Client/Server 数据库模型下的并发控制与恢复技术*)

邢建英[1] 张 艳[2] 李舟军[1]

(国防科技大学计算机学院 长沙 410073)[1] (四川大学数学学院 成都 610064)[2]

摘 要 并发控制与恢复技术在 client/server 结构 DBMS 中有着极其重要的地位,直接关系到运行的正确性以及数据库的健壮性。在 Client/Server 结构中,进行日常事务处理时的首要任务就是保持缓存的一致性,这是进行并发控制的基础。本文详细地分析和研究了 Client/Server 中用到的缓存算法和恢复技术,指出了现有技术存在的不足,并对其发展趋势进行了展望。

关键词 Client/Server,缓存,日志,并发控制,恢复

Concurrency Control and Recovery in a Client/Server DBMS

XING Jian-Ying[1] ZHANG Yan[2] LI Zhou-Jun[1]

(School of Computer, National University of Defence Technology, Changsha 410073)[1]

(School of Mathematics, Sichuan University, Chengdu 610064)[2]

Abstract Concurrency control and recovery play very important role in Client/Server DBMS, which affect correctness and healthy of database systems. Under Client/Server architecture, to maintain cache consistency is indispensable when normal transactions are in progress, which is the basis of concurrency control. This paper analyses the techniques of concurrency control and recovery used in Client/Server DBMS, points out the disadvantage of existing techniques, and prospects the future trends.

Keywords Client/Server, Buffer, Log, Concurrency control, Recovery

1 引言

随着计算机技术的发展、网络的普及以及应用的需求,DBMS 已经由早期的单一分散的结构发展成现在的 Client/Server(以下简称 C/S)结构。现在绝大部分的商用和实验 DBMS 都采用 C/S 结构。由于结构上的差别,C/S 数据库的恢复技术与传统的集中式、分布式以及其他(如共享磁盘、共享内存等)数据库系统的恢复技术还存在着很大的差别。

根据客户端向服务器发送请求的内容可将 C/S 结构 DBMS 分为两种:query-shipping 结构系统和 data-shipping 结构系统。query-shipping 结构系统是指客户端向服务器端发送查询消息(如 SQL 命令),由服务器接受命令并进行相关处理后将结果返回给客户。客户端主要负责与用户进行交互、显示结果,其他工作都由服务器来完成。一般关系数据库都是属于这种结构。这种结构的优点就是通信量小,设计相对简单,不需要复杂的并发控制;缺点是服务器承担的任务过多,导致负载太大,不利于多任务的并行。query-shipping 结构的数据库组织类似于单一集中式数据库。

data-shipping 结构系统是指客户向服务器请求数据,得到数据后将其缓存于本地,在本地进行相关处理后再将结果传回给服务器,服务器主要负责并发控制和恢复的工作,其他应用的处理都由客户来完成。在个人计算机功能日渐强大的现今,此结构充分利用了客户机的资源,能有效地减少服务器带来的瓶颈。现在的面向对象数据库系统(OODBMS)一般都是采用此种结构。虽然相对于 query-shipping 结构,data-shipping 结构系统的通信量要大一些,但在网络带宽日趋增强的情况下,这个问题并不显得那么突出。

与传统的数据库系统相比,query-shipping 结构数据库系统的恢复技术并没有太大的区别,因为二者的结构差别本来就不大。而 data-shipping 结构的数据库系统在恢复和日志子系统以及并发控制上却存在很大的差别,这是由其以下结构特征所决定的:

1)数据的修改主要在客户端完成,而其稳定拷贝却要保存在服务器上;

2)每个客户管理自己的缓存池;

3)客户与服务器间的通信比较奢侈(相对于其他结构访问本地内存);

4)客户机相对于服务器具有不同的硬件特性以及执行和可靠性特征。

本文的重点是分析与研究基于 data-shipping 的 C/S 数据库管理系统的并发控制和恢复技术。

基于 data-shipping 结构的 C/S 结构 DBMS 的 3 个主要功能——数据传输、并发控制和拷贝管理可以在不同的粒度上执行,如文件、页、对象或记录。依据数据传输单位的不同可分为以下 3 种结构[4]:基于对象的服务器(object-server)、基于页的服务器(page-server)和基于文件的服务器(file-server)。但经验[4]表明:页级的数据传输是最有效、最易实现的,并且相对于 object-server 结构,更加充分地利用了客户机的

*)国家自然科学基金项目(60073001,90104026,60473057);国家"十五"科技攻关计划项目"银行计算机灾难恢复系统研究",编号:2001BA102A07-04-01。**邢建英** 硕士研究生,主要研究方向为灾难备份与恢复;**张 艳** 博士研究生,研究方向为计算机网络与信息安全、灾难备份与恢复;**李舟军** 博士,教授,博士生导师,主要研究方向为进程代数理论、安全协议的形式化验证、灾难备份与恢复、数据仓库与数据挖掘。

资源，更大地减轻了服务器的负载。因此现有的许多 DBMS 都采用 page-server 结构。并发控制和拷贝管理可以采用不同的粒度组合[3]，可能存在的组合如表 1 所示。

表 1 并发控制和拷贝管理的粒度组合

	Concurrency Control	Replica Management
Granularity	page	page
	record	record
	record	adaptive
	adaptive	adaptive

但 page-server 结构的划分并不限于这一种方法。其实，针对面向对象数据库(OODBMS)，使用对象粒度更加易于考虑。这样，page-server 结构划分还可以有这几种：PS-OO、PS-OA 和 PS-AA。具体使用哪种结构，应根据需要来选择。

本文第 2 节主要介绍缓存一致与并发控制算法，并比较所列算法的优缺点；第 3 节介绍并分析两种 C/S 结构 DBMS 常用的恢复技术；最后对发展趋势进行了展望。

2 缓存一致与并发控制算法

事务缓存一致性维持算法应能保证访问过废弃数据的事务将无法完成提交。数据项若比该项最新提交的版本旧，就认为它是废弃的。我们根据是否允许事务访问废弃数据而将缓存算法分成两种：avoidance-based 和 detection-based[6]。这两种算法的区别就是 detection-based 类算法是惰性的(lazy)，需要事务来检查所访问数据的有效性；而 avoidance-based 类算法则是热心的(eager)，它们保证无效的数据将迅速而自动地从客户端缓存移除。

detection-based 类算法允许废弃数据在客户缓存保留一段时间，因此事务在被允许提交前必须检查它所访问的缓存数据是否有效。服务器负责维持相关信息来保证客户完成此类检查。detection-based 类算法之所以如此命名，是因为对废弃数据的访问会被明确地检查并检测出来。不同的是，在 avoidance-based 类算法下，事务永远没有机会访问废弃数据。avoidance-based 类算法使用 read-one/write-all(ROWA)方法来管理各缓存副本。ROWA 方法保证当执行修改的事务提交时，修改过的数据项的所有现存拷贝都必须具有相同的值。在 ROWA 方式下，事务允许读取所需数据的任何一个拷贝(如果本地存在，就读本地)，而当事务对数据项进行修改后，就需要将修改结果反映到所有拷贝。

detection-based 类算法根据客户事务何时向服务器发送缓存数据的有效性检查，可以分为以下 3 类：

(1)同步。每次事务对页初始访问时，等待同步消息；

(2)异步。也是对页初始访问时，不需等待；

(3)延迟。等到事务提交时才进行检查。

avoidance-based 类算法根据客户事务何时向服务器发送写意图声明，也可以有 3 种分类：

(1)同步。写动作要开始前，等待服务器回应；

(2)异步。写动作要开始前，不等待服务器回应；

(3)延迟。等到进行修改的事务进入提交阶段。

近 10 年来，对缓存算法的研究已取得了重要进展，并且其中的一些算法还得到了广泛的应用，表 2 列出了一些较为常见的算法。下面将对其中几种具有代表性的算法进行分析和比较。

表 2 常见的缓存算法

	Synchronous	Asynchronous	Deferred
Avoidance-Based	CBL[5] ACBL[3]	AACC[10]	O2PL[2]
Detection-Based	C2PL[2]	NWL[12]	AOCC[1]

2.1 C2PL[2]

C2PL 是最简单的一种缓存算法。早期的一些系统，如 ORION-1SX 原型、EXDOUS 存储系统都使用类似于此算法的缓存策略。

C2PL 中，缓存一致性通过使用“check-on-access”方法来维持。所有页拷贝都标注版本号，来唯一标识页状态。当客户事务需要访问它未获得合适锁的页时，向服务器发送请求并等待回应。如果该页在客户缓存中，就将页版本号附加到锁请求消息中。如果其它客户有相冲突的锁，锁请求就需在服务器等待，直到锁释放。如果请求的是读锁，服务器还需判断客户的缓存拷贝是否是最新的。如果不是，需在授权消息上附加有效的页拷贝，一起发送给客户。C2PL 使用严格的两阶段锁，所有的锁都是在事务提交或撤销后才释放。死锁都是由服务器集中处理的，一般的解决方法是杀掉最新加入的事务。

2.2 ACBL[3]

ACBL 属于同步的 avoidance-based 类算法。客户缓存数据和读锁，但在进行写操作之前需要从服务器获得写许可。ACBL 可以动态获得页或对象级的锁，因此它是 CBL 的自适应版本。客户首先申请获得页级写锁，如果失败就转而申请对象级写锁。如果该页缓存在其它客户，则要求他们进行锁降级(downgrade)或交出锁。ACBL 可以保证事务无法访问到废弃数据，因此不存在由此导致的撤销。

CBL(callback locking)[5]是 ACBL 的基础，是在商业系统中应用最多的算法，下面具体讲述其工作过程。

CBL 中，总可保证本地缓存数据是有效的，事务可以在不与服务器联系的情况下读本地数据。如果在本地缓存无法找到所需页，客户就向服务器发送请求；服务器在确定无其它活跃客户对该页拥有写许可后，向它返回所请求页的一个有效拷贝。写意图是以同步方式声明的，客户在授予事务本地写锁之前必须获得写许可。由于事务在其生命期内获得了写许可，因此当其完成所有操作后，不需任何一致性维持动作就可以提交。

服务器维持整个系统中缓存拷贝所在地信息。当客户从缓存池中丢掉页后，通过在它与服务器的下一个交互信息上附加消息的方式通知服务器。服务器根据消息来重新组织拷贝信息表。事务从执行它的客户的本地锁管理器获得锁，读锁请求以及已经得到写许可的写锁请求都可以在不通知服务器的情况下得到满足。如果没有写许可，请求写锁就会引起写许可错误，客户必须向服务器声明其写意图，并等待服务器授权后才能继续。当写意图到达服务器后，服务器向所有拥有所请求页的缓存拷贝的客户站点发布召回请求。客户将这样的请求作为写锁请求来对待。如果请求无法立即满足，说明它与活跃事务发生了锁冲突。当召回请求最终成功，客户就将所要求页清除并向服务器返回确认信息。当所有的召回请求得到确认后，服务器就将所请求页的写许可授权给相应客户并将此举动通知其它客户。随后，任何其它客户对该页的读或写请求都会被服务器阻止，直到客户释放写许可或服

务器召回写许可。

在事务的末尾,客户会将所修改缓存页的拷贝发给服务器,这样做是为了简化恢复。

2.3 AOCC[1]

AOCC是延迟的detection-based类算法。AOCC中,客户默认对所缓存数据具有读许可。但如果要对缓存数据进行修改,需将所有的写声明延迟到提交时间来进行。AOCC不阻止客户访问废弃数据。由已提交事务带来的修改,会以无效化消息的方式发送给相关的客户。为节省信息量,这些无效化消息是附加在其它消息上发送的。如果收到对象无效消息的客户已经访问了该无效数据,就需执行废弃数据撤销操作。AOCC是一种乐观的算法,不使用锁,不会存在读/写或写/写冲突,故也不会有死锁存在。但是,由于废弃数据引起的撤销,AOCC很容易出现饿死情况,即客户事务不断撤销而无法完成提交。

AOCC中,服务器必须对客户事务访问过的每个对象都进行相应的有效性检查。事务要提交时,服务器检查客户访问的是否是对象的最新版本。服务器对每个客户都维持一个无效队列,队列中存储着可能影响该客户的其它客户已提交的修改值列表。服务器在客户提交修改值列表进行有效性检查时使用此队列。

2.4 AACC[10]

AACC属于异步的avoidance-based类算法。它解决了AOCC(高撤销率)和ACBL(信息量大和信息阻塞开销大)的基本问题。

AACC中,客户可以不受事务约束保留读锁,而必须在事务结束后释放所持有的写锁。服务器主要按页来管理锁,客户则按页和对象来管理。服务器在几个客户同时向同一页写对象时也按对象来组织锁。如果存在锁冲突,由服务器来进行死锁处理。客户不会在执行写操作时阻塞。但它提交时,若使其它客户的缓存数据,废弃就会阻塞。AACC中,页锁有private-read、shared-read及write3种,而对象锁有read和write两种。服务器在满足客户页请求的同时,还会通知其该页是否还在其它客户缓存(以此来决定是采用private-read还是shared-read锁模式)。客户根据private和shared锁的概念来决定是以延迟还是以异步方式发送锁升级消息。

AACC的执行性能可以跟AOCC媲美,而其撤销率却低得多(跟ACBL差不多)。低撤销率是因为它是属于avoidance-based的,执行性能好是因为信息处理量低并且阻塞少。

3 恢复技术

在C/S结构数据库管理系统中,日志是在客户端生成的。但是,如果服务器不统一存储客户生成的日志,在恢复时就需要依赖客户。虽然将日志存储在客户本地磁盘中可以减少服务器的工作量,但是在绝大多数C/S结构DBMS中,这样做都是不可接受的,因为这要求安全性相对较高、可信赖的服务器在进行服务器失效恢复时要依赖不可靠的客户。

如果服务器管理日志磁盘,则客户可以仅返回已修改页(whole-page logging[13])。返回已修改页和日志记录(ARIES/CSA方法[9])时,只返回日志记录或已修改页中的一个(redo-at-server[13])。这几种方法都各有优缺点。目前应用比较广泛的是redo-at-server和ARIES/CSA方法。

3.1 redo-at-server[13]

redo-at-server方法是在ESM-CS的恢复方法[7]上修改而来的。ESM-CS恢复算法将日志记录和已修改页一起发回给服务器,但在事务提交前只保证日志记录写入到稳定存储器中即可。在redo-at-server中,服务器收到日志记录时,利用其中的redo信息来修改服务器中的页拷贝。这样做的缺点就是服务器可能会需要从二级存储器中读取页来应用日志记录。

3.1.1 ESM-CS的恢复算法[7]

ESM-CS是使用page-server结构的基于data-shipping的C/S系统。它恢复的实现部分主要包括两个主要组件:日志子系统管理并负责对稳定存储器上日志的访问;恢复子系统使用日志中的信息来负责事务回退和系统重启。恢复的实现还包括与缓存管理器和锁管理器的协同操作。

客户端通过向服务器发送消息来启动事务,同样靠发送消息来提交或终止事务;服务器可根据系统错误或死锁来决定终止哪个事务,然后在客户的下一个消息到来后通知客户该事务已终止(服务器无法主动建立与某个客户的连接)。事务执行过程中,客户对数据和索引页的修改产生日志,服务器负责管理日志,并根据日志终止某些具有危险性的事务。

ESM-CS的恢复算法是基于ARIES[8]的。

此恢复方法以最小化普通进程中恢复相关开销为目标,并提供合理的回滚和系统重启时间。此方法支持有效的缓存管理方法,允许客户与服务器间的灵活交互,允许客户端执行包括生成日志在内的大量工作。

此方法还存在以下不足:

1)由于此方法要求在事务提交时需将客户修改的页全部发送到服务器,并且在事务终止时间内,要将其在客户端所有的页都从缓存池中清除。这会造成很大的开销。

2)客户端除了写日志记录外不执行任何恢复动作,即使是普通的事务回滚客户端也不执行,导致了系统的复杂化。

3)只支持页级或粗粒度的锁,不支持记录锁。

3.2 ARIES/CSA[9]

ARIES/CSA可以对C/S正确地执行恢复,它支持WAL、细微粒度锁、部分回退和灵活的缓存管理规则(如steal和no-force)。不要求客户与服务器时钟同步,客户端与服务器都使用了周期性的检查点来保证能正确操作。

ARIES/CSA是为适应C/S模型而对ARIES稍加修改而成的,其中精彩之处为:

1)服务器代表失效的客户执行恢复。客户端获得检查点,服务器利用检查点信息来完成对客户的恢复;

2)服务器在其检查点中存放客户的脏页(dirty-page)列表信息,来保证客户恢复的正确性。

ARIES/CSA的主要优点有:

1)它支持细微粒度锁、steal和no-force方法;

2)在事务提交时,既不要求将客户方的页清空,也不要求立即将脏页强制发送到服务器;

3)通过在客户端执行普通的事务回滚,降低了服务器的负载及恢复复杂性;

4)ARIES/CSA支持索引和散列方法,可以实现逻辑undo;

5)服务器与客户端之间不要求时钟同步;

6)支持非常强大的提交LSN优化;

7)客户端可在本地指派LSN值来提高性能。

ARIES/CSA的不足之处是它只支持以页为单位的恢复,可以将其扩展为支持单个对象或记录。

结束语 本文论述了C/S结构数据库管理系统中恢复技术实现所遇到的难点，以及它与传统数据库所存在的差别；对C/S结构DBMS进行了详细的分类，并讨论了不同结构的优劣；分析并研究了C/S结构数据库中所面临的最大问题之一是并发控制和缓存一致性；最后介绍了几种恢复策略。

纵观近10年来对C/S结构数据库的恢复和并发控制的相关研究，发现主要工作还是基于数据页的，而基于索引的相关研究还比较少。下一步的工作就是研究基于索引的缓存一致与并发控制算法，将现有集中式系统的相关算法扩展到C/S结构。随着客户机和服务器性能的增强，适当增加并发控制和恢复的复杂性来换取执行性能是可行的(例如结构和控制的自适应扩展[11])。针对现在P2P技术的充分发展，考虑是否可以将P2P技术应用到缓存算法的改进中。

参 考 文 献

1 Adya A, Gruber R, Liskov B, et al. Efficient optimistic concurrency control using loosely synchronized clocks. In: Proc. ACM SIGMOD International Conference on Management of Data, 1995. 23～34

2 Carey M, Franklin M, Livny M, et al. Data caching tradeoffs in client-server DBMS architectures. In: Proc. ACM SIGMOD International Conference on Management of Data, 1991. 357～366

3 Carey M, Franklin M, Zaharioudakis M. Fine grained sharing in a page server OODBMS. In: Proc. ACM SIGMOD International Conference on Management of Data, 1994. 359～370

4 DeWitt D, Futtersack P, Maier D, et al. A study of three alternative workstation-server architectures for OODBS. In: Proc. 16th International Conference on Very Large Data Bases, 1990. 107～121

5 Franklin M, Carey M, Livny M. Global memory management in client-server DBMS. In: Proc. 16th International Conference on Very Large Data Bases, 1992. 596～609

6 Franklin M, Carey M, Livny M. Transactional client-server cache consistency: Alternatives and performance. ACM Transactions on Database Systems, 1997, 22(3): 315～363

7 Franklin M, Zwilling M, Tan C, et al. Crash recovery in client-server EXODUS. In: Proc. ACM SIGMOD International Conference on Management of Data, 1992. 165～174

8 Mohan C, Haderle D, Lindsay B, et al. ARIES: A transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. ACM Transactions on Database Systems, 1992, 17(1): 94～162

9 Mohan C, Narang I. ARIES/CSA: A method for database recovery in client-server architectures. In: Proc. ACM SIGMOD International Conference on Management of Data, 1994. 55～66

10 Ozsu M T, Voruganti K, Unrau R. An asynchronous avoidance-based cache consistency algorithm for client caching DBMSs. In: Proc. 24th International Conference on Very Large Data Bases, 1998. 440～451

11 Voruganti K, Ozsu M T, Unrau R. An adaptive hybrid server architecture for client caching ODBMSs. In: Proc. 25th International Conference on Very Large Data Bases, 1999. 150～161

12 Wang Y, Rowe L. Cache consistency and concurrency control in a client/server DBMS architecture. In: Proc. ACM SIGMOD International Conference on Management of Data, 1991. 367～376

13 White S, DeWitt D. Implementing crash recovery in quickStore: A performance study. In: Proc. ACM SIGMOD International Conference on Management of Data, 1995. 187～198

14 Bernstein PA, Hadzilacos V, Goodman N. Concurrency control and recovery in database systems. Reading, Mass: Addison-Wesley, 1987

15 Panagos E, Biliris A. Synchronization and recovery in a client-server storage system. AT&T Research, 600 Mountain Avenue, Murray Hill, NJ 07974, USA, 1996

(上接第147页)

FTP Server交换数据[4,5]。客户端三方控制数据传输的主要代码片段如下：

```
public void thtransfer(String host1,
    int port1,
    String sourceFile,
    String host2,
    int port2,
    String destFile,
    GSSCredential cred,
    int parallelism
    ) {
    try {
    GridFTPClient source = new GridFTPClient(host1, port1); //与host1建立连接
    GridFTPClient dest = new GridFTPClient(host2, port2); //与host2建立连接
    setParams(dest, cred); //调用自定义函数setParams()设置被动模式
    setParams(source, cred); //调用自定义函数setParams()设置主动模式
    source.setOptions(new RetrieveOptions(parallelism)); //设置并行性
    long size = source.getSize(sourceFile);
    HostPortList hpl = dest.setStripedPassive(); //设置条状传输
    source.setStripedActive(hpl);
    source.extendedTransfer(sourceFile, dest, destFile, null); //调用GridFTPClient的传输函数
    }catch(Exception e)
    {
    e.printStackTrace();
    }
    }
private void setParams(GridFTPClient client, GSSCredential cred)
    throws Exception{
    client.authenticate(cred); //进行安全认证
    client.setProtectionBufferSize(16384); //设置保护缓冲
    client.setType(GridFTPSession.TYPE_IMAGE); //设置传输类型为IMAGE
    client.setMode(GridFTPSession.MODE_EBLOCK); //设置主机模式为扩展模式
    }
```

结论 GridFTP是Globus网格计算环境中数据管理的一个重要服务，它是一个通用的数据传输协议，提供数据在广域环境中安全、可靠、有效地传输功能。实际上GridFTP由许多下一层的子协议组成，子协议不断扩充诸如双向数据传输、管道指令、统一的Web服务接口、数据预约和空间锁定、精确的传输时间估计等新的功能。在我们的课题研究中，成功地开发了客户端控制软件，在局域网(校园网)中实现了C/S和三方控制两种模式下的多种形式的数据传输。下一步工作中将研究应用GridFTP的新功能，研究网格数据管理中的复制管理服务。

参 考 文 献

1 Globus官方网. GridFTP的白皮书[EB/OL]. http://www-fp.globus.org/datagrid/deliverables/C2WPdraft3.pdf

2 徐志伟，冯百明，李伟. 网格计算技术[M]. 北京：电子工业出版社，2004

3 Globus官方网. Java GridFTP client-programmer guide [EB/OL]. http://www-unix.globus.org/cog/jftp/guide.html

4 Silva V. Transferring files with GridFTP [EB/OL]. http://www-128.ibm.com/developerworks/library/gr-ftp/index.html

5 陈东锋，杨寿保，冯征，郭磊涛. 基于Globus的邮件系统Gmail的设计[EB/OL]. http://www.chinagrid.net/grid/paperppt/USTC/gmail.pdf