达到 PB 级时, 关系型数据库就很难有效地处理. 为了降低数据存储量, 目前网省公司都会从最原始的采集数据中过滤出一些相对“干净”的数据, 再存储至关系型数据库. 这样可以有效地减少了网络传输和数据库存储的数据量, 但是这些过滤掉的数据中依然有着重要的潜在价值, 例如绝缘的放电普.

2) 有限的非结构化数据处理能力. 传统的关系型数据库对数据的处理数值型和字符型等数据类型表现优异, 但对非结构化的数据类型(图片、音视频等)的处理能力比较差. 然而随着用户应用需求的不断提高、互联网技术的发展, 促使多媒体技术不断演进. 用户对多媒体数据的要求也日益提高, 面对日益增长的音频、视频、图像、文本等非结构化的数据, 传统数据库已经无法满足需求.

3) 有限的海量数据快速访问能力. 关系型数据是一种按内容访问的数据模型, 即根据列的值来定位相应的行. 在访问数据时, 这种数据模型会有耗时的输入输出, 从而降低快速访问的能力. 虽然, 传统的数据库会利用一些技术来减少查询时间(如分区技术等), 但是在大数据时代, 这样的优化方案使得性能提升水平并不显著.

4) 扩展性差. 在大数据时代, 传统关系型数据库的扩展性(scalability)并不好. 通常在解决此类问题时, 数据库有 2 种方式: 向上扩展(scale up)和向外扩展(scale out). 面对大数据处理时, 通过提升数据库服务器硬件性能, 实现向上拓展, 这样会使用户成本增加. 关系型数据库也可以通过向外拓展对数据集合进行水平或垂直划分, 然后将数据部署在数据库集群上, 但这种方法通常针对特定应用, 不同的应用其切割设计是不同的.

2 主流开源分布式文件系统

分布式文件系统主要功能可以分为两类: 1)存储图像、文本、音视频等非结构化数据; 2)作为分布式表格、检索等系统的持久化层. 分布式文件系统比较有名是 Google 的 GFS, 它构建在廉价的 PC 服务器上. 其他开源文件系统文件系统, HDFS、TFS 和 FaceBook Haystack 多多少少都借鉴了 GFS 的架构思路[13].

2.1 Hadoop 分布式文件系统(HDFS)

HDFS 是 GFS 的开源实现, 最早是 Apache 项目 Nutch 的基础结构[14]. 目前已经有很多存储 PB 级数据的 HDFS 集群了, 雅虎已经将 Hadoop 集群扩展到 4000 节点了[15]. 它的构建思路以流式数据访问模式来存储超大文件, 一次写入, 多次读取是最高效的访问模式.

HDFS 是一个注重容错性和兼容廉价设备的分布式文件系统, 它采用一次写入多次读取的文件模型, 读取时采用流的方式批量读取(而非用户交互式读取). HDFS 采用主从架构, 由唯一一个目录(元数据)节点(NameNode)和多个数据节点(DataNode)组成[16], HDFS 的体系结构如图 1 所示.

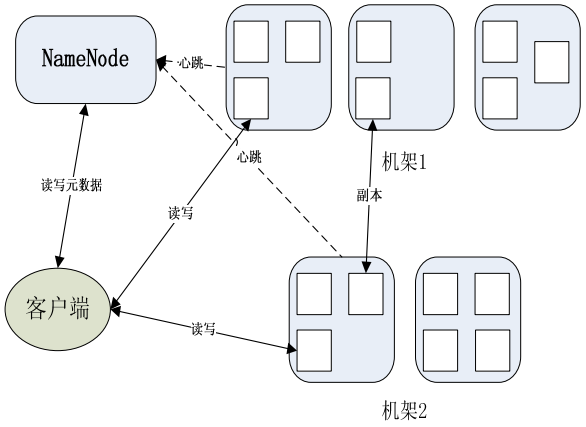

图 1　HDFS 架构图

NameNode 是集群的主节点, 负责文件名的维护管理(包括文件和目录的创建、删除、重命名等), 同时也管理数据块(DataBlock)和数据节点(DataNode)的映射关系. NameNode 用来管理文件系统(包括目录、文件名、文件与 DataNode 中数据块的对应关系)的命名空间, 将所有文件和文件夹的元数据保存在一个文件系统树中, 这些信息会在硬盘上保存成命名空间镜像(namespace image)及修改日志(edit log). NameNode 通过从数据节点收集信息的方式在内存中保存一个文件包括哪些数据块、数据块分布在哪些数据节点上的信息. NameNode 是客户端访问文件的入口, 客户端需要访问 NameNode 后得知文件的所有 DataBlock 所在 DataNode.

对于国网的非结构化数据中的大文件, HDFS 具有天生优势, 大文件的特点是一次写入多次访问. HDFS 批量写入大文件效率很高, 而且少量大文件可以减少元数据, 从而减轻 NameNode 压力. 对于大数据计算问题, HDFS 可以与 MapReduce 无缝连接, 作为 MR 作业的数据源, HDFS 是完美的选择.

但是, 一般地, HDFS 上都是大文件, 而且它的访问机制适合批量读取数据, 因此, 在访问 HDFS 的文件时, 会存在延迟. 对于实时性非常高的的场景, 需要额外一些技术来支撑比如 HBase. 在面对大量小文件时, HDFS 的元数据会非常大, 这样 NameNode 节点压力会很大.

2.2 Taobao 文件系统(TFS)

在 2007 年以前, 淘宝的图片存储系统使用的是昂贵的 NetApp 存储设备, 但是由于淘宝飞速发展, 数据量增长很快, 处于性能与成本的考虑, 淘宝自主研发了非结构化数据存储系统 Taobao File System(TFS). 目前, TFS 中存储的图片规模已经达到数百亿[17].

TFS 的设计思路是多个逻辑图片文件共享一个物理文件[18]. 它主要考虑两个瓶颈问题: 1)海量元数据存储. 百亿张图片的元数据占用空间约为 1TB, 单台服务器是无法满足存储的. 2)减少图片读取的 IO 次数, 在访问某一文件时, 需要依次将元数据、结点数据和文件数据读入内存中. 由于小文件数量众多, 只通过一次 IO 就访问数据的理想状态很难达到.

TFS 的架构如图 2 所示, 一个 TFS 集群由两个 NameServer 节点(主、备)和多个 DataServer 节点组成, NameServer 通过心跳对 DataServer 的状态进行监测. 当主 NameServer 出现故障时, 服务将会被转移到 NameServer 上. 每个 DataServer 上会运行多个 dsp 进程, 一个 dsp 对应一个挂载点, 挂载点通常对应一个独立磁盘.

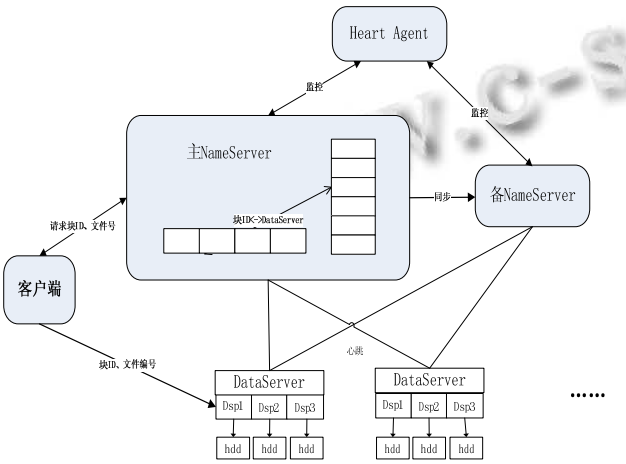

图 2 TFS 架构图

在 TFS 中, 将大量的小文件(实际数据)合并成为一个大文件, 这个大文件称为块(Block), 每个 Block 拥有在集群内唯一的编号(块 ID), 通过<块 ID, 文件编号>可以唯一确定一个文件. TFS 中 Block 的实际数据都存储在 DataServer 中, 大小一般为 64MB, 副本为三份. 客户端应用是 TFS 提供给应用程序的访问接口, 客户端不缓存文件数据, 只缓存元数据.

国网非结构化数据有海量小文件, 特别是小图片文件(表计的照片信息等等). 国网中, 通常用 CIM 数据模型来表示网架设备结构拓扑图, 会有海量 SVG 图片数据, 每张图片数据通常不大, 但是它们的经常需要要展示, TFS 可以很好地适应, 它可以支持海量小图片文件的低延迟访问.

但是 TFS 存储大文件导致项目集合占用的存储越来越大, 浪费了存储控件同时影响了备份效率.

2.3 FaceBook Haystack 文件系统

FaceBook 目前存储了 3000 多亿张照片, 大小为 20 多 PB. 用户每周新增照片数为 10 亿张(约为 60TB), 每秒大约有 3500 次写操作, 读操作峰值可以达到每秒百万次. FaceBook 相册后端早期采用基于 NAS 的存储, 通过 NFS 挂载 NAS 中的文件来提供服务. 后来亦是因为数据量不断增长, 带来了性能和成本压力, FaceBook 自主研发了 FaceBook Haystack 存储相册数据[19].

FaceBook Haystack 的设计思路亦是多个逻辑文件共享一个物理文件. Haystack 架构如图 3 所示.

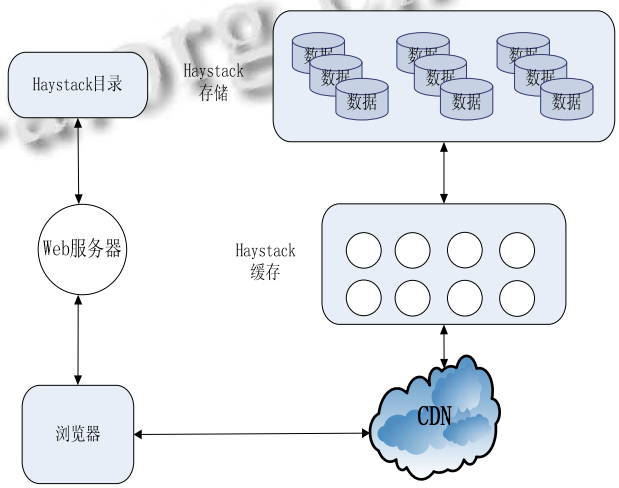

图 3 Haystack 架构图

Haystack 系统主要包括三个部分: 目录(Directory)、存储(Store)以及缓存(Cache). Haystack 存储是物理存储节点, 以物理卷轴的形式组织存储空间,

每个物理卷轴一般很大. 每个物理卷轴对应一个物理文件, 因此, 每个存储节点上物理文件元数据很小. 多个物理存储节点上的物理卷轴组成一个逻辑卷轴, 用于备份. Haystack 目录存放逻辑卷轴和物理卷轴的对应关系, 以及照片 ID 到逻辑卷轴之间的映射关系. Haystack 缓存主要用于解决对 CDN 提供商的依赖问题, 提供最近增加照片的缓存服务.

Haystack 的亮点在于其缓存技术, 国网多级部署可以像 Haystack 架构一样, 总部与网省公司的整个文件系统通过网络互连, 网省公司可以作为 CDN 存在. 不过因为多种原因, 目前在国网中, 尚未使用 Haystack 文件系统.

3 国网非结构化数据存储策略及演进

3.1 不同大小文件存储策略

(1) 文件存储

根据上节分析, 大文件存储 HDFS 非常适应. HDFS 可以部署廉价的 PC 机上, 它能支持硬件出错情况下保证数据完整, HDFS 目标是在成百上千的廉价 PC 机上存储数据, 可以快速检测硬件错误并自动进行数据迁移、修复. 在 HDFS 上一个典型的文件一般都是 GB 或 TB 级的, 对于一次写入多次读取的大文件的存储 HDFS 可以非常好的支持. 大文件历史数据存储, 最好的开源分布式文件系统选择是 HDFS.

目前国网中的一些运行数据, 底层可以直接储存在 HDFS 上, 中等规模网省公司的某些系统数据一年大概几十亿条. 利用这些数据构建数据仓库时, 在 HDFS 之上可以搭建 Hive, 并根据这些大写文件创建 Hive 外表, 方便数据查询分析, 查询速度平均在十几秒.

(2) 文件存储

对于海量的小文件存储, 原生态的 HDFS 是难以满足的, 但是存储小文件的策略在开源分布式文件系统方面可选择方案要更丰富一些, 根据上节分析, 有以下三种方案:

1) 淘宝 TFS 设计初衷就是为海量小文件提供存储解决方案, 小文件通常不超过 1MB. 在淘宝的多个应用中, 海量的小图片的都是存储在 TFS 上的, TFS 部署在普通 Linux 机器集群上. TFS 中没有文件的概念, 它所有小文件拼接成 Block, 因此元数据中只有 Block 的映射信息, 这样就不会想 HDFS 有 NameNode 瓶颈问题.

2) FaceBook 主要用 Haystack 系统存储图片数据. Haystack Directory 是目录的高速缓存, 提供了逻辑卷与物理磁盘的映射, 数据访问都经过此. 部分文件在 Haystack Cache 中缓存中, 它作为 Haystack 存储缓存区. Haystack 存储只需要一次磁盘 I/O 操作就可以完成文件读取. 在 Haystack 架构中, 引入内容分发网络(CDN)可以增加用户访问速度.

3) 针对 HDFS 本身提供了小文件存储方案 Hadoop Archives 文件(HAR 文件)是将多个小文件打包成一个大文件然后进行存储. 打包后的文件依然可以通过 MapReduce 应用程序操作, 打包后的文件由索引和存储两个部分组成, 索引记录了原有文件的目录结构和状态. 由于 HAR 文件不擅长删除与修改, 它只适合做小文件定期归档存储. Hadoop 中的 Sequence 文件可以支持小文件存储. Sequence 文件是由一系列<key,value>组成的, 可以通过将 key 设计成小文件名, value 为文件内容, 来把多个小文件合并成一个大文件.

在国网中, CIM 数据模型文件用来描述电气模型, 它可以构建出设备模型之间的拓扑关系, 一组模型包含 xml 描述和 svg 图片. 对于海量的小 CIM 文件 TFS 可以很好解决问题. 对于一些归档信息, 比如一些档案文件, 它的处理方式通常是通过 HDFS 的 HAR 文件方案解决.

3.2 不同热度文件存储策略

我们根据数据访问频率将数据分为热数据与冷数据, 热数据用户会经常查询, 访问频率高, 冷数据一般指归档历史数据, 访问频率不高.

冷数据通常只做归档数据, 一般地, 可以容忍较高的访问延迟, 这类数据存储方案比较简单, 可以存放在 HDFS 中.

热数据因为访问频次高、要求延迟低, 因此这部分数据可以缓存在国网现有的高端存储系统中. 除此之外, 可以向 Haystack 系统中一样, 引入热数据缓存机制, 并根据成熟的缓存命中算法, 提高热数据替换. 综上, 分级存储是提高性能有效途径, 文件存储系统可以使用 SSD+SAS+SATA 混合存储, 随着热点数据变化, 最热的数据存储在 SSD, 中等热度数据存储在 SAS, 轻热度数据存储在 SATA. 这样可以很好的结合 SSD 的性能与 SAS、SATA 的成本优势.

国网中，每天运行的系统成千上万，实时性的要求也是不尽相同很高的. 比如SCADA 系统，它是为电网调度提供可靠数据，它的实行性要求很高，而且其数据访问热度亦很高，通常这种数据放在访问速度最快的 SSD 中. 对于一些归档的历史数据，比如电网中报废的设备信息,则可以存放在 SAS 或 SATA 上.

3.3 总体存储系统演进

采用多种技术路线，构建面向不同文件大小、不同数据热度、不同访问延时的非结构化数据存储策略有以下演进轨迹:

(1) 前期

考虑到现有高端存储的利旧问题，及分布式文件系统在小文件存储和低延时数据访问等方面的问题，前期任保留高端存储，用于存放小文件、实时数据、在线数据、核心业务数据，将现有的大文件、非实时数据、离线业务数据、非核心业务数据逐步迁移至分布式存储上. 其架构如图 4 所示.

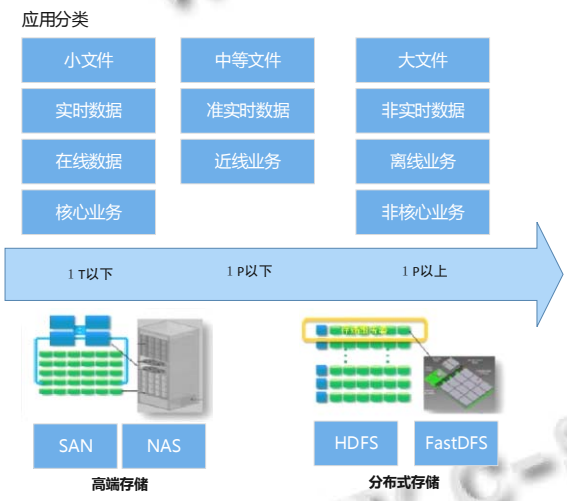

图 4 前期存储架构

(2) 中期

中期采用通过开源集成方式，采用不同的分布式存储产品(HDFS、TFS 等)，解决不同大小、不同延时要求、不同访问频率、不同重要程度的数据存储问题，同时通过自主研发方式，将现有的高端存储纳入到分布式存储的管理范围，最大程度上利用现有设备，减少投资. 其存储架构如图 5 所示.

(3) 后期

最后，通过采用统一的技术路线，选取几种开源的分布式文件系统(HDFS、TFS 等)，通过自主研发和创新，构建统一存储系统，使其满足各类业务数据的存储需求.其存储架构如图 6 所示.

图 5 中期存储架构

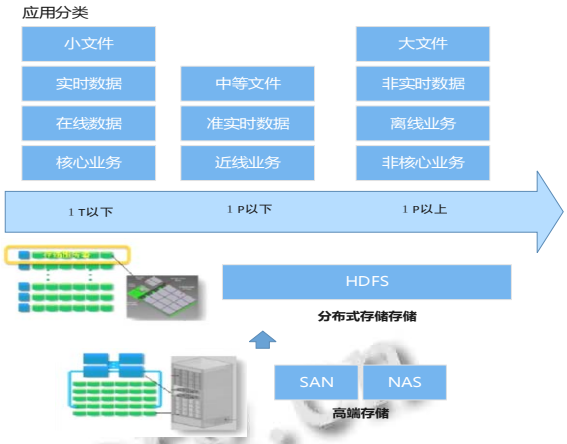

图 6 后期存储架构

4 总结与展望

未来国网需要构建全景实时的电网，这势必将产生大数据. 这些数据将会为电力设备状态检修、电网调度、电网自愈、信息孤岛互通提供良好支撑方案. 构建具有成本低、拓展性好(无限量存储)、可靠性高的大规模分布式文件存储系统将势在必行. 在数据实时性、一致性、安全性等方面依然具有严峻挑战，需要研究者们去找出更好的解决方案.

参考文献

1 张文亮,汤广福,查鲲鹏,等.先进电力电子技术在智能电网中的应用.中国电机工程学报,2010,30(4):1–7.

2 宋亚奇,周国亮,朱永利.智能电网大数据处理技术现状与挑战.电网技术,2013,37(4):927–935.

3 Wang PJ. D-pro: Dynamic data center operations with demand-responsive electricity prices in smart grid. IEEE Trans. on Smart Grid, 2012, 3(4): 1743–1754.

4 周晖,钮文洁,王毅.从缴费行为分析电力客户的信用度.电力需求侧管理,2006,8(6):12–16.

5 Conejo AJ, Morales JM, Baringo L. Real-time demand response model. IEEE Trans. on Smart Grid, 2010, 1(3):236–242.

6 牛东晓,谷志红,邢棉,等.基于数据挖掘的 SVM 短期负荷预测方法研究.中国电机工程学报,2006,26(18):6–12.

7 李晗,萧德云.基于数据驱动的故障诊断方法综述.控制与决策,2011,26(1):1–16.

8 周东华,胡艳艳.动态系统的故障诊断技术.自动化学报,2009,35(6):748–758.

9 Pregelj A, Begovic M, Rohatgi A. Quantitative techniques for analysis of large data set in renewable distributed generation. IEEE Trans on Power Systems, 2004, 19(3): 1277–1285.

10 IBM Corporation Software Group. IBM big data overview for energy and utilities. 2011-06[2012].

11 Versant. NoSQL and the smart grid big data challenge. 2012-08[2013-02].

12 Kligman D. PG&E's Austin kicks off conference on dealing with smart grid data.

13 杨传辉.大规模分布式存储系统原理解析与架构实战.北京:机械工业出版社,2014:66–67.

14 http://nutch.apache.org.

15 White T. Hadoop 权威指南.北京:清华大学出版社,2011:41–42.

16 http://hadoop.Apache.org/common/docs/current/hdfs-design.html.

17 Taobao.淘宝图片存储与 CDN 系统.http://wenku.baidu.com/view.

18 Taobao. Taobao File System. http://code.taobao.org/p/tfs/src.

19 Beaver D, Kumar S, Li HC, Sobel J, Vajgel P. Finding a needle in haystack: Facebook's photo storage. OSDI. 2010.