doi: 10.3969/j.issn.1006-1576.2010.07.030

分布式海量数据管理系统 Hypertable 底层存储结构分析

程俊[1]，杨卓宁[2]，费江涛[2]
(1. 总装重庆军代局驻绵阳地区军代室，四川 绵阳 621000；
2. 北京航天飞行控制中心，北京 100046)

摘要：通过分析 Hypertable 的源代码，描述了 CellStore 存储结构，介绍其读写流程，总结了该结构存在的缺陷，并提出了优化思路。优化步骤主要包括：将关键字数据进行合并，建立关键字到数据值的索引结构；并设置一个关键字快表，用于存储访问频率较高的关键字数据。结果表明：该优化结构可减少冗余数据，提高数据的读写效率。

关键词：Hypertable；CellStore；单元行；缺陷
中图分类号：TP311.12　　**文献标识码**：A

Analysis of Lower Store Structure in Hypertable System for Massive Data Distributed Management

Cheng Jun[1], Yang Zhuoning[2], Fei Jiangtao[2]
(1. Mianyang Representative Office of Chongqing Military Representative Bureau of PLA General Equipment Department, Mianyang 621000, China; 2. Beijing Aerospace Command & Control Center, Beijing 100046, China)

Abstract: By analyzing the source code of Hypertable, the Cellstore's storage structures have been described. The procedures of reading and writing operations have been introduced. The disadvantages of aforementioned structure have been identified; and the development procedures have been recommended. The development procedures include: combining phase information; building up a phase information table, in order to store high-frequency data. The result shows that the optimal structure can reduce overloaded data in order to improve the efficiency and performance of data reading and writing operations.

Keywords: Hypertable; CellStore; cell; default

0　引言

Zvents 公司开发的 Hypertable[1]是一个大规模的结构化数据分布式存储系统，用于存储和管理海量的网络数据。系统采用 CellStore 作为底层存储结构，是系统存储数据的关键部分，直接影响系统的磁盘使用量和读写效率。由于 Hypertable 系统仍处于测试开发阶段，系统的 CellStore 存储结构设计简单，还具有较大的数据冗余。故在分析 Hypertable 的源代码及 CellStore 存储结构的基础上提出该结构的优化思路。

1　存储结构

1.1　单元行结构

在 Hypertable 系统中，数据存储的基本单元是关键字和值组成的数据对，叫做单元行（Cell）。值是任意长度的字节串，关键字是值的多维索引。

如图 1，关键字包括：行关键字、列家族编码、列限定词、标志位和反序时间戳，其中，行关键字是任意长度的字符串。Hypertable 系统对表的所有列家族进行顺序编码，即列家族编码，列家族编码是单字节的字符。列限定词是任意长度的字符串，表结构没有对其进行定义，在数据写入过程中，由用户添加列限定词。标志位用于标识此行数据的删除格式。CellStore 的数据均以添加形式写入，数据一经写入，不再删除。因此，用户通过添加删除标志，删除对应行、列家族、列限定词或单元行的数据。标志位的类型包括[2]：

```
FLAG_DELETE_ROW                 = 0x00;
FLAG_DELETE_COLUMN_FAMILY       = 0x01;
FLAG_DELETE_CELL                = 0x02;
FLAG_INSERT                     = 0xFF;
```

行关键字	列家族编码	列限定词	标志位	反序时间戳

图 1　Key 的结构图

反序时间戳是对用户提交时间戳取反，以字符串的形式保存。取反的目的是将数据按照时间戳降序排列，保证最新的数据存储在前面。

1.2　单元库结构

单元库对单元行数据进行永久存储，包括单元索引映像和单元库文件 2 种结构。单元库文件是位

收稿日期：2010-04-06；修回日期：2010-05-12
作者简介：程俊（1984-），男，重庆人，2006 年毕业于中国人民解放军装备指挥技术学院，从事电子产品质量综合管理监督、电子产品相关标准研究。

于底层文件系统上的文件，它对表数据进行有序的连续存储。单元库文件中的数据由数据块组成，数据块在写入之前附以特定的数据块头。

单元库文件中包括 4 种数据：表数据、索引数据、偏移量数据和跟踪器（Trailer）数据。表数据是按照关键字进行排序的单元行数据；索引数据和偏移量数据实现对表数据块的索引和定位，其中，索引是每个数据块最大关键字，偏移量是每个数据块头相对文件偏移位置；跟踪器是在数据写入库文件过程中，对库文件的属性进行记录的对象，所记录的信息包括：总索引条目数、总单元行数、块大小、压缩率、压缩类型和跟踪器版本号，以及索引数据、偏移量数据和跟踪器数据在库文件中偏移量。

单元库文件包含 3 种数据块头：表数据块头、索引数据块头、偏移量数据块头，用于标识不同的数据，以进行数据的合法性检验。

在数据写回过程中，系统按照关键字顺序依次写入单元行数据，并将其进行分块压缩。如图 2，表数据块存储在库文件的前部，索引数据和偏移量数据依次保存在表数据的后面，文件尾部存储的是跟踪器数据。由于跟踪器数据的大小是固定的，系统无需保存它的偏移位置。

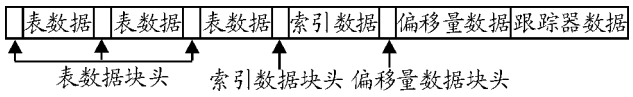

图 2　CellStore 文件的数据结构图

单元索引映像以有序映射的形式保存索引关键字和对应的偏移量，位于内存中。在进行数据读取时，系统将索引数据加载到此映像结构中，以提高系统重复查找的速度。

2　写操作流程

在 Hypertable 系统中，表数据的写入操作包括 4 个阶段：1) 客户端收集更新数据并分组提交；2) 数据服务器接收更新数据，解析并分配给所在的 Range 对象；3) Range 对象将数据添加到对应的访问组对象中，由访问组对象完成数据的写入；4) 访问组对象执行数据整理操作，将缓存的更新数据写回到单元库文件中。

数据整理操作包括：小型整理（Minor Compaction），合并整理（Merge Compaction）和大型整理（Major Compaction）[3]。3 种整理操作的特点是：读取的数据都是有序的，这将会提高数据的写回速度。整个写操作的最后一步是将读取的数据写回到一个新的库文件中。

在 Hypertable 系统中，CellStore 对象设置了 3 个缓冲区和 1 个跟踪器，如图 3。表数据缓冲区用于保存写回的单元行数据，将数据整理成字节串的形式。索引缓冲区和偏移量缓冲区分别保存索引数据和偏移量数据。系统设置缓冲区的作用是：统一数据格式，将数据分块，集中写回，减少底层写操作的次数。

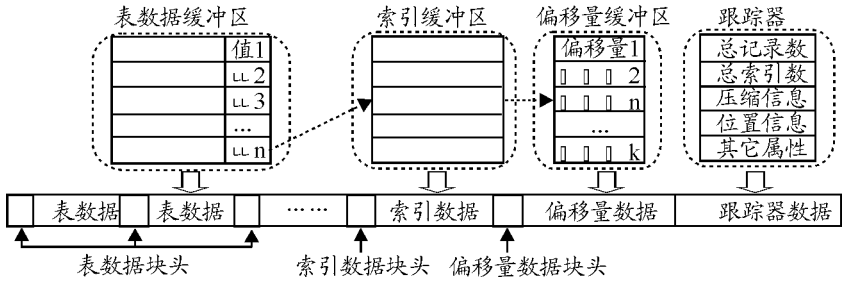

图 3　CellStore 数据写入示意图

当 CellStore 对象接收到数据时，创建一个新的库文件，并清空缓冲区和初始化跟踪器。在数据写回过程中，跟踪器不断记录单元行数目和索引数目。首先写回的是表数据，系统将单元行数据直接写回到表数据缓冲区。这样，多个单元行会含有相同的行关键字、列关键字等，导致关键字数据重复写入。

当表数据缓冲区满时，系统将数据块写回到文件中，同时记录最大的关键字作为索引关键字。这个索引关键字只能标识块中的数据范围，不能确定行关键字和列关键字所对应数据的位置。因此，索引关键字对数据的定位是不准确的。

当所有表数据写完后，系统依次将索引数据、偏移量数据和跟踪器数据写回到文件中。

3　读操作流程

如图 4，用户对数据的读取最终将定位到读取单元库文件，系统将单元缓存和单元库的数据进行合并，返回给上层的数据扫描器。

在介绍读操作原理前，首先对查找条件对象 ScanContext 进行分析。ScanContext 对象指定了需要读取的内容，包括：起止行、行数、列家族、数据的有效时间和版本数等。系统查找起止行范围内

的数据，并将数据和查找条件进行对比，判断是否满足用户需求。

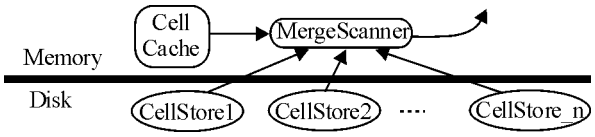

图 4 访问组对象的数据读取流程图

在系统进行数据查找之前，首先打开单元库文件，同时加载跟踪器数据。为了实现数据块的定位，系统将索引数据和偏移量数据加载到单元索引映像中。系统在索引数据中查找所需数据位置。由于索引数据记录的是数据块最大关键字，系统只能获取某一范围内的表数据块位置，无法确定此数据块中是否包含所需数据。然后，系统读取查找到的多个数据块，并逐条解析数据。最后，将数据记录返回给上层扫描器（MergeScanner）。由上层扫描器判断记录的关键字是否满足查找条件，过滤掉无用数据。

从上面的处理流程可见，系统逐条查找所需数据，数据读取效率很低。由于索引关键字的定位不够准确，系统可能会从底层文件系统中读取大量的无用数据，造成系统资源的浪费。

4 CellStore 缺陷

CellStore 作为底层的存储结构具有很多缺陷，突出表现为存储结构简单，数据冗余大，读写效率较低。下面对 CellStore 结构的缺陷进行详细分析：

1) 单元库文件以单元行的形式存储了用户提交所有数据，存在着大量的数据冗余。从图 1 中看出：单元行包括了多维关键字，仅仅索引了单列的单版本数据。在 Hypertable 系统中，一个表包含不超过 256 个列家族，一般为几十个；一个列家族包含多个列限定词；一个列包括多个版本的数据。用户提交一行数据，库文件需要保存上千次行关键字，数百次列家族编码和多次列限定词。底层文件系统需要花费大量的磁盘空间保存冗余的关键字数据。

2) 单元库文件通过索引关键字数据对数据块进行定位，存在 2 个问题：其一，数据的定位不够准确，由于索引关键字只是数据块中的最大关键字，系统无法通过它判断所需的数据行是否存在。其二，系统中的单元库文件不断整理合并，单个文件的数据块数目不断增加，索引数据量将随之增大，最终索引数据量可以达到 GB 级，甚至更大。库文件的查找操作需要将索引数据全部加载到内存中，将花费大量的内存空间，严重影响系统的运行效率。

3) 库文件的读取效率低。系统逐条读取数据块中的记录，进行简单的条件判断后，提交给上层扫描器重新判断，系统花费了过多的时间用来重复读取数据。特别是在查找失败的情况下，用户所需的数据不存在，由于索引定位方法不够准确，导致系统从底层文件中读取多个无效的数据块，造成系统资源的浪费。

4) 由于系统使用分布式文件系统进行数据存储，大量的冗余数据需要额外的网络数据传输，影响了库文件的写操作效率。

5 优化思路

针对 CellStore 结构存在的问题和缺陷，采用多重索引结构进行优化，利用文件偏移量对数据块进行定位。

如图 5，优化的 CellStore 结构内所有行关键字有序排列，每个行关键字指向所属的列家族列表，用户指定的列限定词、时间戳和数据值链接到对应列家族指针上。系统通过查找指定的行关键字，确定所属列家族的内容和数量，然后准确定位到用户所需数据。特别是用户的多数读操作以行或列家族为单元，这种结构可以提高数据定位的准确度，从而达到更高的读速度。

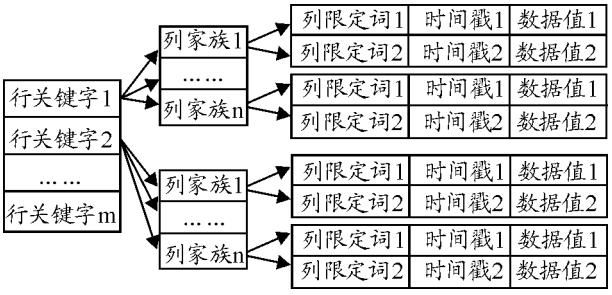

图 5 CellStore 优化结构图

6 总结

描述了 Hypertable 系统的底层存储结构 CellStore，通过详细分析单元行结构的读写流程特点，总结了 CellStore 结构的存在的缺陷：重复存储多维关键字浪费了磁盘空间，索引关键字无法准确定位数据，查找效率较低，冗余的数据影响了系统的读写效率。最后，提出了的优化的 CellStore 结构。

参考文献：

[1] Google Wiki. Overview of Hypertable Architecture [EB/OL]. (2008-5-23). http: //code.google.com/p/hypertable/wiki/ArchitecturalOverview.

[2] Doug Judd. hypertable-0.9.0.4-alpha[CP/OL]. (2008-3-12). http://www.w3c.org/TR/1999/REC-html401-19991224/loose.dtd.

[3] Chang F. Dean J. Ghemawat S. Hsieh W.C. Wallach D.A. Burrows M. Chandra T. Fikes A. Gruber R Bigtable: A distributed structured data storage system[C]. In 7th OSDI, 2006.