浙江工业大学

本科毕业设计文献综述

（2013 届）

论文题目　基于内存数据库的大数据应用系统的设计与实现

作者姓名　陈佳鹏

指导教师　陈　波

学科 (专业)　软件工程

所在学院　计算机科学与技术学院

提交日期　2013 年 03 月

基于内存数据库的大数据应用系统的设计与实现

摘要： 本文是基于内存数据库的大数据应用系统的设计与实现的一篇文献综述，先介绍项目的由来及其研究意思，然后介绍项目的国内外研究现状及难点以定位项目开发的一个大环境，明确当前同类项目的研究情况。接着本文简述内存数据库系统的结构，紧接着介绍系统开发中所需的关键技术。

关键字： 内存数据库，索引结构，并发控制，T 树，数据恢复，影子内存，混合日志

一、引言

在信息化建设中用于对数据进行维护和管理的数据库扮演了一个十分重要的角色，一个合适的数据库系统往往是信息化改造成功与否的关键所在。传统的磁盘数据库（Disk Resident Database，DRDB）由于经过几十年的发展其功能完备，稳定性好，因而经常应用于电信，银行等公司进行客户资料管理，认识部门档案管理等日常应用方面，并且一直以来有着令人满意的表现。然而伴随着科技的发展，涌现出一批新兴行业或传统行业需要新的应用，例如工业控制，数据通信，证券交易，电力调度，航空航天等。这些新的产业所要求的数据库系统通常并不需要数据库具有强大而完备的功能和进行非常复杂的事物处理的能力，却需要数据库能在指定的时刻或时间内对大量的数据进行采集，处理并能正确的响应的高速的性能。传统的数据库不能很好地支持实时性，无法满足工业控制实时性要求。主要原因是传统数据库事务涉及输入输出（I/O）操作、缓冲区管理和页违例等事件存在执行时间的不可预知性和弱实时性，使得事务的实时性和预测性较差[17]。由于“I/O 瓶颈”问题，基于磁盘的数据库系统（Oracle，SQL Server 等）不能满足现代应用对数据库的实耐性处理的要求。内存数据与磁盘数据在访问时间上相差好几个数量级，内存的速度具有嚷显的优势。内存容量增大，而价格却在不断地下降。从 20 世纪 80 年代开始，数据库研究人员考虑把整个或者大部分数据库放在内存中。内存数据库（MMDB）是实时数据库研究的基础，井成为了研究的热点。

MMDB 与 DRDB 最主要的区别就是数据主版本的驻留位置不同，前者驻留在内存，后者驻留在磁盘。由于内存与磁盘在访问速度、易失性、访问模式等方面存在很大的差异，内存数据库与磁盘存数据库中数据是常驻内存，处理前不需要从磁盘读取数据，数据库也存在差别 [16]。如表 1.1 所示。

二、研究意义

内存数据库因为其快速的数据访问能力，使其比磁盘数据库（DRDB）更适合于需要快速响应和高事务吞吐量的应用环境。对于那些需要在严格要求的时间段内完成事务请求的实时应用系统，对于需要支持大数据量并发访问的高性能事

表 1.1: MMDB 和 DRDB 的比较

性能	MMDB	DRDB
数据存储	行、列级储存模型以及段 -分区式储存模型，不要求模型在内存中连续存放	Sybase IQ 采用列级储存，其他数据库系统采用行级存储，在磁盘上连续存放
缓冲管理	不需要	需要
并发控制	采用封锁机制、多版本等方式，一般采用较大粒度的锁，如库级锁、表级锁或采用乐观封锁机制	采用封锁机制、时间戳、多版本等方式，为了提高事务的并发度，一般支持多粒度和多种类型的锁
恢复机制	备份、日志和检查点技术，采用预提交、组提交等提交方式；用稳定内存来存储日志记录	采用备份、日志、检查点、保存点等技术
索引结构	T 树索引、hash 索引	B 树索引、hash 索引
查询优化	基于 CPU 代价以及 cache 代价	基于 I/O 代价

务处理平台来讲，内存数据库都是一个理想的选择。此外，在实际生产中，常常出现不能互相访问其内置实时数据库的信息，从而使大量信息冗余重复存在于各系统中，也就是出现数据孤岛。为了解决这个问题，必须实对时数据库的数据管理进行合理规划以建立开放的实时数据库系统，使之能够提供高速、及时的实时数据服务。

三、研究现状

内存数据库的研究始于 20 世纪 80 年代，并逐渐吸引了越来越多的研究者的研兴趣。经过 20 多年的发展，时至今日研究者们已对它的体系结构，数据组织与存取方法，事务处理，并发控制和恢复备份技术进行了大量的探讨和研究，针对硬件，软件和算法设计提出了许多不同的策略和实现方案，取得了丰硕的成果。目前内存数据库的研究主要集中在一下几个方向：

1. 内存数据库的体系结构，内存数据库的体系结构包括系统的习题结构和存储体系结构两部分内容，系统体系结构方面主要侧重于研究多处理器在 MMDB 中的应用，而存储体系结构方面则侧重于研究非易失性内存（NV-RAM，UPS RAM 等）在 MMDB 中的应用。

2. 事务处理，事务的处理主要对事务的提交及与之相联的日志记录、查询优化，尤其是联机查询的优化进行了多方面的研究，开发了一些有效的技术。如开发了“提前提交”等策略以优化事务的初始如对事务的初始处理、提交处理、并发控制、完整性和安全性检验等，加速并发事务的响应时间，查询的优化主要在于对减少查询中中间关系的算法的研究。

3. 数据组织与存取方法，这方面主要针对 MMDB 存储介质的特性设计了许多适合内存特性的数据存储组织结构和存取方式，索引结构和存储策略，MMDB 的压缩，如 AVL 树索引结构，区段式组织结构，位图分配法等，Hash 结构 [19]。

4. MMDB 的备份和恢复，备份方面更是提出了多种方案，为了预防 MMDB 的崩溃主要是结合检查点和日志来保证 MMDB（MMDB）在崩溃后的可恢复性，提出了许多具体的算法如 FUZZY（模糊）检验点策略，BLACK/WHITE（黑/白）策略，COPY-ON-UPDATE（变更拷贝）检验点等，同时研究了非易失性内存在 MMDB 备份中的应用，如“影子内存”技术等。而备份恢复方面则是对 MMDB 重启之后的装入策略的研究，结合内存数据中数据使用的频率和优先级提出一种最优的装入策略，目前提出了“部分重装”等策略，MMDB 的目录恢复后就启动系统，然后根据要求再继续重装，从而提高 CPU 的使用率和事务的吞吐量。

5. MMDB 的并发控制，在短小事务时，往往采用大力度的锁，MMDB 近似于串行处理，串行化执行事务一方面使得并发控制的代价几乎完全消除，同时也能减少 CPU 的 cache 缓存和虚拟内存页表 TLB 的刷新频率；而对于长事务和多处理机环境串行处理明显不合适，这样就必需应用而合适的锁机制来支持并发操作，目前提出了二级层次封锁协议方案、乐观并发控制方法、使用可扩展的哈希技术的方法等。

近些年来随着大容量廉价的内存投入市场，使得上诉的各种技术也逐渐在内存数据库的设计和实现上得到了应用，内存数据库也不在停留在理论研究阶段，内存数据库走向了实际应用阶段，各高校和研究机构发布了研究模型系统，一些公司也推出了用于工业的商用内存数据库系统。目前工业应用上比较流行的商用系统有 Amazon 的 Dynamo[1]，SAP 的 Hana[8]，Oracle 公司的内存数据库 Berkeley DB，内存数据库 SQLite，开源的内存数据库 FastDB，以及 McObject 公司的 eXtremDB。他们的特点如表 3.1 所示：

表 3.1: 主流内存数据库

内存数据库名	厂商	特点
Berkeley DB	Oracle	Berkeley DB 数据库系统简单、小巧、可靠、高性能，提供了一系列应用程序接口（API），应用程序和 Berkeley DB 所提供的库在一起编译成为可执行程序。每一个记录由关键字和数据（KEY/VALUE）组成的键值对构成
SQLite	开源	SQLite 是一款轻型的数据库，它的设计目标是针对嵌入式系统，提供了很多语言的接口，支持 SQL 语句，支持事务处理，它占用资源非常的低，支持跨平台操作，操作使用简单。每完成一次操作需要进行内存和磁盘之间的同步
ExtremeDB	McObject	ExtremDB 是为实时系统及嵌入式系统而特别设计的，完全工作在主内存中，不基于文件系统，支持事务，支持部分 SQL，支持稳定的 RAM。通过数据库定义语言为应用系统各自的 API。具有工业应用强度
FastDB	开源	FastDB 是一个高效率的内存数据库系统，支持事务、在线备份和系统崩溃之后的自动恢复，支持类似 SQL 语言并提供了 C ++ 接口，算法和结构的优化都是基于数据存放在内存中这个假设上，但物理内存较小时也可使用

四、系统实现技术方法研究

4.1 内存数据的组织结构

在 MMDB 系统设计时，广泛使用了现代操作系统提供的共享内存机制，系统初始化时将整个数据库装入一片共享内存区，运行时，应用进程可以把整个数据库或一部分映射到自己的虚地址空间进行直接访问。针对关系和索引数据全在内存中这一特点，指针在数据结构和数据访问中被广泛的使用。应用进程可以通过指针，也可以通过位置独立的数据库偏移量访问数据，无需像 DRDB 那样与缓冲区管理器交互。另外，由于指针长度固定，因此变长字段问题可以很好解决。其次，若一个大的数据对象在数据库中多次出现，则内存中只需存储一次，其它地方使用指针来引用。

MMDB 中关系的存储通常采用分级结构，同时强调元数据与数据应分区组织，以提高安全性。在 Starburst[16] 中，存储结构分两层：段（Segment）和分区（Partition）。段是可变长的，一个段由多个固定长的分区组成，每个段存放一个关系，但并不要求构成一个段的分区空间上连续。分区则是 Starburst 的基本内存分配回收单元，其长度固定。分区的结构由两部分组成，其中 Free space 存放记录插槽（Record dot），Heap space 存放分区控制信息和记录（即元组）本身。整个数据库维护一个段表，段表中的每一项是一个段控制块。段控制块中包含一个分区表、段 latch、和索引链表的头指针。图 4.1 描述了 Starburst 的内存数据组织结构图。

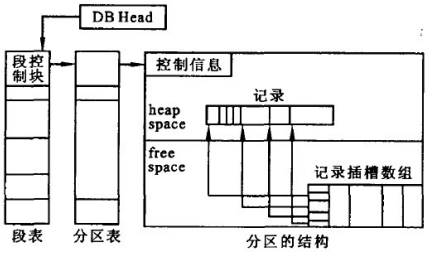

图 4.1: Starburst 内部结构

从图 2 中可以看到：一条记录本身放在分区的 Heap 区，对记录的访问是通过放在 Free 区的记录插槽进行的。记录插槽中包含一个指向相应记录各字段在 Heap 中地址的指针数组，由于指针的长度都一样，因此记录中不等长字段问题可以通过包含字段指针数组的等长记录插槽解决；同时，在关系创建后，若增加关系的字段，则可以通过在记录插槽预留的 Tail 结构进行扩展，所需空间在分区的 Heap 中分配。当一个事务要访问记录时，先根据关键字查索引，找到该记录的 RID。一个 RID 是一个由段号、分区号、和记录插槽在分区中的偏移量组成的三元组，用 RID 可以很快找到记录插槽，从而最终定位一条记录。

4.2 索引技术

4.2.1 T 树

DRDB 采用的索引结构主要是 B/B+ 树，其设计目标是减少访问磁盘数据的 IO 次数。而在 MMDB 中，通常采用了一种新的索引结构 T 树，其设计目标是减少内存开销和 CPU 指令数。T 树 [16] 是由 AVL 树和 B 树发展而来，它是一种一个节点包含多个元素的二叉树，如图 4.2 所示。

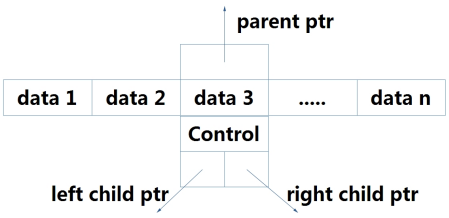

图 4.2: T 树节点

由于是二叉树，T 树保持了 AVL 树二分查找的高效率，同时一个 T 节点包含多个元素，像 B 树一样，每个节点的充满程度保持在半满和全满之间，这样索引一般不再需要溢出块，由插入和删除所引起的数据移动通常只需在一个节点内进行，减少了为保持树的平衡所必须进行的旋转操作，因此 T 树又保持了 B 树优异的更新和存储特性。由于索引和数据全在内存中，在一个 T 树节点中不需要像 B 树那样存放 N 个索引键值一指针对，只需存放指向内存中相应记录对应字段的指针，这样索引中变长字段的存储不再是问题，另外，由于指针一般比它指向的字段要小，大量的空间也被节省了。

4.2.2 Hashing

为了快速地定位数据库的记录，内存数据库中广泛使用的哈希索引有链接桶哈希（chained bucket hashing）、可扩展哈希（extendible hashing）、线性哈希（linear hashing）、修正的线性哈希（modified linear hashing），其中链接桶的哈希使用静态结构处理冲突，速度很快，但不适合动态环境。基于等值的比较，哈希技术能够快速地访问数据库，且易于实现，但不支持范围检索。

4.3 并发控制

内存数据库中使用的并发控制与磁盘数据库中的并发控制基本相同，细节上存在一定差异。Starburst 系统 [16] 使用封锁机制来保证事务的 ACID 特性。Starburst 系统中有两个粒度的锁：表级锁和记录级锁。其封锁机制具有以下两个方面的特点：1）锁信息直接与表和记录本身关联，不需要哈希表来存放锁信息，可消除通过哈希表查找定位锁信息的开销；2）使用动态锁机制来维护表的锁粒度级别。Starburst 根据系统对关系的共享程度的需求，动态地改变每个关系的锁粒度。因为表级锁比记录级锁开销小，因此当不需要较细的锁粒度时推荐表级锁。

如果当一个或者多个事务在访问表时被其他事务阻塞，表级锁被分解成为记录级锁。当不再需要细粒度锁时，记录级锁被转化为表级锁。

对于传统内存数据库，数据存储在内存中，事务执行时间较短，持锁时间也较短，封锁产生的 CPU 代价会对性能产生严重的影响，系统中冲突较少，所以可以采用以下方法减少锁的开销：

1）采用较大的锁粒度（如表级锁），因为数据常驻内存后，对数据的竞争已很低，细粒度锁的优点对改善性能的意义已不大

2）采用乐观加锁方式

3）减少锁的类型

4）将锁信息存储在数据本身

4.4 恢复机制

将日志写在何处以及何时将日志写人磁盘在内存数据库中是一个非常重要的问题。一些研究者提出了预提交、组提交等方法来降低日志 I/O 的代价，并且提出使用稳定内存存储日志的方法：首先将日志存储在稳定内存中，然后提交事务，再异步地把日志写入磁盘。在 MMDB 中，只有在事务提交写日志、执行 Checkpoint、以及系统故障恢复时才需要访问磁盘，基于以上分析，重新设计 MMDB 的事务提交策略和 Checkpoint 方式，对改善系统性能至关重要。

4.4.1 事务提交处理

为保证事务的 ACID 特性，事务提交时必须强制写日志到磁盘，因此写日志成为系统的瓶颈。如何提高 MMDB 中事务提交的速度，常用以下方法：

1）使用稳定内存

2）组提交

3）只记录 ReDo 日志

4.4.2 Checkpoints

在 MMDB 中，Checkpoint 负责将内存数据库映像存储到磁盘，并截短日志。每次 Checkpoint 执行时，检查自上一次 checkpoint 以后，内存数据库发生更新的内存页，并将更新保存到 checkpoint 文件中，然后删除不再需要的日志文件内容。在系统恢复时，联合使用 Checkpoint 文件和日志文件可以加快恢复速度。为减少对正常事务的干扰。在 MMDB 中广泛采用了 fuzzy checkpoint 技术。

fuzzy checkpoin 是一种非阻塞方式，既在任何时间点都可以进行 checkpoint，而不用考虑事务或动作的状态是否一致，但系统恢复时，checkpoint 必须与日志文件配合完成工作。

为了保证恢复时数据的一致性，采用 Ping-Pong 方式：在磁盘上保存两个数据库备份，checkpoint 时交替更新两个备份，系统中记录哪一个是当前最新的

checkpoint 文件。若执行 checkpoint 时系统崩溃，则总有一个完整的 checkpoint 文件可用。

五、总结

内存数据库是一个较新的研究领域．由于它具有传统的磁盘数据库无法比拟的优越性，已经引起了数据库领域的广泛关注与此同时．它也提出了很多需要研究和探讨的新课题．必须研究设计与之相适应的数据结构和算法、相应的各种管理技术．才能最大限度地发挥驻留内存的优越性。相信在不久的将来．随着硬件、软件技术的不断发展．内存数据库将有非常广泛的应用。

参考文献

[1] DeCandia G, Hastorun D, Jampani M, et al. Dynamo: amazon's highly available key-value store[C]. ACM Symposium on Operating Systems Principles: Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles, 2007，14:205--220.

[2] Kemper A, Neumann T. HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots[C]. Data Engineering (ICDE), 2011 IEEE 27th International Conference on, 2011:195--206.

[3] Bernet J. Dictionary compression for a scan-based, main-memory database system[D].[S.l.]: Master thesis, Eidgenössische Technische Hochschule, 2009-2010, 2010.

[4] Qureshi M K, Srinivasan V, Rivers J A. Scalable high performance main memory system using phase-change memory technology[C]. ACM SIGARCH Computer Architecture News, 2009，37:24--33.

[5] Liu M, Fei X D, Hu S, et al. Design and Implementation of Main Memory Database in ATC System[J]. Computer Engineering, 2010, 21:019.

[6] Zhao Y M, Zheng X F, Xu L Z. Design and implementation of index in main memory database system named SwiftMMDB[J]. Journal of Computer Applications, 2011, 9:024.

[7] Diaconu C, Freedman C S, Larson P A, et al. IN-MEMORY DATABASE SYSTEM[R], 2010. US Patent App. 12/756,185.

[8] Färber F, May N, Lehner W, et al. The SAP HANA database-an architecture overview[J]. IEEE Data Eng. Bull, 2012, 35(1).

[9] Ren K, Thomson A, Abadi D J. Lightweight locking for main memory database systems[C]. Proceedings of the 39th international conference on Very Large Data Bases, 2012:145--156.

[10] Niu X, Jin X, Han J, et al. A Cache-Sensitive Hash Indexing Structure for Main Memory Database[M]//Pervasive Computing and the Networked World.[S.l.]: Springer, 2013:400--404.

[11] Cattell R G, Russell C L. Systems and methods for a distributed in-memory database and distributed cache[R], 2012. EP Patent 1,840,766.

[12] Gurajada A P, Eluri A S, Nalawade V A, et al. Managing Data Storage as an In-Memory Database in a Database Management System[R], 2010. US Patent App. 12/726,063.

[13] Hoang C, Lahiri T, Neimat M A, et al. Distributed Consistent Grid of In-Memory Database Caches[R], 2009. US Patent App. 12/562,928.

[14] Han J, Song M, Song J. A Novel Solution of Distributed Memory NoSQL Database for Cloud Computing[C]. Computer and Information Science (ICIS), 2011 IEEE/ACIS 10th International Conference on, 2011:351--355.

[15] Plattner H. A common database approach for OLTP and OLAP using an in-memory column database[C]. Proceedings of the 35th SIGMOD international conference on Management of data, 2009:1--2.

[16] 王珊, 肖艳芹, 刘大为, et al. 内存数据库关键技术研究 [J]. 计算机应用, 2007, 27(10): 2353--2357.

[17] 陆宏. 一种高效内存数据库设计 [J]. 指挥信息系统与技术, 2012, 3(1):81--84.

[18] 郭超, 李坤, 王永炎, et al. 多核处理器环境下内存数据库索引性能分析 [J]. 计算机学报, 2010, 1:33.

[19] 袁培森, 皮德常. 用于内存数据库的 Hash 索引的设计与实现 [J]. 计算机工程, 2007, 33(18):69--71.

[20] 周游弋, 董道国, 金城. 高并发集群监控系统中内存数据库的设计与应用 [J]. 计算机应用与软件, 2011, 28(06):128--130.

[21] 张延松, 王占伟, 孙妍, et al. 内存数据库可控的 page-color 优化技术研究 [J]. 计算机研究与发展, 2011, 48(z2).