HDFS RBF 在车好多的应用

背景

随着集群规模的不断扩张，文件数快速增长，目前集群的文件数已高达2.7亿，这带来了许多问题与挑战。首先是文件目录树的扩大导致的NameNode的堆内存持续上涨，其次是Full GC时间越来越长，导致NameNode宕机越发频繁。此外，受堆内存的影响，RPC延时也越来越高。

针对上述问题，我们做了一些相关工作：

控制文件数增长速度

快速增长很大一部分原因是集群存储大量小文件导致的，为解决该问题：

- 我们对计算组件的参数进行了调整，从源头减少小文件的产生。
- 对于已存在的小文件，采用工具定期进行小文件合并。

另一部分原因是数据的正常增长。针对这部分，我们进行了文件治理。具体措施包括：

- 文件生命周期管理，清理无效数据。
- 文件数Quota、账单管理，push 用户自行清理。

NameNode 优化

HDFS的元数据、数据块与数据节点的映射、RPC队列等都是常驻内存，对于NameNode的内存有着强大的依赖，因此我们调高了NameNode的堆内存，以减少RPC时延以及提高NameNode的稳定性。

- NameNode GC调优
- 堆大小调整

尽管我们做了一系列数据治理和调优工作，但收效甚微。hadoop社区提供了Federation方案，可以通过横向扩展NameNode 来提升元数据的管理上限，所以开始调研 HDFS Federation。

RBF的落地

Hadoop社区提供了两种Federation方案：分别是ViewFS 和 Router-Based Federation。在 Hadoop 3.1.3 版本下两者对比如下表：

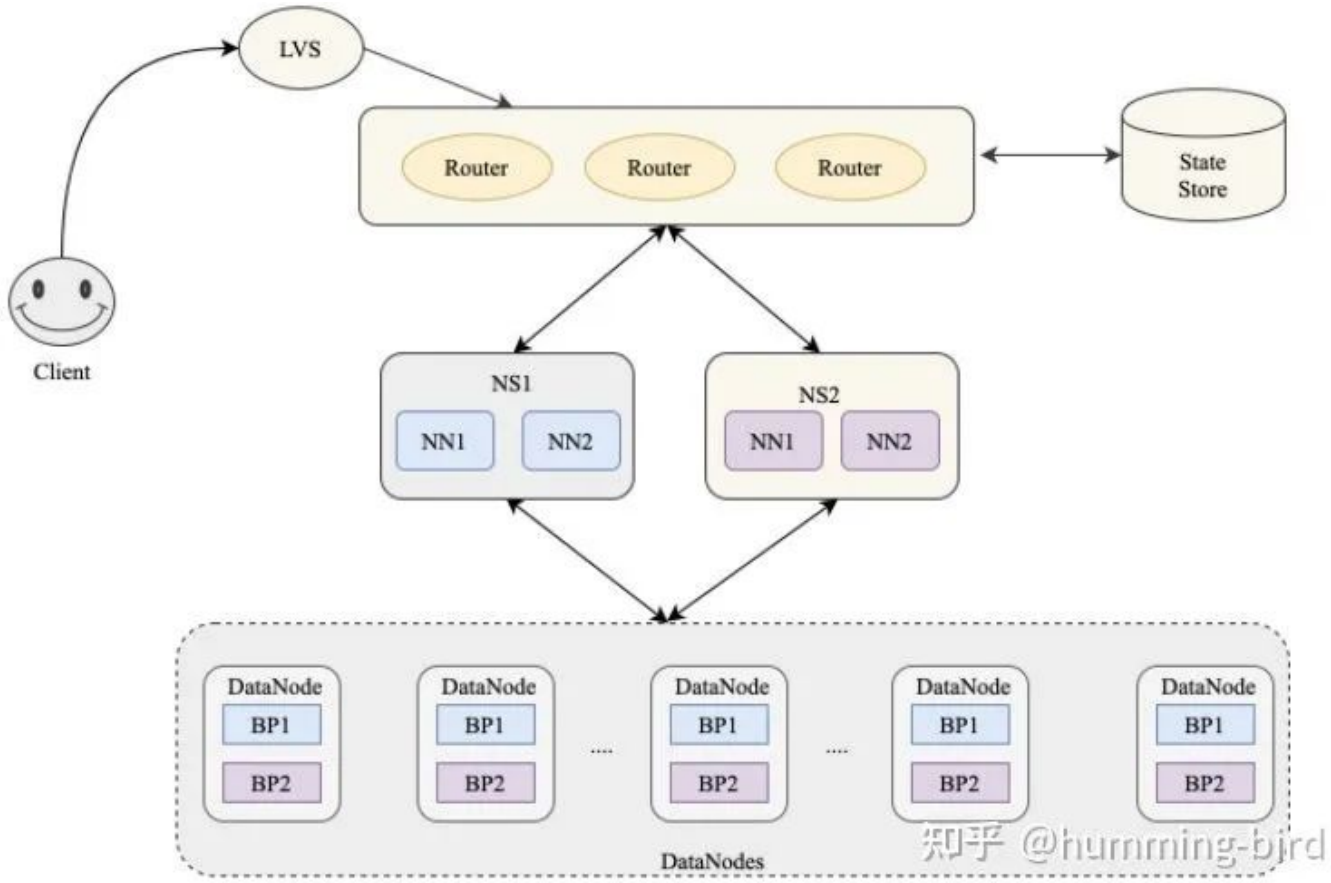
功能	Router-Based Federation	ViewFS	备注

schema前缀	hdfs://	viewfs://	
挂载信息存储方式	本地或zookeeper	配置文件	
客户端配置	轻	重	
调用耗时	适中	无	RBF加了一层router
缓存映射表	支持	支持	cache
权限设置	支持	不支持	
Quota设置	支持	不支持	
多NS挂载	支持	支持（备份方式）	
跨子集群的balance	社区开发中	不支持	
跨子集群的cp/mv	社区开发中	不支持	
安全认证	不支持（3.3开始支持）	支持	
Hadoop版本要求	Hadoop 2.9+	Hadoop 0.23+	

综合来看，Router-Based Federation 和 ViewFS 相比，在Hadoop 版本上要求较高需要2.9以上，不过我们已经完成了HDFS的从2.7 到 3.1.3 的升级，具体请查看这篇文章《HDFS 2.x 升级到 3.x
在车好多的实践》
。在升级管理方面更易于维护，且客户端改动更小，更好推动，因此我们最终选择了Router-Based Federation（RBF）。

方案架构

社区 Hadoop 在 2.9 和 3.0 中发布了 RBF 这个 Feature，瓜子目前的 Hadoop 版本是 3.1.3，我们直接使用社区的RBF，在HDFS集群中，Federation共有2组NameNode。根据我们的性能测试结果来看，一个Router对应服务一组NameNode不存在压力，因此我们选择部署3个Router来服务整个HDFS集群。整体架构如下图所示。



如果想及时了解Spark、Hadoop或者HBase相关的文章，欢迎关注微信公众号：过往记忆大数据

RBF 主要包括两个模块：Router 和 State Store

Router

在HDFS中Router可以有多个，且Router之间相互独立，如果一个Router不可用不会影响其他Router提供服务。Router主要实现了

- Federation 的接口：向客户端提供了一个全局的NameNode接口。负责接收Client 请求，根据 mount-table 中的信息查找正确的子集群，并转发请求到对应子集群 Active NameNode, 在收到 Active NameNode 的响应结果之后，将结果返回给客户端。为了提升性能，Router 可以缓存远程挂载表条目和子集群的状态。
- NameNode信息的维护：Router 定期检查一个 NameNode 的状态和向 State Store 报告其高可用性（HA）状态和负载/空间状态。为了提高 NameNode HA 的性能，Router 使用 State Store 中的高可用性状态信息,以将请求转发到最有可能处于活动状态的 NameNode。一个Router可以检测多个NameNode。
- 实现 可用性 和 容错性。Router 可以独立于 Hdfs 集群部署，并且是无状态的，可以放在负载均衡器后面使用。
- 其他的接口。比如，RPC、 routerAdmin 、WebUI、WebHDFS、JMX。

State Store

State Store有两种存储方式，本地存储和Zookeeper，我们选择的是Zookeeper，它主要维护两种信息：Membership和Mount Table

- Membership ：子集群的状态，比如总容量、DataNode 数量、NameNode的HA状态，Router 的状态等。
- Mount Table：保存路径到子集群的映射关系

落地步骤

- 搭建新的Namespace, 配置Federation
- 搭建Router集群
- Yarn / HBase 等上层集群配置RBF
- Client配置RBF
- Namespace1 集群相关目录distcp Namespace2
- Mount table 中add 映射

经过一段时间的准备工作，在用户无感知情况下，我们完成了RBF搭建、配置、数据迁移工作，解决了当前的痛点。

遇到的问题

在RBF落地的过程中，我们遇到过一些问题，有些问题社区高版本已经解决。这里，我们列举一些典型的问题做简要说明。

执行rm失败

在HDFS集群中，在哪个用户下执行rm操作，数据会被移动到/user/用户/.Trash目录下，默认的该目录挂载在Namespace1中，因此Namespace2中执行rm会报错rm失败。解决方法：将Trash挂载到两个集群。

多集群挂载的问题

以hadoop用户为例，当/user/hadoop/.Trash 多集群挂载到 hdfs://ns1:/user/hadoop/.Trash 和 hdfs://ns2:/user/hadoop/.Trash 并且挂载策略为LOCAL时 ，执行 rm 操作有一定概率失败。失败的原因在于多集群挂载策略的选择。多集群挂载策略有五种：HASH_ALL，RANDOM，SPACE，LOCAL，HASH。

这里解释下这五种策略：

- HASH_ALL：完全哈希策略，会对访问目录path的所有子目录进行hash。这种策略我们可以理解为是一种随机策略。不过它比随机策略更“高级”一些。它的内部实质上是用一致性哈希算法来做目标集群的选择的，用一致性哈希算法的好处是为了减少未来由于集群的变更（添加或移除）导致目标文件数据的重新定位。
- RANDOM：随机选择策略，调用随机数生成函数即可。因为对于相同路径，每次产生的结果是不可预测的，再对这些文件进行读操作的时候，是通过遍历所有潜在的集群位置来

定位实际的位置。
- SPACE：空间选择策略，遵循的一个原则：每次尽可能选择当前目标集群中可用空间最富余的集群，作为目标集群。这在大体上能够保证集群间的数据平衡。
- LOCAL：本地优先策略，优先选择请求发起方所在的集群为首先目标集群。所有此策略，要维护一套 [IP,
集群id]映射关系，然后客户端发起请求时，进行映射判断，选择一个最“近”的集群。
- HASH：局部哈希策略， 此策略的本质与完全哈希策略一样，继承了上面的策略类。唯一不同的在于“局部”，此策略在哈希生成key的时候，用的是第一层级路径，而不是全路径，这保证了一点，在同一父目录的路径，它们的哈希值是相同的，对应所找到的集群位置也会是相同的。

当执行rm操作时，会顺序调用 mkdirs 和
rename，而mkdirs，create，setPermission，setOwner，delete，getFileInfo
这几个方法在router 中进行了重写，策略HASH_ALL，RANDOM 和 SPACE
会调用rpcClient.invokeConcurrent(locations,
method)并行的去所有集群执行，其他策略HASH,LOCAL 会调用
rpcClient.invokeSequential(locations, method)
顺序的去执行，当第一个返回后就返回了，也就是说HASH_ALL ，RANDOM 和
SPACE会在所有集群创建目录，而HASH，LOCAL只会在一个集群创建目录。

当多集群挂载策略是LOCAL时，mkdirs 会以 rpcClient.invokeSequential(locations, method) 的方式执行，只要有一个子集群执行成功就返回true。这样两个集群中，就会有一个集群mkdir失败。所以对应的后续在这个集群上执行rename操作就会失败

解决方式：将多集群挂载设置为HASH_ALL ，RANDOM，
SPACE中的一种，为了负载均衡我们最终选择RANDOM策略。

在采用多子集群挂载时，getContentSummary() 类操作时失败并且报 空指针异常

该问题是由多集群挂载时包含EC路径引起的，目前社区已经修复，因此，我们的解决方式是将该issue合并到我们的Hadoop 3.1.3中。http://issues.apache.org/jira/browse/HDFS-14224

Hive Location 的问题

Hive会以绝对路径的方式存储元数据，因此如果修改fs.defaultFS则需要大批量修改Hive元数据，因此，我们决定保留fs.defaultFS的设置，在HDFS中更改该nameservice的映射关系，即由原来的映射NameNode修改为映射到LVS和Router。

安全认证

Hadoop 3.1.3版本的RBF是不支持kerberos的，社区修复了这一问题。http://issues.apache.org/jira/browse/HDFS-13358

但由于目前集群涉及的安全认证链路较少，因此我们暂时没有将它合并到我们的Hadoop 3.1.3版本。

Router 连接不均匀

首先，明确一点 Router 是无状态服务，可以通过负载均衡的方式来保证多个Router 处理的请求平均一些。需要在 Client 配置文件中声明dfs.client.failover.random.order = true，并且，每次新增或者更改Router ，都需要刷Client 配置。但是该参数在当前2.7 Client 不支持，2.9 之后支持. 因此我们的解决方案是通过LVS实现负载均衡。

将来的计划

当前的RBF不支持集群间的cp，采用distcp的方式迁移数据效率较低，对于集群迁移而言，fastcp是个不错的加速跨集群数据迁移的工具，因此未来我们会朝着这个方向进一步发展。此外，集群的安全认证功能需要完善，我们需要将kerberos安全的代码合并到我们的版本中。后续我们也会继续跟进社区和其他公司的发展，使RBF能够在公司有很好的落地。

原文链接HDFS RBF 在车好多的应用