一种基于 V^3 模型的内存数据库性能分析研究

王寅峰[1,2,3],王龙翔[3]
(1. 深圳信息职业技术学院 软件学院,广东 深圳 518172;
2. 北京航空航天大学深圳研究院,广东 深圳 518001;
3. 西安交通大学 电信学院,陕西 西安 710049)

摘 要:针对大数据时代各种复杂业务对数据处理日益增长的性能要求,以及对数据管理中:模式自由、高可用、轻量级复制、大容量水平可扩展等方面的需要,文中从内存数据库的存储类型、体系结构、规模、并发性、可用性与可扩展性等方面对 19 种主流内存数据库进行了对比分析。提出了一种综合考虑处理速度、规模与可扩展性的 V^3 性能模型,对主流内存数据库进行了分类,并选取了有代表性的内存数据库在高频量化交易测试环境进行性能分析与测试。结果表明,NewSQL 数据库有较好的综合性能。为提高内存数据库在多任务并行情况下处理的速度,文中对多核环境中内存数据库的设计与优化进行了分析,将优化过程分为访存、并发加速和数据划分模式,并对内存数据库发展进行了展望。

关键词:内存数据库;事物处理;性能模型;高频量化;多核计算
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2015)06-0077-07
doi:10.3969/j.issn.1673-629X.2015.06.017

Research on Performance Survey in Memory Database Based on V^3-model

WANG Yin-feng[1,2,3], WANG Long-xiang[3]
(1. College of Software, Shenzhen Institute of Information Technology, Shenzhen 518172, China;
2. Research Institute of Beihang University in Shenzhen, Shenzhen 518001, China;
3. School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China)

Abstract: To satisfy the ever-increasing performance demand of Big Data and critical application's operation, the data management needs to offer the flexible schema, high availability, light weight replica, high volume and high scalability features, the 19 kinds of main memory database has carried on the comparison and analysis from storage type, system structure, scale, concurrency, availability and scalability in the memory database. The V^3 performance model is proposed considering the velocity, volume and scalability comprehensively, classifying the main memory database, and selecting the representative in-memory database to conduct the analysis and testing in the high frequency of quantitative trading environment. Test results clearly demonstrate that NewSQL is better at dealing with high-frequency trading models. To increase the parallel processing speed of memory database under the condition of multitasking, the design and optimization for memory database of the multi-core environment are analyzed, and the optimization process is divided into fetching, concurrent acceleration and data classification mode, and development of memory database is discussed.

Key words: memory database; transaction; performance model; high-frequency trading; multi-core computing

0 引 言

随着"Big data"(大数据)[1]时代的到来,电力、医疗、社交网络、电信、金融等行业数据作为业务的沉淀可以说"数据就是业务本身"[2]。业务能否高效执行与数据处理的实时性和可扩展性这两个方面密切相关,例如证券高频金融交易数据的主要特点为实时性(Real-time)与大规模(high-volume),通常沪深两市每天 4 小时的交易时间会产生 3 亿笔以上成交数据,2014/12/5 两市交易额首次突破万亿元,之后持续多日保持万亿元以上规模,交易记录随着时间累积规模

收稿日期:2014-07-17 **修回日期**:2014-10-24 **网络出版时间**:2015-05-06
基金项目:国家自然科学基金资助项目;深圳基础研究重点项目(JCYJ20120615101127404);深圳科技攻关重点项目(JSGG20140519141854753)
作者简介:王寅峰(1974-),男,博士,香港浸会、香港理工、香港大学博士后研究学者,CCF 会员,研究方向为内存数据库、云计算、人工智能、无线传感器网络等。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20150506.1646.040.html

非常可观。金融投资研究机构需要对历史交易和实时数据进行挖掘,以修正数量化交易模型支持高频量化等复杂自动交易在证券交易中的应用。这些应用的复杂性(包括高可用性、高性能,低延迟实时数据呈现)和数据规模(包括财务,金融与历史汇总交易数据、新闻资讯及研报、每个交易日数据增量等)的叠加又对数据处理的灵活性提出了新的挑战。

传统的数据库管理系统基于磁盘进行管理,需要频繁地访问磁盘来进行数据操作[3]。由于磁盘寻道要进行磁头的机械移动,另一方面受到系统调用(CPU 中断完成,时钟周期的制约)的影响,当操作频繁且复杂时,I/O 瓶颈问题日益突出,数据处理的实时性和可扩展性均无法保证。当前内存(RAM)容量越来越大,内存的价格也越来越低,64 位操作系统已经支持 180 亿 GB 内存的寻址能力,充分利用内存技术提升数据库性能成为普遍趋势。主要方法之一是共享内存技术,在数据库中增大缓冲池,将一个事务所涉及的数据都放在缓冲池中,组织成相应的数据结构进行查询和更新处理以最小化磁盘访问但仍然受到 I/O 操作延迟的影响。另一种是内存数据库(In Memory Database)技术[4],通过对查询处理、并发控制与恢复的算法和数据结构进行重新设计,更有效地发挥 CPU、内存、传输带宽等硬件的性能,通过将数据库装载在内存中以避免事物处理中的 I/O 操作,可以加快数据处理的速度。在内存数据库中,影响数据库性能的决定因素也由传统关系数据库中的磁盘访问转变为如何降低计算时间以及内存访问的延迟上,确保事物处理的高实时性对内存数据库在体系结构设计等方面提出了新的要求。

数据库按对应的需求可分为四类,分别是:嵌入式、传统关系型(RDBMS)、NoSQL 以及 NewSQL 数据库。嵌入式数据库一般不支持网络访问,必须与应用程序在同一机器上运行,具有空间小、使用方便等特点,适合于轻量级应用,但是其扩展性差。RDBMS 是应用最广泛的一类数据库,其代表有 Oracle、SQL Server 以及 MySQL 等,但其架构继承于 System R 数据库,而 System R 数据库设计以磁盘为中心,由于物理器件的因素,磁盘读写速度远低于 CPU 以及内存的速度,该架构也制约了 RDBMS 性能的提升。NoSQL 数据库是一类具有相似特点的数据库,这类数据库一般不支持 SQL 语法,而以 key-value 方式来进行数据存储。这类数据库对于 OLTP 等应用并不适合,因为这种应用场景下对于 SQL 的支持往往是必须的。而 NewSQL 则是一类结合了传统关系型数据库与 NoSQL 优点的数据库,以 VoltDB 为代表,改变了传统的关系型数据库以磁盘为中心的架构,转变成了以内存为中心的架构方式。充分利用内存的访问速度,从而极大提高了数据库的性能。

随着高性能计算的发展,Exascale-computing[5]已经成为下一代计算技术发展的目标。人类对数据处理的性能要求是无止境的,从系统的角度出发围绕集成系统资源,以满足不断增长的对性能和功能的要求;而从应用的角度需适当分解应用,以实现更大规模或更细致的计算。文中提出了一种 V^3 模型,就 Velocity,Volume,Varity(处理速度、容量效率、复杂性)的性能方面,对目前市场主流的 IMDB 进行了分析,对典型系统进行了性能测试,并对多核环境中 IMDB 提高实时性技术的策略进行了分析与展望。

1 内存数据库系统性能分析模型

随着信息化建设的展开和应用的发展,需要处理的数据量越来越庞大,对实时性要求越来越高和复杂。例如股票在线交易系统中做一笔证券买/卖委托,涉及到了客户的基本信息、资金账号信息、股票信息、拥金和税率信息等,对这些数据进行查询、计算、修改等事务操作要满足实时性;而这些信息的数据量都在百万或千万级别,同时这种在线交易有可能瞬时发生成千上万笔。已有的商业内存数据库[6]虽解决了传统数据库的访问效率问题,但在事务实时性处理方面有明显的缺陷。实时内存数据库事务处理的特征是:

(1)资源是有限和共享的;

(2)资源与服务本身具有开放、异构和动态的特点;

(3)用户的数量、行为是随机的,而对响应时间要求是严格的。

应用中的实际业务逻辑复杂,业务之间的关系非常紧密,强调时序性,即业务逻辑执行是有顺序的,反映到数据库上就是其 SQL 语句只能按提交的顺序执行,在处理事务时采取序列化的方式,牺牲了并发性,无法全面满足 IMDB 对事务处理高实时性的要求。针对应用对 IMDB 应用发展的需求,文中参考大数据 3V 模型[7],给出了一个三维性能分析模型对 IMDB 系统进行分析。

在 V^3 模型中,容量维度除表明内存数据库能够处理的规模外,也反映了内存的利用率;处理速度维度则针对简单的 key-value 模式和复杂的事务处理模型分别进行讨论;多样性维度则着重于处理的并行与并发,即应用是简单而频繁的读写操作(parallel)还是复杂有序的数据库并发(concurrency)操作,如:Hash 连接、聚集运算、数据划分和排序等执行代价(计算和访存)较大的执行操作。相关维度的定量化分析如下所述:

Delay(延时)= 初始化开销+路由延迟+通道占用

时间+竞争延迟(排队 v. s. 多线程数目)　　(1)

Complexity(复杂度)= 任务的规模(分支语句数目+数据依赖程度)　　(2)

Scale = 任务的规模(记录数量 * 记录的维度)(3)

依据 V^3 模型,文中对应用范围较广的 19 种 IMDB 的各项特性进行对比,各属性特征列表如表 1 所示[8-11]。

表 1　IMDB 系统特性对比

IMDB 名称	SQL 92 标准	并发性	数据规模	存储类型	系统架构	可用性	可扩展性
eXtremeDB	SQL-89	multi-version concurrency control (MVCC) 0.7 次/ms	Billion	No Translation	嵌入式/ Shared memory	两阶段同步模式	1 对 *N* 的内存镜像
Oracle TimesTen	√	2.5 ms 读 10 ms 写 Transaction isolation Locks,MVCC	Billion	基于行的关系模型	C/S,嵌入式	事务日志记录与数据库检查点	X/Open XA Specification Java Transaction API
Altibase	√	MVCC	Billion	多种数据格式	C/S	复制、切换、Write Ahead Logging	星型复制 1;32
SQLite	√	整个数据库文件进行读取/写入锁定	Million	Typelessness 五种亲缘类型	嵌入式	持久的 ACID	无
Berkeley DB	√	支持数千的并发线程	最大 256 TB	任意类型的键/值对	嵌入式	两段锁技术和先写日志策略	无
FastDB	面向对象	并发读,不能并发写	Million	每行记录被视为一个对象实例	嵌入式单机	内存数据到磁盘文件的备份	无
H2	√	并发读 MVCC	Million	UUID 等 21 种数据格式	C/S,嵌入式	数据库镜像机制	Built-in Clustering / Replication
MEMCACHED	×	多线程锁机制来保证数据更新操作的互斥	Million	Key-Value 单个 Item 最大 1 M	C/S,两阶段哈希	支持数据的相互复制备份	C/S, memcached 间不共享信息,需分布式算法由客户端完成路由选择的工作
Redis	×	不支持,但支持事务处理	Billion	Key-Value	C/S	Snapshotting Append-only file	主从同步(Sina weibo)
MONGO DB	兼容	per-database 级锁 支持 ACID 将降低性能	Billion	类似 json 格式,存储内容是文档型	集群结构	主-从、主-主模式及服务器之间的数据复制	支持水平的数据库集群,可动态添加
HBASE	×	Map-reduce 模式 不支持 ACID	Billion	列存储 Big table	分布式	Hadoop HDFS	支持超大规模集群
TOKYO CABINENT /TYRANT	×	concurrency in multi-thread	Billion	Key-Value 可以是二进制,字符串	C/S	双机互为主辅模式,支持持久记录	C/S,支持 HTTP
LEVEL DB	√	Share Nothing 所有存储过程(事务)均全局有序,避免了锁的使用	Billion	SkipList Key-Value	分布式	Snapshot,记录 log	支持超大规模集群
Aerospike	√	multi-key transaction	Billion	Key-Value Sets/Records	C/S, Cloud	Partition Tolerant Mode multi-server replication	cluster-aware
VOLTDB	subset	并行的单线程处理方式确保数据一致性避免了锁的使用	Billion	structured or unstructured data	Hash 分割数据库	Snapshot,记录 log	支持集群
S-Store	√	streaming OLTP	大于 Million	数据预处理避免锁机制	3 层架构	Stream-oriented transaction model	shared-nothing clusters
MySQL + handler socket	√	聚集请求,同时执行聚集起来的请求和返回结果	Billion	Key-Value structured unstructured	支持插件	同 MySQL	同 MySQL
MySQL + Innode with Memcached	√	同 MySQL	Billion	Key-Value structured unstructured	支持插件	同 MySQL	同 MySQL
MySQL cluster + Memcached	√	同 MySQL	Billion	Key-Value structured unstructured	支持插件	同 MySQL	去中心化的无缝高可扩展性

注:表中的嵌入式结构说明该 IMDB 是可直接链接到进程内的数据库引擎

通过表 1 的分析,可以将这些 IMDB 分为以下 4 种类型,如表 2 所示。

表 2 IMDB 分类

名称	特点	主要系统
企业商用	支持事务处理,规模大,可扩展,接口丰富、管理工具齐全、应用面广、成熟性、可维护和升级性能好,价格昂贵	eXtremeDB, Oracle TimesTen, Altibase
嵌入式	与应用程序直接绑定、轻量、处理速度较快,并发性与可扩展性较差,多种 license	SQLite, Berkeley DB, FastDB, H2
NoSQL	非关系型、分布式、开源、水平可扩展及 Key-Value 存储,处理速度快,操作简便,扩展容易但不支持事务处理,多种 license	MEMCACHED, REDIS, MONGO DB, HBASE, TOKYO CABINENT/TYRANT, LEVEL DB
NEWSQL	具备高度结构化查询,支持 ACID、事务处理,支持 K-V 查询、具备通用接口,具备高可扩展与高性能、速度快,多种 license	Aerospike, VOLTDB, MySQL+Handler Socket, S-Store, MySQL+Innodb with Memcached, MySQL Cluster+Memcached

2 典型数据库性能测试

考虑到数据库种类较多并且不同的数据库运行的平台不一样,很难在相同的环境下进行统一的性能对比,例如 SQL Server 只能运行在 Windows 平台上,而 VoltDB 只能运行在 Linux 平台上,因此文中从每一类数据库中挑选出了具有代表性的数据库作为测试对象,在统一的 SUSE Linux 测试环境下进行性能测试对比。其中嵌入式数据库选择了 SQLite,RDBMS 选择了 MySQL,NoSQL 数据库选择了 MongoDB,NewSQL 数据库选择了 VoltDB。

2.1 测试环境

2.1.1 硬件环境

测试服务端硬件配置如下:2 路 CPU,其中 CPU 为 Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00 GHz,内存 32 G,磁盘为 4 块希捷 ST9300605SS SAS 硬盘,转速为 10 000 r/min,每块磁盘容量为 300 G。

客户端硬件配置:CPU 为 Intel Core i5 3210M@ 2.5 GHz,内存为 2 G,磁盘为西部数据 WD5000LPVT-22G33T0,转速为 5 400 r/min,容量为 500 G。

2.1.2 软件环境

服务器端软件环境如下:测试数据库分别为 SQLite Version 3.7.17, VoltDB-3.2.0.1, MongoDB-Linux-x86_64-2.4.3, MySQL-5.6.11-Linux-glibc2.5-x86_64,操作系统为 SUSE Linux Enterprise Server 11 (x86_64)。

客户端软件环境如下:端操作系统为 Centos 6.4。客户端测试程序通过对 select/update/delete/insert 四类基本操作对服务端进行并发访问以测试数据库的性能。客户端程序并发测试线程设置为 100,测试时间持续 600 s。

2.2 测试数据集

文中选用实际证券交易系统的数据库中的高频量化数据表作为测试数据集,数据集 1customer 的记录数量为 856 497 条,字段 21 个,数据集 2clearmatch 的记录数量为 77 603 条,字段 84 个,分别代表了频繁查询和高频读写的典型证券交易应用场景。数据集 1 中表维度较少但数据量庞大,数据集 2 中表结构复杂但数据量相对较少。

2.3 测试结果与分析

文中通过对在 600 s 内完成的总操作以及平均吞吐率来对三种数据库进行性能对比。图 1 所示为数据集 1 的测试中操作总数以及平均吞吐率对比,可以看出 SQLite 在 select,update,delete 性能上明显优于 MongoDB 与 MySQL,这是由于 VoltDB 与 SQLite 的操作是在内存中完成的。由于没有网络开销,SQLite 在 select,update,delete 性能优于 VoltDB。但是由于 SQLite 不能支持网络访问,必须与客户程序运行在同一节点上,并且可扩展性十分低下,具有很大的局限性。图 2 所示分别为在数据集 2 中四类数据库的总执行操作数与平均吞吐率。可以看到在复杂数据表结构的测试数据集中,VoltDB四类操作的性能远远高于其他数据

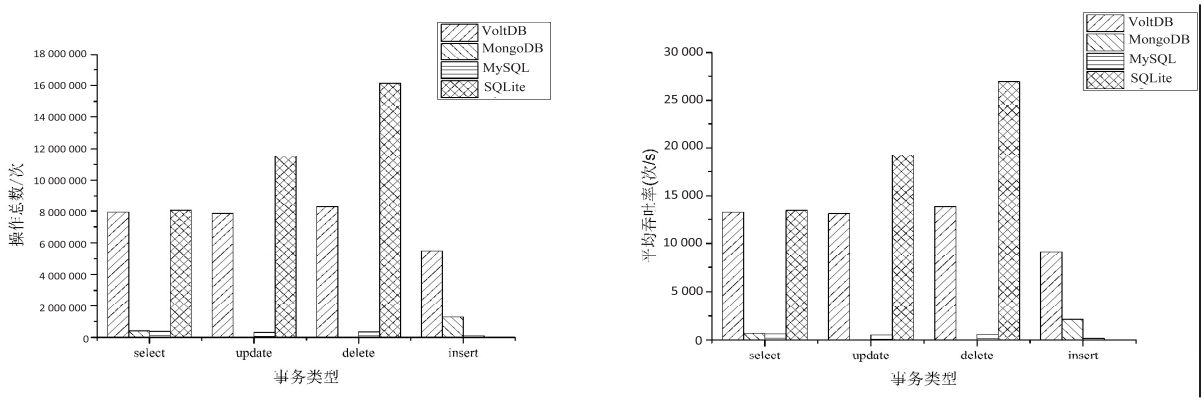

图 1 数据集 1 中四类数据库总执行操作数与平均吞吐率对比

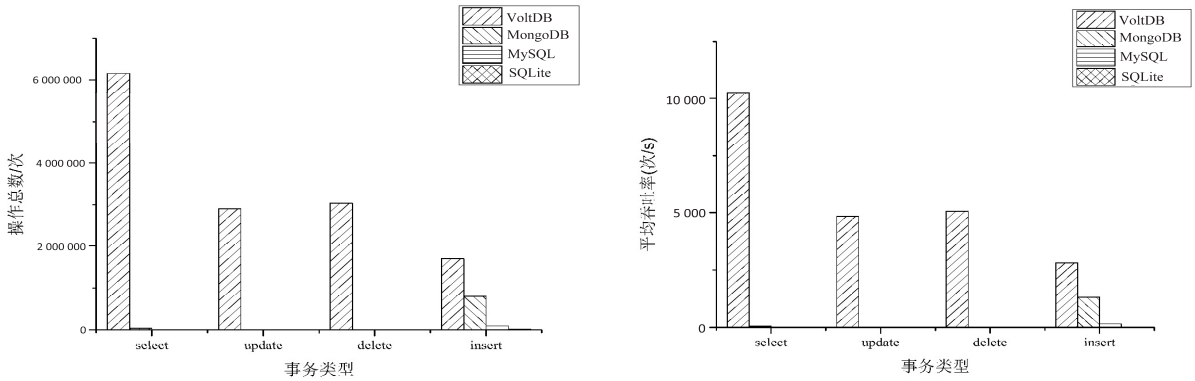

图 2　数据集 2 中四类数据库总执行操作数和平均吞吐率对比

库,这表明 VoltDB 对于不同场景下的 OLTP 应用具有很好的适应性,而 SQLite 四类操作的性能是最低的,这与数据集 1 的测试结果相反,也表明了 SQLite 只适合于表结构简单的应用场景。

从图中可以发现,MySQL 在数据集 1 的测试中 select,update,delete 基本上维持在 600 次/s 左右的吞吐率,而 insert 操作吞吐率比其他三种操作低,维持在 100 次/s。而在数据集 2 中 select,update,delete 操作的性能已经出现了很大程度的降低,这表明 MySQL 适合表结构简单的应用,对于表结构复杂的应用,MySQL 性能不高。MongoDB 在数据集 1 的测试中 select 与 insert 性能较高,能够达到几千次/s 的吞吐率。MongoDB 在数据集 1 的测试中 update 与 delete 性能非常低下,这与文中的测试数据以及 MongoDB 本身的并发锁控制机制有关。MongoDB 在数据字段较多时其 select 操作性能也会出现严重的性能下降,而 insert 操作则始终能够保持相对较高的性能。在数据集 1 与数据集 2 的测试中,VoltDB 的四类操作都能保持很高的性能,没有因为字段数目的增加而导致严重的性能下降。VoltDB 的高性能得益于其舍弃了硬盘,完全将数据置入内存中进行处理,并且针对现有的 OLTP 应用程序访问特点,重新设计了数据库架构,以及改进了并发锁控制机制。

2.4　V³模型对比

通过对四类 IMDB 系统进行分析,可知其对应的 V³模型如图 3 所示。

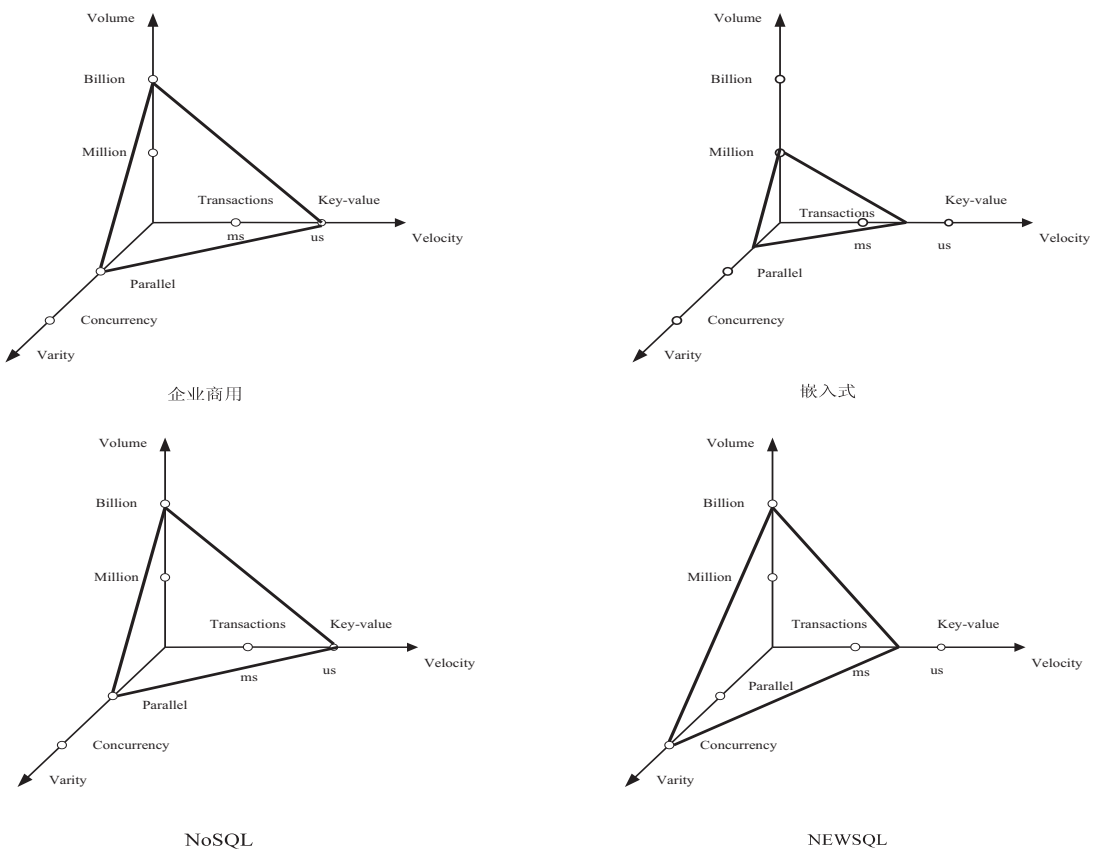

图 3　各类 IMDB 系统的 V³投影模型

由图 3 可知,在性能上 NoSQL 数据库性能已经达到了商用数据库的要求,但在事物处理方面较弱,而 NEWSQL 系统已经在容量和多样性方面取得了很大进展,并且通过扩展 Memcached-like 处理模式在处理速度上有较大提升,由于具有一定的并行性在大数量环境下也有较好的性能,同时具有良好的线性可扩展性和灾备机制,是除 NoSQL 数据库之外一个很好的选择。但是随着技术的发展,CPU 速度的提升远远超过内存处理速度的增长。例如在 Core 2 3.0 GHz 上,大部分简单指令的执行只需要一个时钟周期,也就是 1/3 ns,当 CPU 操作一块内存区域时,其中的信息若未保存在 L1/L2 cache,则需要将其从系统主存中调入 cache,然后再处理需等待大约 80 ~ 250 个时钟周期的延迟。因此,如何更好地提升 IMDB 系统的并行处理,避免或隐藏访存延迟以提升处理速度是 IMDB 在新的多核环境中设计与优化的目标。

3 多核环境中内存数据库设计与优化

随着高性能计算的发展,Top500 计算系统的性能已经超过了 10 Pflops[12],而以 GPU、异构众核(MIC)为代表的新一代体系结构将使单机 T 级计算成为可能。GPU 因其大规模的计算核而拥有强大的浮点计算能力,适合计算密集型任务,而由于其缺少分支预测判断单元,不适合逻辑性事务处理,因此,为发挥其优势,缺点由通用处理器 CPU 来弥补。CPU-GPU 系统具有多级硬件并行性,系统内各个节点(Node)由高速互联网络进行连接和扩展,所有节点并行的执行程序,节点内和节点间不同处理核心(主处理核心与协处理核心)并行的运行程序,形成任务级或程序级的并行。在每一个 GPU 内部,使用数量庞大的 SP(流处理器)并行处理数据,形成单个任务内的数据并行。这种处理方式对提升 IMDB 处理复杂应用时通过多任务并行隐藏访存延迟提供了保证。

为提高 IMDB 在多任务并行情况下的处理速度,可以将优化过程分为访存、并发加速和数据划分三级优化,从访存延迟、任务映射及主机-设备端通信开销等多方面予以分析[13-14],最大程度上发挥系统的性能。

3.1 访存优化

其主要目的在于针对变量访存特性为其进行访存优化,即为其分配合适的存储空间,以减少访存延迟;协处理器的存储器中,常量存储器空间较小,而寄存器属于每个线程私有,线程之间不能共享,多用于存储临时变量。由于 IMDB 处理中访存非常频繁,因此,访存级优化策略主要针对全局存储器上的数据进行共享存储器优化和纹理存储器优化。

3.2 并发加速优化

并发加速级优化主要设计是在访存级优化的基础上加速 GPU 计算,关键优化技术包括线程块整合、循环分块、寄存器暂存等。在 IMDB 中减少线程间的同步与序列化可以有效提升响应速度。进一步通过 Cache 预取有效利用片上存储共享存储器、寄存器和高速缓存,合理映射计算任务至 GPU 硬件,充分发挥 GPU 的计算能力。

3.3 数据划分优化

数据划分级优化主要目的在于充分利用 CPU-GPU 系统中的资源,尽可能地减小由于异构平台所带来的额外通信开销。例如对于 NEWSQL 类型的 MySQL Cluster+Memcached 系统,所有的写操作直接作用于 MySQL Cluster,消除缓存失效并通过数据一致性检查,确保数据库和缓存之间的同步。缓存和数据库中的重复数据被淘汰,使数据的简单重复跨多个应用,减少内存占用。可以先通过主机-设备端通信优化技术以最小化其数据传输;然后再通过采用 GPU 数据复用技术进一步减少主存与显存之间的数据传输;接着采用数据比例划分方法,基于 CUDA 流处理技术和异步拷贝函数、内核函数的异步启动特性以最大程度地实现 CPU 与 GPU 计算的重叠、PCI-E 总线通信和 GPU 计算的重叠。两方面重叠循环进行,以渐近的方式实现延时隐藏,从而缩短程序整体执行时间,充分利用系统资源,提升处理性能。三级优化策略的整体框架如图 4 所示。

图 4 NEWSQL IMDB 在多核环境中的优化策略

4 结束语

自 1984 年 D. J. DeWitt 等发表了《主存数据库系统的实现技术》[3]并提出了 Main Memory Database(主存数据库)的概念以来,IMDB 的发展非常迅速,已经将一定规模的事务和数据处理速度提升到了微秒级,在信息处理、自动化控制等很多关键应用中发挥了极大作用,此外 Google 的 Spanner[15]、淘宝的 OceanBase[16]等已经开始在可扩展的、多版本的、同步复制的、全球级分布式数据库方面具有很高的设计指标:可以扩展到几百万个机器节点,跨越成百上千个数据中心,具备几万亿数据库行的规模,对内存数据的推广应用起到了很好的示范作用。

但是随着数据规模和应用复杂性的增长,以及内存电气结构物理特性的限制,单纯依靠内存处理速度

的提升已经不能满足如金融交易在线事务处理等数据操作对高实时性的要求。文中通过对 19 种有代表性 IMDB 的分析,给出了一种考虑速度、规模与可扩展性的 V^3 性能模型对内存数据库性能进行对比和分析,并结合目前正在兴起的多核环境对内存数据库提高实时性处理技术的设计与优化进行了分析与展望。这些分析和优化方法对于提升金融、证券等行业关键交易应用的处理速度具有重要意义。

参考文献:

[1] Nature. Big Data[EB/OL]. 2008. http://www. nature. com/news/specials/bigdata/index. html.

[2] 孟小峰,慈　祥. 大数据管理:概念、技术与挑战[J]. 计算机研究与发展,2013,50(1):146-169.

[3] DeWitt D J,Katz R,Olken F,et al. Implementation techniques for main memory database systems[C]//Proceedings of annual meeting. Boston:ACM Press,1984.

[4] Kogge P. ExaScale computing study:technology challenges in achieving exascale systems[R/OL]. 2008. http://users. ece. gatech. edu/mrichard/ExascaleComputingStudyReports/exascale final report 100208.

[5] Exascalar results from November 2012[EB/OL]. 2012. http://www. top500. org/.

[6] Grobelink M. Big-data computing:creating revolutionary breakthroughs in commerce, science, and society [R/OL]. 2012. http://videolectures. net/eswc2012_grobelnik_big_data/.

[7] Oracle TimesTen[DB/OL]. 2014. http://www. oracle. com/us/products/database/timesten/overview/index. html.

[8] Srinivasan V,Bulkowski B. Citrusleaf:a real-time NoSQL DB which preserves ACID[J]. The Proceedings of the VLDB Endowment,2011,4(12):1340-1350.

[9] Ravi N,Yang Y,Bao T,et al. Apricot:an optimizing compiler and productivity tool for x86-compatible many-core coprocessors[C]//Proc of the 26th ACM international conference on supercomputing. New York,NY,USA:ACM,2012:47-58.

[10] VoltDB LLC. VoltDB technical overview, whitepaper[EB/OL]. 2014. http://docs. voltdb. com/.

[11] Çetintemel U, Du Jiang, Stonebraker M, et al. S-Store:a streaming NewSQL system for big velocity applications[J]. The Proceedings of the VLDB Endowment,2014,7(13):1633-1636.

[12] 曹仰杰,钱德沛,伍卫国,等. 众核处理器系统核资源动态分组的自适应调度算法[J]. 软件学报,2012,23(2):240-252.

[13] 张　保,董小社,白秀秀,等. CPU-GPU 系统中基于剖分的全局性能优化方法[J]. 西安交通大学学报,2012,46(2):17-23.

[14] Zheng Hao,Wang Endong,Wang Yinfeng,et al. Transparent driver-kernel isolation with VMM intervention[C]//Proc of conference on timely results in operating systems. Pennsylvania,USA:[s. n.],2013.

[15] Corbett J C,Dean J,Epstein M,et al. Spanner:Google's globally-distributed database[C]//Proc of 10th USENIX symposium on operating systems design and implementation. [s. l.]:USENIX Association,2012.

[16] Oceanbase 内存事务引擎[EB/OL]. 2014. http://oceanbase. org. cn/.

(上接第 76 页)

[3] IEEE Std 1609.12-2012. IEEE standard for Wireless Access in Vehicular Environments (WAVE) identifier allocations[S]. 2012.

[4] IEEE 802.11p-2010:wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications amendment6:wireless access in vehicular environments[S]. [s. l.]:IEEE Standards Association,2010.

[5] Ghandour A J,Felice M D,Artail H,et al. Dissemination of safety messages in IEEE 802.11 p/WAVE vehicular network:analytical study and protocol enhancements[J]. Pervasive and Mobile Computing,2014,11:3-18.

[6] IEEE Std 1609.4-2010. IEEE standard for Wireless Access in Vehicular Environments (WAVE)-multi-channel operation[S]. 2011.

[7] 唐　伦,柴　蓉,戴翠琴,等. 车联网技术与应用[M]. 北京:科学出版社,2013:96-116.

[8] Ahmad A,Doughan M,Mougharbel I,et al. A new adapted back-off scheme for broadcasting on IEEE 1609.4 control channel in VANET[C]//Proc of 11th annual mediterranean on Ad Hoc networking workshop. Ayia Napa:IEEE,2012:9-15.

[9] 张　民,李德敏,金　康. 一种多接口多信道 VANET 动态信道分配算法研究[J]. 计算机应用研究,2014,31(5):1516-1519.

[10] 王晨梦,赵云鹏,唐　伦. 基于时分复用的车载自组织网多信道 MAC 协议[J]. 计算机工程与设计,2014,35(5):1521-1525.

[11] 吴福源. 基于 IEEE_802.11p 协议的车载网络接入协议性能研究[D]. 西安:西安电子科技大学,2012.

[12] Han C,Dianati M,Tafazolli R,et al. Asynchronous multi-channel MAC for vehicular ad hoc networks[C]//Proc of vehicular networking conference. Amsterdam:IEEE,2011:109-115.

[13] 须　超,王新红,刘富强. 车联网网络架构与媒质接入机制研究[J]. 中兴通讯技术,2011,17(3):16-20.

[14] 鲁忠辉. 基于 VanetMobiSim_NS_2 的车载自组网的研究与仿真[D]. 武汉:武汉理工大学,2010.

一种基于V3模型的内存数据库性能分析研究

作者： 王寅峰， 王龙翔， WANG Yin-feng， WANG Long-xiang

作者单位： 王寅峰,WANG Yin-feng(深圳信息职业技术学院 软件学院，广东 深圳 518172; 北京航空航天大学深圳研究院，广东 深圳 518001; 西安交通大学 电信学院，陕西 西安 710049)， 王龙翔,WANG Long-xiang(西安交通大学 电信学院,陕西 西安,710049)

刊名： 计算机技术与发展 ISTIC

英文刊名： Computer Technology and Development

年，卷(期)： 2015(6)