文章编号:1001 - 9081(2005)12 - 2814 - 03

基于访问模式的数据库缓冲管理自适应研究

白 洛,王元珍
(华中科技大学 计算机科学与技术学院,湖北 武汉 430074)
(bobdos@126.com)

摘 要:介绍了缓冲管理策略的研究现状,以及数据库管理系统中典型的工作负载特点。通过分析负载的访问模式,构造了一个代价评估模型,并在不同工作负载下验证模型的有效性。结果证明,基于访问模式的数据库缓冲管理自适应策略能使系统性能达到最佳。

关键词:自适应;数据库管理系统;访问模式;缓冲
中图分类号:TP311 **文献标识码**:A

Research on adaptive buffering management of DBMS based on access patterns

BAI Luo, WANG Yuan-zhen
(*College of Computer of Science and Technology*, *Huazhong University of Science and Technology*, *Wuhan Hubei* 430074, *China*)

Abstract: The status of buffer managing methods and the characteristics of the representative workload of DBMS was introduced. By analyzing access patterns, a model of evaluating cost was constructed, and its efficiency under various workloads was validated. The tests show that the adaptive policies based on access patterns of buffering management can get the optimal performance of DBMS.

Key words: adaptive; DBMS; access pattern; buffer

0 引言

在当前数据库系统应用中,存在着如下几种访问模式:1)具有局部性的随机访问模式,比如 OLTP:TPCC,ODB。其特点是查询简单,并发度高,即时更新,随机访问,无重复性;2)LRU 算法在这种访问模式下性能最优。而另一种是顺序访问模式,比如 DSS:TPC-H。其特点是查询复杂,并发度低;大部分时间只读,很少情况下需要批量更新,顺序访问,有重复性。而 MRU 算法在这种访问模式下性能更好。

缓冲区管理策略在数据库管理系统(DBMS)中非常重要。缓冲区管理策略包括缓冲区分配和缓冲区替换两个过程,其中缓冲区替换策略是传统缓冲区管理技术的核心。传统的替换方法以提高事务对内存数据访问的命中率为目的,一般基于局部性原理和事务的倾斜访问原理[7]。针对一个固定长度的缓冲区设计一个替换算法,按照某种策略将内存数据与磁盘数据进行交换,以期在内存中保存事务将要访问的数据,降低事务执行时的缺页率,减少事务对磁盘的访问[2~4]。但是这些文献提出的缓冲区管理策略都只是一种静态策略,没有考虑工作负载变化情况而引起访问模式变化时的性能波动。

文献[6]虽然是针对磁盘硬件内部 cache 预取策略自适应的研究,但是其原理可以部分借鉴到 DBMS 的缓冲管理策略中。其算法正是利用数据的局部性原理,通过缓冲区数据预取提高加速比[4],从而提高 I/O 性能。在实际的 DBMS 中,主要使用优先级作为替换信息,并大多采用多线索查询[8],这将形成复杂的查询序列,很难以单个查询的存取方式达到高效率。

文献[5]虽然介绍的是数据仓库中的缓冲管理策略,但文中提到在多种负载情况下,应该使用不同的分配算法和不同的替换策略以形成完善的全局缓冲管理策略,这个思想可以直接借鉴到 DBMS 本身。

目前大多数商业数据库依旧只采用传统的静态策略和单一的 LRU 替换算法,对连续数据块并没有进行合并优化。大多数商业数据库具有预取功能,但均由管理员手工指定是否使用,不根据系统运行状态的改变而自适应地改变。

本文提出了一种根据访问模式,自适应地调整 DBMS 缓冲预取策略、替换策略,以及在文献[8]提到的多线索查询情况下,充分利用异步 I/O 机制,并自动合并地址连续的请求,以减少 I/O 次数,使系统性能达到最佳。

1 模型建立

访问模式是一种定性的说明,如果某顺序访问模式当中存在极少量的随机请求,同样视为顺序访问模式。相反,一味追求准确判断模式而变更缓冲管理策略,会浪费更多的资源,使系统性能下降。

模型的建立包括两个部分:代价评估算法和策略选择算法。

1.1 代价评估

在时间间隔和数据量的选择上,因为可能存在某一段时间内没有任何请求,这对于模式的判断会造成较大的误差影响,所以选取在一定的数据量中审查访问模式更为可行。代价评估模型建立如下:

r:读请求;
R:r 的集合;
w:写请求;
W:w 的集合;

收稿日期:2005 - 06 - 20;**修订日期**:2005 - 09 - 07

作者简介:白洛(1980 -),男(满族),湖北人,硕士研究生,主要研究方向:数据库存储技术; 王元珍(1945 -),女,湖北人,教授,博士生导师,主要研究方向:数据库和多媒体技术.

$K(X)$:集合 X 中元素的个数。

total_len:总数据量,单位为字节。计算公式如下所示:

$$total\_len = \sum_{i=1}^{K(R)} r_i + \sum_{i=1}^{K(W)} w_i$$

N:在 *total_len* 数据量中,I/O 请求的总次数。计算公式如下所示:

$$N = K(R) + K(W)$$

则 I/O 请求模型可以表示为:

$(addr, rw\_len, rw)$

其中:

addr 表示随机 I/O 在整个 *total_len* 数据量中的比例。*addr* 的值越大,表示访问磁盘的地址越不连续:

$$addr = \{x \mid 0 \leqslant x \leqslant 1\}$$

rw_lem 表示写请求在整个 *total_len* 数据量中的字节比例。*rw_len* 的值越大,表示写请求越多。计算公式如下:

$$rw\_len = \frac{\sum_{i=1}^{K(W)} w_i}{\sum_{i=1}^{K(R)} r_i \sum_{i=1}^{K(W)} w_i}, \quad rw\_len = \{x \mid 0 \leqslant x \leqslant 1\}$$

rw 表示写请求的次数比例。*rw* 越大,表示写请求的次数越多。计算公式如下:

$$rw = \frac{K(W)}{K(R) + K(W)}, \quad rw \in \{x \mid 0 \leqslant x \leqslant 1\}$$

从模型中可以看出,如果 3 个比例的和越小,表示从磁盘读取连续的大数据块到内存,在这种情况下采用 MRU 算法,并采用预取算法,可以最大限度地减少 I/O;反之,3 个比例的和越大,表示从内存写不连续的小数据块到磁盘,在这种情况下采用 LRU 算法,并采用 I/O 块合并算法,可以最大限度地减少 I/O。

1.2　策略选择

通过上述代价评估模型,策略选择算法描述如下:

```
if (addr ->0)
 {
  if (rw_len ->0)
    {
      use MRU algorithm;
      if (rw ==0)
        prefetch read IO pages;
      else
        if (sizeof(page1) + sizeof(page2) <= system block MAX_
          SIZE)
          merge IO pages;
    }
  else
    {
      use LRU algorithm;
      if (rw ==0)
        prefetch read IO pages;
      else
        if (sizeof(page1) + sizeof(page2) <= system block MAX_
          SIZE)
          merge IO pages;
    }
 }
else
 {
  use LRU algorithm;
  if (rw_len ->0)
   {
     if (rw ==0)
       prefetch read IO pages;
     else
       if (sizeof(page1) + sizeof(page2) <= system block MAX_
         SIZE)
         merge IO pages;
   }
  else
   {
     if (rw ==0)
       prefetch read IO pages;
     else
       if (sizeof(page1) + sizeof(page2) <= system block MAX_
         SIZE)
         merge IO pages;
   }
 }
```

其中:

(1)预取操作只能应用于读取请求;

(2)合并操作既可以用于读取请求,也可以用于写回请求;

(3)在 I/O 操作访问的磁盘地址不连续情况下,即随机读写时统一采用 LRU 算法;

(4)I/O 操作中根据事务的性质决定是否使用异步 I/O。如果无需等待,则使用异步 I/O。

该算法的前提假设是系统运行稳定,即如果根据近期的数据请求判断出访问模式是顺序的,则可以假设后继的请求也是顺序的。因此该算法对于系统启动和退出时或不稳定的系统不适用。

结合近期 I/O 统计数据计算出随机 I/O 的比例,如果连续 I/O 的比例大于随机 I/O 的比例,则判断为顺序访问模式。同理比较读写比例,在读操作比例大于写操作比例时,缓冲区替换算法采用 MRU 算法;反之采用 LRU 算法。如果为只读操作,则采用预取策略。如果邻接两块的大小和不大于系统允许的最大块大小,则合并邻接的块。

对于一般文件系统中的文件进行普通的读写操作,数据统计如表 1 所示。

表 1　文件读写吞吐量统计

项	组				平均
	1	2	3	4	
顺序写(MB/s)	6.08	6.01	5.90	5.69	5.92
顺序读(MB/s)	6.51	6.53	6.53	6.43	6.50
随机写(MB/s)	0.16	0.23	0.24	0.25	0.22
随机读(MB/s)	0.20	0.25	0.28	0.28	0.25

可以看到顺序读写与随机读写的差异较大。鉴于这种情况,上述策略选择算法的第(3)点中,在随机读写情况下,其他两个模型因素相对 *addr* 因素已经不重要,因此直接采用 LRU 算法。

2　性能分析

该试验在 P4 CPU 2.0GHz,512M DDR333 内存,Linux 内核版本 2.6.3,DBMS 为 DM4 上进行。

2.1　TPC-H 测试

采用数据规模为 1G(SF = 1)的 tpch 测试数据库,各表数

据量如表 2 所示。

表 2 TPC-H 测试中表的数据量

表名	记录数/条
Region	5
Nation	25
Supplier	10000
Customer	150000
Part	200000
Partsupp	800000
Orders	1500000
Lineitem	6001215

在这里选取的测试语句是：

```
select
        l_returnflag,
        l_linestatus
from
       lineitem
where
        l_shipdate  <= dateadd( day, -79, '1998-12-01')
```

表 3 TPC-H 测试比较

项	类别		
	改进前	改进后	变化量
时间/s	3783	1927	-1856
读请求/次	1685	1136	-594
写请求/次	171	154	-17
读字节数/MB	22.92	23.28	+0.36
写字节数/MB	12.35	12.47	+0.12

TPC-H 测试的特点是大量的顺序读请求，且重复性强。可以看到在采用 MRU 替换算法动态取代改进前 LRU 算法的情况下，进一步经过数据块的预取和合并操作，读写请求次数都有明显下降。而读字节数的上升是由预取策略所带来的，但是并不造成系统性能的下降，时间开销反而减少。

2.2 TPC-C 测试

TPC-C 测试与 TPC-H 测试的目的不同。TPC-H 主要考察固定查询的能力，即在吞吐量一定的情况下，时间开销如何。而 TPC-C 主要考察在一定时间内，系统吞吐量如何。因此在这里作两个试验以验证模型有效性。

(1)时间一定情况下，吞吐量如表 4 所示。

表 4 TPC-C 测试比较

项	类别		
	改进前	改进后	变化量
时间/s	1800	1800	0
读请求/次	380999	416425	+35426
写请求/次	30193	30049	-144
读字节数/MB	1488.277	1626.660	+138.383
写字节数/MB	122.642	133.880	+11.238

在相同的时间内，改进后的数据吞吐量明显优于改进前。这是因为 TPCC 的特点是随机访问，这样改进前后的替换算法都主要采用 LRU 算法，但不同的地方主要体现在改进后算法中的合并操作。表 4 中写请求的次数减少，而写请求的数据量增大，是因为合并操作减少了 I/O 操作，并增大了单个数据块的字节数。

(2)在 TPC-C 标准压力测试中，总体吞吐量如表 5 所示。

表 5 TPC-C 标准压力测试

类别	项			相应的图号
	压力值	终端个数	MQTH(tmpC)	
改进前	10	280	296.4	1
改进后	10	330	360.0667	2
变化量	0	+50	+63.6667	/

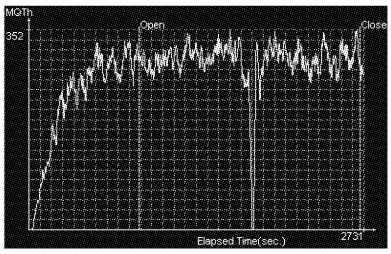

图 1 改进前 TPC-C 标准测试

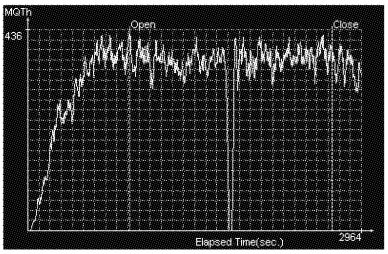

图 2 改进后 TPC-C 标准测试

3 结语

现代数据库应用的特点之一是负载类型多、变化大。不同的工作负载对系统资源有着不同的要求，因而对系统的优化策略有着重要的影响。因此数据库自适应算法日趋重要。在数据库缓冲区管理中，同样可以采用自适应算法，在不同工作负载和访问模式下动态调整策略，使系统性能达到最佳。本文未涉及缓冲区大小动态分配，但该因素对系统的影响也很重要，在接下来的研究中会进一步考虑这个问题。

参考文献：

[1] STANKOVIC J, SON SH, HANSSON J. Misconceptions about Real-time Databases[J]. IEEE Computer, 1999, 32(6): 29 - 36.

[2] DATTA A, MUKHERJEE S, VIGUIER I. PAPER: Prefetching and Priority Cognizant Buffer Replacement Policies in Real-time Active Database Systems[J]. Journal of Systems and Software, 1998, 42: 227 - 246.

[3] DAN A, MDIAS D, YU PS. An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes[A] . Proceedings of the 1990 ACM SIGMETRICS conference on measurement and modeling of computer systems[C]. Boulder, Colorado, USA, 1990. 143 - 152.

[4] ANTHONY K, TUNG H, TAY YC, *et al*. BROOM: Buffer Replacement Using On-line Optimization by Mining[A]. Proceedings of the 17th International Conference on Information and Knowledge Management[C]. 1998. 185 - 192.

[5] 金树东，冯玉才. 数据仓库环境中多级负荷的全局缓冲管理[J]. 计算机学报，1998，21(8).

[6] 薛一波，韩承德. Cache 对加速比的影响[J]. 计算机学报，1997，20(1): 27 - 36.

[7] 国志宏，王宏安，王强. 实时数据库缓冲区管理算法的设计和实现[J]. 计算机应用研究，2005，22(2): 121 - 124.

[8] 金树东，冯玉才，王元珍. 多线索 DBMS 中的全局缓冲管理[J]. 计算机应用与软件，1996.