维普资讯 http://www.cqvip.com

计算机科学 1997Vol. 26№. 2

# 移动数据库中客户机的断接操作*)

Disconnected Operation for Clients in Mobile Database Systems

李 霖 周兴铭

(国防科技大学计算机学院 长沙 410073)

**Abstract** Mobile clients (MC) may be frequently disconnected from database servers in a mobile computing environment. This paper presents a caching mechanism for MC which can support MC's disconnected operation, i.e. mobile users can continue access databases when MC being disconnected. The key component of MC's caching mechanism is the cache manager(CM), which maintains a subset of database on MC, and simulates a database server when disconnected. MC's caching improves the availability of mobile database systems greatly.

**Keywords** Mobile computing, Cache, Disconnected operation, Mobile database

## 1 引 言

在移动计算环境中,移动客户机(Mobile Client,简称 MC)可能经常处于断接状态,即 MC 无法与数据库服务器通信[1,2]。这种断接情况的发生通常有两个原因:一是MC的自愿断接,即MC由于自身的原因,比如电源不足、无线网络代价太高、MC 将移动到一个没有网络连接的地方等,主动断开与服务器的网络连接;二是MC的意外断接,即由于无线网络的故障或服务器发生故障,造成 MC 被动地断开与服务器的网络连接,在目前的移动计算环境中,大部分的断接都属于 MC 的自愿断接。然而 MC 在断接时仍需要访问移动数据库的有关数据,我们称之为移动客户机的断接操作[1]。显然,MC 的断接操作只有通过数据库的复制或缓存技术来支持。

某些移动数据库系统通过在 MC 上采用与服务器完全相同的复制来支持断接操作,如 Bayou 系统[4]。但是,我们认为 MC 不宜采用服务器级复制,其主要原因是:1)服务器之间通常保持固定高速的网络连接,而 MC 经常与其它服务器断接,使数据库复制之间的同步更加困难;2)服务器拥有相对强大的处理能力和存储容量,而 MC 的处理能力和存储容量都有限,难以维持一个完整的数据库复制;3)服务器的硬盘可以看作稳定存储器,并由专门的数据库管理员(DBA)负责维护、备份,即使发生故障,也可以进行恢复,且其安全问题也容易控制;而 MC 由于其移动性,易于发生故障、损坏甚至失窃,因此其硬盘不能再认为是稳定存储器,可能会永久失效,其安全也得不到保障;4)相对于服务器来说,MC 的数量较多,而且随着技术的进步,使用 MC 的用户将会越来越多,如果每个 MC 都作为一个复制服务器,则不仅要求移动数据库系统支持大规模的复制节点,而且还必须支持复制集合的动态加入与退出。因为系统中可能不断地有新的 MC 加入,而 MC 也可能随时离开系统,这两个要求都是难以实现的。因此,我们选择在 MC 上利用缓存技术来支持 MC 的断接操作,与服务器级复制相比,MC 的缓存属于次一级的地位,它与服务器级复制互相配合,共同支持 MC 对移动数据库的访问,从而提高移动数据库的可用性。

## 2 移动客户机的基本结构

采用数据库缓存的 MC 基本结构如图 1 所示,其中,缓存管理器(CM)是移动客户机的核心。用户

*)本课题受九五国防预研项目"并行与分布式数据库"资助。**李霖** 博士研究生,主要研究兴趣为移动计算与移动数据库系统;**周兴铭** 中科院院士,主要研究兴趣包括高性能机体系结构、并行与分布式数据库技术等。

维普资讯 http://www.cqvip.com

对移动数据库的所有访问都必须通过 CM 完成。CM 负责管理 MC 上的缓存及其元数据，并维护两个事务日志，即脱机事务日志(OTL)和夭折事务日志(ATL)。

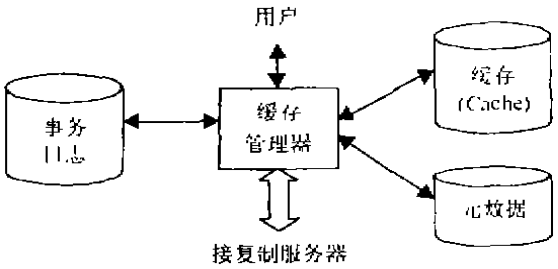


图 1　MC 缓存结构示意图

当 MC 可以与服务器通信时，CM 负责从服务器上更新本地的数据库缓存，同时将用户提出的事务转交给服务器执行；在断接状态，CM 就代替了服务器，利用缓存对用户的事务进行处理，并将更新事务记录在脱机事务日志中，等待下一次与服务器联机时交给服务器执行。CM 还负责维护一些元数据，它们记录了有关缓存集合的描述以及数据库的结构信息等。由于移动计算机技术水平的发展，无论是其处理能力还是存储容量，都已经足以满足个人 DBMS 的基本需求，例如 Sybase 公司的 SQL Anywhere，Watcom SQL 等。因此，在本文中，我们假设 MC 上已有基本数据库管理系统的支持，而将重点放在讨论缓存管理器在缓存管理和断接支持上的特有功能。

与服务器相比，MC 的存储容量仍然是有限的。尽管现在的笔记本电脑已拥有高达几个 G 的硬盘容量，但服务器及其规模也在迅速的膨胀，使得 MC 仍然难以容纳整个数据库系统。另一方面，即使 MC 能够放下整个服务器的数据库，一般用户也不会希望这么做，这既有安全方面的问题，也有代价的问题：将整个数据库的缓存与服务器完成同步将是一个代价高昂的任务。因此，我们提出的 MC 缓存机制允许用户只指定 MC 缓存服务器上数据库的一个子集。

## 3　移动客户机缓存的子集描述

Keller 在文[5]中提出一种基于谓词的缓存机制，它允许根据客户的查询操作，使用任意的 SPJ(选择、投影、连接)谓词来定义客户机的缓存内容。但是，Keller 只在理论上对缓存的定义及维护进行了探讨，而没有涉及任何实现问题，因此是否具有实用性还难以确定。另外，实体化视图技术[6]也可以支持 MC 建立一个数据库子集的本地备份，但是它不支持反向更新，即客户对本地缓存的更新不能反馈给服务器，因此也不适用于 MC 的缓存机制。在我们对 MC 缓存的讨论中，更为关注的是断接操作的支持，因此暂时采用了一种简单的静态缓存子集描述方法。

我们采用的静态缓存子集描述方法如下：MC 以数据库中的关系为基本单位定义缓存的数据库子集，即定义缓存集合为 $\{R_1, R_2, \cdots, R_n\}$，并且允许对每个缓存的关系附加一个或一组谓词演算，谓词演算可以是选取(select)或者投影(project)，其中最多只能有一个投影谓词，而且其所指定的属性集合中必须包括关系的主键。一个缓存关系中的所有记录必须满足该关系的所有谓词，即各个谓词的交集，因此，MC 缓存定义的形式化描述为：

$$CCD=\{(R_i, PS_i) \mid 1\leqslant i\leqslant n\}$$
$$PS_i=\{P_{ij} \mid 1\leqslant j\leqslant Q_i\}$$

其中，CCD 表示 MC 的缓存描述，$R_i$ 为 MC 要求缓存的服务器数据库中的关系，$PS_i$ 为定义在 $R_i$ 上的谓词集合，n 表示 MC 缓存中的关系数，$Q_i$ 表示 $PS_i$ 中的谓词个数。

例如，服务器上的数据库包含库存表 INVENTORY(ID, PID, QUANTITY, FACTORY, INDATE)，产品表 PRODUCT(PID, NAME, QUALITY, COLOR)，定单表 ORDER(OID, UID, PID, QUANTITY, INDATE, OUTDATE)，用户表 USER(UID, NAME, ADDRESS, PHONE)，则一个 MC 可以定义其缓存为：

$$CCD=\{\text{INVENTORY}, \text{PRODUCT}\}$$
$$PS(\text{INVENTORY})=\{\sigma_{\text{QUANTITY}>50}\}$$
$$PS(\text{PRODUCT})=\{\pi(\text{PID}, \text{NAME}, \text{COLOR}), \sigma_{\text{COLOR}=\text{'RED'}}\}$$

虽然上述方法有较多的局限性(例如，它不能支持基于连接的子集定义)，但易于实现，并且足以满足一般断接用户的需要(对于无法直接支持的子集定义，CCD 也可以间接支持，例如连接，CCD 可以定义两个连接关系都缓存全部，缺点是代价可能大大增加)。在今后的工作中，我们将进一步考虑 MC 缓存的描述问题，包括允许一般的 SPJ 谓词，支持 MC 根据用户查询行为动态定义缓存内容等。

## 4　缓存管理器 CM

为了支持移动用户的断接操作，MC 的缓存管

维普资讯 http://www.cqvip.com

理器CM具有三种不同的工作状态，即联机状态、脱机状态、集成状态。根据与服务器的网络连接状况的不同，CM将在这三种状态之间转换，如图2所示。下面我们将分别介绍CM在这三种状态下的任务以及状态之间的转换。

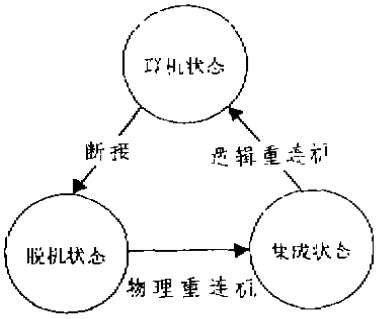


图2 MC缓存管理器的状态转换图

### 4.1 CM的联机状态

当移动客户机MC与数据库服务器连接时，缓存管理器CM工作在联机状态。在联机状态下，CM的主要任务包括联机事务处理、MC缓存的建立与同步等。

**4.1.1 联机事务处理** 在联机状态下，MC的事务处理主要依靠服务器的数据库完成。这时，CM基本上只起到一个转发代理的作用，它将MC用户提出的数据库访问事务直接转交给与之连接的服务器，由服务器负责分析并执行这些MC事务，并在服务器返回处理结果时转发给MC的用户。

**4.1.2 缓存的建立与同步** 在MC与服务器建立连接后，如果用户需要在断接时访问数据库，而此时MC的缓存还没有建立，则CM将负责与服务器一道，建立本地的缓存数据库。首先，CM要求用户定义本地缓存的子集描述，即CCD，然后将CCD报告给服务器S。在接到MC发来的缓存描述CCD之后，服务器S将数据库中所有满足CCD描述的关系记录及相关的元数据一起发送给MC。最后，缓存管理器CM将接到的元数据及记录保存在本地数据库中，从而建立起本地的缓存。

在我们所讨论的移动计算环境中，绝大部分的MC断接都属于MC的自愿(主动)断接。而且，我们设计MC缓存的主要目的是支持MC的断接操作，在MC联机时暂不使用缓存。因此，与传统客户/服务器数据库系统中的客户缓存不同，(如在文[7]中，客户机的缓存主要用于提高联机访问服务器的性能)，MC的缓存并不要求在联机过程中总是保持与服务器的同步，为了保证MC脱机状态的需要，只需在MC主动断接之前与服务器完成一次缓存同步即可。于是，对每个MC，在其联机期间只需进行一次缓存同步，大大减小了服务器的处理开销。

由于MC的缓存只是服务器上数据库的一个子集，因此不能简单地采用直接将服务器的更新事务日志传送给MC，并由MC根据事务日志来更新本地缓存的方法。为此，我们专门为维护MC缓存的同步设计了一种基于删除操作日志的缓存同步算法——CS同步算法。由于篇幅所限，我们将在另一篇论文中专门介绍CS算法的细节(见本期P10～13)。

### 4.2 CM的脱机状态

在MC与数据库服务器断接之后，缓存管理器CM将转入脱机状态，利用本地缓存仿真服务器的数据库管理功能，继续支持用户对数据库的访问和操作。与服务器相同，用户通过CM访问数据库时也以事务为基本单位，包括查询事务(即只读事务)和更新事务。CM在本地缓存允许的范围内，负责解释并执行用户的事务，并将用户提出的更新事务记录在MC的脱机事务日志中。

由于MC已经与服务器断接，因此用户的事务无法发送到服务器上执行。此时缓存管理器CM向用户提供两种选择：严格一致性事务和弱一致性事务。严格一致性查询事务要求读到服务器数据库的最新值，严格一致性更新事务则要求事务立即提交，并将更新结果记入到服务器的数据库中；而弱一致性事务可以在MC本地的缓存数据库上执行。显然，在断接状态下CM只能支持弱一致性事务，只有在联机状态下才支持严格一致性事务。

**4.2.1 查询事务处理** 如果用户要求查询到最新的数据库状态，则必须使用严格一致性查询，对于此类查询，CM将返回一个错误标志，表明MC正处于断接状态，不能执行严格一致性查询。因此，从这个意义上来说，MC的断接对用户不是完全透明的。我们之所以选择让用户了解到MC断接状况的存在，是因为这比实现完全透明性更有实际意义。如果用户知道MC正处于断接状态，他将避免使用严格一致性事务，并且对今后MC重新联机时可能发生的事务回滚有所准备。当然，MC的缓存管理器CM仍然支持一定的应用透明性：对用户来说，在MC断接的情况下，除了执行严格一致性查询和更新事务(将在下一小节中讨论)时会得到失败信息之外，其它事务的执行看上去与MC联机时是完全一样的。

维普资讯 http://www.cqvip.com

如果用户允许读到过时的甚至不一致的数据，则使用弱一致性查询。CM 根据 MC 缓存中的内容完成用户提出的弱一致性查询事务。由于 MC 缓存中的数据记录了断接期间用户更新事务的执行结果，而在稍后 MC 与服务器重新连接时，这些更新事务有可能因与服务器的更新事务发生冲突而被回滚，因此用户弱一致性查询事务的查询结果有可能与服务器上数据库的状态存在一定的偏差(不一致)。但是，在实际应用环境中，如果 MC 处于断接状态，则对用户来说读到略有偏差的数据总比没有任何数据要好，例如移动推销员需要查询客户数据，更何况在设计实际应用时，可以使数据库的更新冲突率降到很低。因此，在 MC 断接时，CM 提供弱一致性查询是具有实际意义的。

当 CM 接到用户的一个弱一致性查询事务 qt 时，首先必须根据本地缓存的子集描述 CCD，判断利用本地缓存是否能完成 qt。这种判断过程可以采用 Keller 在文[3]中描述的技术，在这里我们做一个简单的介绍。

对于一个一般的 select-from-where SQL 查询来说，其 where 子句中指定的查询谓词可以分为两类，即点查询(point-query)和面查询(range-query)，点查询指定一个关系中的最多一个记录，而面查询则指定一个或多个关系中的一组记录(其结果集合中可以有 0 个、1 个或多个记录)。当 CM 接到一个用户查询时，设其查询谓词为 Q。如果 Q 可以分解为一个或多个点查询，则 CM 直接在缓存上分别执行这些点查询，如果所有记录都存在，则 CM 可以向用户返回结果(Q 可以在缓存上执行)；否则，CM 将 Q 看作一个面查询，并继续下面的判断过程：

如果 Q 是一个面查询谓词，则 CM 将其与缓存描述 CCD 进行比较：

1. 若 Q 可以根据 CCD 中所有谓词的交集求出，则说明所有 Q 指定的记录都存在于缓存中；

2. 若 Q 无法根据 CCD 中所有谓词的交集求出，则 CM 无法确定 Q 是否包含于缓存之中，因此 CM 认为 Q 超出本地缓存的范围。

如果 CM 可以确定查询事务 qt 的查询范围完全包括在本地缓存中，则可以根据缓存内容执行 qt，并将查询结果返回给用户。否则，即 qt 的查询范围超出了本地缓存的界限，这时 CM 可以有两种选择：一是不执行 qt，直接向用户返回一个失败信息，告知用户目前无法执行该查询(缓存内容不足)；二是仍然在本地缓存上执行 qt，将结果返回给用户，同时指明该查询结果可能是不完全的。用户在提出查询事务时，可以指定其中一种处理方式。CM 之所以提供第二种选择，是因为在断接条件下，用户可能希望尽可能地得到查询数据，即使查询结果是不完全的，也总比完全没有结果要好。

4.2.2 **更新事务处理** CM 在脱机状态下负责分析并执行用户提出的更新事务。由于在断接时 MC 无法与主服务器通信，因此 CM 不支持严格一致性更新事务；如果用户向 CM 发出严格一致性更新事务，CM 将返回失败信息，并指明原因(MC 正处于断接状态)。因此，在本节中讨论的用户更新事务都指的是弱一致性更新事务。由于在脱机状态下执行的更新事务最终将在服务器上查询执行，因此在提交给服务器之前，我们将这些更新事务称作暂时事务。

在接到用户提出的一个更新事务 t 时，CM 首先判断 t 的读集合和写集合是否都包含于本地缓存中，这与 CM 处理查询事务时对查询谓词 Q 的判断方法基本相同。如果 t 的读集合或者写集合超出了缓存的范围，则 CM 不能执行 t，并向用户返回一个失败信息。否则，说明 CM 可以在本地缓存上执行 t，则 CM 为 t 分配一个本地唯一的递增事务标识符，记为 t.tid，为区别于服务器分配给事务的标识符，我们将 CM 分配的 tid 称作事务的临时标识符。为了在 MC 重新与服务器联机时将断接期间的用户更新事务发送给服务器重新执行，CM 将 t 记录到本地的一个脱机事务日志(OTL)中(在 OTL 中，所有暂时事务都按照 tid 的次序排列)。然后，CM 在本地缓存上执行 t，并将 t 的临时事务标识符 tid 与执行结果返回给用户。

在脱机状态下，CM 将用户更新事务的执行结果记录在本地缓存中数据记录的暂时版本中；相应地，我们把 MC 的缓存中由服务器更新的记录版本称作提交版本。MC 缓存的暂时版本可以允许用户更新事务读取前面执行的暂时事务的结果，也允许用户的查询事务读到该事务之前的暂时事务结果。当 MC 重新联机时，MC 缓存的暂时版本将被作废。

为了方便事务日志的管理，提高事务日志的可靠性，CM 将事务日志存放在本地数据库中，利用本地数据库管理系统的能力维护日志，而且在存取事务日志时也充分利用了数据库管理系统的优点(如索引)，因此 MC 脱机事务日志的存取是高效的，同时也是可靠的。

### 4.3 CM 的集成状态

维普资讯 http://www.cqvip.com

最后，当 MC 与服务器重新建立网络连接时，CM 将转入集成状态。CM 的集成状态是介于脱机状态与联机状态之间的一个过渡状态，在此期间，CM 必须完成以下任务：

● 将断接期间执行的用户更新事务（脱机事务日志）发送给服务器，由服务器重新执行；

● 如果发现暂时事务已被主服务器夭折，CM 将报告给用户，并帮助进行善后处理。

只有在完成集成状态的任务之后，CM 才能进入正常的联机状态。因此我们也把这个过程称为"逻辑重联机"，以区别于 MC 与服务器的物理重联机过程。下面我们将具体介绍 CM 在集成状态下的任务。

4.3.1 **脱机事务日志的发送** 在 MC 断接期间，工作在脱机状态下的 CM 允许 MC 的用户继续访问数据库，包括对数据库的查询和更新操作，但是，在脱机状态下 CM 执行的用户更新事务都暂时记录在 MC 的脱机事务日志中，并且它们的执行结果也是存储在 MC 缓存中的暂时版本中。因此，在 CM 进入集成状态时，需要将这些暂时事务的执行结果合并到服务器的数据库中。CM 采用的合并方式是将断接期间的暂时事务发送给服务器，由服务器负责重新执行这些暂时事务。

在集成状态下，CM 将 MC 本次断接期间执行的更新事务，即脱机事务日志中的所有暂时事务，按照临时标识符 tid 的次序，逐个发送给服务器。服务器接到这些暂时事务之后，依照其本来的顺序，像服务器上普通的更新事务一样，逐个进行处理。然后，如果这些暂时事务顺利提交，服务器将它们的正式标识符与临时标识符 tid 一道，返回给 CM。在 CM 接到服务器返回的成功提交事务的正式标识符之后，从脱机事务日志中删除对应的暂时事务。

在 MC 断接期间，服务器上可能执行了其它更新事务，因此在服务器接到 MC 的暂时事务并执行时，可能出现与前面的更新事务冲突的情况。我们采用时间戳的方法检测事务冲突，即 MC 的暂时事务包含所读记录的更新时间戳，如果服务器在重新执行一个暂时事务 t 时，发现 t 所读记录的更新时间戳已发生改变，即说明 t 与服务器先前执行的某个更新事务冲突，因此服务器必须夭折 t，并将夭折信息报告给 CM。如果 CM 发现暂时事务 t 已被主服务器夭折，则 CM 将 t 从脱机事务日志中移出，并存入到夭折事务日志中，以利于稍后进行的夭折事务善后处理。

为了提高 CM 集成状态下暂时事务的成功提交率，我们还设计并采用了一种基于应用语义的事务冲突检测与消解技术，它可以减少许多不必要的冲突，从而大大降低暂时事务的夭折率。限于篇幅，本文不再赘述。

4.3.2 **事务夭折后的处理** 如果发现有被服务器夭折的暂时事务，CM 将向 MC 用户发送一个消息，同时指出夭折事务日志的地址。如果用户需要对夭折事务进行善后处理，可以选择在联机状态或脱机状态进行。CM 在集成状态只负责检查和通知用户。我们在这里简单介绍一下 CM 辅助 MC 用户对夭折事务的善后处理过程。

在脱机状态和联机状态，用户可以通过 CM 的夭折事务日志获知被夭折事务的所有细节，包括原始操作序列、夭折原因等，然后，在 CM 的帮助下，用户可以选择以下几种处理方法：重新执行该事务，修改后执行该事务，或者放弃该事务。为了方便用户的处理过程，我们在 MC 上设计了一个专门的夭折事务处理程序，它可以和 CM 互相配合，简化用户的操作，使夭折事务的善后处理更加智能化。

## 参 考 文 献


1 Imielinski T, Badrinath B R. Mobile wireless computing: challenges in data management. CACM, 1994, 37(10): 18～28

2 李霖，周兴铭 移动数据库技术：机遇和挑战. 计算机科学，1997，24(3)

3 Kistler J J. Disconnected operation in a distributed file system: [Ph. D. dissertation]. Carnegie-Mellon University, May1993

4 Demers A, et al. The Bayou architecture: support for data sharing among mobile users. In: Proc. IEEE Workshop on Mobile Computing Systems and Applications. Santa Cruz, California, 1994. 2～7

5 Keller A M, Basu J. A predicate-based caching scheme for client-server database architectures. VLDB Journal, 1996, 5(1): 35～47

6 Widom J, editor. Special issue on materialized views and data warehousing. IEEE Data Engineering Bulletin, 1995, 18(2)

7 Franklin M. Caching and memory management in client-server database systems: [PhD Thesis]. Dept. of Computer Science, University of Wisconsin, July, 1993