第36卷 第20期 计 算 机 工 程 2010年10月
Vol.36 No.20 Computer Engineering October 2010
·软件技术与数据库· 文章编号：1000—3428(2010)20—0061—03 文献标识码：A 中图分类号：TP311

一种结构化数据缓存方法

周进刚[1]，邢铁军[1]，纪 勇[1]，赵大哲[2]
(1. 东软集团(大连)有限公司基础软件事业部，辽宁 大连 116085；2. 东北大学计算机软件国家工程研究中心，沈阳 110004)

摘 要：针对现有报表缓存方法在内存消耗和访问速度上相冲突的问题，提出一种结构化数据缓存方法，把具有行列结构的报表数据分块存储到文件中。以文件形式存储的数据被划分为索引区和数据区，通过数据分块算法和写操作将报表数据缓存入文件。在读取报表数据时根据索引区可以直接定位到所在的块，在块中快速查找所需要的数据，从而在内存消耗和访问速度上达到优化。
关键词：缓存；报表；索引；数据分块；结构化数据

Caching Approach for Structured Data

ZHOU Jin-gang[1], XING Tie-jun[1], JI Yong[1], ZHAO Da-zhe[2]
(1. Platform Software Division, Neusoft Corporation, Dalian 116085, China;
2. National Engineering Research Center for Computer Software, Northeastern University, Shenyang 110004, China)

【Abstract】 For the conflicting problem between memory assumption and access speed caused by existing caching methods of report, a caching method is provided for structured data. The main characteristic of such method is caching the report data which has the structure of row and column to file as data blocks. The data to be cached as file is partitioned into indexing area and data area, then written to the file via a data blocking algorithm. Using such approach, the need data can be retrieved from the located block by index, and the data rapidly. Thereby, an optimized result can be obtained to the memory assumption and access speed problem.
【Key words】 caching; report; index; data block; structured data

1 概述

报表是一种具有行列结构的数据表示形式，表示的是一个时间点上业务数据的展现。在很多应用系统中，系统运行结果通过报表来展示，报表已经成为信息系统不可或缺的一部分[1]。

在实际应用中，当系统生成报表后，用户仍有一些对于这个报表的处理，如翻页、打印、导出PDF、Excel格式文件等。在此过程中，如果系统每次重新生成报表，则会增加报表服务器、数据库服务器的负荷，并增加用户的等待时间；同时在两次生成报表的过程中，数据库中的数据可能发生变化，这样就会造成用户通过浏览器看到和打印出的报表内容不一致。因此，一个高效、完善的报表服务器必须具有缓存报表数据的能力。

对于报表数据的缓存，目前有2种典型的方法：(1)把报表数据完整写入内存，然后通过“最近最少访问”或者“加权”等算法[2-5]，管理维护报表的缓存列表；(2)把报表数据完整写入硬盘等外部存储介质，每次在对报表数据访问前，从硬盘加载报表数据。

上述2种方法为应用系统提供了不同的报表缓存机制，但都存在缺点：

第(1)种方法由于把报表数据写入内存，对于并发量大、报表数据量大的应用，会大量消耗应用服务器的内存，导致服务器效率降低，甚至出现内存溢出、系统宕机等现象。

第(2)种方法虽然避免了系统内存的大量消耗，但是每次访问硬盘读取整个报表内容的响应时间远高于从内存读取的操作时间，同时硬盘文件的读写会占用大量的CPU时间，从而影响整个系统的性能。

因此，需要综合考虑I/O与存储等多方因素[6]。本文提出一种数据缓存方法，以解决现有的报表缓存方法在内存消耗和访问速度上相冲突的问题。

2 结构化数据缓存方法

针对现有技术中的报表缓存方式不能同时在内存消耗和访问速度上达到要求的问题，提出了一种结构化数据缓存方法，对要写入文件的报表数据分块，并且记录每块在文件中的位置，这样在读取报表数据时就可以直接定位到所在的块，在块中快速查找到所需要的数据。该方法主要针对报表等具有行列结构的数据，是依据可通过一个行列组成的矩阵表示的数据特性而设计的。

对于大数据量的报表，通常是纵向(行)或者横向(列)扩展，很少有2个方向都大量扩展的情况，所以报表数据的分块采用了按行(纵向扩展)或者列(横向扩展)来分割。最常见的是几千行甚至几万行，每行十几列的情况，以下的说明都以纵向扩展、按行分块为例。对应横向扩展，通过横纵向坐标置换可以用相同办法实现，在此不再赘述。

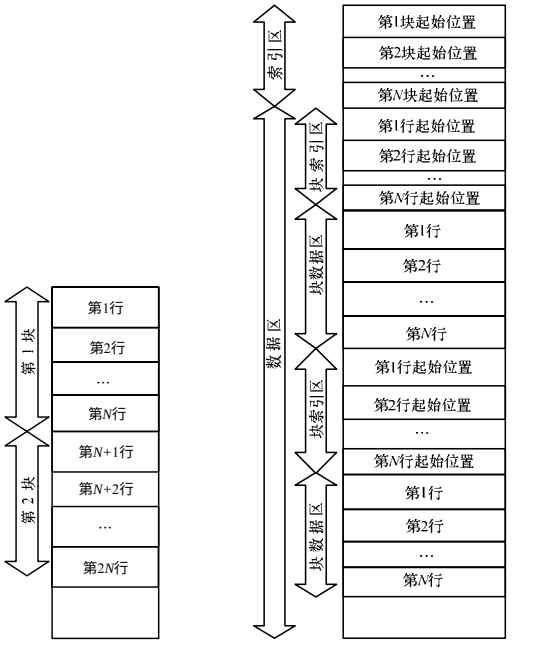
2.1 索引存储机制

本文提出的数据缓存方法的基础是结构化数据的索引存储机制。在存储上，把外部存储空间划分为索引区和数据区，其中数据区又可划分为块索引区和块数据区。当然，如果每

基金项目：国家科技支撑计划基金资助项目(2008BAH32B03)
作者简介：周进刚(1979－)，男，工程师、硕士，主研方向：软件复用，软件构件技术；邢铁军，工程师；纪 勇，高级工程师、硕士；赵大哲，教授、博士、博士生导师
收稿日期：2010-05-20 **E-mail：**zhou-jg@neusoft.com

行数据量小，或实际并不需要，也可以不用引入块索引区。

图 1(a)表示报表数据在存入文件系统前根据行数被分成多块，图 1(b)表示分块的报表数据在写入文件系统后的存储方式。在图 1(b)中，报表数据以块为单位被写入文件，文件索引区依次存储第 1 块、第 2 块、……、第 *N* 块数据的起始位置，文件数据区中每块数据又分割为块索引区和块数据区，其中块数据区存储该块的所有行数据，块索引区存储该块中每行数据的起始位置。在缓存文件的数据区中，每块数据的块索引区依次存储了第 1 行、第 2 行、……、第 *N* 行数据的起始位置，每块的块数据区依次存放了第 1 行、第 2 行、……、第 *N* 行数据，根据块索引区中的数据行起始位置可以在数据块中找到任意一行数据。这种两层索引方式，可以快速实现报表数据的读取，在查找某一行报表数据时，通过所述两层索引就可以快速定位，即先在文件索引区找到该行数据所在的块，然后在该块的块索引区找到这行数据的起始位置，进而找到该行读取数据。



(a)原始数据记录　　(b)经过二级索引后的缓存数据记录

图 1　报表数据存入文件前后结构比较

基于上述存储方式，数据缓存实现主要包括 2 大部分：数据分块和数据读写。

2.2　数据分块

将数据进行分块处理有多种方法，可以制定不同的分块算法来得到符合应用需求的分块结果。采用的分块算法如下：

```
1  初始化：MaxBlockNum←  //最大行数
     MinLineNum←  //每块最小行数
2 IF Lines > MaxBlockNum×MinLineNum goto 3; THEN goto 4;
3 LineNum ← MinLineNum× 2;
   BlockNum ← Lines ÷LineNum;    End
4 LineNum ← MinLineNum;
   BlockNum←(Lines+LineNum–1)÷LineNum;      End
```

其中，*Lines* 为总记录行数；*LineNum* 为每块行数；*BlockNum* 为块数。

初始化最大块数和每块最小行数不仅是计算的需要，同时也保证了块的大小适中。太大的块在块内查找行的效率很低；而如果块太小，会导致块数很多，也相应降低了对块的查找效率。步骤 3 中的参数“ 2 ”可根据实际应用进行调整。

利用分块算法得到的分块结果，包括块数和每块行数，在文件的写操作和读操作过程中都会使用到。在把报表数据写入文件时，需要根据分块算法执行的结果来分割报表数据，建立索引；在从文件中读取报表数据时，同样需要根据分块算法结果获得报表块数，进而根据请求的单元格所在行得到该行存储于哪一块，位于块中的第几行。因此，分块算法是数据缓存方法的基础算法。

在理论上，只需要在写数据时执行分块算法，读取时直接获得分块结果即可。但是在实际应用中，分块算法结果也需要占用系统内存，而且分块算法本身比较高效，执行速度很快。所以，为尽量节省内存空间，在写入和读取数据时都执行一次分块算法，并能够保证两次分块结果相同。

2.3　数据写/读

写数据的过程是将报表数据写入文件。图 2 是以图 1(b)两层索引为例的写数据算法流程。

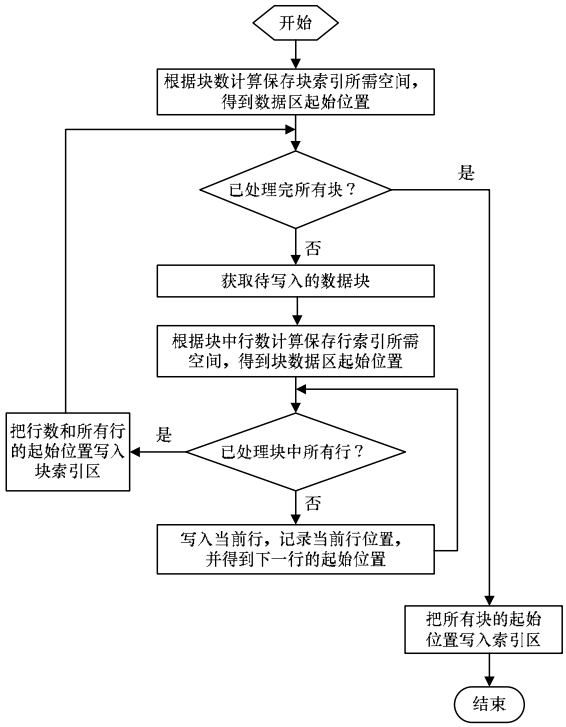


图 2　结构化数据的文件写入流程

如图 2 所示，采用的写数据方式是：先根据块数和块中行数计算得到写入位置，然后写入当前行，同时记录每行的位置，最后把写数据过程中记录的行索引信息写入块索引区；同样，完成所有数据块的写入后，才把块索引信息写入索引区。因为每行数据可能占用的空间大小不同，所以先将数据写入文件后再根据实际占用空间来写入索引信息，这样可以根据写入数据的大小分配合适的存储空间。如果每行数据的大小固定或变化不大，也可以预先设置固定的存储空间来存放索引信息和报表数据，然后根据块数和行数计算写入位置，将块索引信息、行索引信息和每块行数据依次写入文件。

在向块索引区中写入每行起始位置时，将该块的行数作为索引信息也一同写入(并未在图 1 未标出)。由于数据分块后，有时最后一块中的行数少于其他块中的行数，查询时通过块索引区中保存的行数信息，就可以知道该块有多少行数据。如果需要也可以把块数写入文件的索引区。

读数据是从文件中读取指定行或指定单元格的报表数据，依据上述写数据的过程，读数据的步骤如图 3 所示。

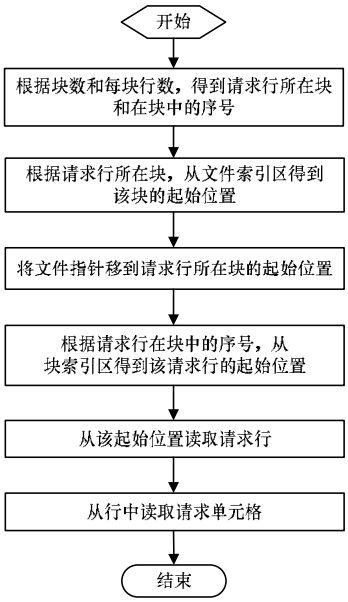


图 3　从文件中读取结构化数据流程

2.4　技术评估

上述报表缓存技术，在内存中只保存报表名称(在硬盘中通过文件名称区分不同文件)，而将报表数据存放在硬盘等外部存储介质(除内存之外的大容量高性能存储介质，包括硬盘、闪存等以及未来可能出现的类似存储介质)中，每次读取时根据报表名称从硬盘加载相应数据。不同的报表系统在处理报表生成方面具有不同的处理机制，导致不同的响应时间。以 UniEAP Report 3.0 产品中进行的测试为基准对其做介绍。

在一台 2 GHz 处理器、1 GB 内存 PC 机(用作数据展示)、Tomcat 5.5.9 和 Oracle 9.2 服务器环境下，利用 LoadRunner8.0 对于一张 5 000 行×16 列的报表缓存数据(总数据量约 4 MB)，分别以 1 个、10 个、20 个、30 个、40 个、50 个、60 个、70 个并发用户全部进行性能测试。

(1)写-读并发：包括从数据库获取原始报表数据、报表数据缓存到外存、数据的翻页读取。

(2)读并发：只包括从外存缓存中读取报表数据。

(3)数据展示：每页 30 行，包括从外部缓存的报表读取、从服务器把数据传输到前台浏览器、浏览器中的数据展示。

(4)系统平均响应时间：如图 4 所示。图 5 显示了 70 个并发读写情况下服务器 CPU 的利用率。

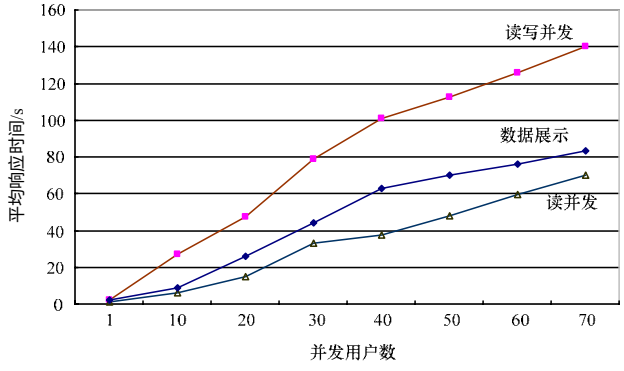


图 4　结构化数据缓存方法性能测试结果

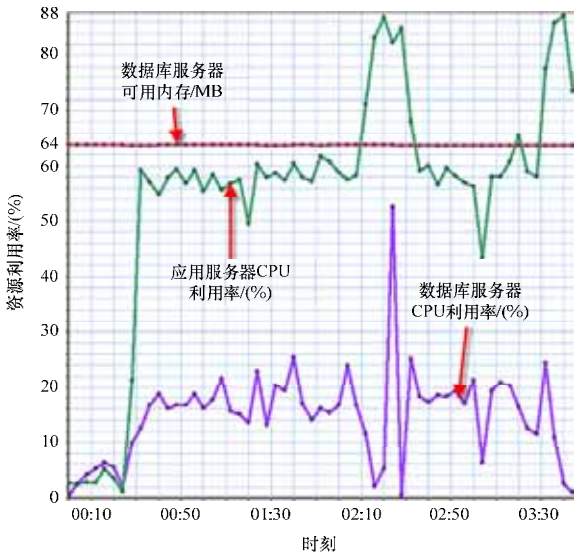


图 5　70 个并发用户下服务器 CPU 利用率

对于报表的访问，由于每次从硬盘读取数据，减少了内存的占用，在多并发、大数据量的情况下，可以大量节省内存空间的使用，提高系统的并发处理能力；同时，根据测试和应用中的经验，通过这种索引方式从硬盘读取报表文件，极大地提高了文件读取的效率，并减轻了对数据库服务器的压力。因此，本方法在内存空间占用和访问速度之间取得了一个平衡。测试过程中所有的数据都是缓存在硬盘，如果结合内存中的缓存配合使用，则效率会更高，这已不是本文关注的重点。

3　结束语

本文提出了一种结构化数据的缓存方法，适用于具有行列结构的大数据量数据的处理，目的是在尽量少地占用内存空间的同时又能保障数据的快速获取。其主要思想依据两层索引和数据分块算法。虽然以行分块进行方法的介绍，但对于列分块也是一样的，同时由于行列结构的特点，可以通过行列坐标的变换获取相应所需的数据。在缓存策略方面，本文提出的方法并不限定任何缓存替换算法，可以根据实际应用灵活选用。

参考文献

[1] 李兴勇, 袁兆山, 汪正海. 复杂报表生成系统实现技术研究[J]. 计算机应用, 2007, 27(7): 1821-1824.

[2] 杨春贵, 吴产乐, 彭鸿雁. 一种有效的 Web 代理缓存替换算法[J]. 计算机工程, 2007, 33(3): 43-44, 47.

[3] 张震波, 杨鹤标, 马振华. 基于 LRU 算法的 Web 系统缓存机制[J]. 计算机工程, 2006, 32(19): 68-70.

[4] 黄　敏, 蔡志刚. 缓存替换算法研究综述[J]. 计算机科学, 2006, 33(12): 194-195.

[5] 吴婷婷, 章文嵩, 周兴铭, 等. 语义缓存的最小权值项 LWI 替换策略[J]. 计算机研究与发展, 2003, 40(8): 1223-1229.

[6] 孟晓烜, 李一鸣, 庆　忠, 等. 一种面向存储服务的缓存管理模型[J]. 计算机工程, 2009, 35(1): 30-32.

编辑　顾逸斐