分布式混合数据库系统①

林金芳, 欧 锋

(江南计算技术研究所, 无锡 214083)

摘 要: 为了进一步提高混合数据库系统的实用性, 本文提出了一种分布式混合数据库系统. 首先, 散布服务器将关系数据库中的实体表复制到本地内存数据库中, 散布客户端从服务器内存中载入数据库的完整快照, 而不涉及到关系数据库. 在完成最初的数据初始化后, 散布客户端与散布服务器通过散布机制持续保持同步及异常恢复. 本文还给出了数据分类和数据存储设计, 重点强调了散布机制设计过程, 它主要涉及散布模块定义、散布客户端 Poll 模式、Delta ID 的用法、散布复原协议、散布客户端的 DB 记录同步等.

关键词: 混合数据库; 内存数据库; 散布机制; 散布复原; 数据库同步

Distributed Hybrid Database System

LIN Jin-Fang, OU Feng

(Jiangnan Institute of Computing Technology, Wuxi 214083, China)

Abstract: In this paper, we propose a distributed hybrid database system. Firstly, the dissemination server copies entity tables in relational database to the local memory database and the dissemination clients load complete snapshot from server's memory. Then, the client and server will keep synchronization through dissemination mechanism after initialization. In this paper, we present the design of data classification and data storage. Especially, we introduce the design porocess of dissemination mechanism, mainly involving dissemination module, polling mode, Delta id usage, dissemination recovery protocol, DB record synchronization.

Key words: hybrid database; in-memory database; dissemination mechanism; dissemination recovery; database synchronization

1 概述

混合数据库系统(Hybrid Database)[1]是将数据存于不同的存储介质并加以管理的新型数据库管理系统, 即同时支持内存数据库(IMDB: In–Memory Database)[2]和磁盘数据库(DRDB: Disk–Resident Database)[3]技术, 从而获得极高的存取速度、极强的并发访问能力、以及大容量存储访问的功能. 近期国内外学者在混合数据库的研究中, 提出了很多有用的设计.

周鹏[4]等人提出了一种带快照的混合数据库系统设计应用, 它采用分存储层次混合存储数据的方式, 除了和基于磁盘的数据库系统一样将全部数据存储在磁盘上以外, 又将部分被访问频率较高或者需要较快响应的数据以快照的形式组织在内存中. 但是该方法有以下几个局限性: 一是缺乏对磁盘数据库中海量数据存储的方案; 二是快照的刷新问题, 当基于多张表的快照需要更新时, 需要重新创建快照, 并且缺乏一个机制通知何时进行快照刷新; 三是快照进内存中数据只能提供查询功能, 并不能进行插入、删除、更新等常规操作, 缺乏将内存数据库中的数据变化保存进磁盘数据库的方法. 因此该方案实际应用范围是较窄, 很难满足当前绝大多数信息系统对数据库处理能力的要求.

韩国 Altibase 公司推出的一款混合型数据库管理系统 ALTIBASE 数据库[5], 它结合了磁盘数据库和内

① 收稿时间:2015-01-19;收到修改稿时间:2015-04-02

存数据库的特性, 支持内存数据库专用模式、磁盘数据库专用模式和混合模式, 为需要大量、高速处理的用户提供了有效的解决方案. 然而当数据量超出内存大小时, ALTIBASE 数据库数据性能将大大减低, 为了使系统不中断运行只能使用硬盘 swapping 技术[6]临时解决, 因为 ALTIBASE 缺乏一种转化机制将内存中的历史数据腾出进磁盘空间.

为此, 本文提出了一种分布式混合数据库系统. 它将数据类型分为动态数据和静态数据, 采用三层存储模式, 除了将动态数据和静态数据存储进磁盘数据库外, 又将动态数据存储进内存数据库中, 而采用散布机制保持数据的实时同步和异常恢复.

本文随后部分组织如下, 第 2 节介绍了分布式混合数据库系统的总体设计, 第 3 节介绍数据存储方式, 第 4 节是散布机制设计, 最后是小结.

2 总体设计

2.1 数据类型

我们将数据分为两类, 静态数据和动态数据.

静态数据具备以下特点:

1) 变动较不频繁(甚至不变动)的数据并且读取也要花较长的时间.

2) 随着时间累积的历史数据.

动态数据具有以下特点:

1) 随着需求次数需要经常变化的数据.

2) 动态数据的存储大小要明显小于静态数据的存储大小.

2.2 数据存储形态

数据分三种形态分布存储:

1) 散布服务器: 散布服务器拥有一个轻巧的内存数据库, 该内存数据库支持标准结构查询语言(SQL)语法, 服务器程序负责执行数据的插入、删除、更新等操作.

2) 散布应用客户服务器: 内存数据库同时也存在于散布应用客户服务器, 该数据库通过 JDBC 对外提供一个标准的 SQL–99 接口, 应用程序可以通过该接口对内存数据库中的数据进行查询、插入、删除、更新等操作. 并且散步应用客户服务器可以多个同时存在. 为了避免与散步服务器的概念混淆, 后续统称为散步客户端.

3) 磁盘数据库系统: Oracle 数据库用于实现数据的永久性存储, 包括 Oracle 关系数据库和 Oracle 数据仓库两部分, 关系数据库存储动态数据, 而数据仓库存储静态数据.

图 1 简略给出了分布式混合数据库系统的设计及主要程序组件的构成.

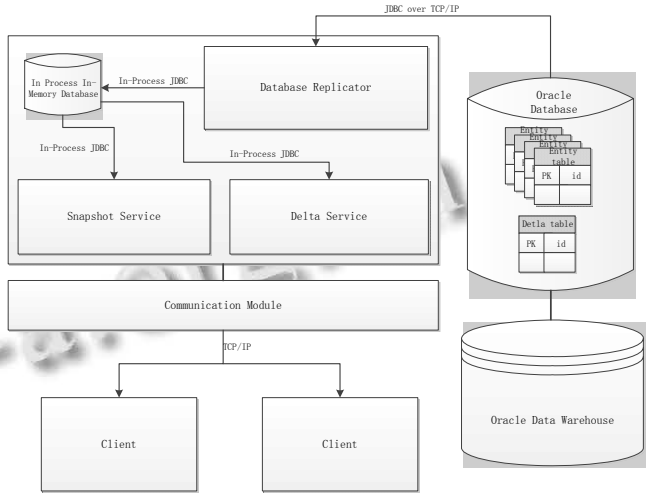

图 1 分布式混合数据库系统设计结构图

2.3 设计思路

1) 当散布服务器启动后, 服务器复制程序会将关系数据库中的数据表内容复制至本地的内存数据库中. 一旦散布客户端启动, 它通过快照隔离服务程序直接从服务器的内存中载入数据库的完整快照, 而不涉及关系数据库. 而静态数据并不会加载至服务端内存数据库, 也不会散布至客户端, 所有客户端请求的静态数据都直接从数据仓库中获取.

2) 在系统初始化完成后, 散布客户端与散布服务器通过散布机制持续保持同步. 当散布客户端数据发生变更(指发生插入、删除、更新等操作)时, 不直接修改本地内存数据库, 而是先将变更内容发送到散布服务器, 服务器程序修改关系数据库中相应的数据表项, 经修改后的变更将立即反映至服务器的内存数据库中. 当散布服务器更新成功后, 通过散布机制, 统一更新所有散布客户端, 整个数据变更过程如图 2 所示.

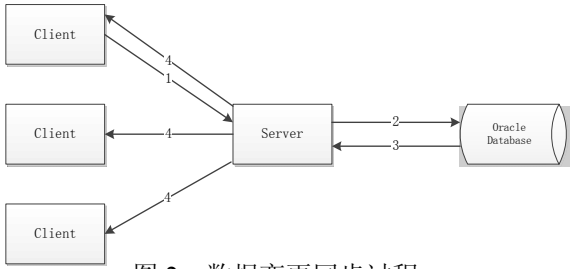

图 2 数据变更同步过程

3) 在散布服务器发生异常崩溃重启后, 散布服务端通过散布复原协议将散布客户端带到一个与服务器同步的状态点, 然后在此状态点后重新进入过程 2.

3 数据存储

3.1 关系数据库

关系数据库内容被分成以下两个逻辑部分:

1) 实体表: 这些表包含了信息系统正常业务逻辑所需的所有相关信息. 即为信息系统所需的所有动态数据设计相应的数据表, 用于动态数据的存储.

2) 散布相关表: 这些表被散布机制所使用, 用于跟踪数据库变更, 并决定散布的相关动作. 在这类表中最重要的一个表是 Delta table, 如表 1 所示. Delta table 用于累积实体表的变更记录, 并在稍后为散布机制将变更散布到相应节点提供依据. 其中 ID 是该表的索引, 是最重要的一个字段; ENTITY_ID 是实体表标识; ENTITY_TABLE 是实体表名称; OPERATION 表示变更操作, 只有三个值 INSERT、UPDATE、DELETE; COLUMN_VALUES 表示实体表具体的变更记录, 详细记录实体表中哪些字段发生了变化以及相应的变化值; 其他三个字段表示主体, 变更时间以及标识发起变更的客户端.

表 1 Delta table 表结构

Name	Type
ID	NUMBER(22)
ENTITY_ID	VARCHAR2(2000)
ENTITY_TABLE	VARCHAR2(30)
OPERATION	VARCHAR2(10)
COLUMN_VALUES	CLOB
TOPICS	VARCHAR2(100)
CREATION_TIME	TIMESTAMP(6)
CLIENT_ID	VARCHAR2(100)

3.2 数据仓库

数据仓库的采用是为了避免 Oracle 数据库实体表的无限制增大而导致加载进内存数据库的体积过大, 从而影响服务器的响应性能. 因此当 Oracle 关系数据库中部分动态数据随着时间的推移, 访问频率逐渐下降以及存储体积的累积增加后, 该部分数据则被转移进 Oracle 数据仓库, 转变成静态数据, 为信息系统提供决策数据支持. 散步客户端通过 ETL 直接从数据仓库中获取.

3.3 内存数据库

内存数据库同时存在于散布服务器和散布客户端中. 该数据库与关系数据库具有相同的 JDBC API, 存储与信息系统紧密相关的动态数据.

1) 散布服务器: 散布服务器使用内存数据库是为了减少加载关系数据库的次数, 降低关系数据库的负载. 当服务器启动时, 散布服务器复制一组数据库表到内存数据库中, 通常都包含实体表和散布相关表.

为了保持内存中的数据库表是最新的, 散布服务器以一定的时间间隔规律地请求关系数据库中的 Delta table, 如果有新的变更, 则将相应的变更插入到内存数据库的 Delta table 中, 并更新内存数据库中相关的数据库表内容.

2) 散布客户端: 散布客户端首先快照请求获取服务器内存数据库的快照镜像, 然后依据 Delta table 从服务器内存数据库中获取相应的数据变更. 该内存数据库在稍后被用于本地信息系统的查询、插入、修改、删除等操作, 而不用客户端直接访问服务器.

4 散步机制

如何保证分布式数据库系统数据的一致性是数据库同步领域的热门问题[7]. 近年国内外学者在数据库同步技术的研究中, 快照隔离[8]的同步方式成为当前研究的主流和热点.

Lin 等人提出了一种“1-copy-si”的数据库同步模型[9], 在每个数据副本提供快照隔离的同时达到全局的快照隔离, 即所有的快照从全局看效果等同于在一个副本中执行.

Elnikety 等人提出了一种 GSI 算法[10], 将单个副本的快照隔离扩展到多副本的数据库同步系统中, 同时保证了快照隔离的特性.

王珏等人结合快照隔离的性能优势和组通信技术的消息定序特性, 提出了一种满足单副本可串行化的数据同步协议[11].

虽然上述基于快照隔离的数据库同步方案通过快照隔离级别扩展到全局、单副本可串行化等方法达到了数据的全局同步, 然后该类方案每次数据同步都需要重建快照, 对系统性能要求较高、开销较大、效率较低. 由于本文设计的分布式混合数据库系统对数据的实时性和并发性有很高的要求, 完全采用基于快照隔离的数据库同步方案并不完全合适. 因此本文提出

了散布机制用于实现分布式混合数据库系统的数据同步, 它每次只需要将数据变更内容散布至各个节点, 而无需重建整个快照.

4.1 散步模块

散布机制是通过一组散布模块来实现的. 散布模块是一个独立的模块, 它通过检测关系数据库的变更, 并将这些变更散布至散布客户端, 从而达到数据同步. 散布模块由以下子模块组成:

1) 关系数据库触发器: 用于检测实体表的变更并记录变更. 每个实体表上都设计有一个数据库触发器, 当一个实体表被修改相应的触发器都将被激活, 并在 Delta table 中增加对应的变更日志.

2) 服务器散布模块: 一个运行在散布服务器的独立模块, 该模块预先读取关系数据库中的 Delta table, 将写入到 Delta table 中的变更记录读入到服务器内存数据库中.

3 客户端散布模块: 一个运行在散布客户端的独立模块, 该模块利用 Poll 模式主动连接到散布服务器, 以接受被读入到服务器内存中的数据变更.

4.2 散步机制阶段

散布机制可以分为变更复制和散布复原两个阶段:

1) 变更复制: 一旦系统初始化完成或者散布复原结束, 变更复制就启动了. 在变更复制过程中, 客户端散布模块依据 Delta table 获取散布服务器中的数据变更.

2) 散布复原: 当服务器发生故障或异常崩溃重启以后, 散布客户端使用散布复原协议同步其与散布服务器的状态.

4.3 变更复制

4.3.1 Poll 模式

客户端散布模块通过 HTTP 协议在一定的时间间隔(根据需要由外部参数设定)连接散布服务器, 这种主动连接的模式称之为 Poll 模式, 采用 HTTP 协议是由于 HTTP 是一种无状态协议, 每个散布客户端连接散布服务器, 询问数据是否有相应的变更, 然后断开连接. 数据变更散布过程通常是由客户端散布模块发起的, 而散布服务器无需保存任何信息去识别请求是由哪个客户端发起的.

客户端散布模块采用 Poll 模式主要基于以下理由:

1) 由于服务器端不需要保持客户端的任何特定信息而使服务器的实现变得简单.

2) 由于采用无状态连接, 使得服务器可以承受更多客户端的同时连接.

3) 保持一定的时间间隔访问频率使得服务器有足够的时间处理每个客户端的请求.

4.3.2 变更的产生和删除

任何一个散布客户端都可以产生数据变更, 但是所有的数据变更都将先发送至关系数据库. 数据库触发器启动在每一个实体表之上, 任何一个实体表发生变化, 都将在 Delta table 中记录相应的变更并产生唯一的标识符号 ID, ID 是 Delta table 表中最重要的一个字段, 该字段是一个自增长正整数, 每次都增加 1. 因此, 依据它们的标识符号 ID, 所有的数据变更都可以进行有序排列, 稍后的变更会比之前变更具有更大的 ID 值.

散布模块同时也包含一个防止 Delta table 无限制增长的机制, 以保证 Delta table 的体积在一个合理的范围之内. 该机制保持 Delta table 的 ID 值不会比预定值要大, 当超出这个值时, 删除最旧的记录来保持 Delta table 容量的上限, 而这些最旧的记录已经在更早的散布过程中被同步到了散布服务器和散布客户端中, 这些记录是可以被删除的. 预定值依据散布服务器的性能可以通过外部参数进行设定.

4.3.3 Delta ID 的用法

变更复制是基于 Delta table 的 Delta ID 进行的. 当执行了最初的快照后, 客户端散布模块就开始规律地 Poll 散布服务器. 在每次的 Poll 请求中, 客户端散布模块发送它本地内存数据库 Delta table 中最大的 Delta ID 值, 该 ID 表示散布客户端的当前数据状态点, 直到其被更新成更大的 ID 值. 散布服务器检查它当前的 ID 值是否要比请求的散布客户端的 ID 值要高, 如果为真, 在 HTTP 响应中发送相应的数据变更记录和当前最大 ID 值到散布客户端. 而除了 Delta ID 值, 客户端散布模块发送的请求中无需包含任何其他信息, 这大大简化了散布机制, 减轻了散布服务器的负担.

4.4 散步复原协议

4.4.1 Delta ID 的用法

散布复原协议用于当散布服务器发生故障后, 散布客户端如何实现其与散布服务器的同步. 该协议的目标是将散布客户端带到一个与散布服务器同步的同步点, 在该同步点, 散布客户端与散布服务器拥有最

后相同的 Delta ID,并且后续的散布变更复制可以被执行.

散布复原协议的算法实现可以用图 3 的流程图来表示.

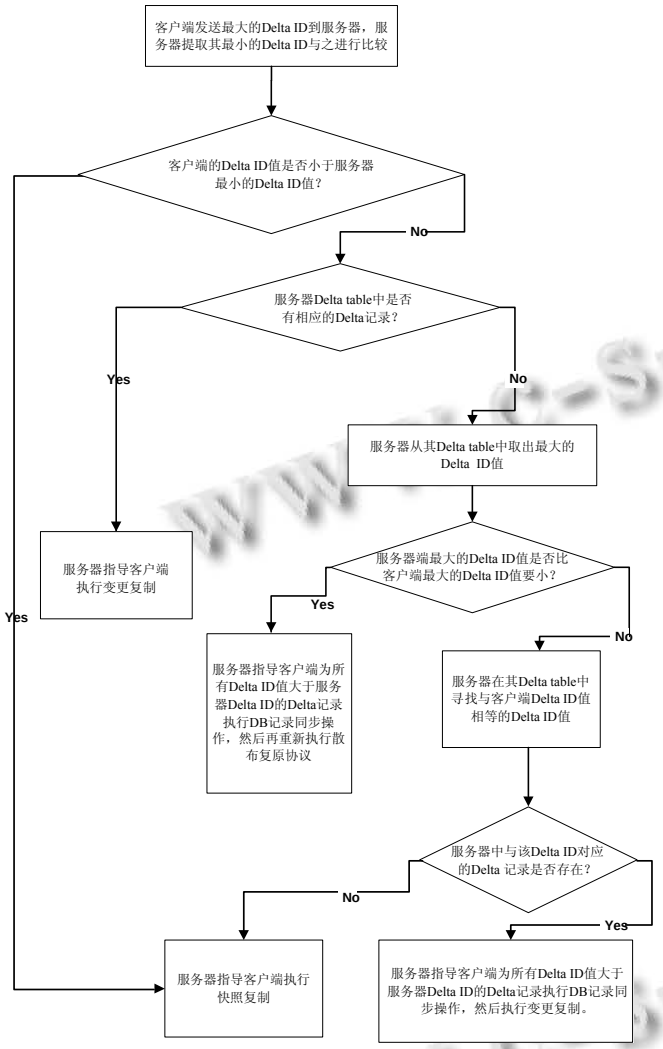

图 3　散布复原协议算法

4.4.2 散布客户端的 DB 记录同步

DB 记录同步是散布复原协议的重要组成部分, 它用于实现散布客户端与散布服务器的数据记录同步. DB 记录同步被用于当最后一次同步点后, 服务器发生异常或崩溃, 而客户端却发生了新的变更的情况. 任何从该同步点之后被应用于散布客户端的数据变更都将被放弃, 散布服务器依据同步点去指导客户端去执行一个 DB 记录同步操作, 具体过程如下:

客户端从本地内存数据库 Delta table 中取出所有的 Delta ID 大于给定 Delta ID 的 Delta 记录, 初始化一个 Delta 列表, 同时也初始化一个与 Delta 对应的实体表 DB 记录 ids 列表, 以便确定如何与服务器达到同步状态. 对于不同的 Delta, 客户端检查的 Delta table 的 Operation 字段的操作记录(INSERT, DELETE 或 UPDATE).

1) INSERT: 客户端从本地内存数据库中删除对应的实体表插入记录. 因为通过这种 Delta 创建的插入记录在散布服务器数据库中不存在, 因此, 客户端不需要询问服务器情况, 直接从本地内存数据库中删除.

同时由于该条 DB 记录已经从本地数据库中删除, 其他针对该条记录的 Delta 记录(UPDATE 和 DELETE)也将被删除.

2) UPDATE: 对于在第一步中没有过滤的 UPDATE Delta 记录, 客户端从本地数据库删除相应的记录, 并在 DB 记录 ids 列表中标记相应记录 ID, 并从散布服务器还原 UPDATE 前的记录.

3) DELETE: 对于在第一步中没有过滤的 DELETE Delta 记录, 散布客户端登记 ENTITY_ID 值, 并从散布服务器还原 DELETE 前的记录.

5　结论及后续研究

本文介绍了一种分布式混合数据库系统. 介绍了它的基本结构和数据同步办法, 与前人的研究相比, 本设计主要解决了以下几个问题:

1) 采用 Oracle 数据库和数据仓库相结合的方式实现了一种磁盘海量数据存储方案, 动态数据能够向静态数据转换的特点, 保证了 Oracle 数据库中实体表大小的合理性, 以及海量历史数据的长期保存.

2) 散布机制使用最小的开销保证了关系数据库、散布服务器、散布客户端三者数据的同步状态, 而快照隔离方法仅仅被用于系统初始化和异常恢复过程中.

3) 散布机制保证了内存数据库中所有的数据变更, 如插入、删除、更新等操作都能与关系数据库保持一致, 即所有的变更都能得到永久性保存. 能够满足信息系统对内存数据库的操作要求.

4) 散布服务器的采用减少了多个客户端直接访问关系数据库的频率, 提高了分布式混合数据库系统的效率, 而散布机制保证了所有散布客户端的数据变更是高度实时并发的.

分布式混合数据库系统还需进一步完善, 其后续

研究工作将主要集中在如何实现多个散步服务器的同步及协同工作, 从而达到提高系统稳定性的目的.

参考文献

1 Wang JL, Ding GR, et al. Spatial-temporal spectrum hole discovery: A hybrid spectrum sensing and geolocation database framework. Science China Press and Springer-Verlag Berlin Heidelberg. 2014, 59(16): 1896–1902.

2 张延松,王占伟,孙妍等.内存数据库可控的 page-color 优化技术研究.计算机研究与发展,2011,48(Suppl.):95–104.

3 丁治明,高需.面向物联网海量传感器采样数据管理的数据库集群系统框架.计算机学报,2012,35(6):1175–1191.

4 周鹏,杨丹,鱼佯训.带快照的混合数据库系统设计应用.计算机科学,2008,35(4):241–243.

5 AltiBase.Hybrid DBMS–ALTIBASE 数据库技术白皮书. http://www.generaldata.com.cn. 2007-01-12.

6 Zhou MQ, Xu C. Optimized data placement for column-oriented data store in the distributed environment. Proc. of Database Systems for Advanced Applications. Berlin. Springer. 2011.

7 Krzysztof K. Experimental B+_tree for GPU. Proc. II of the 15th East European Conference on Advances in Databases and Information Systems (ADBIS 2011). Vienna. Springer. 2011. 232–241.

8 Bereson H, Bernstein P, Gray J, et al. A critique of ANSI SQL isolation levels. Proc. of ACM SIGMOD. New York. ACM Press. 1995. 1–10.

9 Lin Y, Kemme B, Patino-Martinez M, et a1. Middleware based data replication providing snapshot isolation. Proc. of SIGMOD. New York. ACM Press. 2005. 419–430.

10 Elnikety S, Zwaenepoel W, Pedone F. Database replication using generalized snapshot isolation. Proc. of the 24th IEEE SRDS. Washington DC. IEEE Computer Society. 2005. 73–84.

11 王珏,李立新,张绍月等.基于快照隔离的分布式系统同步协议研究与实现.计算机应用研究,2012,29(8):3012 –3017.