浙江工业大学

本科毕业设计开题报告

（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

一、选题的背景与意义

1.1 研究开发的目的

1969 年，埃德加 • 弗兰克 • 科德 (Edgar Frank Codd) 发表了一篇划时代的论文，首次提出关系数据模型的概念。但可惜的是，刊登论文的“IBM Research Report”只是 IBM 公司的内部刊物，因此论文反响平平。1970 年，他再次在刊物《Communication of the ACM》上发表了题为“A Relational Model of Data for Large Shared Data banks”(大型共享数据库的关系模型) 的论文，终于引起了大家的关注。

科德所提出的关系数据模型的概念成为了现今关系型数据库的基础。当时的关系型数据库由于硬件性能低劣、处理速度过慢而迟迟没有得到实际应用。但之后随着硬件性能的提升，加之使用简单、性能优越等优点，关系型数据库得到了广泛的应用。

但是现如今随着互联网的不断发展，各种类型的应用层出不穷，所以导致在这个云计算的时代，对技术提出了更多的需求，传统关系型数据库面临很多问题，主要体现在下面这四个方面：

1. 低延迟的读写速度：应用快速地反应能极大地提升用户的满意度；

2. 支撑海量的数据和流量：对于搜索这样大型应用而言，需要利用 PB 级别的数据和能应对百万级的流量；

3. 大规模集群的管理：系统管理员希望分布式应用能更简单的部署和管理；

4. 庞大运营成本的考量：IT 经理们希望在硬件成本、软件成本和人力成本能够有大幅度地降低；

虽然关系型数据库已经在业界的数据存储方面占据不可动摇的地位，但是由于其天生的几个限制，使其很难满足下面的这几个需求：

1. 扩展困难：当需要表与表之间 Join 操作的时候，数据库就不能很好的进行扩展；

2. 读写速度慢：当数据量达到较大规模的时候，很容易发生死锁的问题，直接导致读写速度急剧下降的问题。

3. 成本高：企业级数据库的 License 价格很惊人，并且随着系统的规模，而不断上升；

4. 有限的支撑容量：现有关系型解决方案还无法支撑 Google 这样海量的数据存储；

而因此思想诞生的内存数据库（MMDB）对数据访问读取有着很高的效率，对于一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，和需要支持大量并发访问的高性能事务处理平台来而言，它都是一个很不错的选择。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，必须对实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。

1.2 研究现状

目前，大数据时代推动了内存数据库的发展，国内外相关人员经过长年的研究，设计并实现了多种内存数据库的模型及商用或开源的产品。商用 MDB 的代表产品有 AltiBase、Timesten，SAP HANA 等，开源产品主要有 FastDB、SQLite、Redis 等，下文对比较热门的 Timesten、SAP HANA[8] 以及本课题所需的 redis 做一个简单的介绍。

1.2.1 Oracle TimesTen

Oracle TimesTen 内存数据库是一个功能全面的关系型内存数据库，旨在通过在应用层的运行，加速处理响应时间和关键任务型应用所需的高吞吐量。从某种角度上来看，TimesTen 也是一种 Cache 机制，是磁盘数据库的‘Cache’，通过物理内存中的数据存储区的直接操作，减少了到磁盘间的 I/O 交互。TimesTen 中的这个 Ten 据说就是指速度能达到基于磁盘的 RDBMS10 倍。

1.2.2 SAP HANA

SAP HANA 是一款面向数据源的、灵活、多用途的内存应用设备，整合了基于硬件优化的 SAP 软件模块，通过 SAP 主要硬件合作伙伴提供给客户。SAP HANA 提供灵活、节约、高效、实时的方法管理海量数据。利用 HANA，企业可以不必运行多个数据仓库、运营和分析系统，从而削减相关的硬件和维护成本。SAPHANA 将在内存技术基础上，为新的创新应用程序奠定技术基础，支持更高效的业务应用程序，如：计划、预测、运营绩效和模拟解决方案。

1.2.3 Redis

Redis 是一个高性能的 key-value 存储系统。和 Memcached 类似，它支持存储的 value 类型相对更多，包括 string（字符串）、list（链表）、set（集合）和 zset（有序集合）。这些数据类型都支持 push/pop、add/remove 及取交集并集和差集及更丰富的操作，而且这些操作都是原子性的。在此基础上，redis 支持各种不同方式的排序。与 memcached 一样，为了保证效率，数据都是缓存在内存中。区别的是 redis

会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且在此基础上实现了 master-slave（主从）同步。

二、研究开发的目标、基本内容，技术关键

2.1 研究目标

通过阅读源码和相关资料来了解 Redis 的工作原理，从数据结构到服务器构造在内的相关知识，并应用于实际的大数据中。本课题的研究目标定位于利用 Redis 基本架构，从而实现一个简单的大数据应用系统。

2.2 基本内容

本课题研究的基本内容可以分成三部分：

（1）了解 Redis 的工作原理，包括内部数据结构（动态字符串、双端列表、字典、跳跃表），内存映射数据结构（整数集合、压缩列表），数据类型（对象处理机制、字符串、哈希表、列表、集合、有序集），功能的实现（事务、订阅与发布、慢查询日志），内部运作机制（数据库、rdb、aof、事件、服务器与客户端）

（2）支持 SQL 语言的存储引擎设计：支持基础的查询修改等 SQL

（3）设计开发图形化的性能测试工具：能准确明显得反应出内存数据库和磁盘数据库之间性能的差距

2.3 技术关键和难点

了解 Redis 的工作原理，支持 SQL 语言的存储引擎设计、图形化性能测试工具的开发。

三、研究开发的方法、技术路线和步骤

（1）Redis 架构

（2）Redis 运行流程 [22]

1. 服务器初始化：从启动 Redis 服务器，到服务器可以接受外来客户端的网络连接这段时间，Redis 需要执行一系列初始化操作。整个初始化过程可以分为六个步骤：
   I. 初始化服务器全局状态
   II. 载入配置文件
   III. 创建 daemon 进程
   IV. 初始化服务器功能模块
   V. 载入数据

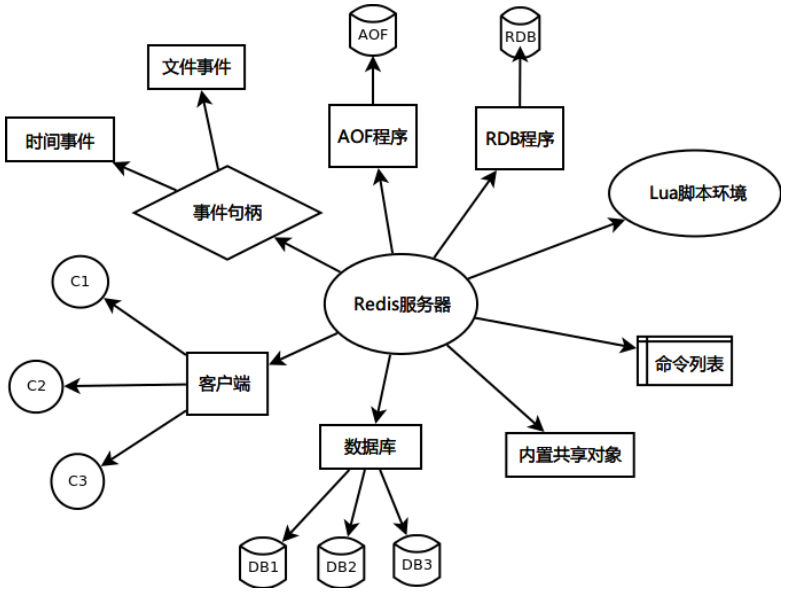

图 3.1: Redis 架构

VI. 开始事件循环

2. 当一个客户端通过套接字函数 connect 到服务器时，服务器执行以下步骤：
   I. 服务器通过文件事件无阻塞地 accept 客户端连接，并返回一个套接字描述符 fd
   II. 服务器为 fd 创建一个对应的 redish/redisClient 结构实例，并将该实例加入到服务器的已连接客户端的链表中
   III. 服务器在事件处理器为该 fd 关联读文件事件
3. 当客户端连上服务器之后，客户端就可以向服务器发送命令请求了。从客户端发送命令请求，到命令被服务器处理、并将结果返回客户端，整个过程有以下步骤：
   I. 客户端通过套接字向服务器传送命令协议数据
   II. 服务器通过读事件来处理传入数据，并将数据保存在客户端对应 redisClient 结构的查询缓存中
   III. 服务器在事件处理器为该 fd 关联读文件事件
   IV. 根据客户端查询缓存中的内容，程序从命令表中查找相应命令的实现函数
   V. 程序执行命令的实现函数，修改服务器的全局状态 server 变量，并将命令的执行结果保存到客户端 redisClient 结构的回复缓存中，然后为该客户端的 fd 关联写事件
   VI. 当客户端 fd 的写事件就绪时，将回复缓存中的命令结果传回给客户端

（3）编程语言：Python

Python 现在是最流行的动态编程语言之一，紧接着是 Perl,Tcl,PHP 和最近的 Ruby。虽然它经常被看做是“脚本语言”，但实际上它是和 List 或者 Smalltalk 类似写法的通用编程语言。今天，python 被用在很多地方，从使用完就扔的脚本到提供 24x7 不间断服务的大型 web 服务。它还用在 GUI 和数据库编程，客户端和服务端 web 编程，程序测试。作为一种即译式的，互动的，面向对象的语言，它包含了异常处理，动态资料形态，高层次的动态结构，和类别的使用。Python 语法简单，但是功能出奇的强大。其表达式语法优美易读。并包含了其他优秀脚本语言的一些特点：解释的，面向对象，内置的高级数据结构，支持包以及模块，多平台，可扩展。

四、研究工作总体安排与时间进度

表 4.1: 工作总体安排与时间进度

任务序号	起止时间	阶段任务要点
1	2013.1.11 - 2013.2.20	了解课题相关内容，查找中、英文资料
2	2013.2.21 - 2013.3.8	查阅文献资料，完成文献综述、开题报告和外文翻译
3	2013.3.9 - 2013.3.25	阅读 Redis 源代码，分析其架构
4	2013.3.26 - 2013.4.1	系统详细设计
5	2013.4.2 - 2013.4.24	编码
6	2013.4.25 - 2013.4.30	系统测试
7	2013.5.1 - 2013.5.15	整理资料，完成论文

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).

[22] huangz. Redis 的设计与实现 [R]. http://www.redisbook.com/.