基于 RMS 查询分片调度的移动数据库同步研究

Mobile Database Synchronization Based on RMS Search Patch Scheduling

刘 巅 尧飘海 张云华 (浙江理工大学 信息与电子学院 浙江 杭州 310018)

摘 要: 针对开发移动设备应用程序的各种受限性和对第三方移动数据库的依赖性,本文提出基于 RMS 查询分片调度的移动数据库同步复制机制,并以杭州地区电力负荷移动查询系统为例,重点分析了该机制中的冲突检测及消解策略,并给出具体的实现算法,同时以事务结果集的方式进行同步处理,为以 RMS 作为移动数据库的数据同步机制提供了一套可行的解决方案。

关键词: 移动数据库 数据同步 事务 冲突检测 分片 RMS

1 引言

移动通讯技术的快速发展和智能手机、PDA、掌上电脑等便携式计算设备的大量普及,促使了移动计算环境的发展,人们可以使用手上的移动设备通过无线网络获取各种服务,而且对信息的检索已经发展到以数据库为中心的阶段。于是,"移动计算"和"移动数据库"的概念应运而生,而将这种移动计算环境应用于数据库领域中,其结果是移动数据库系统的产生和发展。

移动计算环境具有移动性、低带宽、网络稳定性差等特点,如何维护移动数据库和服务器之间的数据一致性是当前研究的焦点。目前,人们已经提出了很多模型与算法来解决数据同步和同步过程中带来的冲突问题,包括两级复制算法、容错型定额同意方法、主动复制机制、基于双时间印的事务级同步模型[1]等等,但这些方法都存在一些问题,如它们的前提是移动终端数据库的存储容量可以存储整个数据库的副本或者应用程序的大小和可用堆栈不受限制等等;另外,当移动用户增加,断连时间增长时,将会导致大量的事务回滚,事务提交的成功率较低。

本文讨论了基于记录管理系统(RMS)查询分片调度的移动数据库同步设计和实现方法,将 RMS 作为一个移动数据库系统,充分考虑了移动计算设备存储容量受限的问题,同时利用对 RMS 操作的单事务作为数据同步粒度来提高事务的提交成功率,特别适合一些数据量较大的记录存储系统,如移动图书管理系统、移动物流配送系统、电力负荷移动查询系统等等。

2 移动数据库与 RMS

2.1 移动数据库系统的概念和基本特点

移动数据库是数据库技术和网络技术相结合的产物,是一种支持移动计算环境的分布式数据库,移动设备上的数据库系统由于涉及数据库技术、分布式计算技术以及移动通讯技术等多个学科领域,目前已经成为一个十分活跃的研究和应用领域。

和传统的分布式数据库相比,无线网络的不稳定性会导致频繁的网络分割,要求移动设备经常在网络断连的情况下工作。为了支持移动设备的断网操作,目前广泛采取了基于数据复制技术的同步处理策略,如乐观复制机制等,它们允许用户在断网的情况对本地的数据副本进行操作,当网络正常时,本地移动终端和远程服务器进行数据同步,最终实现各移动设备上的数据与服务器上的数据保持一致。

2.2 移动数据库的体系结构

移动数据库的体系结构有多种,根据不同的体系结构对应有相关的数据复制机制和相关算法,目前移动数据库的体系结构中使用最多的是如下所示的三级体系结构[1]:

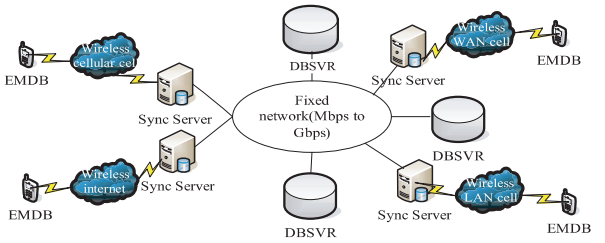

图 1　复制的移动数据库体系结构

(1)**数据库服务器**(DBSVR):**可以是任意大型的数据库系统,例如** Oracle、Sybase、DB2 **等,每个数据库保存有一个完整的数据副本,他们由固网连接起来,构成传统的分布式数据库系统。**

(2)**同步服务器**(Sync Server):**进行分布式事务处理,同时带有无线通讯接口,利用无线网络和固网**(Fixed network)**实现移动设备和** DBSVR **之间的数据同步操作。**

(3)**嵌入式移动数据库**(EMDB):**移动设备上的数据库系统,带有无线通讯接口,允许在断网的情况下对本地保存的数据副本进行操作,在下次连网时使用无线网络和同步服务器进行数据交互。**

2.3 RMS 简介

记录管理系统[2](Record Management System, RMS)**是移动信息设备简表**(Mobile Information Device Profile, MIDP)**的一个主要子系统,是一种应用程序编程接口**(API),**为** MIDP **应用程序提供本地的、基于设备的数据持久存储功能。**

实际上,RMS **就是一个模仿了面向记录的小型的数据**

库管理系统,它以一种简单的,类似表格的形式来组织信息,并存储起来形成持久化存储,以供应用程序在重新启动后继续使用, RMS **的存储结构图如下所示**[2]:

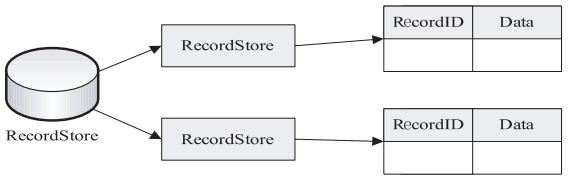

图 2　RMS 存储结构图

应用程序中的 RMS **被看作一个移动数据库,**RMS **本支持增加、删除、更新、查询等操作,当前很多的移动设备上并不支持市场上发布的移动数据库,如非智能手机等,同时对于某些应用无须用到第三方提供的移动数据库,此时,可以将** RMS **作为移动数据库来使用,对于应用数据进行永久性存储。**

3　基于 RMS 数据复制的思想

3.1 分片调度的引入

在基于 RMS **查询分片调度的数据库同步过程中,以** RMS **作为** EMDB**,考虑到应用程序大小和可用堆栈往往是受限的,引入分片调度的机制。移动设备将整个数据库的副本保存在扩充的存储卡中,而应用程序根据自身的需要进行分片调度,这种分片式调度的数据同步模型如下图所示:**

图 3　基于 RMS 分片调度的数据库同步模型

为了说明此模型,以杭州市电力负荷移动查询系统为例。杭州市共有八个区,电力负荷监控点有上万个,每一个监控点对应数据库中的一条记录,移动设备的存储卡上保存了所有测点负荷的数据副本,它们分别存储在八个数据分片中,每个分片用一个分片标示位来表示。由于应用程序的大小和可用堆栈的受限性,应用程序的 RMS **中只保留了一个区的测点负荷数据集。假设移动设备** MH **中** RMS **保存了西湖区的测点负荷信息,部分记录如下:**

表 1　MH 上存储的分片记录

采集点 ID	中文描述	场站名称	电压等级	负荷类型
0051 – 6 – 1	浙大玉泉校区体育馆	浙江大学	260	地区负荷
0051 – 6 – 2	浙大玉泉校区图书馆	浙江大学	500	地区负荷
0051 – 6 – 3	浙大玉泉校区医务室	浙江大学	300	网供负荷

RMS **在分片机制下对本地的数据副本进行如下**

处理:

(1)增加与删除记录:以 MH 为例,MH 保存了西湖区的测点记录,在执行增加(删除)操作之前,先将该记录追加到数据同步缓冲池中,缓冲池里的记录追加"设备 ID","分片标示"和"操作标示"三个字段,使 RMS 中存储的静态记录以单事务的形式存储到同步缓冲池中,追加的三个字段为数据同步时的处理附加字段,后面的五个字段为记录的有效字段,其中采集点 ID 是事务有效字段的主键;操作标示中'A'表示增加,'D'表示删除,'U'表示更新。若记录成功追加到数据缓冲池,该操作才能在本地数据副本或当前 RMS 上执行,如果增加(删除)的记录属于西湖区,则直接在 RMS 中增加(删除)该记录,不属于西湖区,则在数据副本的相应分片上执行该操作。

(2)更新记录:在存储分片的记录中,字段值有字符串类型和数值类型,若更新的字段值为字符串类型,其在缓冲池中的存储形式和增加(删除)的操作是相似的,首先在有效字段部分将需要更新的字段进行修改,然后将附加字段中的操作标示改为'U',若更新的字段值为数值型,由于对数值型的操作都是加减操作,则该字段存储在原始字段值上改动的偏移量,例如将 MH 中采集点 ID 为"0051 – 6 – 1"的测点电压等级增加 35,写入缓冲池中的事务如下表所示:

表 2　更新数值型字段值的缓冲池事务

设备 ID	分片标示	操作标示	采集点 ID	中文描述	场站名称	电压等级	负荷类型
LI – CHENG – MOTOA 1200 – P	HZ – 1	U	0051 – 6 – 1	浙大玉泉校区体育馆	浙江大学	+35	地区负荷

(3)数据查询:由于只有一个分片的数据在 RMS 中,所以在查询过程中很可能查询的记录不在当前的 RMS 分片中,而基于 RMS 的查询效率比基于文件的查询效率要高,为了提高查询效率,采用位置相关的查询分片调度策略,一般情况下,若终端用户处在西湖区,则在查询的过程中,往往会针对西湖区的测点进行连续查询,针对这种情况,设置一个查询调度标示数 N,若查询次数超过 N,且在这 N 次的查询过程中,映射到某一分片的记录数大于 | N/2 | ,则认为当前用户在针对西湖区的数据进行连续查询,此时,若西湖区的数据分片不在 RMS 中,则调用西湖区的数据取代当前 RMS 中的数据;否则,直接执行基于数据副本文件的查询操作。

3.2 冲突检测与处理机制

为了提高移动事务的提交效率和提交成功率,采用事务结果集和语义[3]相结合的冲突检测机制来进行冲突检测和冲突处理。

(1)移动设备端:EMDB 在本地的数据同步时要进行本地的冲突检查,在将记录写入缓冲池时,通过"操作标示"和"采集点 ID"这两个字段可以检测是否对已经增加(删除)的记录进行重复增加(删除)。对于更新记录产生的冲突,主要体现在数值型的字段上,可以通过应用程序的操作语义来进行检测,将更新记录写入到缓冲池之前,使用记录主键(采集点 ID)来检查该条记录是否已经在缓冲池中,若在,则直接在这条事务上再次更新相应的字段,并通过全局限制来判断是否产生了冲突,例如:电压等级字段的全局限定条件为电压等级不小于零。假设对采集点 ID 为"0051 – 6 – 1"的记录有如下三次更新,电压等级字段对应的值分别为"+35","–100","–210",冲突检测条件为:原始值 + 更新值满足全局限定条件,在本例中,第一次更新 260 +35 +0 > =0 成立,不存在冲突,如表 2 的事务成功追加到缓冲池中,并更新本地的数据副本,第二次更新 260 +35 –100 > =0 成立,不存在冲突,更新缓冲池中对应的事务,将"电压等级"字段的值改为"–65",然后更新本地数据副本,第三次更新 260 –65 –210 < 0 不成立,说明存在冲突,此次对该记录的更新不成功,缓冲池中的对应事务不做更改,即"电压等级"字段的值仍为"–65",本地的数据副本也不做更改。

(2)服务器端:Sync Server 上请求缓冲队列中保存着各移动设备上载过来的事务结果集,Sync Server 依次取出这些结果集,根据事务中的"操作标示"字段来刷新本地的数据副本,同时使用和移动设备端相同的方式来进行冲突检查。

4 同步服务器算法

基于图 3 所示的同步模型中，移动设备和同步服务器之间的同步处理由两个独立执行的子过程，即上载子过程和下载子过程，它们分别在 Sync Server 上串行执行，在下载过程完成后，移动设备端再执行一次本地的同步过程。

定义 1（移动设备上传事务的成功写入集）：设为移动设备上缓冲池中的事务结果集，那么定义 SuccRecordSet( ) 为中成功同步到 Sync Server 中的事务集，即 Sync Server 在同步 SuccRecordSet(MH_i) 中的事务时没有发生冲突。

定义 2（移动设备同步下载事务集）：定义 downloadRecordSet(MH_i) 为移动设备的同步下载事务集，对于 downloadRecordSet(MH_i) 中的任意事务 T，满足。$T \in \sum_{i=1}^{n} SuccRecordSet(MH_i) \wedge T \notin SuccRecordSet(MHi)$。downloadRecordSet(NH_i) 中保存的事务集即为移动设备需要在本地进行同步的事务集。

算法 1：上载算法

```
CancelSet = ∅ ;/ * 初始化 * /
/ * 如果事务请求队列不为空，则依次处理队列中的事务 * /
If ( RequestQueue. Size 0 )
   For i = 1 to RequestQueue. Size
     / * 检测到冲突，增加（删除）的记录已经存在 * /
      If ( ( ( OperationSign ( Ti ) = = ‘A’ ) ∧ IsExist ( Ti ) )
∨ ( ( OperationSign ( Ti ) = = ‘D’ ) ∧ IsExist( ) ) )
        CancelSet = CancelSet ∪ { Ti }
     / * 如果是更新操作，通过全局限制来检测冲突 * /
      Else if ( OperationSign ( Ti ) = = ‘U’ ∧ IsExist ( Ti ) )
       / * 如果当前值 + 改变值（偏移量）> = 限制值，
说明不存在冲突 * /
        If( current_val + change_val > = restrict_val )
          Issue( Ti ) ; / * 处理该事务 * /
          SuccRecordSet( MHi ) ∪ { Ti }
        Else / * 否则数据更新过程中产生了冲突 * /
          CancelSet = CancelSet ∪ { Ti }
      Else
       / * 增加或删除操作过程中没有检测到冲突 * /
       Issue( Ti ) ; / * 处理该事务 * /
       / * 根据设备 ID 将该事务加入到相应移动设备 MHi 的成功同步集中 * /
        SuccRecordSet( MHi ) ∪ { Ti }
   Endfor
Else
/ * 如果事务请求队列为空，则什么也不做 * /
Donothing ;
```

在下载的过程中，依次生成各个移动设备需要下载的事务集 downloadRecordSet(MH_i)，由于 Sync Server 上成功执行的事务集都已刷新到本地的数据副本中，因此对于 downloadRecordSet (MH_i) 中的更新事务，其数值型的字段值只需保存 Sync Server 上对应记录的结果值即可，如表 2 的记录在经过 +35，−100 的成功操作后，其值为 195，则 downloadRecordSet(MH_i) 保存这条记录对应的值即为“195”，而不再是数值的偏移量“−65”，这样可以减少事务在移动设备端的计算量，提高了数据同步效率。

定义 3（Sync Server 成功同步事务结果集）：将 $T \in \sum_{i=1}^{n} SuccRecordSet(MH_i)$ 中成功同步到 Sync Server 中的事务结果集定义为 SyncSuccRecordSet，则对于 SyncSuccRecordSet 中的任意条记录 T，有 $T \in \sum_{i=1}^{n} SuccRecordSet(MH_i) \wedge T \notin SuccRecordSet$。

算法 2：下载算法

```
/ * 依次生成 n 个移动设备的下载事务集 downloadRecordSet ( MHi ) * /
For i = 1 to n
    downloadRecordSet( MHi ) = ∅ ; / * 初始化 * /
    If ( ∃ SuccRecordSet( MHi ) ) / * 如果 SuccRecordSet( MHi ) 存在 * /
      For j = 1 to T ∈ ( ∑ SuccRecordSet( MHi ) . Size )
        If ( ∃ Ti ( Ti ∈ ∑ SuccRecordSet( MHi ) ∧ Ti ∉ Suc-
```

cRecordSet(MH_i)))

/ * 将添加到同步下载事务集中 * /

downloadRecordSet(MH_i) $\cup \{T_i\}$

EndIf

EndFor

Else

/ * 如果 SuccRecordSet(MH_i) 不存在,说明移动设备没有成功同步到 Sync Server 上的记录,或移动设备 MH_i 在断网期间没有对本地的数据进行操作 * /

downloadRecordSet (MH_i) = $\sum_{i=1}^{n}$ SuccRecordSet (MH_i) ;

EndFor

由于各移动设备只保留整个数据副本的一个分片,对于移动终端 MH_i 有如下的本地数据刷新过程:

算法 3:EMDB 本地同步算法

/ * 如果 downloadRecordSet(MH_i) 不为空 * /

If (downloadRecordSet(MH_i) $\neq \varnothing$)

/ * downloadRecordSet( ) . Size 为 downloadRecordSet (MH_i) 的记录数 * /

For j = 1 to downloadRecordSet(MH_i) . Size

/ * 如果事务 T_i 在当前分片中 * /

If(IsCurrentPatch(T_i))

/ * 将 T_i 更新到当前的 RMS 中 * /

UpdateToRMS(T_i) ;

Else

/ * 将 T_i 更新到存储卡上数据副本的相应分片中 * /

UpdateToFile(T_i) ;

/ * 将 T_i 从 downloadRecordSet(MH_i) 中删除 * /

Delete(T_i) ;

EndFor

Else

/ * 如果 downloadRecordSet(MH_i) 为空,则什么也不做 * /

Donothing

5 结束语

本文在移动设备上应用程序大小和可用堆栈受限的前提下,在实际例子的基础上,提出基于 RMS 分片调度的移动数据库同步机制,找到在应用开发过程中实现数据库同步复制的一种有效方法。这种以 RMS 作为移动数据库、结合语义进行冲突检测和使用单事务结果集进行数据同步机制,有效地提高了事务的提交成功率,为基于受限的移动设备开发提供了一种可行的解决方案。

参考文献

1 丁治明,孟小峰,王珊. 复制的移动数据库系统事务级同步处理策略. 软件学报,2002,13 (2):258 – 265.

2 廖雪峰,廖济舟,顾庆军,杨仲伟,顾雨生,张永跃,鲁磊,詹建飞,夏彦端,澎湃,魏祖英,黄晔. J2ME 中文教程. 1.01 修正版,北京:清华大学出版社,2005:71 – 84.

3 祝庆,张倪. 基于语义的移动数据库同步服务器的设计. 计算机工程与设计,2005,26(1):212 – 214.

4 丁治明,孟小峰,王珊. 移动数据库系统乐观事物处理策略. 计算机研究与发展,2002,39(10):1379 – 1386.

5 李霖,周兴. 移动数据库系统的三级复制体系结构. 计算机学报,1997,20(增刊):213 – 222.

6 王文琴,费贤举,鞠时光. 基于数据复制技术实现移动数据同步. 计算机应用,2006, 7(26):1676 – 1678.

7 王珊,丁志明,张孝. 移动数据库及其应用. 计算机应用, 2000,20(9):1 – 5.

8 李霖,周兴铭. 非对称网络环境中数据广播的启发式多盘调度算法. 计算机学报,1999,22(1):45 – 50.