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*)

涂小朋　汪林林
(重庆邮电学院软件学院　重庆400065)

摘　要　空间数据通过 Intranet 或 Internet 从服务器传送给客户端，由于涉及到大量的空间数据，繁重的网络传输可能成为系统的瓶颈。客户端高速缓存技术能较好的解决这个问题。论文提出了客户端缓冲技术应用在分布式空间数据系统中的体系结构；研究了基于重叠区域的客户端缓冲技术在分布式空间数据库中具体应用，并给出适合空间数据特点的缓冲置换算法；同时讨论了高速缓存技术相关高速缓冲一致性问题。

关键词　事务，ACID，高速缓冲，分布式空间数据系统，R-tree，缓冲一致性

Transactional Client-Cache Techniques in Distributed Spatial Database

TU Xiao-Peng　WANG Lin-Lin
(The Software Institute of Chongqing University of Post and Telecom, Chongqing 400065)

Abstract　Spatial data is shipped to client through Internet or Intranet from server in distributed spatial database system. Heavy communications between client and server may become bottleneck because of high volume of spatial data. Client caching techniques can effectively solve the problem. In the paper we present the architecture of spatial database in which client cache technique will be applied. Then we give the method based on overlapped region, which is applied for spatial cache techniques. Replacement algorithms considering spatial data criteria are devised combined with conventional LRU algorithms. Cache consistency related to cache techniques is discussed in succession.

Keywords　Transaction, ACID, Cache, Distributed spatial database system, R-tree, Cache consistency

1. 引言

几乎所有关系数据库系统都是采用由客户端发送查询命令，服务器执行查询生成结果集并传输给客户端，而客户端负责处理用户接口。在空间数据库中，由于数据量非常大、空间对象间的复杂关系，这种方法会给网络造成沉重的负担，繁重的网络传输就可能成为应用的瓶颈。因此，有必要采取一些技术来减少不必要的网络数据传输。客户端高速缓存技术就是一种有效的解决方法。当采用客户端高速缓存技术时，客户端首先分析哪些数据能满足用户的需求，并且只有在本地缓存区域中没有找到所需数据的情况下，才向服务器发出数据请求。采用客户端高速缓存技术的分布式空间数据库系统具有以下优点。

·由于本地保存有数据的副本，这使得用户程序可以更高效地访问数据。

·减轻了数据库服务器的工作量，增强了系统的性能和可伸缩性。

在实践中存在两种不同的高速缓冲技术：事务内高速缓冲和事务间高速缓冲。事务内高速缓冲指在事务结束(提交或终止)后，高速缓冲中的数据就不再有效。事务间高速缓冲指客户端能够越过事务边界将高速缓冲保留在本地，也就是事务结束后，高速缓冲数据继续保留在客户端，可以参与执行下一个事务。事务内高速缓冲可以采取常见的两阶段提交(2PC)来解决。事务间高速缓冲由于越过事务边界，不能采取常规的面向事务的并发控制机制，需要特殊的高速缓冲一致性协议来保证数据的正确性。本文提出客户端高速缓冲技术在分布式空间数据库中应用的体系结构并讨论高速缓冲中数据传输粒度问题，详细分析高速缓冲技术在空间数据库中的具体应用方法和高速缓冲置换策略，讨论了高速缓冲一致性问题，最后进行总结。

2. 带高速缓冲的体系结构

在分布式空间数据库中，由于应用的复杂性和大量的地图数据，客户端除了要处理用户界面、业务流程外，还要负责部分数据处理任务，这种客户端被称为胖客户端。图1给出了带高速缓冲的客户端在分布式空间数据库系统中应用的总体结构图。

从图1可以看出，客户端除了处理与数据库服务器的接口外，还有部分数据处理功能。包括基于 R-tree 的空间查询分析器和高速缓冲管理器。客户端应用程序通过使用可视化组件和本地查询分析以及高速缓冲管理组件相连，所有和服务器的通讯都是通过数据桥组件。

可视化组件：可视化组件接受用户输入并显示相关信息。这个组件是用户管理和操纵地理信息的接口。

基于 R-tree 的空间查询分析器：主要完成对用户的查询请求(包括空间数据和非空间数据)进行处理，分析得出所需的数据(所需数据的 OID 列表)，返回给应用程序，应用程序调用相应的底层组件(如 DataBridge)从数据源获取数据。

高速缓冲管理器：高速缓冲管理器负责管理在本客户端高速缓冲区域中的数据，如数据页置换等。在图1中，本地高速缓冲区域中包含有两种高速缓冲类型：缓冲区和磁盘高速缓冲。高速缓冲管理器会将从数据库服务器下载的地图数据临时存放在磁盘上，在需要的时候会首先在缓冲区和磁盘上找(当然是通过空间查询分析来决定)，只有在本地没有找到的

*)基金项目：重庆市信息产业局"GIS 组件库"200112028发展基金项目资助。涂小朋　硕士研究生；汪林林　教授，从事数据库、计算机网络、GIS 技术研究。

情况下，才从服务器下载到本地高速缓冲区域中。

数据桥接口：负责和数据库服务器通信，从服务器获取数据或将更新的数据传到服务器上。

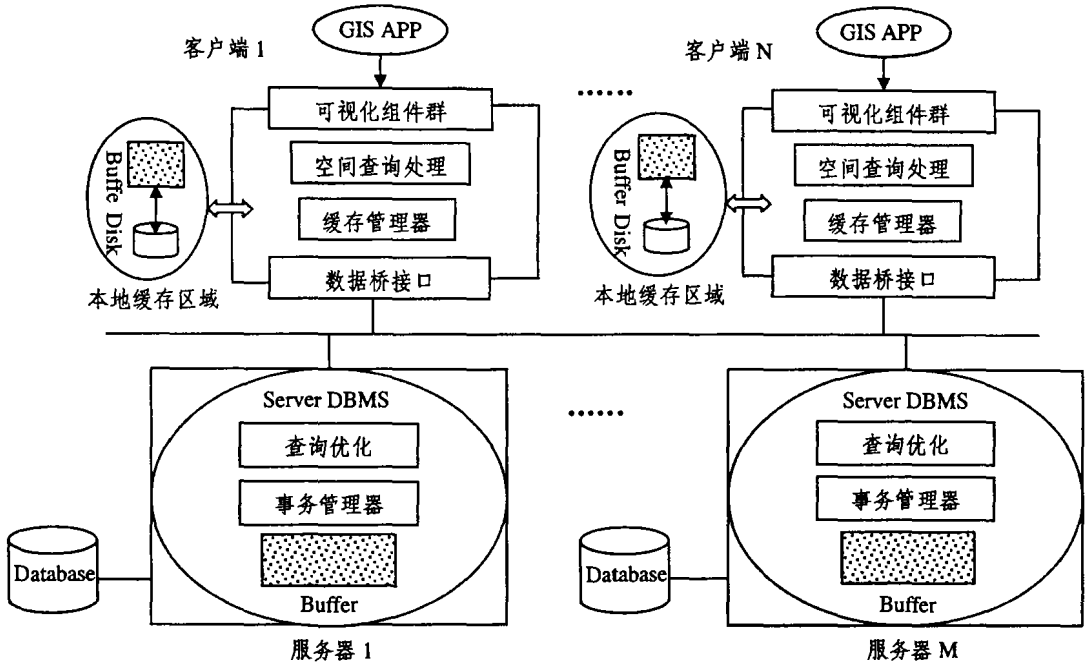

图1　带客户端客速缓冲的分布式空间数据库系统体系结构

3. 客户端高速缓冲技术在分布式空间数据库中的应用

高速缓冲的基本原理是将需要频繁访问的数据保存在相对能够快速存取的高速缓冲区域中，这样可以避免不必要的系统的开销。考虑到空间数据的特殊性，应该根据空间特性对用户所需的数据进行预测。在分布式空间数据库中采用基于重叠区域的高速缓冲技术。这种预测方法只是根据用户访问数据的某种特性进行测试的，不可能完全准确地预测用户所需的数据，同时由于高速缓冲区域的容量相对较小，不可能容纳所有数据，因此还必须采取一定的高速缓冲置换策略，将暂时不用的数据移出高速缓冲以容纳新加入的数据。

3.1　基于重叠区域的高速缓冲技术

由于客户端的存储容量和性能限制，客户端通常只能够将空间数据库服务器上全局地图数据中用户经常使用的一部分数据保存在高速缓冲中。重叠区域的高速缓冲技术是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。因此空间数据库服务器每次不是把用户所需的单个空间对象传输给用户，而是将包含用户所需地图数据的一个空间连续区域传输给用户。在分布式空间数据库中，这种的方式称为重叠区域策略。从服务器传过来的这部分区域的地图数据将保留在客户端的高速缓冲区域中。这样就大大减少了不必要的网络传输，加快用户访问数据的速度。如图2所示。

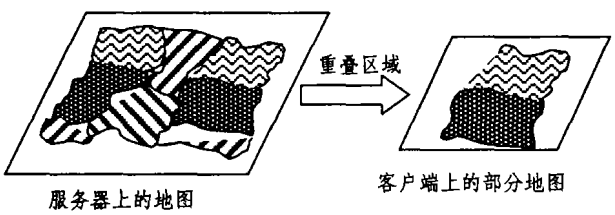

图2　基于重叠区域的客户端高速缓冲技术

在图2中，客户端高速缓冲中只保留服务器上地图的一块重叠的区域。用户程序在存取地图数据时，首先访问本地高速缓冲区域，只有在本地高速缓冲区域中没有找到的情况下，才向空间数据库服务器发送数据请求。由此可见采取本地高速缓冲技术能够大大减少网络延时，增强网络的吞吐量，减少网络瓶颈发生的机率，提高系统的性能和系统的可伸缩性。

3.2　客户端高速缓冲置换策略

由于客户端存储容量的限制，当在高速缓冲区域中再也没有多余的空间时，必然导致用户访问空间数据，高速缓冲管理器将丢弃缓冲区域中的一些地图数据，同时在空出的区域中写入新的地图数据。这就需要采取适合于空间数据特点的数据置换策略。最近最少使用算法(LRU)是应用最为广泛的置换策略，它的基本规则是丢弃最近一段时间内最长时间没有被访问过的数据项。LRU 在操作系统中被广泛采纳。但 LRU 存在一些本质的缺陷：(1)不能分辨出在相当长的时间段内被频繁访问的页。(2)它不能根据用户访问的数据对访问模式进行预测。鉴于 LRU 的这些缺陷，出现了一些适合于空间数据特点的置换算法。在空间数据库中，如果通过分析数据页的内容来判断选择应该丢弃的页，则显得更为有效。空间数据访问方法(SAM)对不同的数据结构有不同的优化标准。文[4]给出了针对 R*-tree 的四种优化标准：(1)最小化目录矩形的面积；(2)最小化目录矩形间的重叠；(3)最小化目录矩形的边框长度；(4)最小化存储器的利用率。

根据上面的四个标准，可以开发出符合空间数据特点的高速缓冲置换策略，称之为空间置换策略。假设每个数据页是一组数据项的集合，在每个数据项中，可以确定数据项的最小边界矩形(MBR)。如果数据页是存储的几何对象的一部分，则 MBR 值为此对象的最小边界矩形的一部分。在上述的四个标准中，高速缓冲置换策略可以使用第一个标准，就是将在高速缓冲中目录矩形面积最大的页尽可能的保存在高速缓冲中。因为目录矩形越大的页，被访问的可能性就越大，所以置

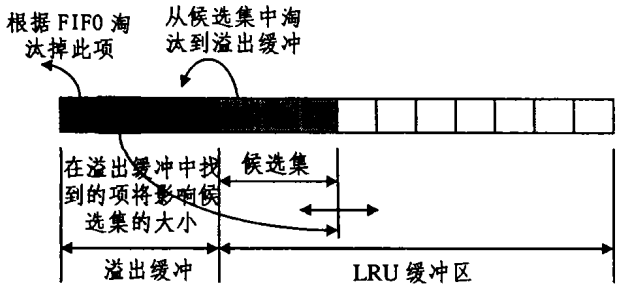

图3 适合空间数据的缓存置换

换策略应该将这些页尽可能长时间的放在高速缓冲中。空间置换策略是通过比较数据页的空间特性进行置换的。在大多数情况下，空间置换策略能够比LRU更准确的预测出被替换的页。但是，如果单纯地采用空间置换策略，在某些情况下却不能获得较好的性能。因此一个折中的办法就是结合空间置换策略和LRU。图3是采用了结合FIFO策略和空间数据特点的置换结构。当后边缓冲区满时，通过LRU策略淘汰的数据项进入到候选集中；候选集中的数据项通过空间置换策略淘汰到溢出缓冲区。溢出缓冲区采用FIFO(先来现服务)策略删除其中的数据项。若所需数据项在溢出缓冲中，则应适当调整候选集的大小。针对此数据项，若此时LRU策略优于空间置换策略，则减小候选集；若空间置换策略优于LRU策略则增大候选集，否则不变。实验表明采取这种结合的置换策略比单纯采用LRU(或LRU的变异算法)可以获得更好的性能。

4. 基于事务的高速缓冲一致性

本地数据高速缓冲技术改善系统性能的同时，也给空间数据库系统的设计带来新的问题。基于事务的高速缓冲一致性就是其中一个最主要的问题，事务高速缓冲一致性和并发控制是紧密相连的。如图4所示。

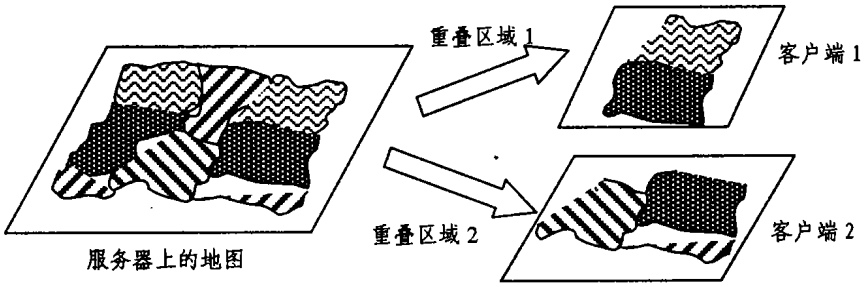

图4 客户端高速缓冲的高速缓冲一致性

在图4的两个客户端中，如果这两个客户端中同时对这部分公共数据进行存取、访问，而其中至少有一个客户端修改数据时，就必然会导致如何协调这些访问动作，以确保数据的正确性的问题，也就是并发控制问题。若其中的一个客户端对这部分的数据进行了修改，将面临如何把修改的数据传播到另外含有这个的客户端，也就是保持多个高速缓冲副本一致性的问题。

多个客户端上事务的并发执行必须要确保数据库的正确性。客户端上事务的执行不能破坏事务的ACID属性，同时在多高速缓冲副本多个事务的并发执行必须等价于在服务器上非副本数据的某种串行执行，这称之为单一拷贝可串行化。客户端高速缓冲一致性算法必须保证单一拷贝可串行性。

4.1 基于事务的高速缓冲一致性

保持高速缓冲一致性的算法大致可以分为两类：基于冲突检测的算法和基于冲突避免的算法[1]。基于冲突检测的算法允许事务访问本地高速缓冲区域中旧数据，但在事务提交之前必须要验证数据的有效性。若数据无效，则要向数据库服务器发出请求以获取更新后的数据，或者终止事务。基于冲突避免的算法保证所有本地高速缓冲中的数据都是有效的，也就是访问本地数据时不需要向服务器提出数据验证请求。冲突避免的算法采取单一读、写全部的方法(ROWA)管理在各个客户端上的高速缓冲副本。ROWA算法确保在更新事务提交后，在各个客户端高速缓冲区域所有更新数据的副本都是相同的。在ROWA算法模式下，客户端的事务可以读取相应数据的任何一个副本，通常情况下如果数据缓存在本地客户端，则只需读取本地高速缓冲中的数据(前提是这部分数据没有被另外的事务加X锁住)，对于更新操作，在事务提交时，数据的改变要被反映到当前所有的客户端副本中。基于冲突检测的算法和基于冲突避免的算法按照与服务器通讯的方式不同又可以分为同步、异步和延迟。在同步算法中，客户端发送一个所需的锁请求到服务器，并阻塞等待服务器的应答；在异步算法中，客户端发送一个所需的锁请求，但客户端并不阻塞等待服务器的应答，而是乐观地继续执行另外的操作；在延迟算法中，客户端直到事务提交的时候才向服务器响应所有的锁请求。在上述的三种算法中，延迟算法的通信开销最小，因为它在事务提交的时候才将所有的锁请求发送给服务器。

5. 客户端高速缓存技术的实现

在GIS组件库项目中，客户端高速缓存包含在数据访问组件群(Data Access Components)中，其中 主要包含有两个接口：DataBridge和DataCache。DataBridge用于访问数据源中的数据，因此针对每种不同类型的数据库服务器(包括本地数据源和远程数据源)，都应有一个与之相对应的DataBridge的实现。而DataCache用于访问存储在Cache Area(存放在本地磁盘)或者已经在内存中的数据，对数据读取起缓冲作用。从服务器传送来的数据都将缓存在Cache Area区域中，客户端的请求将会先在Cache Area中进行搜索，若没有找到就将请求发送到远程服务器中，若找到就直接返回不用再向服务器发送请求。也就是说，当访问数据时，将首先搜索缓冲区(即调用DataCache组件的方法)，若没有找到才从磁盘文件或数据库中读取数据，这样就减少了数据通过网络传输的次数，提高了系统的性能。客户端数据缓存接口采用IDL语言描述，用COM技术实现。

客户端数据缓存接口说明：

interface IDataCache: IDispatch

(下转第81页)

接用户的信息。

下面我们给出 telnet 服务的状态变迁集合的部分片断，如表2所示。

表2 telnet 服务状态变迁集合

```
#port 23 (telnet) finite state machine translation set table
# State    Input    Next  Parm  Cmd    Message
0          START    1     1     cat    @23.banner
0          NIL      0     1     cat    @23.banner
0          ERROR    0     1     cat    @23.banner
# initial prompt
1          NIL      2     1     cat    @23.login
1          ERROR    2     1     cat    @23.login
# user IDs
2          fc       3     1     echo   Password:
2          guest    3     1     echo   Password:
2          root     3     1     echo   Password:
2          daemon   3     1     echo   Password:
# passwords
3          root     4     1     echo   HOST_NAME $
3          guest    4     1     echo   HOST_NAME $
3          fc       4     1     echo   HOST_NAME $
3          daemon   4     1     echo   HOST_NAME $
# some commands
4          ls       4     1     cat    @23.ls
4          df       4     1     cat    @23.df
4          pwd      4     1     cat    @23.pwd
```

从以上的片断中，我们可以清晰地看出有限状态机的变迁集合的工作机制，以一个用户 telnet 登录为例，假如他连接上之后依次输入的是 telnet ×.×.×.×(登录)，guest(用户名)，guest(密码)，ls(命令)，他将依次看到的返回信息是文件@23.login 中的信息、字符串"Password:"、全局变量 HOST_NAME $ 的值，文件@23.ls 中的信息。

4 增强诱骗服务器的欺骗质量

我们的系统能实施有效的欺骗，但是欺骗质量并不高，攻击者经过多次试探之后很可能就能发现系统是个诱骗服务器。我们可以采取以下的手段来提高系统的欺骗质量：

(1)状态爆炸问题的解决。利用有限状态机分析和模拟协议所面临的主要问题是状态爆炸。为了控制状态爆炸，应寻找一定的方法减少状态的生成。使用人工智能从当前网络中的网络服务中的协议状态中学习，总结出最常使用的状态能有效减少状态数。

(2)模拟网络数据流，使得一些网络数据流分析系统不能发现欺骗。一种办法是获取当前内部网络的真实网络数据流进行简单的重放，第二种办法是获取外部网络与内部网络交换的数据流进行重放。网络数据流的重放能有效地迷惑攻击者的监听行为。

(3)动态配置欺骗服务器。真实的网络总是在不断变化中，诱骗服务器应当尽可能地反应出真实系统的一些变化，一成不变的诱骗服务器很容易被攻击者发现。我们可以使用一定的算法来随机产生返回给客户的信息。

(4)建立一个欺骗性的网络。单一的诱骗服务器不能有效地实施诱骗服务器的诱骗功能，如果攻击者没有发现诱骗服务器，那么诱骗服务器就失去了它的价值。我们可以在系统中设置多台诱骗服务器，并且在网络的 DNS 中增加相应的条目，提高诱骗服务器的主动诱骗功能。

结束语 诱骗服务器是网络安全的热点课题，本文详细分析了基于有限状态机的诱骗服务器的理论可行性，并就在 Linux 平台上实际架构诱骗服务器所必须处理的关键技术做了详细的阐述。当然，目前的诱骗服务器缺少动态的变化，自身的隐藏能力差，主动诱骗能力缺乏等等都影响其进入实用阶段，这些也是我们未来研究的重点内容。

参 考 文 献

1 Dill D L, The Murphi Verification System. In: Int'l Conf. Computer Aided Verification, 1996. 390～393
2 Klug D. HoneyPot and Intrusion Detection. Http://www.sans.org/infosecFAQ/Honeypots.html
3 RFC(Request For Comments)－764, Telnet Protocol specification
4 王建国，吴建平. 基于扩展有限状态机的协议测试集生成研究. 软件学报，2001，12(8)：1197～1204
5 夏春和，吴震，等. 入侵诱骗模型的研究与建立. 计算机应用研究，2002，4：76～79
6 刘湘辉，等. 利用有限状态机分析 TCP 协议握手过程的安全问题. 计算机工程与科学，2002，24(4)：21～23

(上接第78页)

```
{
[property,int] CacheSize;
// 从缓冲区中读取所需要的对象，并返回未查找到的对象 OID 列表
[method,long] ReadObjects([in,out,int] *pNum, [in,OID,size_is(*pNum)] *pOIDList, [in,boolean] bSpatial, [in,string] TBLName, [out] void **ppResult,[out,int] pNum2, [out,OID,size_is(*pNum2)] *pOIDList2 );
// 向缓冲区中写入对象
[method] WriteObjects ([in,out,int] pNum, [in,OID,size_is(pNum)] *pOIDList, [in,boolean] bSpatial, [in,string] TBLName, [in,out,long] *pLen, [in] void *pValues);
// 更新缓冲区中的对象
[method] UpdateObjects([in,out,int] *pNum, [in,OID,size_is(*pNum)] *OIDList, [in,boolean] bSpatial, [in,string]TBLName, [in,out,long] *pLen, [in] void *pValues);
// 删除缓冲区中的对象
[method,long] DeleteObjects([i,out,int] pNum, [in,OID,size_is(*pNum)] *OIDList,[in,string] TBLName);
};
```

总结 高速缓冲技术作为一种能够提高系统性能的有效技术广泛的应用于各种系统中。在分布式空间数据库中，基于重叠区域的客户端高速缓冲技术能够减少不必要的网络数据传输，提高分布式空间数据库的性能和可伸缩性。本文给出了客户端高速缓冲技术在分布式空间数据库中应用的系统结构，提出符合空间数据特点的高速缓冲置换策略，采取这种置换策略比单纯采用 LRU 策略能够获得更好的性能，最后讨论了基于事务的缓冲一致性问题。

参 考 文 献

1 Franklin M J, Carey M J, Livny. Transactional Client-Server Cache: Alternatives and Performance. ACM, 1997, 0362-5915/97/0900-0315
2 Güting R H. An Introduction to Spatial Database Systems. In processdings: VLDB Journal, 1994, 3(4)
3 Brinkhoff T. A Robust and Self-Tuning Page-Replacement Strategy for Spatial Database Systems. In: Proc. 8th Intl. Conf. on Extending Database Technology, Prague, Czech Republic, 2002
4 Beckmann N, et al. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. In: Proc. ACM SIGMOD Intl. Conf. on Management of Data, Atlantic City, NJ, 1990. 322～331
5 Guttman A. R-trees: A Dynamic Index structure for Spatial Searching. In: Proc. ACM SIGMOD Intl. Conf. on Management of Data, Boston, 1984. 47～57