一种基于四叉树的空间数据缓存策略

李东军[1,2],曾国荪[1,2]
LI Dong-jun[1,2],ZENG Guo-sun[1,2]

1.同济大学 计算机科学与技术系,上海 201804
2.国家高性能计算机工程技术中心 同济分中心,上海 201804
1.Department of Computer Science and Technology,Tongji University,Shanghai 201804,China
2.Tongji Branch,National Engineering & Technology Center of High Performance Computer,Shanghai 201804,China
E-mail:lidongjunlhf@163.com

LI Dong-jun,Zeng Guo-sun.Spatial data buffer policy based on quadtree.Computer Engineering and Applications, 2008,44(22):162-165.

Abstract: This paper presents a quadtree as the buffer data structure,improves and combines LFU and LRU algorithm,gives an efficient buffer strategy-spatial data buffer strategy based on quadtree,depicts a detailed description of buffer frameworks and strategies.This buffer strategy takes into account temporal locality and spatial locality of the spatial data access,which are considered by both LFU and LRU algorithm.Finally,this paper designs the spatial data accessing model,then verifies the effectiveness of the method.

Key words: quadtree;spatial data;buffer strategy

摘 要:提出了以四叉树作为缓存数据结构,结合广泛应用的 LRU 和 LFU 算法,给出了一种高效的缓存策略—基于四叉树的空间数据缓存策略,并详细描述了缓存框架和缓存策略。提出的缓存策略充分考虑了空间数据访问所具有的时间局部性和空间局部性,兼有 LRU 和 LFU 算法的优点。最后设计了空间数据请求模型,通过实验对算法的有效性进行了验证。

关键词:四叉树;空间数据;缓存策略
DOI:10.3778/j.issn.1002-8331.2008.22.048 **文章编号**:1002-8331(2008)22-0162-04 **文献标识码**:A **中图分类号**:TP338

1 引言

空间数据的研究随着地理信息系统(GIS)、计算机辅助设计(CAD)、多媒体系统、卫星影像库、定位服务系统[1]的广泛应用而展开,其访问效率成为研究的热点问题。

空间数据的数据量比较大,在网络带宽有限和磁盘速度较慢的条件下处理空间数据很难保证系统的效率,提高空间数据的访问效率非常迫切。目前存在多种提高空间数据访问效率的方法,但大部分工作集中在空间数据库的构建、空间数据的索引以及查询处理方法的优化上,对空间数据的缓存策略研究涉及不多。以往对缓存策略的研究集中在非空间数据上,可以分为两类[2]。一类研究关注访问频率,其中的代表性算法是 LFU(Least Frequently Used),另一类则关注最近访问的历史,LRU(Least Recently Used)是这一类算法中的经典。这些经典的缓存管理技术主要利用了时间局部性原理,即最近访问的数据在最近的将来被访问的概率比较大。空间数据访问有它自己的特性,即空间局部性。空间局部性是指,若一个区域中的数据被访问,那么此区域中其它数据被访问的可能性也很大。因此在空间数据的缓存管理技术中空间局部性也要被考虑在内,但传统的缓存管理技术对此没有涉及。

最近对空间数据的缓存策略有零星的研究,但其目的在于提高空间数据库的存访效率,其缓存的对象是一个个空间对象实体。如 Jun-Ki Min 提出的 BEAST 算法(Buffer Replacement Algorithm using Spatial and Temporal locality)[1]是在 LRU 的基础上,充分考虑空间局部性,利用两个 LRU 缓存队列得以实现的,提高了空间数据库的存访效率。但这些算法针对的是空间数据库系统,很难扩展到应用级的系统。如在 Web GIS 中的客户端缓存很难实现这个模型。本文综合考虑 LRU 和 LFU 两种算法并对其进行改进,利用本文提出的两个空间局部性原理,实现了一种基于四叉树的空间数据缓存策略模型 SLRFU(Spatial Least Recently/Frequently Used)。不同于其它缓存模型,SLRFU 以区域为单位缓存空间对象,可以应用于各种涉及空间数据的系统,特别是应用级的系统。实验表明,SLRFU 模型有效地提高了系统的访问效率。

基金项目:国家高技术研究发展计划(863)(the National High-Tech Research and Development Plan of China under Grant No.2007AA01Z425);国家重点基础研究发展规划(973)(the National Grand Fundamental Research 973 Program of China under Grant No.2007CB316502);国家自然科学基金(the National Natural Science Foundation of China under Grant No.90718015,No.60673157)。

作者简介:李东军(1982-),男,硕士生,主要研究领域为并行计算;曾国荪,博士,教授,博导,主要研究领域为网格计算、信息安全。

收稿日期:2007-03-06 **修回日期**:2008-06-02

2　四叉树缓存

用户对空间数据的检索大多是基于区域的，空间数据的访问在很大程度上体现空间位置的局部性，如在 GIS 系统中查询某大学附近 1.5 km 半径内所有的书店信息，或某矩形区域内所有的停车场信息。基于这种查询特性，空间数据一般采取分块缓存的方式，也就是说缓存的不仅仅是一个个孤立的空间对象，还包含了它们之间的空间位置关系。

传统的网格缓存对于点对象可以很大地提高检索效率。但线、面等具有一定空间范围的几何对象，可能落在多个网格区域中，特别是当对象范围很大时，会有很多记录项记录了该实体的信息，造成重复存储。重复的记录信息不仅造成了存储空间的浪费，而且由于查询时要去掉重复的对象，增加了查询的时间开销[7]。

本文提出的四叉树缓存是针对网格缓存的缺点进行的改进方案。其基本思想是将整个矩形对象空间对应到一棵完全四叉树，在缓存中保存这棵四叉树中经常被访问的一部分以提高检索效率。

四叉树中的每个节点代表一个矩形区域，孩子节点代表的矩形区域对应于将父亲节点的矩形区域均衡地划分为 4 个矩形区域中的一个，即

$$p.r=c1.r\cup c2.r\cup c3.r\cup c4.r \quad (1)$$

其中 p 为父亲节点，$p.r$ 为其表示的矩形，$c1$，$c2$，$c3$，$c4$ 为 4 个孩子节点，$c1.r$，$c2.r$，$c3.r$，$c4.r$ 分别为将父亲节点表示的矩形划分后的左上、右上、左下、右下小矩形。

完全四叉树的构建是一个自上而下的过程，一棵指定高度 k 的完全四叉树构建过程描述如下：

(1)将整个矩形对象空间作为树的根，并记为树的第一层($L1$)，其节点集合记为 $R1$，置 i=1；

(2)若 $i>k$，转(5)；

(3)将节点集合 Ri 中的每个节点所表示的矩形 r 均衡划分为 4 个小矩形，作为 r 的孩子节点，并加入节点集合 Ri+1，记为树的第 i+1 层(Li+1)；

(4)$i=i$+1，转(2)；

(5)构造结束。

空间对象 e 从属于四叉树节点 node(或其表示的矩形)的充分必要条件是 e 的最小外包矩形完全包含于 node 所表示的矩形，但不包含于比 node 层次高的任何节点所表示的矩形：

$$e\in node\Leftrightarrow$$
$$e.r\subset node.r\wedge(\forall n\in Ri)e.r\not\subset n.r,\text{其中 } node\in Rj,i>j \quad (2)$$

同样可以采用自上而下的方式将空间对象关联到四叉树。构造结果如图 1 所示，其中展示了整棵四叉树，灰色节点表示为了存储所有空间对象必须保留的节点。

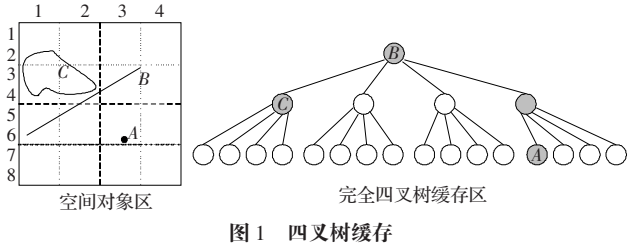

图 1　四叉树缓存

相对于传统的网格缓存，四叉树缓存有如下优点：

(1)由于每一个空间对象都属于唯一的一个节点，不存在重复的存储信息，减少了缓存的存储空间。在图 1 中，对象 B 只在根节点有记录，对象 C 只在根节点的一个孩子节点上有记录。

(2)查询空间对象时，没有冗余的记录信息，省去了排重操作，提高了缓存速度。

在实际应用中，空间数据一般很大，而内存容量又很有限，所以在用四叉树作为缓存时，不可能保存完整的四叉树，而只能保存树的一部分。应该保存树的哪一部分以及如何通过四叉树检索空间数据涉及到缓存策略，在下面有详细地论述。

3　SLRFU 缓存策略

下面详细叙述用四叉树缓存空间对象时的策略，充分考虑了时间局部性和空间局部性。

空间局部性存在如下两个启发式规则：

规则 1　优先淘汰小的空间对象

从概率上看，当在矩形对象空间随机地选取一个点时，相对于大对象来说，小对象包含这个点的可能性要小，即小的空间对象比大的空间对象拥有更小的访问频率。这里的大小指的是空间上的跨度。

规则 2　尽可能淘汰空间上临近的对象

此规则基于空间区域连续性访问原理，也就是如果用户访问了某个空间对象，用户很可能继续访问这个对象或者访问和这个对象在空间分布上邻接的地理数据，反过来说就是如果某个对象没有被访问，那么它邻近对象被访问的可能性也不大。所以在空间上临近的对象一般具有相似的访问频率。

基于这两个规则，结合时间局部性上经典的 LRU(Least Recently Used，最久未使用)算法和 LFU(Least Frequently Used，最少使用)算法，以四叉树为基础建立空间数据缓存模型 SLRFU。

本文首先介绍 SLRFU 缓存模型的框架，然后讨论其具体实现。

3.1　SLRFU 缓存框架

缓存中的数据结构如图 2 所示，主要包括 3 个部分：空间对象区、四叉树缓存区(节点中保存指向空间对象的指针)、SLRFU 缓存队列。

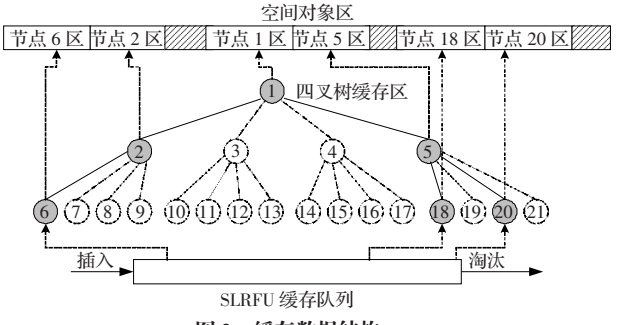

图 2　缓存数据结构

空间对象区：采用伙伴算法管理空间对象占用内存的分配和回收。每个节点包含的空间对象数目可能非常庞大，单独为每个对象分配空间会非常费时，且易产生内存碎片。所以，为每个四叉树节点包含的所有空间对象集中分配空间。如图 2 中空间对象区所示，每个节点包含的空间对象存储在连续的内存空间中。

四叉树缓存区：保存了一棵不完整的四叉树，每个节点包含指向空间对象的指针。节点包含的对象由式(2)决定。

SLRFU 缓存队列:SLRFU 缓存队列并不是保存四叉树缓存区中的所有节点,而只是保存对应于第 2 章介绍的完全四叉树中的叶子节点,即层数最高的节点。

初始状态时,空间对象区、四叉树缓存区、SLRFU 缓存队列都为空。

3.2 SLRFU 缓存策略

LRU 和 LFU 是两种广泛应用的淘汰算法。

LRU 采用的启发式规则是:最近访问过的对象在最近将来被访问的可能性比较大,而最近一次访问时间离现在时刻越远的对象,在最近将来被访问的可能性也越小。因此,LRU 淘汰最久没有使用的对象[5]。

LFU 则认为过去一段时间内最经常访问的对象,将来对其访问的可能性也最大,所以将它们保存在缓存中。其开销是"使用次数计数器"的维护。问题是刚刚调入缓存的对象其使用计数器值相当小,很可能被选中淘汰出去[5]。

访问频度和最新的访问历史是反映时间局部性的主要指标。LFU 和 LRU 具有互补的特性。LRU 算法利用上一次访问的时间特性,对访问特性的变化比较敏感,但没有考虑数据访问的全局特性;LFU 算法则使用所有访问的频率信息,考虑数据访问的整体特性,但不能适应数据访问模式的变化和阵发性的访问。它们分别代表了两个极端。本文综合这两个因素,以四叉树为基本数据结构,利用两个空间局部性启发式规则,构建出 SLRFU 模型。

在 SLRFU 中,每个在缓存队列中的对象和一个量化对象最近将来被访问可能性的 CRF(Combined Recency and Frequency)值相关联。设缓存对象被访问 n 次,则其 CRF 值为:

$$CRF=\sum_{i=1}^{n}\left(\frac{1}{2}\right)^{i} \tag{3}$$

可以证明 CRF 是一个介于 1/2 和 1 之间的数。

SLRFU 总是选择缓存队列头的对象作为淘汰的牺牲者。在 SLRFU 中,并不是根据 CRF 值来选择淘汰的对象,而是当某个对象被访问时根据 CRF 值来决定对象在队列中的位置。设当前队列长度为 L,则将被访问对象插入到队列的 $CRF\times L$ 处。当对象第一次被访问时,n=1,CRF=1/2,则将此对象插入到队列的中间;随着对象被访问次数的增多,CRF 渐渐趋向于 1,最终插入到队尾。可见 SLRFU 兼有 LRU 和 LFU 的优点。

结合空间局部性的两个启发式规则,在 SLRFU 模型中主要操作如下:

(1)加载:对象(在 SLRFU 模型中指的是树节点)刚刚载入内存时,有两种存放策略。第一种策略是所有节点都要放入 SLRFU 缓存队列参加淘汰选择,而不论它是否叶子节点。这是一种通用的策略,即每个在缓存中的对象都要参加淘汰,它们之间没有优先关系。在 SLRFU 模型中这个策略的失效点在于,对象之间是存在空间局部性关系的,反映在四叉树上就是根节点的访问次数明显地高于叶子节点,这个策略仅仅从时间局部性出发而完全没有考虑空间方面的因素。

第二种策略是只把叶子节点加入 SLRFU 缓存队列参加淘汰选择。一般说来,远离树根的节点中所含有的对象跨度要小于上层节点对象的跨度,根据空间局部性的启发式规则 1,这些节点应优先淘汰。让低层节点首先参加淘汰选择无疑是此规则的体现。

在 SLRFU 模型中选择第 2 种策略。需要注意的是,虽然非叶子节点不参加淘汰选择,但将它们载入内存可能需要淘汰其它节点才能完成,因为无论是否叶子节点,它们所包含的实际空间对象都公用一个空间对象区。

在实际应用中加载属于某个节点的对象时会遇到一个问题:空间数据库并不直接提供此类操作,而只是提供获取与某个区域相交的所有对象的操作。如图 2 所示,无法直接获得属于根节点的对象 B,而必须将所有和根节点相交的对象 A,B,C 都提取出来,然后再将 B 提取出来。这在实际中的效率太低以至于无法实施。采用给四叉树编码的方式来解决这个问题,即为完全四叉树的每个节点编码。如图 2 所示,对四叉树从上到下,从左到右编码,每个节点内的数字为其编码。结果是每个空间对象唯一对应一个编码,通过数据预处理,为每个空间对象编码,加载数据时,只需按编码提取即可。

(2)淘汰:基本操作为选择 SLRFU 缓存队列中的头节点将其淘汰。

如果被淘汰节点的父亲节点变为叶子,对其有以下两种处理策略:

策略 1 将其加入 SLRFU 缓存队列进行淘汰选择。在这种策略下,虽然非叶子节点在缓存队列中没有记录项,但其也要记录自身的访问历史信息,因为一旦其变为叶子将被加入缓存队列,这些信息将被用作计算 CRF 值。

策略 2 将其删除。这是一个递归的过程,一旦删除这个节点后又产生新的叶子则也要将其删除。这样做主要基于以下考虑:根据空间局部性的启发式规则 1,大对象的访问概率要大于小对象,因此淘汰时采取自底向上的策略,即从叶子节点向上淘汰;根据启发式规则 2,淘汰时应该尽量以子树为单位,因此父亲节点变成叶子后则直接将其删除。在 SLRFU 模型中采取这种策略。

(3)访问:若缓存队列中的对象被访问,则更新其 CRF 值并将其插入缓存队列的适当位置。

4 实验与分析

笔者在 Web GIS 系统中实现 SLRFU 缓存模型,验证了其有效性。该 Web GIS 系统的数据为整个中国地理空间数据,数据量为 10 GB。数据服务器的配置为:四核英特尔®至强® 5310 系列处理器,主频为 2.00 GHz,内存容量为 4 GB。

实验的关键在于数据请求的模拟。空间对象密集的区域被访问的频率更加频繁,基于此建立如下请求模型。

以矩形框为请求单位,得到的是与矩形框相交的空间对象。每生成一次请求都要确定矩形的中心点和长宽。

将中国地图划分为边长为 1 km 的方格子,统计每个格子包含的空间对象外包矩形中心点的个数,记为 n。n 越大,则请求矩形框中心点落入此格子的概率越大,其概率为 n/N,其中 N 为空间对象总数目。

设请求矩形框边长在区间[m,M]中,每一个请求矩形框的边长服从[m,M]之间的均匀分布。

用 1 台 PC 机客户端以此模型向数据服务器发送数据请求,每秒钟发送 10 个请求,统计处理请求的平均时间。图 3 展示了采用 SLRFU 缓存策略和没有采用缓存策略的平均响应时间对比。

可以看出,无缓存模式下的平均响应时间基本上为一条水平的直线。而采用 SLRFU 缓存策略的系统,曲线开始时呈现下

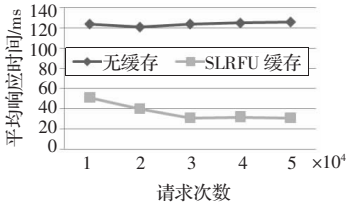

图 3 SLRFU 缓存和无缓存性能对比

降的趋势，到达某个关键点后开始水平延伸，开始的一段下降曲线是由于当请求次数很少时缓存中数据很少，很难命中，大部分数据需从磁盘读取。而随着请求次数的增加，缓存中的数据也越来越多，命中率增加，平均响应时间降低，当缓存满时，平均访问时间基本维持不变。

当系统稳定时，采用 SLRFU 缓存策略系统的平均响应时间约为无缓存系统的 1/3，大大提高了系统性能。

5 结束语

本文分析了空间数据存取在时间局部性和空间局部性上的特征，给出了空间局部性的两个启发式规则，综合并改进经典的 LRU 和 LFU 算法，在四叉树的基础上给出了空间数据缓存策略——SLRFU 策略。四叉树作为空间数据缓存数据结构具有天生的优势，很好地体现了两个空间局部性原理：跨度小的对象放在叶子节点，将优先被淘汰；属于同一子树的对象在空间上比较临近，具有相似的访问频度，可以将它们作为一个整体进行淘汰和加载。首先分析了以四叉树作为缓存数据结构弥补了传统网格技术的不足，然后结合空间局部性原理给出了缓存数据结构，并改进 LRU 和 LFU 算法，给出了淘汰策略，最后在 Web GIS 系统中对该模型进行了验证。实验证明，SLRFU 可以大幅提高空间数据的存取效率。

本文所设计的 SLRFU 模型仅考虑只读数据，今后的工作将集中在如何对四叉树进行加锁来实现可读写的缓存策略模型。

参考文献：

[1] Min J K.BEAST：a buffer replacement algorithm using spatial and temporal locality[C]//LNCS 3981：ICCSA 2006，2006：67-76.

[2] Lee D，Choi J.LRFU replacement policy：a spectrum of block replacement policies，SNU-CE-AN-96-004 [R].Seoul National University，1996-03.

[3] Hanan Samet.The quadtree and related hierarchical data structures[J].Computing Surveys，1984，16(2).

[4] Lee D，Choi J.Implementation and performance evaluation of the LRFU replacement policy[C]//Proceedings of the 23rd Euromicro Conference on New Frontiers of Information Technology：Short Contributions，EUROMICRO 97，1-4 Sep 1997：106-111.

[5] 尤晋元.UNIX 操作系统教程[M].西安：西安电子科技大学出版社，2004：114-115.

[6] 涂小朋，汪林林.分布式空间数据库中基于事务的客户端高速缓存技术研究[J].计算机科学，2004，31(6)：76-81.

[7] 舒忠玲，汪林林，王佐成.利用层次网格索引提高 WebGIS 性能[J].计算机应用，2004，24(9)：150-152.

[8] 王映辉.一种 GIS 自适应层次网格空间索引算法[J].计算机工程与应用，2003，39(9)：58-60.

（上接 123 页）

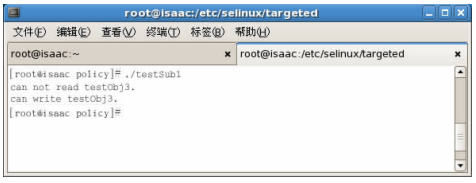

图 4 Biba 模型下 testSub1 对 testObj3 的访问请求

7 总结

本文提出了基于向量矩阵的多级安全访问控制实现方法，在 SELinux 访问控制机制的基础上，引入了访问控制向量矩阵，对 SELinux 中的访问控制机制进行了描述。并且引入了 BLP 和 Biba 模型，实现了与原有 SELinux 机制的结合，对 SELinux 安全操作系统中访问控制机制的研究具有良好的参考价值。

下一步研究方向主要在该访问控制机制的基础上，进一步完善各种访问控制方法和控制权限，例如将除了读、写之外的诸多权限加入并予以实现等。另外，每次对 SELinux 中对象访问控制权限的修改都需要重新启动，这点也有待进一步改进。

参考文献：

[1] National Security Agency.SELinux[EB/OL].http://www.nsa.gov/selinux/.

[2] DoD 5200.28-STD，Department of Defense Standard，Department of Defense Trusted Computer System Evaluation Criteria[S].1995.

[3] Huang Xian-zhi，Wang Hai-yang，Chen Zhen-xiang，et al.A context，rule and role-based access control model in enterprise pervasive computing environment[C]//2006 1st International Symposium on Pervasive Computing and Applications，Shandong University，2006.

[4] Chen Xiao-su，Lin Zhi.Policy-based access control model for mobile agent system[C]//Wireless Communications，Networking and Mobile Computing，2006.

[5] Srivatsa M，Iyengar A.An access control system for Web service compositions[C]//IEEE International Conference on Web Services 2007，ICWS 2007，2007.

[6] Bell D E，LaPadula L J.Secure computer systems：mathematical foundations and model，M74-244m[R].The MITRE Corporation，Bedford，MA，1973.

[7] Biba K.Integrity considerations for secure computer systems，MTR-3153[R].MITRE Corporation，Bedford，MA，1977.

[8] Mayer F，MacMillan K，Caplan D.SELinux by example：using security enhanced linux[M].[S.l.]：Prentice Hall，2006-07.

[9] 刘伟，孙玉芳.若干安全操作系统中的基于角色的访问控制特性[J].计算机工程与应用，2004，40(4)：41-44.

[10] Hanson C.SELinux and MLS：putting the pieces together[C]//Third Annual Security Enhanced Linux Symposium，Mar 2006.

[11] Vance C，Miller T，Dekelbaum R.SPARTA，security-enhanced darwin：porting SELinux to Mac OS X[C]//Forth Annual Security Enhanced Linux Symposium，Mar 2007.

[12] MacMillan K，Hat R.Madison：a new approach to automated policy generation[C]//Forth Annual Security Enhanced Linux Symposium，Mar 2007.