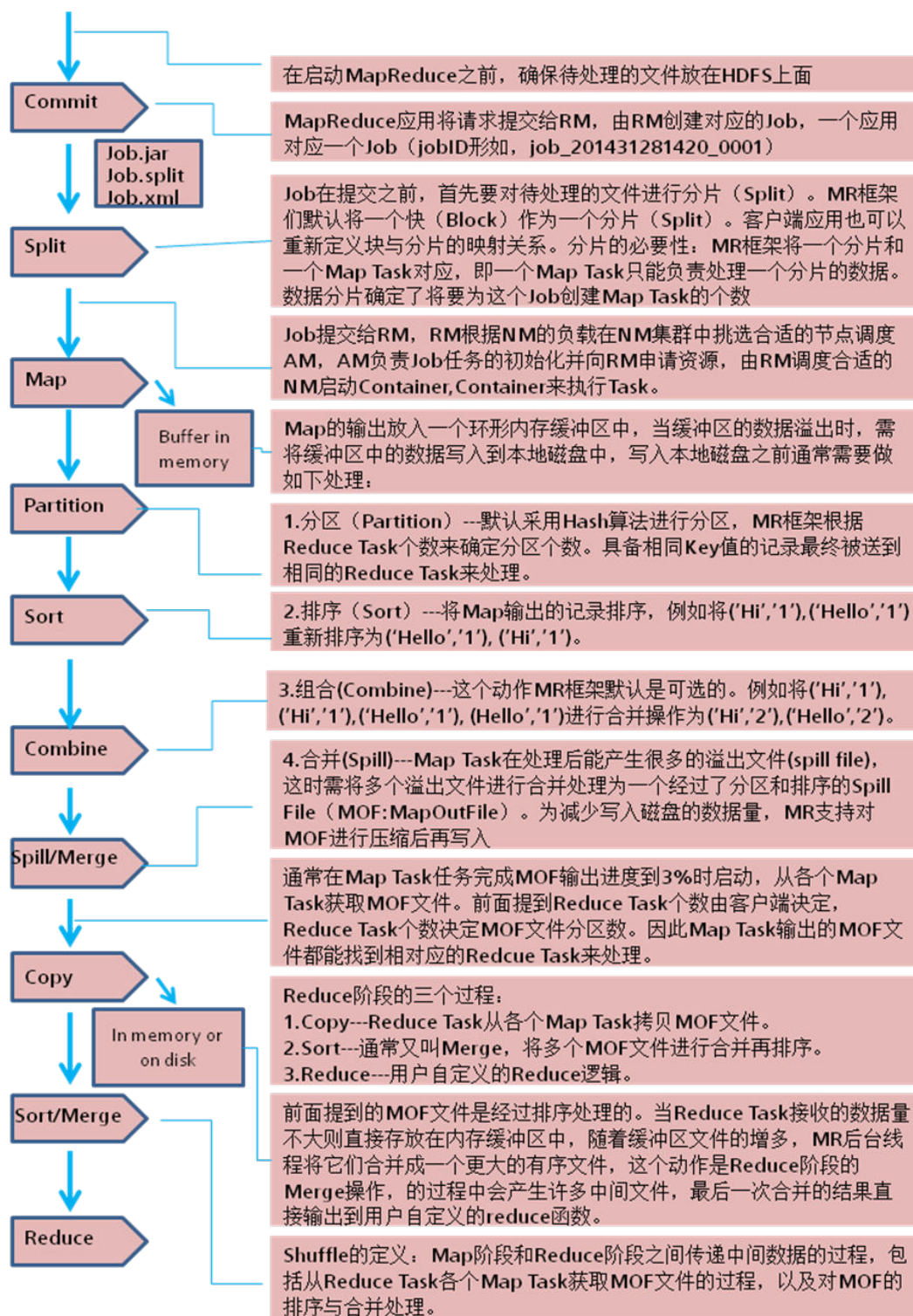
MapReduce基于Google发布的分布式计算框架MapReduce论文设计开发，用于大规模数据集（大于1TB）的并行运算，特点如下：

易于编程：程序员仅需描述做什么，具体怎么做就交由系统的执行框架处理。

良好的扩展性：可以添加机器扩展集群能力。

高容错性：通过计算迁移或数据迁移等策略提高集群的可用性与容错性。

图 1-1 MapReduce 的执行过程



1.2 性能衡量指标

1.2.1 衡量指标

系统资源利用率

吞吐量：单位时间内完成的计算任务数量

资源利用率：对分配到资源的使用率，关注节点任务分配完成后，怎么把资源使用率上升，主要关注CPU的使用率。

1.2.2 指标观测方法

- 1.资源利用率：执行计算任务，查看在不同负载情况下，cpu、内存、网络的使用率。
- 2.节点Container 使用的物理cpu和物理内存。观察监控项。

2 集群服务部署规划

- 2.1 JobHistory的部署
- 2.2 MapReduce任务提交资源使用配置
- 2.3 MapReduce任务CPU使用配置
- 2.4 队列与多租户

2.1 JobHistory 的部署

操作场景

Jobhistory管理历史的任务归集，实例资源配置内存和归集的历史任务。

操作步骤

参数名	描述
GC_OPTS 服务端参数	JVM前4个值为，-Xms512M -Xmx2G -XX:NewSize=128M -XX:MaxNewSize=128M需要根据大集群的任务做调整。 对于大集群建议调整为： 历史任务数10000和内存的对应关系 -Xms30G -Xmx30G -XX:NewSize=1G -XX:MaxNewSize=2G

mapreduce.jobhistory.joblist.cache.size 给个表

在150万历史任务下，内存需要配置30GB以上。建议用户配置20k的分页阈值，能够在10s左右显示结果；

2.2 MapReduce 任务提交资源使用配置

操作场景

yarn.nodemanager.resource.memory-mb，设置为NodeManger所在节点分配给任务使用的内存。

操作步骤

参数名	描述	
mapreduce.map.java.opts 客户端参数	map的jvms最大堆大小，当map阶段出现oom时调整此参数。此参数受到mapreduce.map.memory.mb的限制，一般为mapreduce.map.memory.mb的0.75倍。 -Xmx2048M -Djava.net.preferIPv4Stack=true 说明 客户端参数	默认为4G，运行一个map的container分配的内存，当所需内存超出此大小时，container将被kill掉。用yarn.nodemanager.resource.memory-mb除以该参数得到的值可以认为在内存限制下一个nodemanager最多并行运行的map的container数量。配置的越小，系统能支撑的MAP数越多，但是要考虑map任务的内存限制。 mapreduce.map.memory.mb=mapreduce.map.java.opts 设置的堆内存+direct buffer+unsafe的内存+256MB。
mapreduce.reduce.java.opts 客户端参数	-Xmx3276M -Djava.net.preferIPv4Stack=true reduce的子jvms最大堆大小。同map类似参数。	
mapreduce.reduce.memory.mb 客户端参数	默认为4G，需要大于reduce的jvms最大堆大小（mapreduce.reduce.java.opts）。同map类似参数。	
yarn.app.mapreduce.am.command-opts 客户端参数	传递到MapReduce ApplicationMaster的JVM启动参数。默认值-Xmx1024m .. 运行的一个大任务，map总数达到了10万的规模，最终超时失败。此任务的问题是，task数量变多时，AM管理的对象也线性增长，因此就需要更多的内存来管理。ApplicationMaster（以下简称AM）反应缓慢AM默认分配的内存堆大小是1GB。 对于大集群建议设置为-Xmx4G。	
yarn.app.mapreduce.am.resource.mb 客户端参数	AM的container的设置大小，默认1536。单位M。需要比yarn.app.mapreduce.am.command-opts参数的-Xmx更大。 对于大集群建议设置为5120。	

2.3 MapReduce 任务 CPU 使用配置


操作场景

Yarn的节点资源配置调整部分的参数yarn.nodemanager.resource.cpu-vcores设置了当前节点逻辑上可以分配的总CPU核数，以下为MapReduce任务map reduce 和 am阶段task 对应的CPU使用配置。

对于CPU密集型的任务，可以增加对应任务的cpu使用值。

操作步骤

参数名	描述
mapreduce.map.cpu.vcores 客户端参数	默认值1 每个map任务需要的CPU核数。
mapreduce.reduce.cpu.vcores 客户端参数	默认值1 每个reduce任务需要的CPU核数
yarn.app.mapreduce.am.resource.cpu-vcores 客户端参数	默认值1 每个MRApplicationMaster需要的CPU核数。 对于大集群建议设置为4。

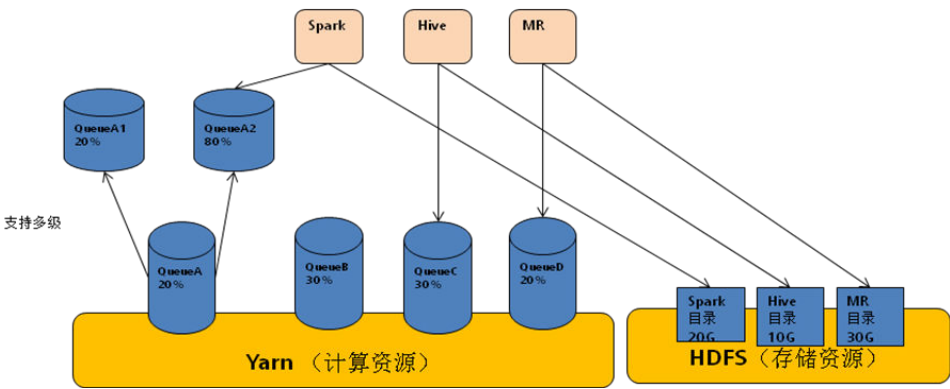
 **说明**

yarn.nodemanager.resource.cpu-vcores除以map reduce 和 am的cpu核数的设置，即为当前节点cpu限制下并发能处理的任务。注意：一般现网关于cpu的调整是很少的，因为cpu配置不合理也不会导致任务失败，只会导致速度会慢，但是当内存不够时，会出现oom导致任务失败。

单节点最大支持的MAP可以通过 $yarn.nodemanager.resource.memory-mb / mapreduce.map.memory.mb$ 和 $yarn.nodemanager.resource.cpu-vcores / \{mapreduce.map.cpu.vcores\}$ 取最小值。

2.4 队列与多租户

操作场景



租户均可从总集群中分得一定的计算资源。各租户均可设置一个专用队列，该队列可以配置一定比例的计算资源。这些队列进一步进行划分，以使租户能在各自不同的用户群之间共享其集群计算资源。同一个队列中默认使用FIFO方法来进行调度。在租户队列资源配置足够，并且比例合理的情况下，不会对具体任务的性能造成影响，以下简单对租户和队列的参数进行描述。

操作步骤

运行在大数据平台之上的服务可以分为两类：

第一类主要指Yarn和HDFS，他们是多租户存在的基础，作为租户资源的提供者和管理者。租户需要向它们申请资源才能在大数据库平台上运行作业。所以这类服务与租户的关系是作为资源提供者（Resource Provider）。

其它服务（Hive,Spark, HBase）面向不同的使用场景和业务需求，这些服务可以被多个租户共享使用（Hive,Spark）或者能被某个租户独占使用（HBase）。一个服务能不能被多个租户共享使用主要看服务运行过程中是否能对多个租户做到资源隔离。

FusionInsight 支持定义租户与服务的关系，分为共享和独占两种模式。如果一个租户与服务建立的共享模式，其它租户也可以关联和使用这个服务；如果租户与服务建立了独占模式，服务就不能被其它租户使用。

主要分为以下几个步骤：

添加租户

根据业务需求规划租户的名称，不得与当前集群中已有的角色、HDFS目录或者Yarn队列重名。规划当前租户可分配的资源，确保每一级别父租户下，直接子租户的资源百分比之和不能超过100%。

1. 在FusionInsight Manager页面，单击“租户管理”。
2. 单击“添加租户”，打开添加租户的配置页面，参见下图和表格内容为租户配置属性。

表 2-1 租户配置参数

参数名	描述
“名称”	指定当前租户的名称，长度为3到20，可包含数字、字母和下划线。
“类型”	可选参数值为“叶子租户”，当选中时表示当前租户为叶子租户，无法再添加子租户。
“动态资源”	为当前租户选择动态计算资源。系统将自动在Yarn中以租户名称创建任务队列。
“容量”	配置当前租户计算资源使用的资源百分比。
“储存资源”	为当前租户选择存储资源。系统将自动在HDFS根目录中以租户名称创建文件夹。
“存储空间配额”	配置当前租户使用的HDFS存储空间配额。最大值可填写“8796093022208”。此参数值表示租户可使用的HDFS存储空间上限，不代表一定使用了这么多空间。如果参数值大于HDFS物理磁盘大小，实际最多使用全部的HDFS物理磁盘空间。
“服务”	配置当前租户关联使用的其他服务资源，支持HBase。单击“关联服务”，在“服务”选择“HBase”。在“关联类型”选择“独立”表示独占服务资源，选择“共享”表示共享服务资源。
“描述”	配置当前租户的描述信息。

修改队列配置

3. 在租户管理中修改指定租户的队列配置

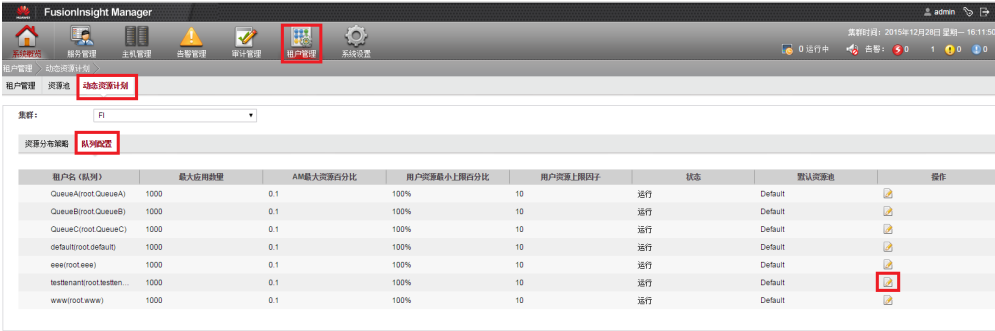


表 2-2 队列配置参数

参数名	描述
“最大应用数量”	表示最大应用程序数量。
“AM最大资源百分比”	表示集群中可用于运行application master的最大资源占比。
“用户资源最小上限百分比”	表示用户使用的最小资源上限百分比。
“用户资源上限因子”	表示用户使用的最大资源限制因子，与当前租户在集群中实际资源百分比相乘，可计算出用户使用的最大资源百分比。
“状态”	表示资源计划当前的状态，“运行”为运行状态，“停止”为停止状态。
“默认资源池”	表示队列使用的资源池。默认为“Default”。

3 典型业务的调优

上层组件业务一般分为可以分为IO密集型业务，计算密集型业务，低延迟业务，高吞吐量业务。

4 二次开发业务应用指导

MR 的应用开发所涉及的主要接口如下表所示。

参数名	描述
setInputFormatClass(Class< extends InputFormat> cls)	指定MapReduce作业的InputFormat类，默认为TextInputFormat。也可以在“mapred-site.xml”中配置“mapreduce.job.inputformat.class”项。该设置用来指定处理不同格式的数据时需要的InputFormat类，用来读取数据，切分数据块。
setOutputFormat(Class<? extends OutputFormat> theClass)	指定MapReduce作业的OutputFormat类，默认为TextOutputFormat。也可以在“mapred-site.xml”中配置“mapred.output.format.class”项，指定输出结果的数据格式。例如默认的TextOutputFormat把每条key，value记录写为文本行。通常场景不配置特定的OutputFormat。
setPartitionerClass(Class<? extends Partitioner> theClass)	指定MapReduce作业的Partitioner类。也可以在“mapred-site.xml”中配置“mapred.partitioner.class”项。该方法用来分配map的输出结果到哪个reduce类，默认使用HashPartitioner，均匀分配map的每条键值对记录。例如在hbase应用中，不同的键值对应的region不同，这就需要设定特殊的partitioner类分配map的输出结果。
setMapOutputCompressorClass(Class<? extends CompressionCodec> codecClass)	指定MapReduce作业的map任务的输出结果压缩类，默认不使用压缩。也可以在“mapred-site.xml”中配置“mapreduce.map.output.compress”和“mapreduce.map.output.compress.codec”项。当map的输出数据大，减少网络压力，使用压缩传输中间数据。

5 MapReduce 优化参数配置

- 5.1 确定Job基线
- 5.2 Shuffle调优
- 5.3 推测执行
- 5.4 容器可重用性提高任务的完成效率
- 5.5 Slow Start

5.1 确定 Job 基线

操作场景

确定Job基线是调优的基础，一切调优项效果的检查，都是通过和基线数据做对比来获得。

- 充分利用集群资源
- reduce阶段尽量放在一轮
- 每个task的执行时间要合理

操作步骤

- **原则一：充分利用集群资源**

Job运行时，会让所有的节点都有任务处理，且处于繁忙状态，这样才能保证资源充分利用，任务的并发度达到最大。可以通过调整处理的数据量大小，以及调整map和reduce个数来实现。

Reduce个数的控制使用“mapreduce.job.reduces”。

Map个数取决于使用了哪种InputFormat，以及待处理的数据文件是否可分割。默认的TextFileInputFormat将根据block的个数来分配map数(一个block一个map)。通过如下配置参数进行调整。

参数名	描述
mapreduce.input.fileinputformat.split.maxsize 客户端参数	用户定义的分片大小的设置及每个文件block大小的设置，可以计算分片的大小。计算公式如下： $splitSize = \text{Math.max}(\text{minSize}, \text{Math.min}(\text{maxSize}, \text{blockSize}))$ 如果maxSize设置大于blockSize，那么每个block就是一个分片，否则就会将一个block文件分隔为多个分片，如果block中剩下的一小段数据量小于splitSize，还是认为它是独立的分片。
mapreduce.input.fileinputformat.split.minsize 客户端参数	可以设置数据分片的数据最小值。
CombineTextInputFormat	解决大量小文件map的问题,将多个小文件合并为一个切片,减少提交任务时的任务数。 <code>Job.setInputFormatClass(CombineTextInputFormat.class)</code>

- **原则二：控制reduce阶段在一轮中完成**

避免以下两种场景：

- 大部分的reduce在第一轮运行完后，剩下唯一一个reduce继续运行。
这种情况下，这个reduce的执行时间将极大影响这个job的运行时间。因此需要将reduce个数减少。

- 所有的map运行完后，只有个别节点有reduce在运行。

这时候集群资源没有得到充分利用，需要增加reduce的个数以便每个节点都有任务处理。

- **原则三：每个task的执行时间要合理**

如果一个job，每个map或reduce的执行时间只有几秒钟，就意味着这个job的大部分时间都消耗在task的调度和进程启停上了，因此需要增加每个task处理的数据大小。建议一个task处理时间为1分钟。

5.2 Shuffle 调优

5.2.1 Map 阶段调优

操作场景

Map运行阶段分为:Read、Map、Collect、Spill、Merge五个阶段。

Map任务执行会产生中间数据,但这些中间结果并没有直接IO到磁盘上,而是先存储在缓存(buffer)中,并在缓存中进行一些预排序来优化整个map的性能,存储map中间数据的缓存默认大小为512M, 由“mapreduce.task.io.sort.mb”参数指定。

“mapreduce.task.io.sort.mb”大小可以根据需要调整。当map任务产生了非常大的中间数据时可以适当调大该参数,使缓存能容纳更多的map中间数据,而不至于大频率的IO磁盘,当系统性能的瓶颈在磁盘IO的速度上,可以适当的调大此参数来减少频繁的IO带来的性能障碍。

由于map任务运行时中间结果首先存储在缓存中,默认当缓存的使用量达到80% (map “reduce.map.sort.spill.percent” 默认0.8)的时候就开始写入磁盘,这个过程叫做spill(也叫溢出),进行spill的缓存大小可以通过“mapreduce.map.sort.spill.percent”参数调整,这个参数可以影响spill的频率。进而可以影响IO的频率。

当map任务计算成功完成之后,如果map任务有输出,则会产生多个spill。接下来map必须将这些spill进行合并,这个过程叫做merge, merge过程是并行处理spill的,每次并行多少个spill是由参数“mapreduce.task.io.sort.factor”指定的默认为64个。但是当spill的数量非常大的时候, merge一次并行运行的spill仍然为64个,这样仍然会频繁的IO处理,因此适当的调大每次并行处理的spill数有利于减少merge数因此可以影响map的性能。

如何判断该参数配小了, Job的Counters中, spill的记录与Map的中间结果record数量是否一致。Spill record大于Map的输出时,说明发生了多次spill,即写磁盘操作。

“mapreduce.task.io.sort.mb”的大小, 需要考虑几个因素:

- 分配给jvm的堆大小
- 用户的Map函数中的业务代码本身需要多少内存
- Map中间结果多大

Jvm堆大小是在用户执行任务时候指定的 (mapreduce.map.java.opts)。Map的中间结果大小可以通过Job运行结束后的JobCounters中找到。同样, 用户的业务代码使用了多少内存可以在JobCounters中找到Total committed heap usage (bytes)来估算 (约等于Total committed heap usage减去Map buffer的大小)。

计算出一个合理的buffer大小, 可以用一个简单的公式:

Buffer的大小=jvm堆大小-业务代码占用量-200M

预留200M的意思是给其他可能的内存占用提供空间，避免jvm频繁gc。

操作步骤

参数名	描述
mapreduce.task.io.sort.mb 客户端参数	I/O排序内存缓冲，默认512M。
mapreduce.task.io.sort.factor 客户端参数	I/O排序因子，默认为64。文件排序时可一次合并的流。
mapreduce.map.sort.spill.percent 客户端参数	默认值0.8，磁盘IO是主要瓶颈，合理配置“mapreduce.task.io.sort.mb”可以使溢出至磁盘的内容最小化。如果map在内存中sort的结果达到一个特定的值，就会被spill进入硬盘。具体这个值是等于mapreduce.task.io.sort.mb * mapreduce.map.sort.spill.percent.具体参见操作场景描述。
mapreduce.reduce.shuffle.parallelcopies 客户端参数	默认值10，洗牌期间并行传输的默认数量,调大需要考虑造成网络的拥堵。
mapreduce.shuffle.max.connections 服务端参数	最大洗牌连接。

5.2.2 Combiner

操作场景

在Map阶段，有一个可选过程，将同一个key值的中间结果合并，叫做combiner。一般将reduce类设置为combiner即可。通过combine，一般情况下可以显著减少Map输出的中间结果，从而减少shuffle过程的网络带宽占用。可通过如下接口为一个任务设置Combiner类。

操作步骤

类名	描述
org.apache.hadoop.mapreduce.Job	public void setCombinerClass(Class<? extends Reducer> cls) 为Job设置一个combiner类

5.2.3 Copy 阶段的优化

操作场景

对Map的中间结果进行压缩，当数据量大时，会显著减少网络传输的数据量，但是也因为多了压缩和解压，带来了更多的CPU消耗。因此需要做好权衡。当任务属于网络瓶颈类型时，压缩Map中间结果效果明显。

shuffle 阶段为reduce 全面拷贝map任务成功结束之后产生的中间结果,如果上面map任务采用了压缩的方式,那么reduce 将map任务中间结果拷贝过来后首先进行解压缩,这一切是在reduce的缓存中做的,当然也会占用一部分cpu。为了优化reduce的执行时间,reduce也不是等到所有的map数据都拷贝过来的时候才开始运行reduce任务，而是当job执行完第一个map任务时开始运行的。

reduce 在shuffle阶段 实际上是从不同的并且已经完成的map上去下载属于自己的数据，由于map任务数很多,所有这个copy过程是并行的,既同时有许多个reduce取拷贝map，这个并行的线程是通过“mapreduce.reduce.shuffle.parallelcopies”参数指定，默认为10个，也就是说无论map的任务数是多少个，默认情况下一次只能有10个reduce的线程去拷贝map任务的执行结果。所以当map任务数很多的情况下可以适当的调整该参数，这样可以让reduce快速的获得运行数据来完成任务。

操作步骤

参数名	描述
mapreduce.map.output.compress 客户端参数	默认为ture开启
mapreduce.map.output.compress. codec 客户端参数	默认为org.apache.hadoop.io.compress.SnappyCodec
mapreduce.reduce.shuffle.paralle lcopies 客户端参数	默认为10，增大参数配置reduce拷贝map文件的线程数，带宽较大时可以增大此配置

5.2.4 Merge 阶段的调优

操作场景

通过调整如下参数减少reduce写磁盘的次数。

操作步骤

参数名	描述
mapreduce.reduce.merge.inmem.threshold	允许多少个文件同时存在reduce内存里。当达到这个阈值时，reduce就会触发mergeAndSpill，将数据写到硬盘上。 默认值1000
mapreduce.reduce.shuffle.merge.percent	当reduce中存放map中间结果的buffer使用达到多少百分比时，会触发merge操作。 默认值0.66
mapreduce.reduce.shuffle.input.buffer.percent	允许map中间结果占用reduce堆大小的百分比。 默认值0.70
mapreduce.reduce.input.buffer.percent	当开始执行reduce函数时，允许map文件占reduce堆大小的百分比。 当map文件比较小时，可以将这个值设置成1.0，这样可以避免reduce将拷贝过来的map中间结果写磁盘。 默认值0

5.3 推测执行

操作场景

Hadoop发现一个任务运行比预期慢的时候，它会尽量检测，并启动另一个相同的任务作为备份，即“推测执行”(speculative execution)。

推测执行是一种优化措施，并不能使作业运行更可靠。默认启用，但可以单独为map/reduce任务设置，“mapred.map.tasks.speculative.execution”和“mapred.reduce.tasks.speculative.execution”。开启此功能会减少整个吞吐量，在集群中倾向于关闭此选项，而让用户根据个别作业需要开启该功能。

当集群规模很大时（如几百上千台节点的集群），总体软硬件故障的概率就变大了，并且会因此延长整个任务的执行时间（跑完的任务都在等出问题的机器跑结束）。推测执行通过将任务分给多台机器跑，取先运行完的那个，会很好的解决这个问题。由于会额外占用资源，对于小集群，不建议使用此功能。

操作步骤

参数名	描述
mapreduce.map.speculative 客户端参数	是否开启map的推测执行。默认为false表示关闭。

参数名	描述
mapreduce.reduce.speculative 客户端参数	是否开启reduce的推测执行。默认为false表示关闭。

5.4 容器可重用性提高任务的完成效率

操作场景

容器可重用性可以提高任务完成的速度。一些优势如下所示：

- 容器可重用性的特性是可作用于一些Map以及Reduce任务使之能快速的完成从而缩短HBase数据批量加载的时间。使用容器重用场景是map或reduce task比较短120秒以内，同时单个task处理的数据量不大，不超过256MB。
- 减少容器调度的时间和初始化的时间。

一旦MapReduce作业被提交。它将分发至Map和Reduce任务中。然后应用管理器（以下简称AM）将执行如下操作。

 - a. AM将向资源管理器（RM）申请容器去执行任务。所有容器将一起完成这个请求。当RM指定了容器，AM将会联系节点管理器（NM）去执行容器。
 - b. 当容器启动完毕，NM将从AM拉取任务。
 - c. 当一个任务完成了，运行该任务的容器不会被立即终止。该容器将会试着向AM拉取下一个任务。如果AM能分配一个新任务给这个容器，该容器会自我清空以及初始化本身，以适用于新任务。
 - d. 如果AM不能分配新任务，容器会请求终止自己的运行。

操作步骤

参数名	描述
mapreduce.container.reuse.enabled	该配置项打开则容器可重用，反之容器不可重用。 false
mapreduce.container.reuse.enforce.strict-locality	当启用了强制本地执行策略，该配置将告知是否遵循严格的数据本地化。如果启用，重用容器只能被分配至本节点上的任务。 false

使用约束

当strict-locality为false时，容器重用的降低了数据本地性。

容器的重用将受限于job中任务的总数。容器不会被不同的job共享。

容器只能被尝试成功的任务重用，不能被尝试失败的任务重用。

任务应该设计成，当一个任务完成后堆栈里将不会存留任何对象。这对于容器重用中的数据一致性以及内存的优化非常重要。如果容器重用是激活状态，job将不会释放容

器给资源管理器，除非所有属于该job的任务都完成了，这种情况会影响到公平调度的原则。

当强制本地执行策略被激活，只有本地节点的任务可以分配给可以重用的容器。

5.5 Slow Start

操作场景

Slow Start特性指定Map任务完成度为多少时Reduce任务可以启动，过早启动Reduce任务会导致资源占用，影响任务运行效率，但适当的提早启动Reduce任务会提高Shuffle阶段的资源利用率，提高任务运行效率。例如：某集群可启动10个Map任务，MapReduce作业共15个Map任务，那么在一轮Map任务执行完成后只剩5个Map任务，集群还有剩余资源，在这种场景下，配置Slow Start参数值小于1，比如0.8，则Reduce就可以利用集群剩余资源。

操作步骤

参数名	描述
mapreduce.job.reduce.slowstart.completedmaps 客户端参数	当多少占比的Map执行完后开始执行Reduce。默认100%的Map跑完后开始起Reduce。 1

6 性能瓶颈监控及调优

6.1 监控手段

6.2 常见性能问题及解决方案

6.1 监控手段

6.1.1 资源监控

MapReduce 程序的运行监控

Garbage Collection (GC) Time of ResourceManager ResourceManager的GC时长

周期（30s）内的GC时长值小于1s，否则就要根据ResourceManager部署章节建议部署。

6.1.2 性能监控

6.2 常见性能问题及解决方案

6.2.1 ResourceManager 的处理能力

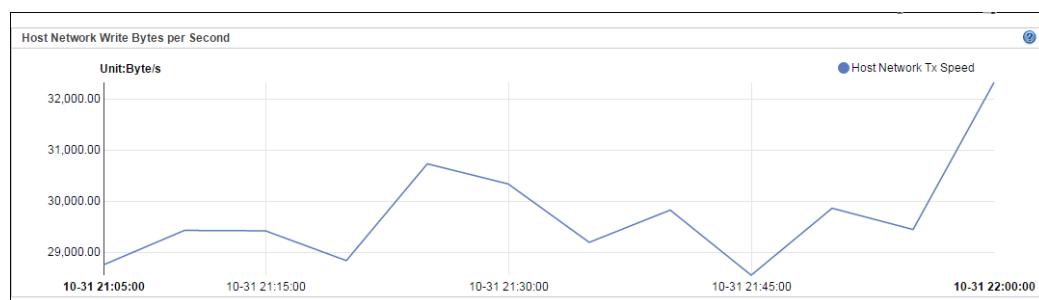
如果容量监控或者性能监控Average RPC Queuing Time和Average RPC Processing Time比较慢，当前发现的调度慢的处理基本和内存配置的不合理有关系，参考**ResourceManger的部署**章节部分进行调整。

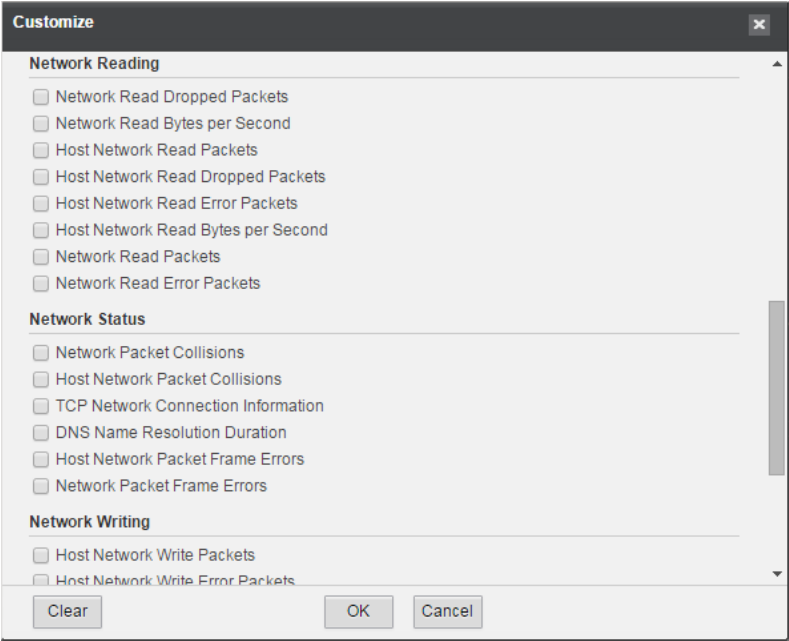
6.2.2 网络瓶颈

节点写入监控

查询节点的网络监控是否达到网卡的吞吐能力。也需要关注节点对应的汇聚交换机的处理能力是否通过以下监控节点网络的处理情况。

图 6-1 Host Network Write Bytes per Second





7 常见 MR 标准测试场景样例

操作场景

以下针对hadoop的标准benchmark的用例测试描述。

操作步骤

用例	描述
MR bench	<code>hadoop jar hadoop-mapreduce-client-jobclient-**-tests.jar mrbench \ -Dmapreduce.reduce.merge.memtomem.enabled=true - Dmapreduce.job.reduce.slowstart.completedmaps=0.0f \ -numRuns 50 -inputLines 8000000 -maps 100 -reduces 50</code>
teragen	<code>hadoop jar hadoop-mapreduce-examples.jar teragen \ -Ddfs.block.size=268435456 \ -Dmapreduce.job.maps=1000 10000000000</code>
terasort	<code>hadoop jar hadoop-mapreduce-examples.jar terasort \ -Dmapreduce.job.reduce.slowstart.completedmaps=0.1 - Dmapreduce.job.reduces=100 \ -Ddfs.client-write-packet-size=262144 - Dmapreduce.map.output.compress=true - Dmapreduce.map.output.compress.codec=org.apache.hado op.io.compress.SnappyCodec \ /terasort-input /terasort-output</code>

用例	描述
wordcount	<code>hadoop jar hadoop-mapreduce-examples.jar wordcount -Dmapreduce.job.reduce.slowstart.completedmaps=0.05f \ -Ddfs.client.write.packet.size=262144 -Dmapreduce.map.output.compress=true -Dmapreduce.map.output.compress.codec=org.apache.hadoop.io.compress.SnappyCodec \ -Dmapreduce.reduce.merge.memtomem.enabled=true \ -Ddfs.datanode.max.transfer.threads=8192 \ -Dmapreduce.job.reduces=70 /wordcount_input /wordcount_output</code>
randomtextwriter	<code>hadoop jar hadoop-mapreduce-examples.jar randomtextwriter \ -Dmapreduce.randomtextwriter.totalbytes=1099511627776 \ /wordcount_input</code>