文章编号:1000-5641(2014)05-0082-07

内存数据库的可用性综述

江泽源, 刘辉林, 吴 刚, 王国仁
(东北大学 信息科学与工程学院,沈阳 110004)

摘要: 随着计算机硬件技术的高速发展,内存的成本不断降低,数据库管理系统将其工作数据集完全放入内存变得可行. 相比于常规的磁盘数据库,内存数据库具有更快的数据存储速度、更高的吞吐量和更强的并发访问能力,满足了许多应用的快速响应需求. 然而,由于内存是易失性存储介质,与磁盘数据库在可用性方面有一定区别. 本综述重点讨论了适用于内存数据库提高可用性的主要策略,包括快速恢复策略、冗余备份和容错等.

关键词: 内存数据库; 可用性; 可靠性; 快速恢复; 容错
中图分类号: TP392 **文献标识码**: A **DOI**:10.3969/j.issn.1000-5641.2014.05.007

Survey of main-memory database availability

JIANG Ze-yuan, LIU Hui-lin, WU Gang, WANG Guo-ren
(*School of Information Science and Engineering, Northeastern University, Shenyang* 110004, *China*)

Abstract: With the development of hardware technology, the cost of main memory is decreasing, which makes it possible to let DBMS (Database Management System) put the whole data into main memory. Compared to traditional DRDB (Disk-Resident Database), MMDB (Main-Memory Database) provides much faster of data storage, higher throughput of applications, stronger ability on concurrent access, and meets the demand of timely response. However, due to its volatility, MMDB has differences on system availability with DRDB. The survey focuses on main strategies of improving the availability of MMDBs, including fast recovery, redundant backup and fault tolerance mechanism.

Key words: main-memory database; availability; reliability; fast recovery; fault tolerance

0 引 言

内存数据库(Memory-Resident Database 或 Main Memory Database, MMDB)将工作用的数据集放置于物理内存中,由于存储介质的特性不同,相对于常规的磁盘数据库(Disk-

收稿日期:2014-06
基金项目:国家自然科学基金(61332006,61370154,61025007,61328202); 973 计划(2011CB302200-G); 863 计划(2012AA011004)
第一作者:江泽源,男,硕士研究生,研究方向为内存数据库.
通信作者:刘辉林,男,教授,研究方向为数据库. E-mail: liuhuilin@ise.neu.edu.cn.

Resident Database, DRDB)，内存数据库具有更高的访问速度和更低的系统延迟，并且不受磁盘 I/O 瓶颈限制. 在内存价格不断降低、容量迅速扩大的今天，富有竞争力的内存数据库产品已经应用于网络、通信、实时处理等多种行业领域.

根据用户需求的不同，内存数据库产品在追求高性能的同时，也需提供相应的可用性保证. 内存是一种易失性的存储介质，断电后数据丢失，且处理复杂任务时易崩溃. 保证系统实现不间断的服务，尽量缩短停机(Downtime)时间，减少用户和服务提供者的损失，是内存数据库面临的重大挑战.

本文主要关注内存数据库的可用性问题，首先通过归纳相关文献中关于系统的可用性定义，给出了内存数据库可用性的解释，包括可用性的描述、度量、表现方面；接着从内存数据易失的特性入手，探究了内存数据库提高可用性的方法，着重关注了包括检查点、日志、恢复协议的快速恢复策略，和以热备份、集群内复制为代表的容错机制；最后进行概括总结，并做出了展望.

1　内存数据库概述

1.1　内存数据库概念

内存数据库将工作数据集存放在内存中，相对于常规磁盘数据库，其存取速率有了极大的提高. 磁盘中存储日志文件和相关映像备份，数据的存取工作尽量不再涉及磁盘的 I/O 操作，以达到更高的效率，此时系统的主要矛盾将不再是磁盘的 I/O 操作时间.

由于存储介质不同，内存数据库在索引结构、存储管理、并发控制等方面有许多技术上的改进[1].

1.2　内存数据库系统面临的挑战

高存取速率带来的高吞吐量，以及特殊应用的实时性响应需求，导致内存数据库系统的负载往往过高；易失性存储器带来数据丢失风险，意外事故导致的数据破坏，可能直接导致财产的严重损失，这也是系统所面临的挑战. 从内存数据易失的角度来看，提升有关可用性水平的技术是内存数据库研究的重要方向之一.

2　可用性概念

2.1　通用数据库的可用性

数据库的可用性(Availability)概念经常出现在文献中，但是至今尚未有标准的、统一的定义. 从传统数据库系统的角度来说，可用性指系统提供的特定的服务等级，当用户或某个进程访问数据库时，它是可用的[2]. 或者描述为在给定的时间内，数据库可用的时间比例，即数据库按照要求能够正常运行的概率. 可用性的概念常常和可靠性(Reliability)一起出现，可靠性多指一个数据库在一个给定的时间间隔内不产生失败的概率[3]. 同时，可恢复性、可持续使用等概念也是构成可用性的相关概念.

系统应当具有即时响应的能力，但当断电(Outage)情况出现，系统服务失效，数据库变得不可用，经常使用停机一词来描述系统不可用时的那段状态. 停机可被分为两大类，即系统遇到意外停机(Unexpected Downtime)，如应用程序运行错误、硬件故障、断电，和计划内的停机(Scheduled Downtime)，如系统硬件更换、软件升级. 设计者需要在一定的成本约束下，在用户的容忍范围下，将系统尽可能地保持可用[4].

2.2 可用性的度量和等级

相对于不同应用的系统,可用性要求也不尽相同.学校的成绩管理系统可能只需要一个较低的可用性等级就可以满足学生查询分数的要求,而银行的管理系统则需要很高的可用性等级,保证各项业务的周转正常,否则可能导致严重的损失.

通常用平均故障间隔时间(MTBF)和平均修复时间(MTTR)来度量一个计算机系统的可用性,数据库系统也参照这些概念,它们的定义见表 1.

可用公式(MTBF / (MTBF + MTTR))对系统的可用性水平进行计算,计算结果可近似作为可用性等级的标准.在文献[5]中,作者根据当时的系统情况描述了一些典型的系统可用性等级[5],见表 2.

表 1 MTBF 和 MTTR 概念图

Tab. 1 The concept of MTBF and MTTR

名称	定义	计算
平均故障间隔时间	系统在失败前运行的平均时间长度	小时数/失败次数
平均修复时间	失败后修复和还原所需的平均时间长度	修复小时数/失败次数

表 2 几个典型的系统可用性等级

Tab. 2 Availability of typical systems classes

系统类型	不可用时间(分钟/每年)	可用性%	可用性等级
非管理	50 000	90	1
管理	5 000	99	2
良好管理	500	99.9	3
可容错	50	99.99	4
高可用	5	99.999	5
极高可用	.5	99.999 9	6
最高可用	.05	99.999 99	7

通常用“N 个 9”来描述表 2 中可用性一列,“5 个 9”,即 99.999%的高可用性(High Availability,HA)[6],意味着系统每年约有 5 分钟的停机时间,这是数据库系统中常使用的概念,这一特性也被许多内存数据库产品写入其技术文档中.

2.3 NoSQL 中的可用性

近年来持续增长的海量数据,催生了 NoSQL 非关系型数据库的迅速发展,其中也有 Redis 这样的内存型 NoSQL 数据库. Eric Brewer 在 2000 年提出的 CAP 理论[7],其定义是:一致性(Consistency)、可用性(Availability)、分区容忍性(Partition tolerance)这个三个属性,只能同时满足其中两个属性. CAP 理论数年后被证明[8],并成为 NoSQL 数据库设计和实现的重要理论之一.其中,“可用性”的定义与上述有微妙差别.这里的可用性指,如果客户可以同集群中的某个节点通信,那么该节点必然能够处理读取及写入操作.

2.4 内存数据库可用性

内存数据库中,特别是集群环境中,内存数据易失,复杂处理易崩溃,这与传统的磁盘数据库在可用性和可靠性方面有一定差别.

显然,内存数据库提升可用性的措施涉及到多个方面:使用更可靠的硬件,如使用更稳定的内存;在系统运行时通过监督等机制预防或及时发觉可能出错的地方,如在应用程序中加入异常状况的处理;当意外停机发生时,寻找更快速的恢复策略;或使用备份的方式,在当前节点中断服务后备份节点立即替代.

本文主要阐述在单机或集群环境中,内存数据的可恢复性及任务执行的可用性,主要关

注数据快速恢复、系统容错，涉及检查点技术、日志技术、复制技术.

3　快速恢复策略

在内存数据库中，磁盘中不再存储工作数据集，而是主要用于持久化操作（多指日志和映像备份）与其他附加信息的存储. 由 IBM 提出的 ARIES[9]（Algorithm for Recovery and Isolation Exploiting Semantics）算法是以日志记录、事务表、脏页表为主要信息进行分析、撤销、重做的算法，是经典的数据库恢复策略. 很多内存数据库产品使用这一原则并加以改进，以追求准确的恢复结果和更高的恢复速度.

检查点是避免消耗大量时间检查日志的策略，检查点配合相关日志可以进行数据库的恢复. 优秀的检查点技术和日志恢复策略可以减少检查点对正常事务的影响，减轻系统恢复的开销，也可以减少日志文件的大小，是提升恢复速度的关键.

3.1　检查点技术

检查点作为事务回滚的起点，在内存数据库中通常作为内存映像保存起来.

模糊检查点（Fuzzy Checkpointing）[10]是一类支持检查点和事务同时进行的技术，在检查点开始的同时记录一个活动中事务的列表，与此同时进行检查点的生成. 由于列表事务可能对某些数据元素进行了更改，故需要事务日志来记录上述活动部分，以表明将来是撤销还是回滚，并结合检查点用于恢复. 由于这种检查点生成的映像是不明确的，故称为模糊检查点. “静态的”检查点技术不支持检查点和事务的同时进行，显然，模糊检查点记录避免了可用性损失，但也增加了管理上的复杂性.

乒乓（Ping-Pong）模糊检查点也是实现模糊检查点的一种措施，该算法[11]将模糊检查点算法和乒乓算法结合，利用两个映像文件，每次仅更新其中的一个，如此交替进行备份. 其目的是避免制作检查点时发生故障，导致该检查点既无法完成，又无法回到上个状态. 虽然消耗了更多的存储空间，但乒乓模糊检查点是一种更加可靠的、适用内存数据库的检查点策略.

文献[12]提出了一种改进的分区模糊检查点策略（Partition Fuzzy Checkpointing），该策略考虑了事务和数据“定时约束”，有以下几条原则：(1)较短有效期的数据需尽早写入到磁盘，减少数据损失；(2)短时限时序的数据无需写入至磁盘，减少不必要开销；(3)高频的数据应该优先写入至磁盘，减少日志处理时间；(4)关键的数据应该优先写入到磁盘，保证重要数据不会丢失；(5)优先级高的数据应该优先写入到磁盘，保证重要任务在时限内完成. 此外，基于分区模糊检查点策略的数据库，对数据段检查点的优先级进行逻辑分区，依照较高优先级的分区安排较高检查点频率的原则，在实验中表现出较好的性能.

模糊检查点技术及其改进技术广泛应用于 TimesTen、Altibase 等常见的内存数据库产品中，有效降低了检查点工作时系统的负载，增强了系统的并发访问能力.

3.2　日志协议

常见的数据库恢复“写前日志”协议，要求被更改的数据元素要先写入日志，以利用日志的内容进行恢复. 而一些内存数据库使用了 LAW（Logging After Writing）日志协议[13]，即“写后日志”协议，该协议通常需要一个非易失的存储器来记录更改的数据元素.

在使用模糊检查点“写前日志”协议中，利用 redo 日志来进行已提交事务的重做时，最近的检查点加上 redo 日志可能无法正确恢复，此时需要向前扫描日志，找到上一个检查点. 一个典型的例子[14]见图 1. A 的旧值为 5，redo 日志（A，7）表示 A 的值将被更新为 7. 检查点

发生在此日志之后,由于采用了模糊检查点算法,在检查点记录的过程中 A 的值可以更改,而恰巧这一更改未被检查点记录到.故障发生在事务 T 已经结束后,故 T 需要重做.这样一来,真正记录下 A 真实值的日志出现在了比这个检查点更早的时候,如果从检查点向前至日志(A,7)这段范围过长的话,系统需要付出更多的扫描代价.

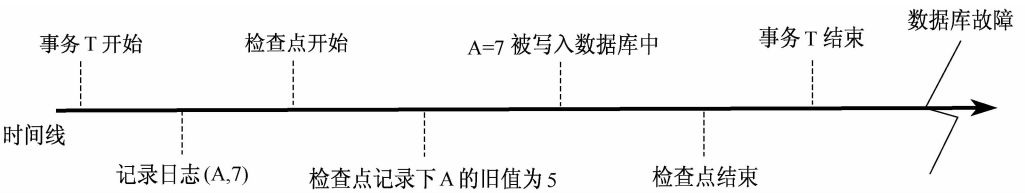

图 1 一个典型的检查点例子

Fig. 1 A typical instance of checkpoint

采用 LAW 日志协议后,最近的检查点一定可以作为重做的起点,redo 日志的起点一定在该检查点之后,这样就避免了过多的向前查找日志,提高了数据库系统恢复速率.

文献[15]提出了一种不同于 ARIES 思想的 command logging 技术,并将算法部署在 VoltDB 中.该主要特点是在日志中只记录事务命令,而不是主要记录数据元素的改变.当需要恢复时,从检查点向后执行事务命令,实现重新执行事务的效果,其目的是降低恢复时的系统负载,但实验测试中相对于 ARIES 并没有缩短系统恢复的时间.

4 主从备份与复制技术

冗余(Redundancy)是容错系统设计中的基本原则,其思想就是通过冗余数据实现更高的可用性.内存数据库系统可以采取主从备份的方式,主节点(工作节点)在停机后,备节点及时接上,保证服务离线时间最短化.此外,在集群环境下,复制技术将主机数据备份到多个节点上,不仅能满足可用性,实现快速恢复,也防止了单个节点性能瓶颈的问题,实现负载平衡.

4.1 主从备份

主从备份是常见的冗余策略,下面以 Oracle 公司的 TimesTen 内存数据库的备份方案作为示例,见图 2.

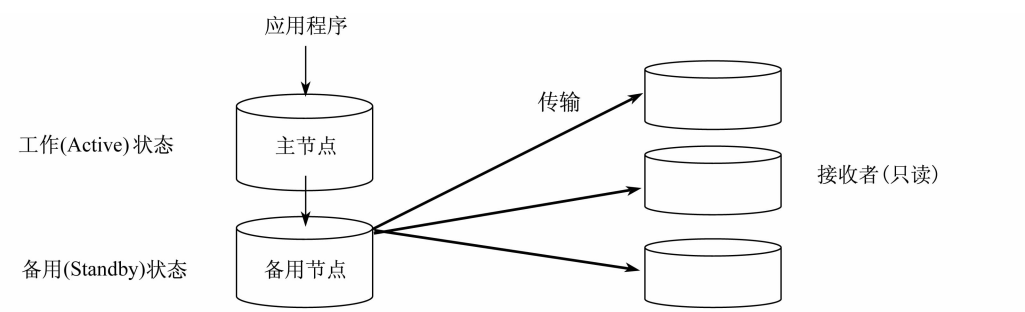

图 2 TimesTen 的 Active-Standby 方案

Fig. 2 Active Standby pair of TimesTen

TimesTen[16]是 Oracle 的优化内存的关系数据库,在其备份方案中,主节点执行更新操作,并将操作同步到备用节点上.备用节点同时将更改传播至只读的接收者(一般是远程的灾难备份).主节点失效后,备用节点升级为主节点,接受应用程序的直接操作.若是备份节

点失效,那么主节点直接将更改发送至只读接收者,待备用节点恢复之后,备用节点通知主节点将刚才缺失的更新再发送过来. TimesTen 还利用了产品中的 IMDB Cache 技术,提供了跨层次(Cross-Tier)的可用性方案,包括只读缓存组和异步写入缓存组,提供了 Oracle 和 TimesTen 的高效连接.

4.2　同步复制与异步复制

同步复制指每次复制必须等到所有拷贝结束后才算完成,使得复制数据在任何时间,任何复制节点均保持一致. 而异步数据复制允许所有节点在某个时刻内的数据不是同步的,它们之间存在延迟. 异步复制技术使得应用的响应时间加快,适用于需要高性能、低延迟的场景.

在 TimesTen 的 Replication 版本中,实现了高可用性和负载平衡. TimesTen 提供了灵活的复制配置,用户有多粒度级别的复制选择,包括表级别复制和整个数据库复制;多重形式的复制方式,多种包括单向复制,双向复制,表分片复制,N 路复制.

4.3　基于事务日志的复制

Altibase 是一个高性能、高通用性的内存数据库,适用于电信、交通等实时应用和嵌入式系统领域. Altibase 的复制功能是通过事务日志实现的[17],在主节点通过一个发送者(sender)线程,将所有新产生的事务日志发送至备份节点,备份节点的接收者(receiver)线程接收并反映到数据库上,同时记录已经发送成功的日志位置. 相比于发送查询语句或是执行计划的复制方式,复制日志的方式仅需要将日志在本地转化为执行计划,使节点的负荷达到最小.

5　研究近况与展望

本文从快速恢复策略和冗余容错两个方面介绍了内存数据库提升可用性的一些方案. 这些方案中,有的将常规磁盘数据库的方法直接运用到内存数据库中,有的利用内存存储的优势,结合实际情况,提出了改进优化算法,取得了良好的效果. 内存数据库产品所适用的范围越来越广,其性能和可用性的要求也会越来越高,更有效的可靠性保障和更快的恢复速度,一直都是提升可用性的重点方案.

随着硬件成本的不断降低,TB 级别内存的服务器已经出现;64 位系统的成熟,计算机寻址范围扩大,内存数据库有了更好的应用环境,基本摆脱了上世纪 80 年代有理论而缺少实际产品的状况. 商业用的产品不断出现和壮大,为更多行业提供了更快速的数据管理方案. 同时,有关可用性的研究工作也在不断进行. 下面关注了一些当今及将来可能的研究热点.

首先,越来越多的研究在针对特定的应用环境提出一类新的算法. 文献[18-19]等提出了一些适用于内存数据库的具有频繁一致(Frequently Consistent,FC)特点的应用程序恢复算法,FC 指在分布式数据库系统中每个节点频繁地更新数据,不仅数据更新到位,且要求所有节点达到数据一致的情形,大规模多用户的在线游戏(Massively Multiplayer Online Games,MMOs)就是 FC 的一个实例,降低系统延迟和减少系统开销是提出新算法的目的,利用更多的内存容量配合逻辑检查点为手段,实现更快速的恢复速率.

其次,减少持久化数据的大小,防止日志成为性能瓶颈,采取更有效的数据持久化方式,追踪可恢复数据的来源,是实现高速恢复的可行性措施.

再次,SSD 一类的快速非易失性存储器成本的不断下降,很多家用级 PC 产品已经使用了 SSD 硬盘作为主要存储选择. 其访问速度快,而且数据断电不丢失,即拥有远超磁盘的速

度和非易失的良好特性,虽然容量不足以和磁盘相比,但这使得今后的内存数据库实现方案有了更广泛的选择,例如将需要持久化到磁盘的高频数据暂时放置在 SSD 中.

最后,更高性能、更高可用性的服务器的使用,将数据库以纵向拓展,可以有效减少集群中备份的数目,降低系统设计的复杂性,实现快速无缝的切换恢复,保证任务的高效执行.此外,将复杂任务进行分解,每个部分承担事务的一部分,如何处理子事务崩溃时与总事务的关系,也是研究的方向之一.

[参 考 文 献]

[1] DEWITT D J, KATZ R H, OLKEN F, et al. Implementation techniques for main memory database systems[R]. Berkeley: University of California 1984.

[2] BOTTOMLEY J. Clusters and high availability[EB/OL]. Presentation, January, 2002.

[3] 邵佩英. 分布式数据库系统及其应用[M]. 北京:科学出版社,2000.

[4] RAGHAVAN A, RENGARAJAN T K. Database availability for transaction processing[J]. Digital Technical Journal, 1991, 3(1).

[5] GRAY J, SIEWIOREK D P. High-availability computer systems[J]. Computer, 1991, 24(9): 39-48.

[6] AHLUWALIA K S, JAIN A. High availability design patterns[C]//Proceedings of the 2006 conference on pattern languages of programs. ACM, 2006: 19.

[7] BREWER E A. Towards robust distributed systems[C]//PODC, 2000: 7.

[8] GILBERT S, LYNCH N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services[J]. ACM SIGACT News, 2002, 33(2): 51-59.

[9] MOHAN C, HADERLE D, LINDSAY B, et al. ARIES: a transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging[J]. ACM Transactions on Database Systems (TODS), 1992, 17(1): 94-162.

[10] HAGMANN R B. A crash recovery scheme for a memory-resident database system[J]. Computers, IEEE Transactions on, 1986, 100(9): 839-843.

[11] SALEMK, GARCIA-MOLINA H. Checkpointing Memory-Resident Databases[C]//Proceedings of the Fifth International Conference on Data Engineering. IEEE Computer Society, 1989: 452-462.

[12] 廖国琼, 刘云生, 肖迎元. 实时内存数据库分区模糊检验点策略[J]. 计算机研究与发展, 2006, 43(7): 1291-1296.

[13] LI X, EICH M H. A new logging protocol: LAW[M]. Department of Computer Science and Engineering, Southern Methodist University, 1992.

[14] 陈安龙. 内存数据库中数据恢复技术的研究与实现[D]. 杭州:浙江大学,2006.

[15] MALVIYA N. Recovery algorithms for in-memory OLTP databases[D]. Massachusetts Institute of Technology, 2012.

[16] LAHIRI T, NEIMAT M A, FOLKMAN S. Oracle TimesTen: an in-memory database for enterprise applications [J]. IEEE Data Eng. Bull., 2013, 36(2): 6-13.

[17] JUNG K C, LEE K W, BAE H Y. Design and implementation of replication management in main memory DBMS ALTIBASE TM[M]//Parallel and Distributed Computing: Applications and Technologies. Springer Berlin Heidelberg, 2005: 62-67.

[18] LOMET D, TZOUMAS K, ZWILLING M. Implementing performance competitive logical recovery[J]. Proceedings of the VLDB Endowment, 2011, 4(7): 430-439.

[19] CAO T, VAZ SALLES M, SOWELL B, et al. Fast checkpoint recovery algorithms for frequently consistent applications[C]//Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data. ACM, 2011: 265-276.

(责任编辑 林 磊)