HDFS Federation在美团点评的应用与改进

HDFS Federation为HDFS系统提供了NameNode横向扩容能力。然而作为一个已实现多年的解决方案，真正应用到已运行多年的大规模集群时依然存在不少的限制和问题。本文以实际应用场景出发，介绍了HDFS Federation在美团点评的实际应用经验。

背景

2015年10月，经过一段时间的优化与改进，美团点评HDFS集群稳定性和性能有显著提升，保证了业务数据存储量和计算量爆发式增长下的存储服务质量；然而，随着集群规模的发展，单组NameNode组成的集群也产生了新的瓶颈：

- 扩展性：NameNode内存使用和元数据量正相关，180GB堆内存配置下，元数据量红线约为7亿，而随着集群规模和业务的发展，即使经过小文件合并与数据压缩，仍然无法阻止元数据量逐渐接近红线；
- 可用性：随着元数据量越来越接近7亿，CMS GC频率也越来越高，期间也曾发生过一次在CMS GC过程中由于大文件getBlocklocation并发过高导致的promotion fail；
- 性能：随着业务的发展，集群规模接近2000台，NameNode响应的RPC QPS也在逐渐提高。越来越高并发的读写，与NameNode的粗粒度元数据锁，使NameNode RPC响应延迟和平均RPC队列长度也在慢慢提高；
- 隔离性：由于NameNode没有隔离性设计，单一对NameNode负载过高的应用，会影响到整个集群的服务能力；

HDFS Federation是Hadoop-0.23.0中为解决HDFS单点故障而提出的namenode水平扩展方案。该方案允许HDFS创建多个namespace以提高集群的扩展性和隔离性。基于以上背景，我们在2015年10月发起了HDFS Federation改造项目。

HDFS Federation是以客户端为核心的解决方案，对Hadoop客户端影响较大，在落地应用时也有较多的限制，对上层应用模式有较强的依赖。本文分享了在此次改造的过程中，基于美团点评的业务背景，我们对HDFS Federation本身做出的改进和对拆分过程的流程化处理，希望能为需要落地HDFS Federation的同学提供一个参考。

上层应用与业务

基础架构方面，美团点评Hadoop版本为2.4.1，使用了Kerberos作为认证支持；相关技术栈中，Spark应用版本包含1.1、1.3、1.4、1.5，同时使用了Zeppelin作为Spark notebook的开发工具；在查询引擎方面Hive有0.13和1.2两个版本，同时重度依赖Presto和Kylin；除此之外，也对dmlc提供了平台性支持。

工具链建设方面，基于Hadoop生态，数据平台组自研了各类平台工具，其中受Federation影响的部分工具有：

- 数仓管理：满足各类Hive表的DDL需求，同时支持UDF和文件上传建表；
- 原始数据接入：支持日志抓取和MySQL数据接入数据仓库；
- 非结构数据开发：支持作业托管，提供MR/Spark作业编译、管理、测试、部署一站式服务；
- 数仓开发：支持ETL的一站式开发和管理，同时在任务状态、诊断、SLA保证方面也有强力的支持；针对流程测试以及数据回收进行了隔离，使用统一的test.db和backup.db；
- 调度系统：自研的调度系统支撑了每天数万个调度作业，准确的处理作业间的强弱依赖关系，有效的保证了按天数据生产；
- 查询平台：统一了Hive和Presto的查询入口；

自研的数据平台基本覆盖了90%的数据开发需求，借此，一方面有效的控制了Hadoop客户端的数量，收紧了用户入口，对于发放的客户端，配合Kerberos，也具有很高的掌控力；另一方面实现了对用户行为的源码级掌控力。

数据开发方面，美团业务一直持续着爆发式增长，集群规模和数据生产流程增量每年都接近double。业务发展也推动了组织结构的发展，进而也影响到了相应的大数据资产：

- 一个hadoop账号可能经历过多个业务线，用户应用中，对其他hadoop账号的数据进行读写、move较为常见，对这类行为也没有进行过梳理和限制；
- 完成平台接入后，对生产流程管理的规范较多，但对用户代码的规范较少，用户代码风格多样；

应用与改进

Federation的局限性

在解决NameNode扩展能力方面，社区虽然提供了Federation，但是，这个方案也有很强的局限性：

- HDFS路径scheme需要变为viewfs，viewfs路径和其他scheme路径互不兼容，比如DistributedFileSystem无法处理viewfs为scheme的路径，也就是说如果启用，则需要将Hive meta、ETL脚本、MR/Spar
- k作业中的所有HDFS路径均的scheme改为viewfs；
- 如果将fs.defaultFS的配置从hdfs://ns1/变为viewfs://ns/，将导致旧代码异常，通过对用户上万个源码的分析，常用的HDFS路径风格多样，包括hdfs:///user、hdfs://ns1/user、/user等，如果fs.defaultFS有所更改，hdfs:///user将会由于缺失nameservice变为非法HDFS路径；
- viewfs路径的挂载方式与Linux有所区别：

  1、如果一个路径声明了挂载，那么其同级目录都需要进行挂载，比如/user/path_one挂载到了hdfs://ns1/user/path_one上，那么/user/path_two也需要在配置中声明其挂载到哪个具体的路径上；
  2、如果一个路径声明了挂载，那么其子路径不能再声明挂载，比如/user/path_one挂载到了hdfs://ns1/user/path_one上，那么其子路径也自动并且必须挂载到hdfs://ns1/user/path_one上；

- 一次路径请求不能跨多个挂载点：
  1、由于HDFS客户端原有的机制，一个DFSClient只对应一个nameservice，所以一次路径处理不能转为多个nameservice的多次RPC；
  2、对于跨挂载点的读操作，只根据挂载配置返回假结果；
  3、对于跨挂载点的rename(move路径)操作，会抛出异常；
- Federation架构中，NameNode相互独立，NameNode元数据、DataNode中块文件都没有进行共享，如果要进行拆分，需要使用DistCp，将数据完整的拷贝一份，存储成本较高；数据先被读出再写入三备份的过程，也导致了拷贝效率的低效；
- Federation是改造了客户端的解决方案，重度依赖客户端行为；方案中NameNode相互独立，对Federation没有感知；另外hdfs为scheme的路径，不受Federation挂载点影响，也就是说如果对路径进行了namespace拆分后，如果因为代码中的路径或客户端配置没有及时更新，导致流程数据写入老数据路径，那么请求依然是合法但不符合预期的；

对其中一些名词的解释:

- 在HDFS中namespace是指NameNode中负责管理文件系统中的树状目录结构以及文件与数据块的映射关系的一层逻辑结构，在Federation方案中，NameNode之间相互隔离，因此社区也用一个namespace来指代Federation中一组独立的NameNode及其元数据。
- scheme是URI命名结构（[scheme:][//authority][path][?query][#fragment]）中的一部分，用于标识URI所使用的协议，HDFS路径也是一个URI，常见的scheme为hdfs，在Federation的方案中，HDFS路径scheme为viewfs。
- 挂载点(mount point)，它在HDFS Federation中和Linux中的概念近似，指在HDFS客户端上下文中，将viewfs为scheme的一个路径，比如viewfs://ns/user，映射到一个具体的HDFS路径上，比如hdfs://ns2/user，这个路径可以是任意scheme的HDFS路径，这样对于viewfs://ns/user实际上会被转换为对hdfs://ns2/user的操作。

局限性带来的问题和解决

scheme兼容性问题

scheme的兼容问题要求在上线时全量替换业务方代码中的路径，虽然对业务方大多数源码具有掌控力，但是由于不可灰度带来的全量修改带来的测试、上线、修复工作的成本，全量操作带来的运维时间，以及对数据生产稳定性的影响都是不能接受的。为此，以能灰度启用Federation特性为目标，对HDFS客户端进行了修改：

1、增加了viewfs和hdfs两种scheme路径的兼容性:
修改了org.apache.hadoop.fs.FileSystem.fixRelativePart(Path)，该函数在DistributedFileSystem各类请求处理中均有调用，原本用于处理相对路径，而ViewFileSystem不会调用；在这里，如果遇到了viewfs为scheme的路径，则利用ViewFileSystem中的挂载信息返回真正的hdfs路径；
修改了org.apache.hadoop.fs.viewfs.ViewFileSystem.getUriPath(Path)，该函数在ViewFileSystem各类请求处理中均有调用，原本用作判断路径scheme为viewfs，同时处理相对路径；一方面，由于Federation的挂载配置中，只有通过挂载点查询真实路径的数据结构，逆向查询比较复杂，改动也比较大；另一方面，从运营角度看我们也不希望维持非常复杂的挂载配置。所以在这里，做了一个限定，对于hdfs为scheme的路径与其在Federation的挂载点路径相同，所以在此函数中如果遇到了hdfs为scheme的路径，直接使用org.apache.hadoop.fs.Path.getPathWithoutSchemeAndAuthority(Path)去掉scheme即可；

2、fs.defaultFS变更会对原有代码带来影响，但是将其配置为viewfs为scheme的路径才能使hdfs scheme的应用逐渐收敛，因此，我们增加了用于指定默认namespace的配置fs.defaultNS，使hdfs:///user这样即使没有提供Authority的路径也能路由到正确的NameNode；
针对scheme局限性的改造，虽然提高了兼容性，使方案能够进行灰度，但却使DistributedFileSystem和ViewFileSystem耦合，又增加了一条ViewFileSystem挂载限制，因此只适合在过度期间应用。

挂载配置限制

viewfs的挂载方式与Linux有所区别，如果完全继承现有HDFS不变，则需要非常多的挂在配置项，并且后续每次增加Hive库、用户目录，初期我们使用了运营手段解决了这个问题：

- 将迁移路径放到独立的目录下，比如/user/hivedata/xx.db，迁移到/ns2/hivedata/xx.db，这样挂载声明则不会太过复杂；
- 由于用户组路径大都应用于MR、Spark作业中，修改路径需要重新编译，因此初期应用时，只对Hive库路径；
- 由于跨namespace不能进行rename，所以分析NameNode审计日志，得到Hive库路径和用户组路径没有rename关系的库，只对这些库进行迁移；
- 通过以上三个种手段，对于ETL流程这种不需要编译的代码，可以直接替换，对于MR、Spark作业来说推动修改的成本也有所降低；
- 为了进一步降低后续拆分成本，我们在ETL和作业开发两个方面提供并推广了根据库表信息从Hive meta中取得库表HDFS路径的工具，减少了代码中对库表路径的硬编码；

以上的运维手段，能满足美团侧常规的拆分需求，但是随着点评侧数据融合，点评侧数据也作为整体集群的一个namespace加入进来。然而点评侧平台掌控力没有深入到源码级别，因此无法统一推动更改HDFS路径，所以如果不对挂载逻辑进行修改，在合并重复路径时，需要将美团侧/user路径合并到点评侧/user路径中，但是由于跨namespace无法进行rename，势必会造成用户作业的失败。因此，我们对挂载逻辑进行了修改，使其同Linux的挂载方式相同。

同namespace, 不同挂载点不能rename

业务方很多Hive库表数据会先生成在测试库表或用户目录中，验证完成后将数据加载到对应时间分区中，在挂载配置中，业务方Hive库、Hive测试库、用户组目录一般不会挂载到同一目录下，因此即使三者在同一namespace下，由于不同挂载点间不能rename的限制，也无法进行加载。
在源码分析的过程中，发现以下注释：

```
// Note we compare the URIs. the URIs include the link targets.
// hence we allow renames across mount links as long as the mount links
// point to the same target.
if (!resSrc.targetFileSystem.getUri().equals(
        resDst.targetFileSystem.getUri())) {
  throw new IOException("Renames across Mount points not supported");
}


//
```

```
// Alternate 3 : renames ONLY within the the same mount links.
//

if (resSrc.targetFileSystem !=resDst.targetFileSystem) {
  throw new IOException("Renames across Mount points not supported");
}
```


可以发现社区是有考虑相同namespace路径可以进行rename操作的（注释掉的原因没有找到），因此，我们将这段逻辑打开，替换掉了“renames ONLY within the the same mount links”。

存储成本与拷贝效率问题

使用Federation方案时，集群节点规模为2000多台，元数据已达6亿，存储使用已近80%，按照规划，存储容量不足以支撑全部待迁移数据，但是拆成多次操作周期和运维成本都比较高，因此我们计划调研FastCopy。

FastCopy是facebook开源的数据拷贝方案，他通过以下方式在不增加存储成本的情况下对数据进行拷贝：

- 通过getBlockLocation获取源文件块分布；
- 通过ClientProtocol（HDFS包中的接口，下同）创建目标文件；
- 通过ClientProtocol addBlock，在参数中，指定源块分布作为favoredNodes，常规情况下NameNode会优先选择favoredNodes中的DataNode作为块的保存位置，特殊情况下（比如存储空间不足，DataNode负载过高等）也有可能返回不同位置；
- 整理源和目标块位置，使相同DataNode的位置能一一对应；
- 通过ClientDatanodeProtocol向源DataNode发送copyBlock请求；
- 在DataNode中，如果copyBlock请求中的源和目标相同，则通过在Linux文件系统中建立硬链的方式完成拷贝，否则通过原有逻辑完成拷贝；

但是，在计划合入时，该方案也有自身的问题：

- 社区path为HDFS-2139，一直处于未合入状态，且当时patch内容相对facebook的方案来说，部分细节没有考虑，例如文件lease，无法构造硬链时的降级，DFS Used的统计问题等；
- facebook的源码相对成熟，但其源码基于0.20（facebookarchive/hadoop-20），已有四年没有更新，已经有很多源码发生变化，DFS Used的统计问题也没有解决；
- 虽然facebook将FastCopy合入DistCp，但也有部分缺陷：
  1、每个路径生成一个maper，每个mapper只处理一个路径，如果目录层次过高，容易导致数据倾斜，如果目录层次太低，容易产生过多mapper；

2、只对迁移路径进行属主同步，其父目录没有处理；
3、与DistCp耦合定制比较复杂；

所以，综合以上内容，我们完善了HDFS-2139，并更新了issue，在合入facebook实现的基础上解决了DFS Used的统计问题；除了这个patch，我们也实现了独立的FastCopy MR作业，解决了上述问题。最终，在拆分时15小时完成14+PB数据拷贝，保证了方案的可行性。

另外需要注意的是，对于HDFS来说，无法感知哪个块是通过硬链构造的，因此，一旦源和目标文件同时存在时，开启balancer，会因为块的迁移导致存储使用的增加，因此，迁移期间，一般建议暂停相关namespace的balancer。

重度依赖客户端

基于以上几点改进，虽然降低了拆分成本和兼容性，使Federation的应用成为可迭代方案，但是如果没有对客户端强大的掌控力，客户端实例不能完全更新，HDFS路径硬编码不能得到彻底梳理，反而会造成数据生产方面的混乱，成为此方案的掣肘。

经过美团侧数据平台的多年运营，对客户端以及业务代码有非常强的掌控力，有效避免了上述问题的发生。

计算和查询引擎的问题和解决

一方面，虽然Federation已出现了多年，但Hive、Spark等上层应用对Federation的支持仍然存在问题；另一方面，随着应用的逐渐加深，虽然有些问题并不是代码bug，但在美团点评的应用场景下，仍然产生了一定问题；我们针对这些问题，也进行了探索和改进。

安全问题

安全方面，计算引擎包括MR和Spark，在提交作业时，会向NameNode发送RPC，获取HDFS token，在ViewFileSystem中，会向所有namespace串行的申请token，如果某个namespace的NameNode负载很高，或者发生故障，则任务无法提交，YARN的ResourceManager在renew token时，也会受此影响。随着美团点评的发展YARN作业并发量也在逐渐提高，保存在HDFS上的YARN log由于QPS过高，被拆分为独立的namespace，但由于其并发和YARN container并发相同，NameNode读写压力还是非常大，经常导致其RPC队列打满，请求超时，进而影响了作业的提交。针对此问题，我们做出了一下改进：

- container日志由NodeManager通过impersonate写入HDFS，这样客户端在提交Job时，就不需要YARN log所在namespace的token；
- ViewFileSystem在获取token时，增加了参数，用于指定不获取哪些namespace的token；
- 由于作业并不总是需要所有namespace中的数据，因此当单个namespace故障时，不应当影响其他namespace数据的读写，否则会降低整个集群的分区容忍性和可用性，ViewFileSystem在获取token时，即使失败，也不影响作业提交，而是在真正访问数据时作业失败，这样在不需要的token获取失败时，不影响作业的运行；

另外，客户端获取到的token会以namespace为key，保存在一个自定义数据结构中(Credentials)；ResourceManager renew时，遍历这个数据结构；而NodeManager在拉取jar包时，根据本地

配置中的namespace名去该数据结构中获取对应token。因此需要注意的是，虽然namespace配置和服务端不同不影响普通HDFS读写，但提交作业所使用的namespace配置需要与NodeManager相同，至少会用到的namespace配置需要是一致的。

已存在patch问题

https://issues.apache.org/jira/browse/HADOOP-12253
https://issues.apache.org/jira/browse/TEZ-2600
https://issues.apache.org/jira/browse/HIVE-11364
https://issues.apache.org/jira/browse/HIVE-10790
https://issues.apache.org/jira/browse/HIVE-6152
https://issues.apache.org/jira/browse/HIVE-11920
https://issues.apache.org/jira/browse/HIVE-7529

其他问题

Hive create table .. as .. 会导致临时文件所在目录和表目录不在同一namespace，导致move结果失败，目前已修复，思路同HIVE-6152，将临时文件生成在表目录中；

Hive表的元数据中，SERDEPROPERTIES中，可能会存在对HDFS路径的依赖，在梳理路径硬编码时，容易忽略掉；

Spark 1.1在启用viewfs时，会产生不兼容问题；

开源分布式机器学习项目dmlc目前也尚不兼容viewfs；

拆分流程与自动化

随着namespace拆分经验的积累，其流程也逐渐清晰和明确：

1、当namespace的NameNode逐渐接近瓶颈（包括RPC和元数据量）时，对hadoop用户对应的用户组目录和Hive库目录进行分析，得出元数据量（通过分析fsimage）和一天内RPC量（通过分析审计日志），进而得出需要拆分的用户数据；

2、对于需要拆分的数据，分析其和不需要拆分数据的rename关系，如果存在rename关系，则需要重新选择拆分数据；

3、如果需要，则搭建新namespace环境；

4、关闭相关namespace balancer；

5、根据fsimage，分析出待拆分路径元数据分布，得出一个路径列表，使列表中每个路径下的文件块数基本接近；

6、基于第四步的结果进行首轮拷贝，首轮拷贝中针对不需要比较验证的情况作出了优化：FastCopy
MR工具会递归的拷贝路径，如果目标路径已存在说明之前已拷贝成功过，则不进行拷贝；

7、之后进行多轮补充拷贝：通过ls -r得到文件和目录列表；拷贝过程中开启-delete -update，非递归的进行检测与拷贝，这样对于源目录有更新的文件和目录会进行覆盖（包括权限和属主的更新），源目录新增的目录和文件会进行拷贝，源目录删除的文件和目录会进行删除；这样，可以会每一层的目录进行检测，可以同步目录权限和属主发生的变化，同时也不会产生较大的数

据倾斜；

8、准备好新挂载配置，找一个非工作时间，进行最终一轮的操作：

a. 禁止源目录的权限（FastCopy使用hdfs身份运行不受影响）；

b. 进行最后一轮补充拷贝；

c. 由于数据大多数情况下基于硬链进行拷贝，所以存在文件长度相同，但内容有问题的可能性极低，拷贝完成后，可以通过du路径，校验并逐渐找到数据长度不一致的文件，进行重考；

d. 对客户端分发新挂载配置；

e. 对NodeManager分发
新挂载配置，并进行decommission，重启（YARN已支持recovery）；

f. 更新Hive meta；

g. 开放目标目录权限；

9、观察一周，如果没有问题则删除源目录；

10、重启balancer；

以上是已经固定下来的步骤，其中第1、2、5、6、7步，第8步中的a~c是可以进行自动化的，这也是后续工作过程中，有待完善的部分。

总结

HDFS Federation作为以客户端配置为核心的NameNode横向扩容解决方案，对业务背景有较强的依赖，另一方面方案本身也有较多的局限性。本文以美团点评实际应用场景出发，介绍了方案局限性在业务背景下的影响，分享了对局限性的解决和实施经验。对HDFS Federation应用到已运营较长时间的大规模HDFS集群有一定的借鉴意义。

作者介绍

美团点评离线存储团队，深耕Hadoop生态中HDFS、HBase、CarbonData、Alluxio等泛存储领域，致力于为美团点评提供稳定、高效、易用的大数据存储服务。