**一、概述**

影响hadoop集群性能的因素主要有：   
（1）磁盘IO速率  
（2）网络带宽   
（3）集群节点硬件配置（包括内存和cpu）  
（4）集群参数配置和节点系统设置  
因此，一个hadoop集群的性能调优不仅仅是hadoop参数的调优，还包括硬件配置、网络环境等等因素。我们需要做的是在给定的一个硬件网络条件下，通过系统和集群配置的优化，达到性能的提升。

**二、OS调优**

这部分基本上在集群部署时候就已经默认调优。

**2.1、关闭swap分区**

内存不足时会写swap分区，可能会对性能产生影响，所以一般情况下要禁用swap分区。  
swappiness=0的时候表示最大限度使用物理内存;swappiness＝100的时候表示积极的使用swap分区。从性能角度考虑，应尽量避免使用swap分区，故关闭swapiness。 关闭方法：

|  |
| --- |
| $ vi /etc/sysctl.conf vm.swappiness=0 $ sysctl -p |

或者：

|  |
| --- |
| cat /proc/swaps  Filename    Type     Size       Used      Priority  /dev/sda2   partition  4194296     141264      -1  swapoff /dev/sda2 |

其中/dev/sda2为cat /proc/swaps显示的swap分区实际的设备号。  
检查当前值：

|  |
| --- |
| cat /proc/sys/vm/swappiness |

**2.2、系统limits**

修改文件Handler数：

|  |
| --- |
| $ vi /etc/security/limits.conf |

增加一行：

|  |
| --- |
| \*  -  nofile  65535 |

**2.3、关闭selinux**

修改/etc/selinux/config里SELINUX=disabled。

**2.4、永久关闭Transparent Huge Page**

低版本的CentOS中，如果开启了Transparent Huge Page，会造成内存碎片整理的时候占用CPU资源，导致JAVA GC非常慢，影响正在运行的hadoop任务的性能。 因此，需要采用如下的命令关闭：

|  |
| --- |
| echo never > /sys/kernel/mm/redhat\_transparent\_hugepage/enabled echo never > /sys/kernel/mm/redhat\_transparent\_hugepage/defrag |

上述命令只能临时关闭，机器重启将会失效。 执行如下命令永久关闭：

|  |
| --- |
| echo 'echo never > /sys/kernel/mm/redhat\_transparent\_hugepage/enabled' >> /etc/rc.local echo 'echo never > /sys/kernel/mm/redhat\_transparent\_hugepage/defrag' >> /etc/rc.local |

**2.5、磁盘预读readahead buffer**

调整linux文件系统中预读缓冲区的大小，可以明显提高顺序读文件的性能。 默认buffer大小为256 sectors，可以增大为1024或者2048 sectors（注意，并不是越大越好）。 可使用blockdev命令进行调整:

|  |
| --- |
| $ blockdev --getfra /dev/sdb256 $ blockdev --setfra 2048 /dev/sdb $ echo 'blockdev --setfra 2048 /dev/sdb' >> /etc/rc.local $ blockdev --getfra /dev/sdb2048 |

也可以：

|  |
| --- |
| echo 16384 > /sys/block/sda/queue/read\_ahead\_kb |

其中/sda为指定磁盘

**2.6、打开文件描述符最大数量**

对于大多数hadoop节点，改变打开文件描述符的最大值是有帮助的，默认一个进程只能打开1024个文件描述符实在是有点低，会导致hadoop任务失败。 修改为更大的值:

|  |
| --- |
| $ cat /proc/sys/fs/file-max 32832 $ vi /etc/sysctl.conf fs.file-max = 13131465 $ sysctl -p |

**2.7、确保/etc/resolv.conf配置正确**

        通常情况下，hadoop集群的机器不需要连接外网，因此不用配置DNS。 如果需要连接外网，则要配置**可以连得上的DNS**。 如果配置了DNS，但是却连不上，会造成连接变慢的问题，集群内部调用getAddress的时候，会去访问DNS，会造成30s左右的超时，超时之后才能正常连接。

**2.8、dns缓存**

hadoop节点，有很多DNS查找，尤其是HBase & distcp，开启DNS缓存服务，可以减小请求延迟。 开启DNS缓存服务：

|  |
| --- |
| $ chkconfig --level 345 nscd on $ service nscd start |

查看运行状态:

|  |
| --- |
| $ nscd -g |

**2.9、TCP socket的accept队列缓存大小**

          参数net.core.somaxconn限定了TCP socket的accept队列最多允许缓存多少报文，这个参数限制了listen()函数中的第二个参数的最大值。默认值是128。提高net.core.somaxconn，可以支持来自NameNode和JobTracker的大量爆发性的HTTP请求。对繁忙的服务器,增加该值有助于网络性能。 默认的 128 太小，大多数环境这个值建议增加到 1024 或者更多：

|  |
| --- |
| echo 1024 > /proc/sys/net/core/somaxconn |

**三、Hadoop集群调优**

集群调优有两个配置入口：

* + 服务器端配置：在Hcontrol选择HDFS(Yarn/MapReduce）-->Configs,然后在搜索框内搜索参数名称进行修改，若搜索不到，则需要新增参数进行配置。
  + 客户端配置：直接在客户端修改相应的配置文件。

**3.1、HDFS线程数**

| **参数** | **默认值** | **描述** |
| --- | --- | --- |
| dfs.datanode.max.transfer.threads | 4096 | 指定datanode用来数据传输的最大线程数。 推荐值：4096 |
| ipc.server.listen.queue.size | 128 | server接收client连接的监听队列长度。 推荐值：100\*dfs.namenode.handler.count |
| dfs.namenode.handler.count | 10 | Namenode上起的服务器线程数。 推荐值：20 \* log2(DN节点数) |
| dfs.datanode.handler.count | 10 | Datanode上起的服务器线程数。 推荐值：64 |

**说明**：ipc.server.listen.queue.size这个参数表示ipc call queue的队列长度，会受到linux内核参数net.core.somaxconn的制约。 所以如果要增大该参数，同时也要调整内核参数net.core.somaxconn，否则配置是不会生效的。

**3.2、调度框架相关的参数**

* 服务端yarn-site参数

| **参数** | **默认值** | **描述** |
| --- | --- | --- |
| yarn.scheduler.fair.assignmultiple | False | NM一次心跳可以调度多个container。 推荐值：true |
| yarn.resourcemanager.nodemanagers.heartbeat-interval-ms | 1000 | NM向RM心跳的间隔时间(单位：ms)。 测试集群推荐值：100；生产集群推荐默认值 |

* 客户端mapred-site参数

| **参数** | **默认值** | **描述** |
| --- | --- | --- |
| mapreduce.client.progressmonitor.pollinterval | 1000 | 客户端向RM请求作业状态间隔时间(单位：ms)。 推荐值：1000 |
| [yarn.app.mapreduce.am](http://yarn.app.mapreduce.am/).scheduler.heartbeat.interval-ms | 1000 | AM向RM心跳的间隔时间(单位：ms)。 推荐值：1000 |

**说明**：上述参数适合作业本身开销少而主要开销在调度框架上的case，获取到更好的benchmark结果。通过调整组件间通信间隔时间来减少作业调度过程中等待的时间。但是调整这些参数会使组件更频繁的处理请求，从而增加系统压力。

**3.3、提升ShuffleHandler和IFile Reader性能**

MapReduce shuffle handler和IFile reader用Linux原生调用(posix\_fadvise(2) 和 sync\_data\_range）

可以开启shuffle readahead提升shuffle性能。可以让Nodemanager在map输出通过socket发送给reduce之前提前获取。

| **参数** | **默认值** | **描述** |
| --- | --- | --- |
| mapreduce.shuffle.manage.os.cache | true | 推荐开启 |
| mapreduce.shuffle.readahead.bytes | 4M | 可以适当调大，专家属性 |
| mapred.tasktracker.shuffle.fadvise | true | 推荐开启 |
| mapred.tasktracker.shuffle.readahead.bytes | 4M | 可以适当调大，专家属性 |
| mapreduce.ifile.readahead | true | 推荐开启，可以提升merge操作性能 |
| mapreduce.ifile.readahead.bytes | 4M | 专家属性， |

**3.4、开启Nativetask**

| **参数** | **默认值** | **描述** |
| --- | --- | --- |
| mapreduce.job.map.output.collector.class | org.apache.hadoop.mapred.MapTask$MapOutputBuffer | nativetask是把MapOutputCollector通过C++实现，从而达到提升性能的作用。 推荐值：org.apache.hadoop.mapred.nativetask.NativeMapOutputCollectorDelegator |

**说明**：该参数为客户端参数，而且只对部分作业有效，通常在客户端配置，不要在hcontrol中配置成全局变量，否则执行作业可能会报错。 适用场景：

* + 1）有reduce的MR程序
  + 2）非自定义comparator，即mapreduce.job.output.key.comparator.class必须为空
  + 3）map的sort必须是QuickSort，即 map.sort.class必须是org.apache.hadoop.util.QuickSort
  + 4）不支持ssl shuffle，即mapreduce.shuffle.ssl.enabled必须是false
  + 5）map输出的key类型必须是以下几种

必须同时满足以上5点。

**3.5、超时相关参数**

| **参数** | **默认值** | **描述** |
| --- | --- | --- |
| dfs.client.socket-timeout | 60000 | Client socket超时时间（单位：ms） 推荐值：120000 |
| mapreduce.task.timeout | 600000 | Task超时时长（单位：ms） 对于大作业推荐值：1200000,一般默认600s就行 |

**3.6、开启map中间结果压缩**

压缩可以节省磁盘和网络的IO,提高作业性能。Gzip/Snappy/Lzo/Bzip2都是常用的压缩格式，根据需要选用。

* + 服务端core-site参数：

| **参数** | **默认值** | **描述** |
| --- | --- | --- |
| io.compression.codecs | - | 一些用逗号隔开的压缩类的列表，用来对数据压缩/解压缩。 推荐值：[org.apache.hadoop.io](http://org.apache.hadoop.io/).compress.GzipCodec,[org.apache.hadoop.io](http://org.apache.hadoop.io/).compress.DefaultCodec,[org.apache.hadoop.io](http://org.apache.hadoop.io/).compress.SnappyCodec |

**说明**：也可以加入[org.apache.hadoop.io](http://org.apache.hadoop.io/).compress.BZip2Codec和[org.apache.hadoop.io](http://org.apache.hadoop.io/).compress.Lz4Codec。此参数表示集群支持的压缩方式。 在客户端根据不同的作业场景配置参数：

* + 客户端mapred-site参数

| **参数** | **默认值** | **描述** |
| --- | --- | --- |
| mapreduce.map.output.compress | false | Map的输出在传送之前是否启用压缩 推荐值：true |
| mapreduce.map.output.compress.codec | [org.apache.hadoop.io](http://org.apache.hadoop.io/).compress.DefaultCodec | Map输出的压缩方式。 推荐值：[org.apache.hadoop.io](http://org.apache.hadoop.io/).compress.SnappyCodec |

**说明**：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。  
一般采用snappy压缩，也可以根据不同场景使用[org.apache.hadoop.io](http://org.apache.hadoop.io/).compress.DefaultCodec、[org.apache.hadoop.io](http://org.apache.hadoop.io/).compress.GzipCodec、[org.apache.hadoop.io](http://org.apache.hadoop.io/).compress.BZip2Codec、[org.apache.hadoop.io](http://org.apache.hadoop.io/).compress.Lz4Codec 其中DefaultCodec是同时使用了LZ77算法与哈夫曼编码（Huffman Coding）的一个无损数据压缩算法。  
另外，hadoop是支持lzo压缩的，但是需要自行安装lzo库和hadoop-lzo工具。

**3.7、使用snapshot减少distcp的copy list**

版本：bc1.3.0 开始 适用场景：HDFS集群备份、定时get或put数据等。 使用方法：

以从src集群拷贝目录/dist到target集群为例， 假设target集群目录 hdfs://10.133.17.28:8020/dist\_backup：  
（1）src集群执行：

|  |
| --- |
| hdfs dfsadmin -allowSnapshot /dist |

在src集群开启/dist目录的snapshot，默认snapshot功能是关闭的。

（2）src集群执行：

|  |
| --- |
| hdfs dfs -createSnapshot /dist s1 |

该步骤为src集群的/dist目录创建了一个snapshot，名字为s1。

（3）src集群执行：

|  |
| --- |
| hadoop distcp /dist hdfs://10.133.17.28:8020/dist\_backup |

该步骤为初次拷贝，使用默认的distcp命令。 注意：初次拷贝时，target集群<hdfs://10.133.17.28:8020/dist>*backup目录应该不存在， 如果该目录存在，会导致拷贝后目录为*[*hdfs://10.133.17.28:8020/dist*](hdfs://10.133.17.28:8020/dist)backup/dist/。

（4）distcp执行完成后，target集群执行：

|  |
| --- |
| hdfs dfs -createSnapshot /dist\_backup s1 |

该步骤为target集群目标目录/dist\_backup创建一个与src集群/dist目录命名相同的snapshot s1。

（5）对src集群的/dist目录做一些修改，然后执行

|  |
| --- |
| hdfs dfs -createSnapshot /dist s2 |

该步骤在src集群完成，创建新的snapshot s2表示目录做了修改。

（6）src集群执行：

|  |
| --- |
| hadoop distcp -update -diff s1 s2 /dist hdfs://10.133.17.28:8020/dist\_back |

该步骤为使用提升性能的distcp进行拷贝，比较snapshot内容，仅对不同的地方做修改，并且提升了创建copy list性能。

（7）distcp执行完成后，target集群执行：

|  |
| --- |
| hdfs dfs -createSnapshot /dist\_backup s2 |

该步骤为target集群目标目录/dist\_backup创建一个与src集群/dist目录命名相同的snapshot s2。

（8）以后只要重复步骤5-7即可，注意每次创建的snapshot命名。

**3.8、HDFS短路读**

推荐值：/var/lib/hadoop-hdfs/dn\_socket

|  |  |  |
| --- | --- | --- |
| **参数** | **默认值** | **描述** |
| dfs.client.read.shortcircuit | False | 最短路径读取。 推荐值：true |
| dfs.domain.socket.path | - | unix domian socket存储路径 |

**说明**：当client和datanode在一起时，数据以码流的方式从datanode节点传到客户端，这种方式比较低效，并且导致大量的上下文切换，相反，客户端可以选择最短路径读取，直接从磁盘读取数据。开启该参数后，Client读操作会通过unix domian socket与DN建立IPC连接，而不走TCP协议，可以提升读性能。

**3.9、资源配置调优**

**3.9.1、nodemanager节点资源配置**

可用内存

除了分配给操作系统、保证其他服务运行的内存外，剩余的资源应尽量分配给YARN。通过如下配置参数进行调整。

|  |  |  |
| --- | --- | --- |
| **参数** | **默认值** | **描述** |
| yarn.nodemanager.resource.memory-mb | 8192 | 设置可分配给容器的物理内存数 量（单位：MB），推荐节点内存总量的80%。 |

CPU虚拟核数

| **参数** | **默认值** | **描述** |
| --- | --- | --- |
| yarn.nodemanager.resource.cpu-vcores | 8 | 表示该节点上YARN可使用的虚拟 CPU个数，建议将此配置设定在物理核数的1.5~2倍之间。如果上层计算应用对cpu的计算能力要求不高，可以配置为2倍的物理cpu。 |

**3.9.2、作业资源配置**

| **参数** | **默认值** | **描述** |
| --- | --- | --- |
| mapreduce.map.memory.mb | 1024 | 每个map任务申请内存资源大小（单位：MB）。 推荐值：4096 |
| mapreduce.reduce.memory.mb | 1024 | 每个reduce任务申请内存资源大小（单位：MB）。 推荐值：4096 |
| mapreduce.map.java.opts | - | 每个map任务实际jvm启动参数设置最大堆内存。 -Xmx3072m |
| mapreduce.reduce.java.opts | - | 每个reduce任务实际jvm启动参数设置最大堆内存。 -Xmx3072m |
| mapreduce.map.cpu.vcores | 1 | 每个map任务申请cpu资源大小。 推荐值：1 |
| mapreduce.reduce.cpu.vcores | 1 | 每个reduce任务申请cpu资源大小。 推荐值：1 |

**说明**：

（1）这里对mapr/reduce资源的设置还需要根据 集群整体 资源情况和运行作业偏好设置，最好保证单个map(reduce)占用的memoryGB: vcore个数 = nodemanager配置的memoryGB: vcore个数，这样集群资源能够充分利用。  
（2）内存申请大小只是container使用资源的一个上限，实际使用内存由jvm参数决定。通常我们设置jvm最大堆内存为申请内存的0.75-0.95之间。另外对于性能要求非常高的task，可以在jvm参数中加入-XmsXXXXm，设置值与-XmxXXXXm相同，这样使得jvm初始化时候就已经是最大值，可以避免jvm堆空间扩容造成额外开销。 当Map或Reduce内存不够时，需要更多的GC时间，从而影响作业性能。判断Map分配的内存是否足够，一个简单的办法是查看运行完成的job的 Counters中，对应的task是否发生过多次GC，以及GC时间占总task运行时间之比。通常，GC时间不应超过task运行时间的10%，即GC time elapsed (ms)/CPU time spent (ms)<10%。 Map需要的内存还需要随着Map buffer的调大而对应调整。

**3.9.3、作业的map/reduce数量**

       Job需要充分利用集群资源； Job运行时，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，任务的并发度达到大。可以通过调整处理的数据量大小，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** | **默认值** | **描述** |
| --- | --- | --- |
| mapreduce.input.fileinput.format.split.maxsize | - | 可以设置数据分片的数据大值。 由用户定义的分片大小的设置及每个文件block大小的设置，可以计算得分片的大小。 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.fileinput format.split.minsize | 0 | 可以设置数据分片的数据小值 |

**说明**： 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。

对于集群测试mr benchmark相关场景需要考虑：reduce阶段尽量一轮跑完； 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。

**3.9.4、大任务的 AM 调优**

**任务场景**： 运行的一个大任务，map总数达到了10万的规模，但是一直没有跑成功。经过查询，发现是ApplicationMaster（以下简称AM）反应缓慢，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和cpu来管理。就需要修改以下参数值。

| **参数** | **默认值** | **描述** |
| --- | --- | --- |
| [yarn.app.mapreduce.am](http://yarn.app.mapreduce.am/).resource.mb | 512 | AM需要的内存资源。推荐值：1536（根据集群规模和作业规模可以更大） |
| [yarn.app.mapreduce.am](http://yarn.app.mapreduce.am/).command-opts | -Xmx312m | 传递到AM的Java命令行参数。推荐值：1024 |
| [yarn.app.mapreduce.am](http://yarn.app.mapreduce.am/).resource.cpu-vcores | 1 | am需要的vcore资源。推荐值;1 |

**3.10、通过“Slow Start”调优**

MapReduce的AM在申请资源的时候，会一次性申请所有的Map资源，延后申请reduce 的资源，这样就能达到先执行完大部分Map再执行Reduce的目的。在有些场景下，需要确保Map执行完，再执行Reduce，或者提前执行Reduce。

| **参数** | **默认值** | **描述** |
| --- | --- | --- |
| mapreduce.job.reduce.slow start.completedmaps | 1 | 百分之多少的Map执行完后开始执行 Reduce。默认100%的Map跑完后开始起Reduce。推荐值：0.85（根据作业类型可调整） |

**3.11、Active NN宕机，减少put操作和MR任务等待时间**

| **参数** | **位置** | **默认值** | **推荐值** | **说明** |
| --- | --- | --- | --- | --- |
| **参数** | **位置** | **默认值** | **推荐值** | **说明** |
| ipc.client.ping | core-site.xml | true | false | 客户端参数，当发生SocketTimeoutException时，是否发送ping去确认服务器网络是否可达。 |
| ipc.client.connect.max.retries.on.timeouts | core-site.xml | 45 | 5 | 客户端参数，client连接超时情况下重试的次数 |
| ipc.client.connect.max.retries | core-site.xml | 50 | 5 | 客户端参数，client连接重试的次数，重试多少次算超时 |
| ipc.client.connect.timeout | core-site.xml | 20000 | 10000 | 客户端参数，client连接超时时间 |

**3.12、Active RM宕机，减少MR任务等待时间**

| **参数** | **位置** | **默认值** | **推荐值** | **说明** |
| --- | --- | --- | --- | --- |
| ipc.client.rpc.timeout | core-site.xml | 900000 | 30000 | 客户端参数，client连接RM的超时时间 |
| yarn.client.nodemanager-connect.max-wait-ms | yarn-site.xml | 900000 | 180000 | 客户端参数，client连接nodemanaer超时时间 |
| yarn.nm.liveness-monitor.expiry-interval-ms | yarn-site.xml | 900000 | 180000 | 服务端参数，nodemanager检测线程判定NM失效的时间 |
| yarn.resourcemanager.container.liveness-monitor.interval-ms | yarn-site.xml | 900000 | 180000 | 服务端参数，RM认定container超时时间 |