唯品会 HDFS 性能挑战和优化实践

作者：唯技术

> 本文以唯品会 HDFS 实际应用场景和问题导向触发，介绍了优化方案的局限性，分享了这些局限性的解决和实施经验。这对于技术运营较大规模的 HDFS 集群有一定借鉴意义。

1. 性能挑战

HDFS 是一个分布式系统，只要有足够的资源，可以扩容上千个节点支持 100PB 以上的集群。我们发现 Hadoop 集群升级（2.5.0-cdh5.3.2–>2.6.0-cdh5.13.1）以后，NameNode RPC（remote procedure call）queue time 在持续的在间隔一周左右性能恶化，在极端环境下出现一个 RPC 查询需要等待好几分钟的情况，Hive 作业出现大量的同一类型错误失败：

复制代码

```
1 Error in org.apache.Hadoop.Hive.ql.exec.mr.MapRedTask. Unable to close file because the last
```

重启集群以后问题可以得到缓解，但是这个问题需要从根本上考虑如何解决。

2. 性能优化

RPC 变慢的根源在于 HDFS 的 NameNode 吞吐量和性能瓶颈。NameNode 存在最大吞吐量限制，每一次写的请求都会产生排他性“写锁”，强制其他任何操作必须在队列里等待它完成。NameNode 的 RPC queue time 指标可以显示表达这个系统当前状态。对此我们主要从代码和业务两方面进行优化。

3. Datanode 延迟块汇报

1 .Datanode 的块汇报

datanode 这种快汇报请求，会频繁地持有锁，其实非常影响其他 rpc 的处理和响应时间。

2. 优化方案

通过延迟快汇报配置可以减少 datanode 写完块后的块汇报次数，提高 namenode 处理 rpc 的响应时间和处理速度。

配置：

```
1 <property>   <name>dfs.blockreport.incremental.intervalMsec</name>   <value>300</value></pro
```

目前我们 HDFS 集群上此参数配置为 300 毫秒，就是当 datanode 新写一个块，不是立即汇报给 namenode，而是要等待 300 毫秒，在此时间段内新写的块一次性汇报给 namenode。

4. 删除块个数可配置

由于 HDFS 的单一锁设计，NN 对于大目录删除行为并没有表现出很好的执行效果，严重时甚至会出现长时间 block 其它应用的正常请求处理。Hadoop 新版本引入新结构 FoldedTreeSet 来存储 DN 的块数据，但是它并不利于 update 操作，因此删除问题在升级后的版本中体现更为明显了。我们也在社区上提了相关 issue。

后续我们在研究 HDFS 删除块的行为中，发现 NN 在每次 batch 删除块的时候，是以固定 size 按照 batch 方式定期删除收集到的块信息。在每次 batch 间隙，其它请求就有机会得到 NN 锁的机会。于是我们考虑到一个改进手段，即是否能让 batch size 变得更加灵活可配置化，以此来控制给其它请求得到 NN 锁处理的概率。

基于这个思路，我们新建了以下配置项，并改动了相关代码逻辑。

```
1 <property>  <name>dfs.namenode.block.deletion.increment</name>  <value>1000</value>  <descri
```

此优化也已经被我们贡献到 Hadoop 社区，相关 JIRA 链接。

5. HDFS Federation

1. 独立集群模式弊端

在日常 HDFS 集群维护过程中，我们发现 HDFS 集群独立运行模式存在着许多弊端：

- 独立集群模式运维成本高，上下线机器每次都要制定分配所属集群。
- 多独立集群模式无非良好均衡资源和请求，经常发现 A 集群平时负载要远远高于 B 集群，这本质上是资源共享利用的问题。
- 单集群模式性能瓶颈问题。

综上，我们对现有大集群独立运行模式进行了 Federation 改造。Federation 改造的关键前提是不同 namespace 的 Cluster ID 必须保持一致，否则 DN 在上报过程中会抛出异常而注册失败。鉴于我们内部集群在初始搭建时指定了统一的 Cluster ID，所以并没有在前期再对 Cluster ID 做额外人工转换工作。

不同集群拓扑结构不一致导致 DN 注册上报错误，错误如下：

复制代码

```
1 2019-01-29   14:12:10,821 ERROR [Thread-30] org.apache.Hadoop.HDFS.server.datanode.DataNode:
```

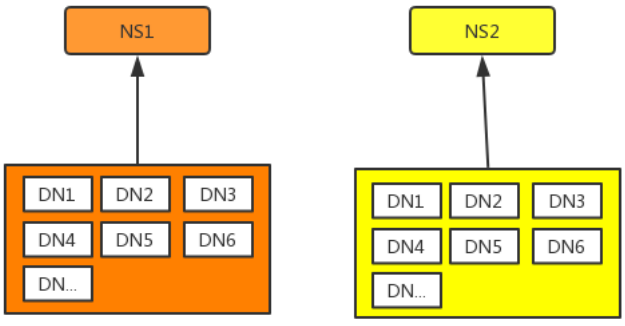
上述错误产生的根本原因是 DN 在 Federation 注册时在不同的 namespace 拥有不同 level 层级。后面经过原因排查，是由于我们没有完全同步好 2 个集群 rack-awareness 的脚本映射关系，由配置项 net.topology.script.file.name 所配置。

后续在 DN Federation 上报过程中，我们又遇到了因为本地 du 命令不准确导致 DN capacity 容量 double 的异常，继而导致 DN 无非正常进行写数据块行为。因为 DN 在上报自身 capacity 容量时，需要依赖于本地系统 du 命令来计算实际使用空间大小。后面我们对系统 du 命令进行了校准修复，最后 DN 能正常 Federation 上报注册。

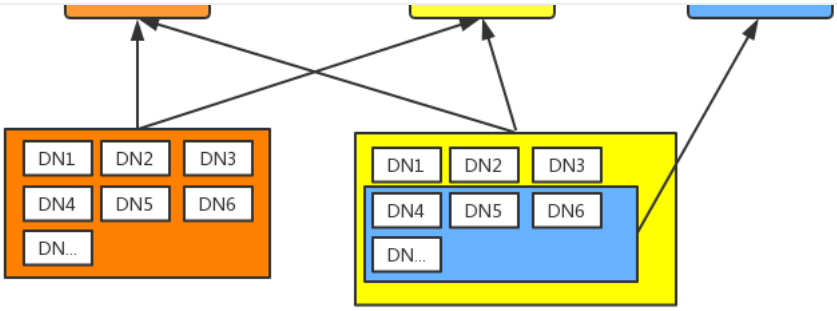
如今，我们已经完全打通 2 个独立大集群，同时加入第三套 NN，来做新的 namespace 存储，在未来会对数据进行业务划分，将数据均衡打散在不同 namespace 下，充分利用每个 namespace 下 NN 的处理能力。

另外一个问题是在 Federation 完成后发现的。因为 Federation 过程是将已有独立大集群模式改造成 Federation 模式，而不是直接搭建新 Federation 集群模式，我们发现 NN 元数据膨胀地比较厉害，即使 block 的元数据没有发生多大变化，但是实质上 DN 和 block 的映射是会得到膨胀的，因此后期马上对 NN 的 JVM 参数进行了相关调整。

我们原有主集群的运行模式如下，两个独立大集群运作模式：



经 Federation 改造完成的结构如下， 最终效果是所有 datanode 向全部三套 NN 汇报：

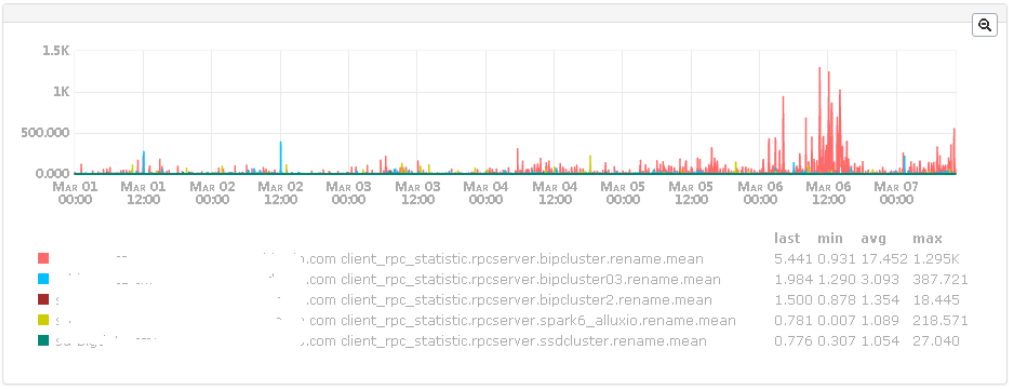


6. 客户端监控以及 temp 目录分流，Hive 本身降低 HDFS 请求

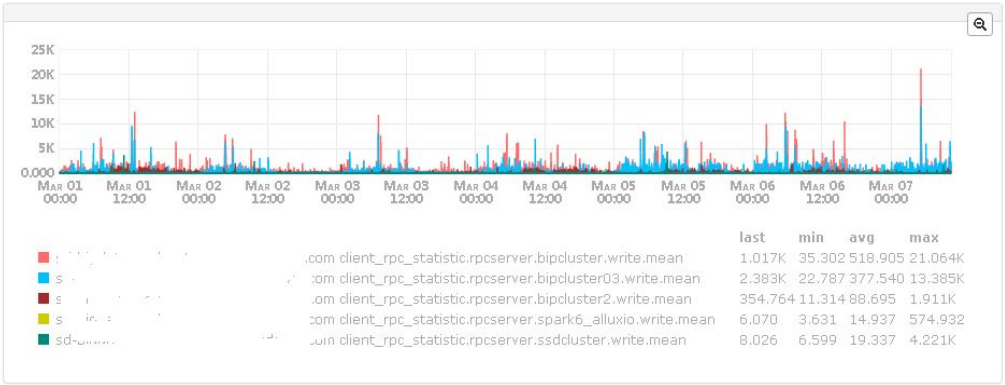
1.HDFS 客户端监控

客户端监控主要是从 HDFS 的客户端角度出发，监控 HDFS 的 rename、create 等部分 rpc 操作或者 write 这种涉及 datanode 操作的操作时长。这是补充 HDFS 服务端 rpc 监控的手段之一。

出发点是，有时服务端这边的监控比较正常，但是从任务（Hive，spark 或者 presto）角度来看，发现一些 move 或者 load 等操作依旧花费很长时间。这意味着服务端监控仅能够体现服务端处理性能，并不能很好地衡量整个集群向外提供服务的性能。



上图是 rename 的平均时长，考量的是一个文件被 rename 后的平均时长。



上图 write 的平均时长，考量一个只有少量数据的文件被创建时的平均时长，通过这个指标可以评估当前 namenode 的 8022 端口以及 datanode 性能。

2. temp 目录分流

从上面分析 bip 以及 bip03 的文件操作以及 rpc 情况来看，可以得出如下两个结论：

- 切换 defaultFs， 明显影响到 /mr/staging, /tmp/Hive/HDFS/ ，/mr/intermediate_done， 也非常明显影响了 rpc。

- 如此，对 temp 目录进行分流将会很大程度影响集群的 rpc 情况。

解决方案如下:

- 在 Hive 引擎层面（或者在调度层面也 ok），平衡切换 defaultFs，确保临时目录均衡地分布在 bip 或者 bip03 上面。

- 采用第三个集群，将 /mr/staging, /tmp/Hive/HDFS/，/mr/intermediate_done 迁移到上面，简而言之，就是把 defaultFs 设置成第三个集群 (最好可以通过 Federation 进行分流，这样不会太大影响数据的本地性)。

- 使用双报，通过自动化的方式平衡 bip 以及 bip03 的压力。

3. Hive 本身降低 rpc 请求

Hive 有很多地方都调用了 HDFS 的 rpc 接口，并发出大量 rpc 请求。如果能够从 Hive 的 rpc 客户方降低 rpc 请求，也能够很大程度缓解 HDFS 的压力。

- Hive 的 insert、create 等操作产生的临时数据，需要统一放到非表下，这样能够大量减少在最后 rename 的操作。

- 因为暂时用不上 HDFS 的 Encryption，所以多次的 Encryption 检测显然非常浪费性能，可以设置参数选择性关闭。

7. 小文件治理

小文件问题在大规模 HDFS 集群中是经常会遇到的问题。小文件过多引发的各种性能瓶颈在一定程度上影响了集群稳定性。我们采取了以下措施进行优化改善。

1. 改进措施

- HDFS Federation。相当于横向扩展 namenode 的处理能力，增加 namenode 数量来共同分担元数据管理的压力。但这并不十全十美，只是暂时隐藏了小文件多的问题。

- 合并小文件。这个方案说起来简单，却也并不容易。针对 Hive 相关任务，针对由历史任务产生大量小文件的作业，首先使用 CombineHiveInputFormat，将多个小文件作为一个整体 split，从而减少 map 数量，然后配置 mapred.min.split.size.per.node 和 mapred.max.split.size 增加 map 处理的文件大小。这个方案我们已经做成可配置化，用定时任务合并用户历史作业产生的数据。其次 orcfile 格式的 Hive 表，推荐使用 CONCATENATE 语义，orcfile 的合并是 stripe 级别，节省了解压和格式化数据的开销，增加效率。

经过一段时间的努力小文件数量得到有效改善，如下图所示:

2. 未来终极解决方案：Hadoop Ozone?

Hadoop Ozone 是基于 HDFS 实现的对象存储服务，支持更大规模数据对象存储，支持各种对象大小并且拥有 HDFS 的可靠性、一致性和可用性。Ozone 的一大目标就是扩展 HDFS，使其支持数十亿个对象的存储。目前这个项目已经成为 Apache Hadoop 的子项目，我们也会持续关注。

本文由唯技术公众号（VIP-Tech）授权 InfoQ 中文站转载。

大数据 Hadoop 性能调优

2 人喜欢

收藏 评论 微信 微博