Hadoop优化与调整

io.file.buffer.size

hadoop访问文件的IO操作都需要通过代码库。因此，在很多情况下，io.file.buffer.size都被用来设置缓存的大小。不论是对硬盘或者是网络操作来讲，较大的缓存都可以提供更高的数据传输，但这也就意味着更大的内存消耗和延迟。这个参数要设置为系统页面大小的倍数，以byte为单位，默认值是4KB，一般情况下，可以设置为64KB（65536byte）。

dfs.balance.bandwidthPerSec

HDFS平衡器检测集群中使用过度或者使用不足的DataNode，并在这些DataNode之间移动数据块来保证负载均衡。如果不对平衡操作进行带宽限制，那么它会很快就会抢占所有的网络资源，不会为Mapreduce作业或者数据输入预留资源。参数dfs.balance.bandwidthPerSec定义了每个DataNode平衡操作所允许的最大使用带宽，这个值的单位是byte，这是很不直观的，因为网络带宽一般都是用bit来描述的。因此，在设置的时候，要先计算好。DataNode使用这个参数来控制网络带宽的使用，但不幸的是，这个参数在守护进程启动的时候就读入，导致管理员没办法在平衡运行时来修改这个值。

dfs.block.size

很多人都会认为HDFS中块的大小都是一样的，其实这不正确。因为每个文件在创建的时候，都会确定相关的数据块大小。参数dfs.block.size定义了所有新建文件的默认数据块大小。这个参数的设定并不会影响文件系统中现有的文件，客户端在创建文件的时候，如果有特殊需要，可以重写该参数。

dfs.block.size的单位是byte，默认值是67108864 (64MB)。对于很多情况来说，134217728 (128MB)更加合适。对于一个Mapreduce作业（尤其是用子类FileInputFormat定义输入格式的作业），对文件的每个数据块会启用一个map任务来处理。这就意味这数据块的大小显著地影响Mapreduce作业的效率。

dfs.datanode.du.reserved

当DataNode想NameNode汇报可用的硬盘大小的时候，它会把所有dfs.data.dir所列出的可用的硬盘大小总和发给NameNode。由于mapred.local.dir经常会跟DataNode共享可用的硬盘资源，因为我们需要为Mapreduce任务保留一些硬盘资源。dfs.datanode.du.reserved定义了每个dfs.data.dir所定义的硬盘空间需要保留的大小，以byte为单位。默认情况下，该值为0.也就意味着HDFS可以使用每个数据硬盘的所有空间，节点硬盘资源耗尽时就会进入读模式。因此，建议每个硬盘都为map任务保留最少10GB的空间，如果每个Mapreduce作业都会产生大量的中间结果，或者每个硬盘空间都比较大（超过2TB），那么建议相应的增大保留的硬盘空间。

dfs.namenode.handler.count

NameNode有一个工作线程池用来处理客户端的远程过程调用及集群守护进程的调用。处理程序数量越多意味着要更大的池来处理来自不同DataNode的并发心跳以及客户端并发的元数据操作。对于大集群或者有大量客户端的集群来说，通常需要增大参数dfs.namenode.handler.count的默认值10。设置该值的一般原则是将其设置为集群大小的自然对数乘以20，即20logN，N为集群大小。如果前面的描述你仍然觉得很不清楚，可以看下面的python程序（其中的200表示集群的大小）

```
esammer:~ hadoop01$ python -c 'import math ; print int(math.log(200) * 20)'
105
```

如果该值设的太小，明显的状况就是DataNode在连接NameNode的时候总是超时或者连接被拒绝，但NameNode的远程过程调用队列很大时，远程过程调用延时就会加大。症状之间是相互影响的，很难说修改dfs.namenode.handler.count就能解决问题，但是在查找故障时，检查一下该值的设置是必要的。

dfs.datanode.failed.volumes.tolerated

当DataNode的任何一个本地磁盘出故障时，它的默认行为认定整个DataNode失效。在一个中到大型的集群中，硬盘故障是相当常见的，所以这种行为不是最优的。一个DataNode的丢失会导致一些数据块备份数下降，因此，NameNode会命令其他DataNode复制这些丢失的数据块来增加被附属。参数dfs.datanode.failed.volumes.tolerated定义整个DataNode声明失败前允许多少个硬盘出现故障。

很多人会问，为什么不能容忍所有磁盘失效的情况，这样就可以把整个DataNode的失效推迟到没有任何可工作的硬盘为止。对于一个永久的时间窗口来说，这看上去很合理的，但是实际上，对于所以的磁盘来说，管理员对于磁盘故障的处理都会早于由于正常磨损而出现的故障。只有一种情况例外，所以的硬盘在一个极短的时间内全部出现故障，这种异常情况需要立即调查。在实践中，快速的磁盘故障通常意味着驱动控制器或者某些部件故障。正因为罕见，但如果磁盘在短时间内开始出现一连串故障，最好的办法就是立即隔离。先把整个集群设置为不可用，直到找到失败的原因为止。参数dfs.datanode.failed.volumes.tolerated默认值为0，也就意味着只要有一个磁盘出现故障就会导致整个DataNode不可用，管理员可以增大该值来保证在出现部分磁盘故障时，DataNode仍能持续运行，但是需要保持谨慎的是，在极短的时间范围内出现一个或者两个磁盘故障表明一个更大的问题存在。

dfs.hosts

所有的DataNode都可以连接到一个NameNode并加入到集群。在第一次连接到NameNode时，DataNode会获取一个命名空间ID，并可以立即接收数据块。管理员可以通过一个含有DataNode主机名列表的文件，来确认允许连接并加入集群的DataNode。在这种情况下，其他DataNode则不允许加入集群。对于安全需要有强烈要求的，或者对于访问权限有控制的，都会用到这个功能。
该文件中dfs.hosts的格式是用换行符来分割主机名或者IP地址，这主要看集群通过什么来识别机器。

dfs.host.exclude

类似dfs.hosts，HDFS可以通过指定文件把相关节点排除在外，这个文件是一个以换行符分割的列表，每行包括一个主机名或IP地址。如果一台主机先被包含在内，又出现在排除列表中。即，如果一个机器名字同时出现在两个文件中，最终结果是被排除掉的。dfs.host.exclude的参数还有一个作用，它能优雅地卸载DataNode。

fs.trash.interval

用户经常会意外删除文件。HDFS支持回收站功能，这类似于大多数操作系统的回收站，当这个功能被启用，文件被移到用户的HDFS主目录中一个名为.Trash目录中，来保留被删除的文件，

而不是立即彻底删除。
fs.trash.interval定义.Trash目录下文件被永久删除前保留的时间。在文件被从HDFS永久删除前，用户可以自由地把文件从该目录下移出来并立即还原。默认值是0说明垃圾回收站功能是关闭的。

要清楚，回收站功能不是万能的，推迟删除意味着要文件所占据的空间仍不可用，除非它被永久删除。用户可以通过运行hadoop fs -expunge命令。或者干脆等待指定的时间来明确回收站清空。可以在hadoop fs -rm命令通过指定-skipTrash参数来跳过回收站从而立即删除文件。

以上摘自《Hadoop.Operations》