DOI:10.16836/j.cnki.jcuit.2015.03.009

文章编号: 1671-1742(2015)03-0254-05

基于 Hadoop 的海量气象雷达小文件存储研究

杨芙容， 王永丽， 王文明
(成都信息工程大学信息安全工程学院,四川 成都 610225)

摘要: 针对气象雷达观测产生的文件多、数据量大、计算复杂等特点,通过研究分布式存储计算平台 Hadoop,提出将雷达观测中实时生成的一次产品文件直接存储在 HBase 中; 而对立体扫描文件和计算生成的二次产品文件,采用 SequenceFile 技术合并处理后存储在 Hadoop 分布式文件系统中。根据雷达文件的标识特点,将 HBase 的行主键设计成时间 + 产品参数,利用随机散列与预分区保证 HBase 中 regionserver 的负载均衡; 通过与直接存储在 Hadoop 分布式文件系统的方案进行比较,从 NameNode 的内存占用和文件读取时间两个方面进行测试和方案评估。实验证明该方案节省约 60% 的 NameNode 的内存空间,同时提高近 18% 的文件读取速率。

关 键 词: 计算机应用; 大数据; 雷达数据; 小文件问题; 分布式存储; Hadoop; HBase; SequenceFile
中图分类号: TP311 文献标志码: A

0 引言

近年,气象现代化业务飞速发展,为广大用户提供更加全面、更加精细化的气象服务,其中,气象雷达近年呈高速发展的态势, 受到世界多数国家和国际气象组织的高度重视[1]。尤其是多普勒天气雷达技术日益普遍的使用,获得了更多的探测数据信息,提高了天气的监测能力,对预报强对流天气等具有非常重要意义。同时,气象现代化造成气象数据的成倍增长,对系统的存储和处理能力提出很高的要求。

传统的关系型数据库系统在存储和管理数据上出现负载饱满读写性能不理想等问题[2],迫切需要新的海量数据处理方案,在众多的海量数据处理平台中,Apache Hadoop[3] 已经迅速成为存储和处理海量数据的首选平台之一。国际上著名的互联网门户网站雅虎、社交网络服务网站 FaceBook,以及国内淘宝、百度等众多知名的 IT 企业,先后采用 Hadoop 开发出适合自己的数据管理平台。Hadoop 具有吞吐量大、效率高、容错性高、可靠性高、成本低等优点,能够扩展到云环境,适用于气象雷达这种有多种处理需求的应用场景。

1 Hadoop 核心组成

Hadoop 是 Apache 软件基金会旗下的一个开源分布式计算平台,核心组件包括 HDFS、MapReduce、HBase 等[4], HDFS 形成 Hadoop 的文件分布式系统; HBase 是一个面向列的分布式数据库,可靠性高、可伸缩、提供实时读写; MapReduce 是 Hadoop 的数据处理模块,能对存储在 HDFS 和 HBase 中的大规模数据进行高效的分布式并行处理。

HDFS 采用主从式架构设计模式(master/slaver structure),一个名称节点(NameNode) 和若干数据节点(DataNode) 构成 HDFS 集群。NameNode 是 HDFS 中的管理者,负责存储和管理文件系统的命名空间(namespace),文件目录的元数据,集群配置(cluster configuration) 和存储块的复制情况等信息。Namespace 是一个分层结构的文件和目录,记录权限,修改和访问时间,命名空间和磁盘空间配额等属性。DataNode 是 HDFS 的基本组成部分,提高文件数据的存储服务。它将文件以分块形式存储到本地文件系统,并周期性地将所存块的信息发送给 NameNode。

HBase 的设计源于 Big Table 的启发,在 HDFS 上开发的一个分布式数据库。数据集,主要存储处理大规模的非结构化和半结构化的数据。Hbase 使用 LSM 树型结构,将需要修改的数据直接写入内存。操作数据时直接定位到相应的 HRegion server,在 region 上找到匹配的数据,而且引入高速缓存,使 HBase 能提供实时计算服务。

2 小文件问题

把海量小文件(小于 16MB 的文件) 直接存储在 HDFS 上存在的问题简称为小文件问题[5]。为节省内存空间和提高传输效率,气象雷达文件在存储和使用前通常先做压缩处理,压缩后文件大小在几 KB 到几百 KB 之间,属于小文件范畴,这就可能引发小文件问题。

(1) NameNode 内存溢出

收稿日期: 2015-05-23
基金项目: 四川省科技厅应用基础资助项目(2014JY0093)

NameNode 直接把文件系统的元数据存储在主存中，通常一个文件的元数据占 250bytes 内存，每个块的 3 个复制文件的元数据占 368bytes[6]，文件数目越大，占用的 NameNode 内存空间也成倍数增加。此外，DataNodes 定时向 NameNode 发送块报告信息，NameNode 收集这些信息，作为块映射信息存储在内存中。当文件数目很大时，这部分信息所占内存不容忽视。由式(1)可以看出减少 NameNode 内存占用的主要方法是减少 NameNode 管理的文件个数和块数目。

$$M_{NN} = 250 \cdot N + (360 + \beta) \cdot \sum_{i=1}^{N} \left[\frac{Li}{HBS} \right] + \alpha \qquad (1)$$

其中 M 是 NameNode 的内存占用，N 文件数，β 代表块的映射信息，HBS 是设定的块大小，默认取整为64 M，Li 是每个小文件的大小，$\left[\frac{Li}{HBS} \right.$ 表示比值向上取整，a 为 NameNode 所占内存，这部分内存很小。

(2) 访问延迟高

当读取大量的小文件出现高访问延迟主要有 3 个原因: 首先，HDFS 客户端每操作一个文件都需要访问一次 NameNode 元数据，引发元数据服务器的频繁相互作用的延迟。如640 MB的大文件(10 个64 MB文件) I/O 吞吐量，HDFS 客户端只需要访问 NameNode 5 次，而在相同大小下(如 10240 个64 KB文件)，客户需要访问 NameNode 5120 次，这种延迟很明显。其次，HDFS 失去块间关系。HDFS 有自己的放置策略，可以在可扩展性、读效率、写带宽之间达到很好的平衡。但没有考虑文件的连续性，连续文件不一定连续放置，甚至可能被放在不同的块中[7]。再者，HDFS 目前没有提供预取函数降低 I/O 延迟，没有为数据的放置和预取机制考虑文件的相关性，从 HDFS 中读取小文件时，通常需要多次查找和在 DataNode 和 DataNode 之间反复来回跳转。

(3) Map 任务过多

在 HDFS 中文件按块存储，文件不满64 MB(Block 默认64 MB) 也占用一个 Block。Map 任务通常一次处理一个块输入，这样 Block 数目越多，Map 任务越多，而实际上，每个 map 只处理很少的数据量，这就造成额外的簿记开销，大量时间浪费在任务启动和结束上。如处理1 GB的一个大文件需要 16 个64 MB的块，而处理1 GB的 1 万个左右100 KB的小文件需要 1 万个64 MB的块。这 1 万个小文件的 map 效率要比大文件 map 效率低几十到几百倍。

3　优化方案设计

目前，基于 HDFS 的海量小文件存储和访问效率问题的解决方法主要分为 3 类。

(1) 利用 Hadoop 自身提供的方法合并小文件，如 hadoop 归档[8] (Hadoop archive，HAR)、SequenceFile[9] 等，但这些方法存在很多问题，最突出的是访问效率低下。

(2) 针对具体的应用提出对应的文件合并、组合方法，通常在 HDFS 上添加一个小文件处理模块[10]，由处理模块完成数据的合并、更新、删除、缓存等操作: Xuhui Liu 等[11] 将地理位置信息相邻的多个小文件合并成一个大文件，为检索到小文件的位置，为其建立一个全局的位置索引，减少 NameNode 的内存占用。但是当小文件数据量非常大时，全局的位置索引文件就变得十分庞大，每次定位一条数据耗费大量时间，文件的检索速率低; Bo Dong 等[12] 提出将同一个课程 PPT 的多个图片小文件合并成大文件，在 DataNode 上为每个文件建立一个本地的单独索引查询方法，分担 NameNode 的查询负担。但合并文件超过默认块大小时，该方法比较复杂，需要人为干预小文件的合并和建立索引操作。

(3) 改进系统的架构。文献[13]提出一种增强型的 HDFS，扩展单一的 NameNode 成分层 NameNode 的架构，同时扩展的 HDFS 架构中集成高速缓存，提高文件的读取效率。刘小俊等[14] 提出将 RDBMS 和 Hadoop 结合，前端 RDBMS 作为小文件访问入口和合并工具，后端 Hadoop 存储和处理海量数据，发挥各自的优势。但总的来说，改进架构非常复杂，成本高而且较难实现。

3.1　基本思想

多普勒天气雷达通常每几分钟(6 min) 完成一个体扫，每个体扫文件里包含 $N \cdot (Z、V、W)$ 一次产品文件、一个 VOL 和根据需要设定的多个二次产品文件，其中 N 表示每次抬高的仰角层数，一天 24 小时不间断采集雷达数据。

为节省磁盘空间和网络带宽，把每个体扫的产品文件采用压缩方式存储。针对气象雷达数据的特征，压缩后将文件格式命名为: 年月日_时分秒. 扫描层号. 产品 Type. 扫描模式. 产品参数 20141214 _ 125233. 00 . 004. 000 _ 0. 47. zdb，表示 20141214. 12: 52: 33. 表示: 第 0 层. v 产品. 000 扫描式. 0. 47 产品参数(仰角) 的一个 V 产品文件。

表 1　压缩文件命名格式

年月日 _ 时分秒	扫描层号	产品 Type	扫描模式	产品参数
20130123 _ 114513	00	003	000	0. 47

压缩后体扫文件大小在几 KB 到几百 KB 之间，其中一次产品文件压缩后文件大小为通常几到几十 KB，这部分数据对系统的数据处理实时性要求比较高，在

存储时选择 HBase。HBase 基于 HDFS 之上,可以将操作分散到多个节点上并行处理,执行效率很高,能够提供实时计算服务。其他的 VOL 文件和二次产品文件通常对系统的实时性要求不高,是由一次产品文件根据需要计算整合。对于这些文件采用 SequenceFile 合并技术,以 VOL 文件和二次产品文件的文件名为 key、文件内容为 value 的形式进行合并,减少元数据的内存占用量,节省 NameNode 的内存空间。全文存储系统架构模型如图 1 所示。

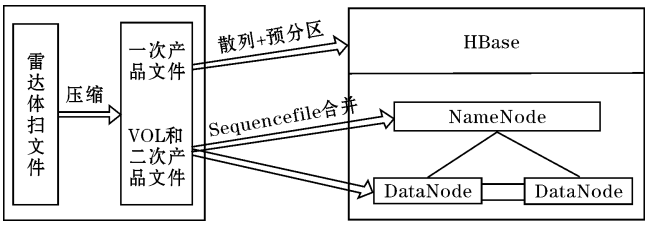

图1 存储系统架构模型

3.2 一次产品文件的存储

HBase 无模式无类型,由行和列组成。行和列的坐标交叉决定表格单元格(cell)。cell 的内容是未解释的字符数组。HBase 具有很好的可伸缩性,表可以数十亿个数据行"高";表可以很"宽"(数百万个列)。表的模式直接反映存储形式,可以提供高效的数据结构的序列化、存储和检索。HBase 处理几十 KB 的小文件时,查询效率很高[15],把 ZVW 一次产品文件直接存储在 HBase 中有利于提高文件的读取效率。

HBase 行键设计:HBase 通过行键(row key)、列族(Column Family)、列和版本 4 个维度定位某条数据,它不能像传统数据库一样支持 where 等条件查询,只能全盘扫面或按 row key 检索数据。HBase 存储模型设计时,首先需要根据业务设计行键,检索记录的主键,可以利用其存储排序特性提高性能。

在天气雷达业务中,雷达产品文件一般以时间和产品参数作为标识,这两个维度也是常用的检索条件,因此设计时间 + 产品参数的行键设计方案。为能将 regionserver 的负载均衡,防止出现所有新数据都在一个 regionserver 上堆积的现象,设计 rowkey 时采用随机散列与预分区二者相结合的方法。预分区开始就预建好一部分 region,这些 region 都维护着自己的 start-end keys,由 MD5 方式生成随机字符串,写数据能均等地命中这些预建的 region,解决写热点问题,大大地提高性能。

HBase 列族设计。HBase 表中列族需要提前定义,是表 chema 的重要组成部分。不同列族的数据是存储在不同的 HFile 中,所以一般把同时访问的数据放在一个列族中,而在气象雷达应用中,因为业务需要访问全要素,所以把同一类型的数据都放置在一个列族中;以首字母和产品参数作为列族和列的标识符。例如列 *FP*: *Z*、*FP*: *V* 都属于 *FP*(FirstProduct)这个列族。

3.3 VOL 文件和二次产品文件的存储

由于 VOL 文件和二次产品文件的存储和访问对系统的实时性要求不高,而且经过压缩后的 VOL 文件和二次产品文件大小通常为几十到几百 KB,采用 Hadoop 提供的应用程序编程接口 API: SequenceFile[16],主要由一个 Header 后跟多条 Record 组成。Header 主要包含版本、Key、value 类名、压缩方式判断、自定义信息和同步标识等,同步标识用于快速定位到记录的边界。每条 Record 以键值对的方式进行存储,表示字符数组。

SequenceFile 根据压缩与否有 3 种存储形式,Value 压缩、block 压缩和不压缩存储,通过 SequenceFile 类的内部类 CompressionType 表示。采用不压缩方式存储 io. seqfile. compression. type = NONE。

SequenceFile 通过创建 SequenceFile. Write 实现写入,然后调用 writer. append(key, value)打包记录。它就像为存储提供一个容器,将多个小文件打包统一存储。在存储气象雷达数据时,采用 SequenceFile 技术将文件进行合并处理,将 VOL 和二次产品文件的文件名作为 key,文件的内容作为 value 序列化合并成一个大文件,合并后的文件元数据信息存储在 NameNode 中,节省 NameNode 的内存空间。

4 实验测试

4.1 测试环境

测试使用 4 台服务器构建集群,其中 1 台作为主服务器 master,3 台从节点 node1、node2、node3;服务器配置 Intel(R) Core(TM) 2 Quad CPU Q8200 @ 2.33 GHz 2.34 CHz,内存 4GB 硬盘 500GB,操作系统 Red Hat Enterprise Linux Sever release 6.5,Hadoop 版本是 1.2.3,HDFS 副本数 dfs. replication 设置为 3,HDFS 最小块设置为默认值64 MB,SecondNamenode 配置在 node1 上。HBase 版本是 0.94.1,使用 HBase 自带的 zookeeper。

4.2 实现方法

实验测试中数据是某台多普勒气象雷达 2014 年 6 月观察的数据,701134 条记录,压缩后总大小为 11.2 GB,每条记录都在1 ~ 500 KB,其中一次产品文件压缩文件大都小于100 KB。

NameNode 的内存受限问题一直是制约其对海量

小文件存储的关键因素，因此减少 NameNode 的内存占用具有重要意义。HDFS 设计建立在更多地响应“一次写入，多次读出”任务的基础之上，因此提高文件的读取速率也至关重要。故从 NameNode 的内存占用和文件读取时间两个方面进行测试和评估方案。这里把提出的方法和直接存储 HDFS 方案进行比较。

4.3　实验结果

NameNode 内存占用(MB/文件个数)

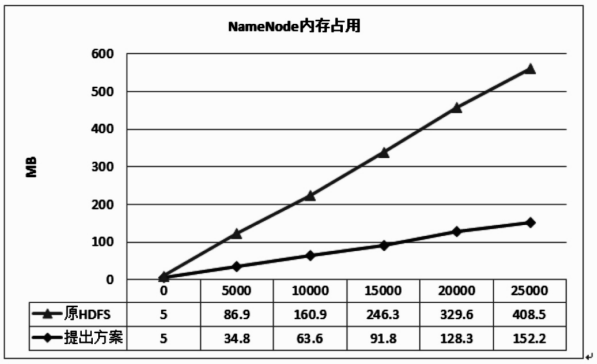

图 3　NameNode 内存占用

图 3 表示 2 种方案对不同数量的文件存储时，NameNode 内存使用情况，文件数量由 0、5000、10 000 依次递增至 25 000。绿色线表明在对小文件不做任何处理直接存储时，随着小文件数目的增加，内存占用量呈现线性增长趋势。采用将 VOL 文件和二次产品 SequenceFile 合并技术打包存储极大地减少元数据的内存占用，同时一次产品文件是存储在 HBase 上的，节省了 NameNode 的内存空间。

读操作(s/文件个数)

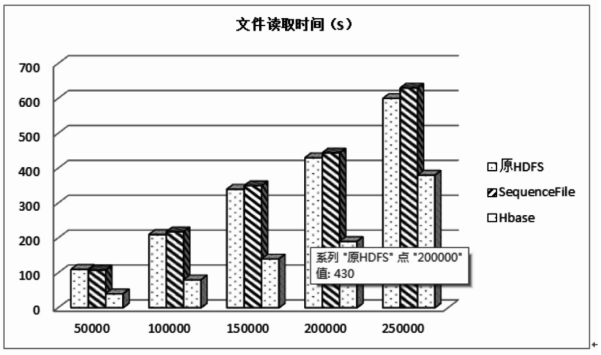

图 4　读取时间

由于气象数据在存储和调用时多数时间操作时间连续的那些文件，所以分别测试时间连续的50 000 ~ 250 000个文件。而在存储方案中把一次产品和其他产品文件采用两种不同的方式存储，在测试读取时间时需要分别讨论。分别将 HBase 和 sequenceFile 访问时间与直接 HDFS 读取时间比较。测试结果表明，HBase 对一次产品文件的响应时间远小于直接从 HDFS 读取气象文件的时间。而对气象雷达数据的操作频繁牵涉到对一次产品文件的访问，HBase 提供实时操作服务，极大地提升文件的访问速率，节省了文件的读取时间。

5　结束语

针对气象雷达数据的特征，先把每个产品文件采用压缩方式存储，压缩后每个产品文件大小为几 KB 到几百 KB，节省了 DataNode 的内存空间。由于每个体扫中的一次产品文件对数据处理的实时性要求比较高，针对这些特殊原因，在存储时选择 HBase，它提供实时计算，处理速度非常快。采用时间 + 产品参数的行主键设计方案，利用随机散列与预分区保证负载均衡，最终提高一次产品文件的读取效率。

此外，一个体扫周期除了生成 ZVW 一次产品文件，还生成一个 VOL 文件和根据需要计算生成多个二次产品，对系统的实时性要求不高，而且大小通常为几十到几百 KB。对于这些文件直接存储在 HBase 上加重其负担，同时 HBase 更适合处理几十 KB 的文件，文件过大降低其处理速度，所以利用 Hadoop 自身提供的 SequenceFile 技术先小文件合并为大文件，NameNode 中只需要存储合并后的文件元数据，节省 NameNode 的内存空间。

HBase 不支持辅助索引，但在气象数据检索时效性上仍需要使用辅助索引[7]，在接下来的工作中，将考虑设计适合气象数据的辅助索引模块; 同时对 SequenceFile 中的 VOL 和二次产品文件设计索引和查找算法，以提高其查找速度。

参考文献:

[1] 伍志方，曾沁，易爱民，等. 短时大暴雨的多普勒雷达探测及暴雨预警信号发布 [J]. 灾害学，2006，21(2)：59 - 63.

[2] 陈东辉，曾乐，梁中军，等. 基于 HBase 的气象地面分钟数据分布式存储系统 [J]. 计算机应用，2014，34(9)：2617 - 2621.

[3] Shvachko K，Kuang H，Radia S，et al. The Hadoop Distributed File System [C]. In: IEEE 26th symposium on mass storage systems and technologies (MSST). IEEE; 2010: 1 - 10.

[4] Jilan Chen，Dan Wang，Lihua Fu，et al. An Improved Small File Processing Method for HDFS [C]. International Journal of Digital Content Technology and its Applications(JDCTA). 2012，(6)：

200 – 205.

[5] Tom White. The Smal Files Problem [EB/OL]. Http://www. clouder. com/blog/2009/02/02/the small files problem/.

[6] Tom White. The Smal Files Problem [EB/OL]. Http://issues. apache. org/jira/browse/Hadoop – 1687.

[7] Bo Dong, QinghuaZheng, FengTian, et al. An optimized approach for storing and accessing small files on cloud storage [C]. elsevier. 2012, (6): 1847 – 1862.

[8] Chatuporn, Vorapongkitipun, Natamut, Nupairoj. Improving Performance of Small File Aceessing [C]. International Joint Conference on Computer Science and Software Engineering (JCSSE) 2014 (11): 200 – 205.

[9] 赵晓勇,杨扬,孙莉莉,等. 基于 Hadoop 的海量 MP3 文件存储架构 [J]. 计算机应用,2012,6(1): 1724 – 1726.

[10] K P Jayakar, Y B Gurav. Managing Small Size Files through Indexing in Extended Hadoop File System [C]. International Journal of Advance Research in Computer Science and Management Studies. 2014, 8(8): 161 – 167.

[11] Liu X, Han J, Zhong Y, et al. Implementing web-gis on hadoop: a case study of improving small file i/o performance on hdfs. [C] In: IEEE international conference on cluster computing and workshops, 2009, 12, (1): 1 – 4.

[12] BoDong, Qiu Jie, Qinghua Zheng, et al. A novel approach to improving the efficiency of storing and accessing small files on hadoop: a case study by PowerPoint files [C]. Proceedings of the 7th Int ernational Conference on Services Computing. Piscataw ay, N J, USA: IEEE, 2010: 65 – 72.

[13] Xiayu Hua, Hao Wu, Zheng Li, et al. Enhancing throughput of the Hadoop Distributed File System for interaction – intensive tasks [C]. 2014: 2770 – 2779.

[14] 余思,桂小林,黄汝维,等. 一种提高云存储中小文件存储效率的方案 [J]. 西安大学报, 2011, 45(6): 60 – 63.

[15] Ganggang Zhang, Min Zuo, Xinliang Liu, et al. Improving the efficiency of storing SNS small files in HDFS [C]. CCIS 426. 2014: 154 – 160.

[16] 薛胜军,删寅. 基于 Hadoop 的气象信息数据仓库建立与测试. 计算机测量与控制 [J]. 2012, 20(4): 926 – 928.

[17] Chen P, AN J. The key as dictionary compression method of inverted index table under the HBase database [J]. Journal of Software, 2013, 8(5): 1086 – 1093.

Hadoop-based Storage Research for Massive Weather Radar Small File

YANG Fu-rong, WANG Yong-li, WANG Wen-ming

(College of information safety Engineering, Chengdu University of Information Technology, Chengdu 610225, China)

Abstract: According to the special features of weather radar data, such as many files, huge datas, complex computation, by study of the Hadoop distributed computing platform, a solution using HBase to store the radar data first products in real time is proposed. While Calculating the radar data resulting secondary products and volscan files, there adopted Sequence File technology to package files into a larger one, after that it is stored on The Hadoop Distributed File System (HDFS). Based on the characteristics of radar identification documents, HBase's row key was designed into time and product parameters. In order to ensure the regionserver's load balancing, random hash and pre-partition was used. By comparison with the proposal that the files are stored directly in Hadoop Distributed File System, NameNode memory footprint and file read time were regarded as a standard to test and evaluate the quality of the program. Experiments showed that this method can save 60% of NameNode memory footprint and improve 18% efficiency of the file to read.

Key words: computer application; big data; radar data; small files problem; distributed storage; Hadoop; HBase; Sequencefile