文章编号:1000-5641(2014)05-0216-12

内存集群计算:交互式数据分析

黄 岚[1], 孙 珂[1], 陈晓竹[1], 周敏奇[2]
(1. 电子科技集团第三十二研究所,上海 200233;
2. 华东师范大学 数据科学与工程研究院,上海 200062)

摘要: 本文围绕大数据分类中决策数据的管理和分析进行展开.重点分析了大数据时代关于商务智能(Business Intelligence,BI)技术新的应用需求;讨论了计算机硬件和体系结构的发展为决策数据管理和分析带来的挑战和机遇;通过对新兴典型应用的分析和相关技术和系统特点的总结,说明了基于内存计算的高性能数据管理和分析技术是当前亟待解决的问题,具有广阔的应用前景.在全内存式(in-memory)数据管理环境下,网络通讯将成为整个系统的主要瓶颈.结合内存的特点(数据易失性、内存墙瓶颈),设计针对高性能服务器的无共享分布式内存系统拓扑结构;研究面向异构、多层次缓存和内存结构的分布式数据布局与索引策略,跨核、跨处理器、跨服务器的多粒度并行处理框架,缓存感知、内存感知的分布式数据一致性维护等关键技术,轻量级面向按列存储的数据压缩机制及压缩感知的数据处理机制,将是基于内存计算的高性能数据管理与分析技术的重点研究内容,并将最终实现实时交互式分析处理.

关键词: 内存数据管理; 缓存感知; 迭代式处理
中图分类号: TP303 **文献标识码**: A **DOI**:10.3969/j.issn.1000-5641.2014.05.019

In-memory cluster computing: Interactive data analysis

HUANG Lan[1], SUN Ke[1], CHEN Xiao-zhu[1], ZHOU Min-qi[2]
(1. *No.32 Institute, China Electronics Technology Group Corporation, Shanghai* 200233, *China*;
2. *Data Science & Engineering Institute, East China Normal University, Shanghai* 200062, *China*)

Abstract: This paper discussed the management and analysis over data for decision support, which is defined as one of the three categories of big data. In this big data era, business intelligence creates tremendous large market values, while the enhancement in the computer hardware further stimulate the emergence of new data analysis applications, which require interactive data analysis. Based on the detailed analysis of the typical applications, we find that the in-memory cluster computing system will be the future trends for interactive data analysis. In the environment of in-memory cluster computing systems, the network communication has become the main bottleneck when comparing to memory data access and disk I/Os. Hence, the further research topics within the in-memory cluster computing aspects, including the system topology of the distributed shared-nothing in-memory computing systems when considering the characteristics of

收稿日期:2014-06
基金项目:国家自然科学重点项目(61332006)
第一作者:黄岚,女,硕士,研究方向为刀片服务器设计. E-mail: huang.moomoo@gmail.com.
通信作者:周敏奇,男,博士,研究方向为内存数据库系统. E-mail:mqzhou@sei.ecnu.edu.cn.

memory (e.g., volatility, memory wall) as well as communication bottleneck, the data placement and index strategies for isomerism, multi-level cache, the parallel computing framework of multi-granularity over multi-core, multi-processor and multi-computer, the data consistency of the distributed data management, data compression and process mechanism over the column-wise data storage.

Key words: in-memory data management; cache sensitive; iterative data processing

0 引　　言

大数据(big data)处理目前已经成为时髦词汇和研究热点.按照应用的类型,大数据大致可以分为三大类:WEB 数据、决策数据和科学数据.以谷歌、亚马逊为代表的互联网公司由于其独特的技术需求和成功的商业应用模式直接催生了云计算和大数据的概念.搜索引擎及其他互联网信息服务所面临的大数据问题是典型的 WEB 大数据问题;被称为科学研究第四范式,数据密集型科学发现要处理的数据属于科学数据;用于政府部门和商业机构决策支持的数据则是决策数据.这三类数据的应用环境和应用需求各有特点,相关的技术研发和应用部署构成了当前大数据研发和应用的亮丽风景.相比而言,决策数据这类大数据更多地沿袭了传统数据管理的技术和理念.但由于应用环境和硬件技术的发展,相关研究面临新的挑战和机遇.众所周知的决策数据应用就是耳熟能详的商务智能(Business Intelligence, BI).据 Gartner 统计,2010 年全球 BI 市场已达 570 亿美元,并预计 2014 年将达 810 亿美元[57].而 BI 是内存计算系统的一个非常重要的应用.

2011 年,SAP 公司推出"更强劲、更简捷、更睿智"的 HANA 内存计算(In-memory)系统,引起了广泛关注.该内存计算系统具有两大特点:(1)与传统数据仓库相比,具有更高的分析灵活性(flexibility),能够胜任各种即兴(ad hoc)的分析任务,这是传统数据仓库无法做的;(2)与传统的基于磁盘的数据分析系统项目相比,具有更高的处理性能.内存计算系统的主要特点是基于内存构建分布式系统,以内存计算方式存储并处理所有数据.内存计算系统主要依据内存的数据存取效率比磁盘的数据存取效率高 200 倍左右,进而通过内存计算可以大幅提升系统的数据处理能力,特别地,当数据访问量越大时,其性能提升越明显.内存计算系统正是由于上述两大特点,从而能够较好地满足目前或者未来 BI 对于海量数据的分析处理需求.

当下,内存计算在研究和系统开发方面已经成为极度热门的热点.然而,内存计算的研究已经有 50 多年的历史.内存计算数据管理系统的发展不仅与具体的数据管理应用的发展和需求息息相关,还与计算机硬件的发展密不可分;计算机组件、体系架构、拓扑结构的发展、演化推动了分布式内存数据管理系统的发展.内存数据管理系统的发展,大致可以分成四个阶段:共享内存多处理机、分布式共享内存、集中式混存优化、分布式全内存数据管理系统.

本文通过分析内存数据管理的发展历程与应用、计算机硬件发展历程之间的关联关系,论证网络通信将成为未来内存集群计算系统的新瓶颈,需要从系统的拓扑结构,数据部署,多核、多处理器、集群等多粒度并行处理等方面研究并设计新系统,最后实现交互式分析处理的功能.

1 内存数据管理发展历程

基于内存计算的数据管理和分析技术的发展与计算机硬件的发展历程密切相关,它大

致可以分成如下四个阶段：

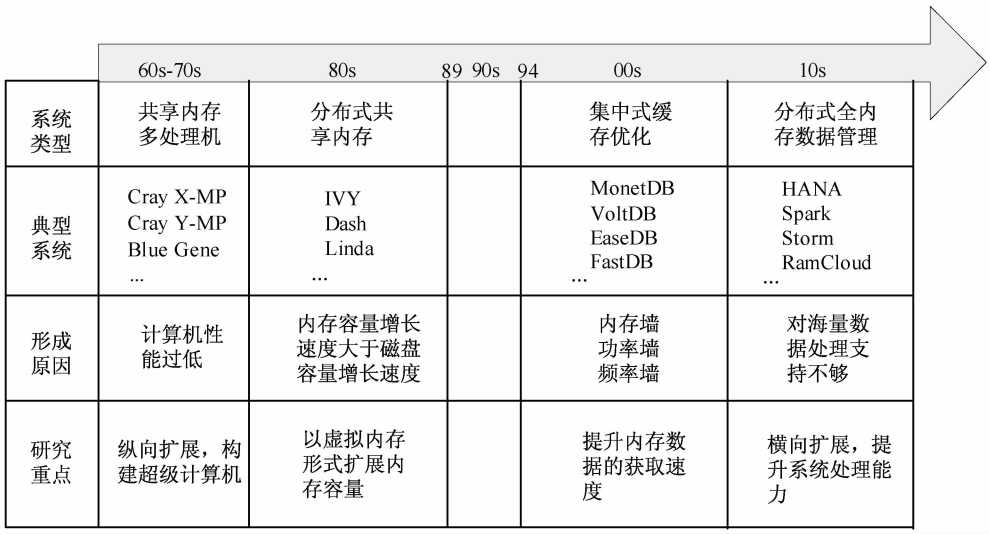


图 1 内存数据管理的发展历程

Fig. 1 History of in-memory data management

1.1 共享内存多处理机

上世纪 50 年代，随着第二代计算机(晶体管计算机)的出现，大幅度促进了计算机的发展，但是当时计算机的性能与需求之间存在着巨大的差距. 于是，构建大型计算机(mainframe)成为了当时的一种热潮，并主要采用通过增加处理器数量，扩大内存容量，提升计算机的整体处理能力.

2.2 分布式共享内存系统

20 世纪 80 年代，涌现出大量的关于分布式共享内存系统的研究，假设的前提是内存容量会赶上超过磁盘的容量. 系统大致可以分成三类：(1)多处理器共享内存系统，如 Encore 的 Multimax，SGI 的 Iris 等，这些系统的处理器数量受到共享总线带宽的限制，系统可扩展性较低；2)分布式共享内存系统，这类系统主要通过如下三种方式实现对共享内存中不同粒度、不同结构数据的访问：通过硬件实现类似于虚拟内存式的分布式内存共享，如 ivy[9]，Mirage[10]，Dash[11]等；通过操作系统实现类似于虚拟内存式的分布式内存共享，如 Mermaid 等；通过编译器利用消息传递方式实现分布式内存共享，如 Munin[12]等；(3)主存数据库，这类系统考虑到内存容量有限性，将频繁访问的数据构建主存数据库(MMDB)，非频繁问的数据构建磁盘数据库，或者根据不同的访问要求将数据分成更多的存储层次[13]，并实现不同层次数据库的统一访问，即联邦数据库[14].

1.3 集中式缓存优化系统

自 Bill Wulf 和 Sally McKee 于 1994 年提出内存墙[15]问题之后，涌现出大量的关于集中式内存数据管理方面的研究. 内存墙是指处理器性能与内存访问效率之间的差距越来越大(CPU 频率每年增长 60%，而内存访问速度每年增长 10%[32])，以及随后显现的内存访问延迟、访问带宽、应用数量可扩展性、缓存可扩展性也逐步成为性能瓶颈. 为了缓解内存墙的影响，针对缓存层次结构[16]，出现了以缓存为中心的数据管理技术方面的大量研究，并由此构建了大量的主存数据库系统，如 MonetDB[33]，EaseDB[33]，FastDB[35]，TimesTen[36](后两

者未实现缓存优化).这类研究主要可以分为两类,即缓存感知技术和缓存参数无关技术.缓存感知技术需要知晓系统缓存的结构、缓存容量、缓存位宽等系统参数,并通过分块[17],缓冲[18,19],分区[20,17]等技术提高缓存数据的时间局部性;通过压缩[21]、聚类[22]等技术实现缓存数据的空间局部性;由此可大幅提升索引结构[23]、表连接操作[24]、数据库架构[20]等的性能.这类技术的系统性能度量标准主要采用外部内存模型(external memory model,也称为缓存感知模型)[25].缓存参数无关技术则是在无需了解系统缓存结构、缓存容量等系统参数的情况下,通过如下两类方法提升系统系能,即分而治之方法和分摊算法.分而治之方法主要通过递归聚类[26]实现空间局部性和递归分区[27]实现时间局部性,并可大幅提高缓存无关 B+树[29,30]等的性能.这类技术的系统性能度量主要采用缓存参数无关模型[28].分摊算法则是针对一组操作进行优化,使得这一组操作的平均开销比较低[26,27].实验证实即使高度优化后的缓存参数无关算法的性能依旧远低于缓存感知算法的性能[31].

1.4　分布式全内存数据管理系统

内存的容量大幅增长,特别是分布式集群环境下内存容量已经能够满足数据库事务处理和数据分析的需求;2006 年之后,出现了面向事务处理和数据分析的两大类内存数据管理系统.面向事务处理的内存数据管理系统主要通过全内存操作、去除事务处理中断、内嵌高可用特性及灾难恢复机制等方式实现高吞吐量的联机事务处理,这类系统包括 H-store[37],VoltDB[38],HANA[39].而现有的面向数据分析的内存数据管理系统又可以进一步分为数据存储系统和分布式执行框架.Memchached[40]及其扩展 Membase(或 Couchbase)[41]是最早实现的全内存式数据存取系统,该类系统使用 DHT 实现网络拓扑的构建以及数据的布局及查询,具有较高的可扩展性.RamCloud[42]采用联邦方式实现了内存数据的并发控制、事务处理、一直维护及多租户特性(multi-tenancy),并通过日志结构文件系统[43]实现了内存数据的快速恢复[44].Piccolo[46]提供了分布式内存互斥表的存储架构,而 RDD (Resilient Distributed Datasets)[45]提出了针对迭代式计算的分布式数据集抽象,并同时实现了批量式和细粒度的读操作、写操作、恢复操作等.分布式处理框架主要可以分成两大类,即无环数据流式处理和有环迭代式处理.DryadLINQ[47],Pig[48],FlumeJava[49],Storm[50],Ciel[51]实现了类似于 MapReduce 式的无环数据流式处理,而 Twister[52]、HaLoop[53],Pregel[54],Spark[56]则提供了迭代式数据处理框架.但是,这类系统均没有考虑到内存墙问题,即均没有考虑分布式环境下,结合异构的缓存、内存特性提升系统的处理性能.

综上所述,几十年来内存数据管理技术的发展与应用的需求、计算机硬件的发展密不可分.内存数据管理技术的发展与演化,其主要目的是要解决海量数据处理、复杂数据分析与当时硬件内存容量的有限性、硬件处理瓶颈之间的矛盾关系,进而依次采用了共享内存多处理器,分布式共享内存处理系统,以提升数据存储容量和处理性能.然后当数据存储媒介的发展技术发生较大变革之后,即硬盘容量和内存容量的发展速度从 1989 年之后开始倒挂,导致内存数据管理技术研究方向的重大转折,即重回以磁盘为主要存储媒介的数据管理技术.然而,1994 年内存墙的提出,则是进一步促进了集中式内存数据管理技术的发展.然而,当前大数据环境下(特别是商务智能),又进一步为基于内存计算的分布式数据管理系统提出了迫切需求.

下面从当前的大数据分析需求和计算机硬件发展(特别是内存)两方面展开研究,最后确定内存计算的主要研究内容(分布式环境下,异构的缓存、内存系统体系架构,内存数据易

溃等特性)并以实现海量数据的实时交互式分析为最终目标.

2 实时交互式数据分析

几十年来,数据分析领域的目标从未改变过,即实现对大规模数据的实时交互式分析.随着计算机硬件技术的发展,以及数据管理技术在商务智能领域的应用越来越广泛,并形成了巨大商业价值.特别是内存数据库系统时代的来临,使得实时交互式分析成为了可能,进一步拓展了数据分析的领域,使得某些原先无法实现的数据分析任务成为了可能.

2.1 巨大的 BI 市场和新的应用需求

众所周知的决策数据应用就是耳熟能详的 BI.首先是新兴实时分析应用的出现.新型设备,如智能电网中的实时电表[58],供应链中物流的 RFID[59] 等的出现必将产生新的应用需求.譬如,通过每家每户的实时电表数据分析当前用电量情况,精确调整各地发电厂的供电量;通过 RFID 芯片实时跟踪并追溯各类商品,优化商品供应.类似的应用均需以数据流方式处理大量数据,并实时获得分析结果,进而优化各类服务.其次是用户对实时分析的要求.这要求数据分析由以前的批处理式分析(如 Hadoop 处理框架)向实时交互式分析(interactive analysis)转变.所谓实时交互式分析是指,用户期待零等待的分析响应时间.针对这样的应用需求,为支持在线事务处理、交互式分析,近实时挖掘,有必要对传统的数据管理架构进行重新审视[2].另外的是针对操作型数据直接进行复杂、即席的分析型应用需求.传统数据集市、数据仓库等均针对预先定义的分析服务类型,抽取、转换、加载数据,最终实现相关分析;周期性检查操作数据存储中的新增数据,优化分析结果,因而无法满足实时、即席的复杂分析型服务.再一个变化就是数据存储模式的变化.存储由 OLAP 方式向按列存储模式转变.数据量激增之后,传统的在线分析的局限性越来越明显,如数据存取性能下降,连接处理复杂化等.最后值得一提的是自助式商务智能(self-service BI)的提出.传统 BI 服务需要专业计算机人员通过复杂的 SQL 语言实现,自助式 BI 需要提供更为简单易用的分析语言,以支持非专业人员的分析应用需求.

2.2 潜在的实时交互式数据分析应用

随着内存数据库时代的来临,实时交互式分析拓展了 BI 的应用领域,主要体现在提升传统 BI 数据分析时效性,由新硬件设备的引入而带来的新型 BI 服务.这类技术和系统可能的应用主要包括:

- 传统的 BI 分析与决策支持.不论是不同地域还是不同行业的企业,也不论是小型公司还是跨国企业,均产生了大量与货币、交易、监管、服务等相关的数据,并需实时管理、分析、处理,以辅助他们企业规划、实时决策.例如,实时预测未来商品需求、销售订单处理、销售分析、过期销售追踪等[6].而分布式全内存数据管理系统可以大幅提升各类企业 BI 分析和实时决策的能力.

- 由新兴设备引入的新型 BI 服务.举例来说,智能电网中为每家每户安装的实时电表,电表每隔一小段时间发送一次电量消耗数据,电力公司分析上述数据,可以实时优化各地发电厂的供电量,提供电子耗电账单,优化每家每户的用电设备(如监测到住户外出时,提供自动断电服务等),而用户则可以实时查看用电情况.供应链管理中通过为物流商品部署 RFID 后,则可以实时跟踪并追溯商品的物流状态,销售状态,库存情况等,优化商品供应.

• 面向分析的自动订货决策.大型购物广场通过分析历史销售数据,当前商品的流行趋势,预测未来商品的销售数量,并自动完成商品的订购,包括选择上游商品厂商,确定订购商品的型号,数量,送货时间,送货地点等.

目前,分布式内存数据管理系统受到了越来越多的关注,并已出现了一些商业系统,如 SAP 的 HANA[39]、IBM 的 solidDB[55]、Oracle 的 TimesTen[36]、Stonebraker 的 VoltDB[38]等,但是这类系统远未成熟,尚需大量的完善工作.

3　网络通信将成为内存集群计算的瓶颈

四十多年来,计算机硬件的高速发展,特别是内存容量和内存访问性能方面的提升,使得通过内存管理海量商务数据,并提供实时分析服务成为可能.计算机硬件体系架构方面遇到的功耗墙、频率墙、内存墙等问题,使得计算机体系结构发生了较大的变化.为此,有必要重新研究并设计基于内存计算的高性能数据管理与分析技术.

计算机硬件在性能、体系结构方面的演化,使得内存数据管理系统成为必然选择.40 多年来,计算机各组件(CPU,内存,磁盘,网络等)在容量和性能上有了极大的提高,特别是内存容量的大幅增长,使其具备了存储、管理海量数据的能力;而内存容量的激增又加剧了冯诺依曼瓶颈问题[7].为了缓解这一问题,人们设计了多层次缓存架构、共享缓存架构;芯片制造方面遇到的功耗墙、频率墙等问题,使处理器逐步由单核向多核再到众核发展,并出现了多核共享缓存的架构、多处理器共享内存的架构;内存墙问题[15],大容量闪存的出现,如固态盘(SSD)、相变存储器(PCM)等,又使计算机存储结构发生了较大变化.可见,计算机在体系结构和组成原理上已发生了较大变化,势必导致数据管理构架的重大变革.另一方面,多年以来针对当时计算机硬件架构设计、开发的 DBMS[1],以及对 OLAP 系统的优化、修补,已经无法适应当前计算机的存储架构,更无法充分发挥计算机的性能.为此,需要一套全新的数据管理架构,而分布式内存数据管理则是一个较好的选择,逐步实现数据管理方式由以处理器为中心到以内存为中心的转变.

3.1　非统一内存访问(NUMA)

随着处理器多核技术的发展,单台服务器拥有的能够访问内存处理核心(core)的数量越来越多.如果所有的处理核心通过共享总线的方式访问内存,那么该总线必将成为内存访问的瓶颈.于是,非统一内存访问(NUMA)的服务器体系架构应运而生.在该非统一内存访问机制下,每个处理器均有从属于自己的内存,如图 2 所示.在 NUMA 环境下,处理器与处理器之间通过 QPI(Quick Path Interconnect)总线连接,实现跨处理器的内存访问.

在 NUMA 环境下,服务器的内存容量、内存带宽得到了极大的提升.例如,一台包含有 2 个处理器的服务器,可以容纳 768 GB 内存;而一台包含有 8 个处理器的服务器的内存可以达到 3TB.如果按照 2014 年年中的内存价格计算,购买 768 GB 内存仅需 6 000 美元,而 3 TB 的内存价格仅有 24 000 美金.内存价格大幅下降和容量的大幅上升,使得内存数据库系统成为可能.NUMA 的服务器架构不仅内存容量大幅上升,同时也使得内存访问带宽有了极大的提升.内存的多通道连接方式能够大幅提升整个服务器中内存访问的带宽,内存多通道连接方式如图 3 所示.对于一台包含有两个处理器的服务器而言,如果单个处理器在其四个通道上均安装有内存,那么该台服务器能够具有的总内存访问带宽可以达到 102.4 GB/s.由此可见 NUMA 环境下,服务器可以安装大容量的内存并可获得足够的访问带宽.

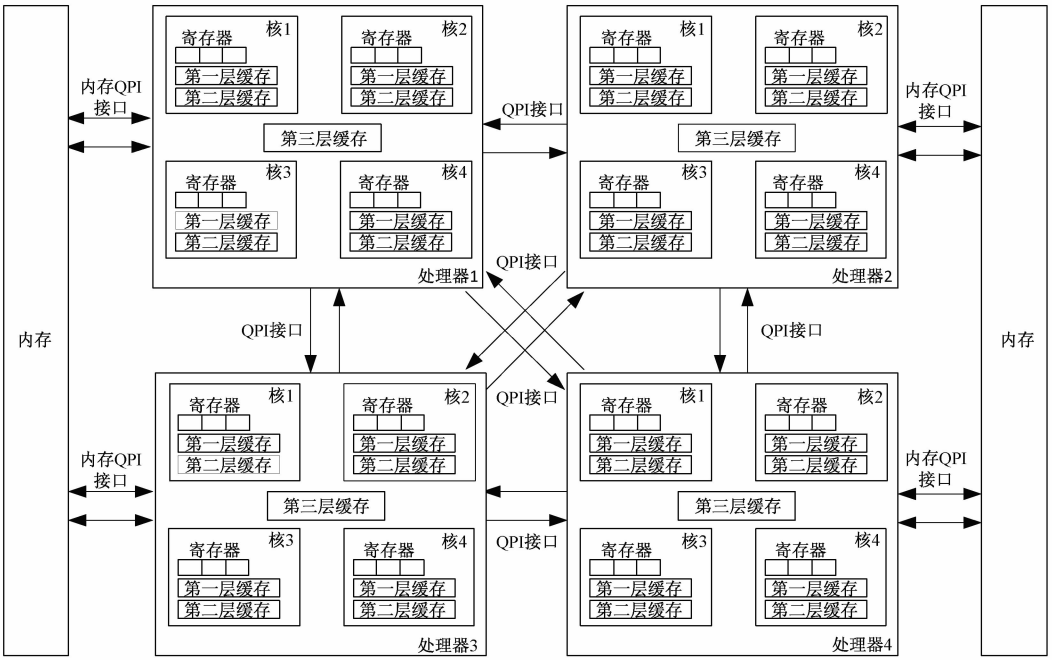


图 2 单节点处理器连接架构

Fig. 2 Architecture of the single processor

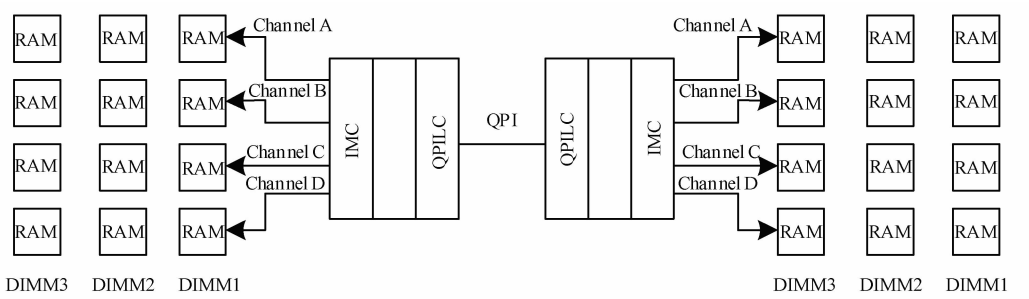


图 3 NUMA 内存访问

Fig. 3 NUMA memory access

内存容量和访问带宽均获得了极大的提升能力,但是内存访问延时方面,其性能提升的幅度有限. 尽管处理器在内存访问方面已经做了很多优化(如内存数据预取等),但是内存访问延迟依旧较大. 针对于当前包含有多级缓存的处理器架构,内存数据访问延迟可以分成三大类,即内存数据的顺序访问、随机访问、全域随机访问. 由于 cache 的存在导致访问延时的较大变化,具体内存访问延时如表 1 所示.

表 1 Intel E5－2697,2.7 Ghz,内存访问延迟

Tab. 1 Memory access latency for Intel E5－2697, 2.7 Ghz

	L1 D Cache(clk)	L2 Cache (clk)	L3 Cache (clk)	Memory(ns)	≈(clk)
顺序访问	4 clk	11 clk	14 clk	6 ns	16.2
随机访问	4 clk	11 clk	18 clk	22 ns	59.4
全随机访问	4 clk	11 clk	38 clk	65.8	178

3.2　网络访问与延迟

随着网络技术的发展，当前数据中心的网络带宽已经有了大幅的提升，截止 2014 年，大部分的数据中心已经大范围内普及万兆网，而一些 20 GB/s，40G B/s 的 InfiniBand 网络也逐步在各大数据中心使用. 在部署并使用了上述网络的环境下，其与机器之间的数据传输的带宽是足够的.

但是，按照当前计算机系统发送消息的机制，待发送的消息需要在操作系统、以太网卡的缓存中进行缓存并进行转发，同时待发送的网络包在经过交换机的时候需要缓存、寻址、转发等操作，致使网络数据包在经过网络传输的时候存在较大的延迟，如图 4 所示. 据统计不同种类的网络延迟如表 2 所示.

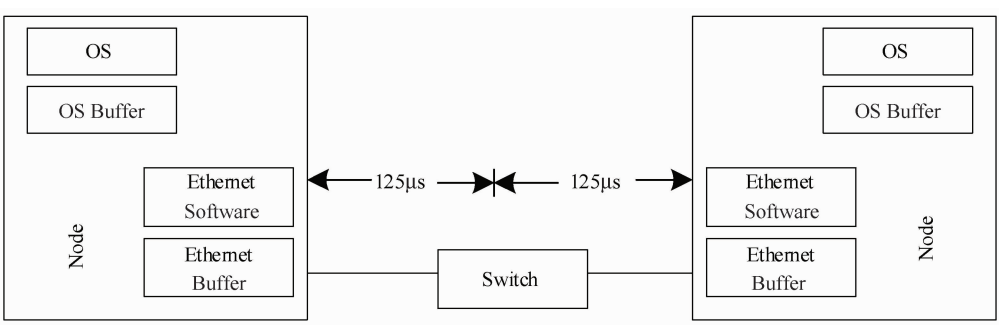


图 4　数据包在计算机间的传输

Fig. 4　Package transmission between nodes

表 2　各类网络延迟

Tab. 2　Latency of networks

	带宽/GB	延迟/μs
1 GB Ethernet	1	50－125
10 GB Ethernet	10	5－50
RDMA，RoCEE，iWARP	10	3－5
InfiniBand	40	2－3

3.3　通信瓶颈

按照当前的计算机硬件配置内存集群计算系统，可以通过如下方法计算整个系统的瓶颈，在获悉整个系统的瓶颈之后，即可进一步改良算法以提升整个内存集群系统的性能. 然而，整个内存集群计算系统瓶颈的获悉，主要可以通过分析整个系统 I/O 性能获得. 以某一内存集群计算系统为例，该集群中单台服务器包含有 2 个 Intel E5－2697 处理器，24 条 32 GB 的内存条，10 块 7 200 转的磁盘，各台机器通过 10 GB 的以太网互联，针对上述内存集群计算系统计算处理器处理不同媒介 I/O（如内存、磁盘、网络）时的效率. 通过计算可以获得，处理器处理内存中的每比特位的数据平均需要 4 个时钟周期，处理磁盘中的每比特位数据需要 27 个时钟周期，而然处理网络中另外一台机器内存中的每比特位数据需要 13 个时钟周期. 由此可以推出，在内存集群计算环境下，网络通信将成为整个系统的瓶颈（见图 5）.

综上所述，网络通讯是整个内存集群计算系统的瓶颈，上层的商务智能分析系统需要着重针对该系统瓶颈进行优化.

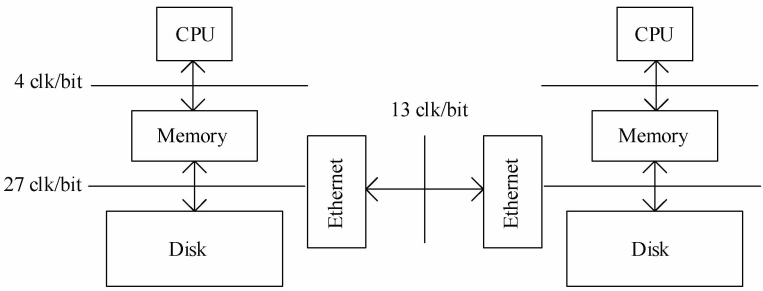


图 5 系统的瓶颈(2 * Intel E5 - 2697 处理器[2.7 Ghz,12 核,24 线程],
24 * 32 GB RAM[1600],10 * 72 k HDD,10 GB 以太网)

Fig. 5 System bottleneck (2 * Intel E5 - 2697 Processor [2.7 Ghz,12 Cores,24 Threads],
24 * 32 GB RAM[1600],10 * 72 k HDD,10 GB Ethernet)

5 内存数据库系统值得探索方向

根据上述对内存集群系统的分析,在如下方面仍需较多的研究投入,如可以针对目前的硬件体系架构,以及用户对数据分析的需求,构建分布式全内存数据管理系统,重点研究分布式系统的拓扑结构、数据存储管理策略、分布式并行调度模式等.

(1) 针对高性能、高可靠服务器的无共享分布式全内存系统拓扑结构 针对分布式系统所采用服务器的高性能、高可靠性,以及内存数据的易失性,研究并确定分布式系统的总体架构(如主从架构、基于 DHT 的分布式架构等),并以实现整体分布式系统的处理高效性、可靠性、可扩展性、可容错性为主要研究目标.

(2) 面向异构和多层次缓存、内存体系结构的分布式数据布局与索引策略 为了缓解内存数据访问效率与处理器频率之间的不匹配问题(即内存墙),以及处理器高功耗问题(即功耗墙),目前服务器大多采用多处理器、多核、多层缓存架构(如核内独立第一、二层缓存,核间共享第三层缓存),并通过 QPI(quick path interconnect)[8] 连接内存与各处理器,实现非统一存储器存取(non-uniform memory access)的共享内存架构;而且服务器之间存在着缓存层次、缓存容量、缓存介质、缓存对齐模式等不统一、不一致的情况.针对上述共享缓存、共享内存架构、异构缓存内存等特性,设计全新的分布内存数据组织结构,考虑单服务器内缓存感知的数据结构对齐(align)、填补(padding)方法,以避免对不同层次结构中高速缓存行的并发访问;提出全新的内存数据布局策略、预取策略、数据访问策略,以提升缓存的命中率针对内存墙瓶颈;考虑服务器间或缓存参数无关数据备份策略、数据调度方法;定义全新的内存数据访问性能的度量标准(如缓存未命中次数替代传统的磁盘 I/O 次数).针对内存访问随机访问的特性(随机访问复杂度为 O(1)),设计全新的缓存常驻索引策略,以提升数据检索效率.在新的硬件环境下,先前所有针对磁盘特性设计的数据布局、索引策略均不再适用,需要针对内存、缓存设计全新的算法.

(3) 跨核、跨处理器、跨服务器的多粒度并行处理框架 分布式全内存数据管理系统是共享缓存、共享内存、无共享系统的复合系统,结合各类系统的优缺点,以及应用服务的复杂度、所需的时效性,设计有效的并行处理策略,提供应用与应用之间、应用内部、操作与操作之间、操作内部的并行策略,以提升系统的并发度,提高内存数据的局部性(节点局部性、空

间局部性、时间局部性），降低网络信息交互，提升内存数据访问性能；设计基于流水线式内存化并行处理（如 MapReduc 等）框架；基于迭代式的内存化并行处理框架（暂无相应系统），以满足各类网页数据、商务智能数据、科学数据的处理需求.

（4）缓存感知、内存感知的分布式数据一致性维护　内存数据具有易失性，服务器断电或者崩溃时，数据将丢失；研究内存数据的备份策略、数据更新的日志策略、无共享系统环境下备份的并发更新，以保持数据备份与备份之间的按需一致性、数据的容错性、数据的可恢复性等. 研究共享缓存、共享内存环境下，缓存、内存中共享数据操作的并发可见性，即缓存、内存数据操作一致性（coherence）. 研究合理的数据一致性维护策略（如加锁、加时间戳等）实现系统细粒度、可扩展的并发控制.

（5）轻量级数据压缩机制及压缩感知数据处理　考虑到内存容量的有限性，以及内存数据与 CPU 处理频率之间的不平衡性，通过有效的数据压缩技术，可以提高内存的使用效率，减少 CPU 的等待时间，增加 QPI 带宽的利用率. 研究通过轻量级压缩技术，如共有值压缩、行程编码、聚集编码、间接编码等压缩技术，而非重量级编码技术如 Lempel-Ziv 编码等，并结合内存、缓存的结构特性实现对按列存储的结构化数据或非结构化数据的压缩处理，提高数据处理性能. 研究压缩感知的查询执行技术、压缩感知的数据分析技术等，以进一步提升数据处理时效性.

5　总　　结

商务智能不断对实时交互式数据分析提出了更高的需求，而计算机硬件的发展，特别是内存技术的发展（如内存容量、NUMA 内存架构等）使得内存集群计算已经成为可能并且是未来发展的主要趋势. 同时内存集群计算为实时交互式数据处理提供了硬件基础. 通过仔细分析当前内存集群计算系统的瓶颈问题，推出网络通信是该类系统的主要性能瓶颈. 为了解决这一瓶颈，同时考虑到数据存储媒介的变化（即有磁盘转向内存），内存集群计算系统的网络拓扑、并行处理框架等均是为了急需重点研究的问题.

[参　考　文　献]

[1] STONEBRAKER M, CETINTEMEL U. "One size fits all": an idea whose time has come and gone[C]//Proceedings of ICDE'2005.

[2] STONEBRAKER M, MADDEN S, ABADI D, J, et al. The end of an architectural era:(it's time for a complete rewrite)[C]//Proceedings of VLDB'2007.

[3] GRAY J. A Conversation with Jim Gray[J]. Queue Storage, 2003, 1(4).

[4] THACKER C P. Improving the future by examining the past[C]//Proceedings of ISCA'10 .

[5] BONCZ P A, KERSTEN M L, MANEGOLD S. Breaking the memory wall in MonetDB[J]. Communications of the ACM, 2008, 51(12).

[6] HASSO P, ALEXANDER Z. In-memory data management: an inflection point for enterprise applicaions[M]. Springer, 2011. 11.

[7] Von Neumann Bottleneck[EB/OL]. http://c2. com/cgi/wiki? VonNeumannBottleneck,1973.

[8] INTEL. An introduction to the Intel QuickPath interconnect[EB/OL]. http://www. intel. com/ technology/quickpath/introduction. pdf,retrieved January 14th 2011.

[9] LI K, HUDAN P. Memory coherence in shared virtual memory systems[J]. ACM trans. Computer Systems, 1989, 74(4): 321-359.

[10] Fleisch B, Popek G. Mirage:a coherent distributed shared memory design[C]//Proceedings of 14th Symposym Operation system Principles, 1989: 211-223.

[11] LENOSKI D, et al. The directory-based cache coherence protocol for the Dash multiprocessor[C]//Proceedings of 17th Int Symp Computer Architecure,1990:148-159.

[12] BENNETT J K, CARTER J B, ZWAENEPOEL W. Munin: distributed shared memory based on type-specific memory coherence[C]//Proceedings of principles and practice of parallel programming,1990.

[13] STONEBRAKER M. Managing persistent objects in a multi-level store[C]//Proceedings of SIGMOD,1991:2-11.

[14] SHENTH A P, LARSON J A. Federated database systems for managing distributed, heterogeneous, and autonomous databases[J]. Journal ACM Computing Surveys, 1990, 22(3).

[15] WUIF W A, MCKEE S A. Hitting the memory wall: implications of the obvious[J]. ACM SIGARCH Computer Architecture News, 1994, 23(1): 20-24.

[16] HENNESSY J L, PATTERSON D A. Computer architecture: a quantitative approach[M]. Morgan Kaufmann Pub,2002.

[17] SHATDAL A, KANT C, NAUGHTON J F. Cache conscious algorithms for relational query processing[C]//Proceedings of VLDB'1994, 510-521.

[18] HE B, LUO Q, CHOI B. Cache-conscious automata for XML filtering[J]. TKDE,2006, 18(12): 1629-1644.

[19] ZHOU J, ROSS K A. Buffering accesses to memory-resident index structures[C]//Proceedings of vldb'2003.

[20] BONCZ P, MANGEGOLD S, KERSTEN M. Database architecture optimized for the new bottleneck: Memory access[C]//Proceedings of VLDB'1999.

[21] BOHANNON P, MCLLROY P, RASTOGI R. Main-memory index structures with fixed-size partial keys[J]. SIGMOD Record, 2001, 20(2).

[22] CHILIMBI T M, HILL M D, LARUS J R. Cache-conscious structure layout[C]//Proceedings of SIGPLAN'1999.

[23] bOHANNON P, MCLLROY P, RASTOGI R. Main-memory index structures with fixed-size partial keys[C]//Proceedings of Sigmod Record, 2001, 30(2).

[24] CHEN S, AILAMAKI A, GIBBONS P B. Improving hash join performance through prefetching[J]. TODS, 2007, 32(3).

[25] AGGARWAL A, VITTER J, et al. The input/output complexity of sorting and related problems[J]. Communications of the ACM, 1988, 31(9).

[26] HE B, LUO Q. Cache-oblivious nested-loop joins[C]//Proceedings of CIKM'2006.

[27] HE B, LUO Q. Cache-oblivious hash join, Technical report HKUST-cs06-04,2006

[28] FRIGO M, LEISERSON C E, PROKOP H, et al. Cache-oblivious algorithms[C]//Proceedings of FOCS'1999.

[29] BENFDER M A, DEMAINE E D, Farach-Colton M. Cache-oblivious B-trees[C]//Proceedings of FOCS'2000.

[30] BENDER M A, DUAN Z, LACONO J, et al. A locality-preserving cache-oblivious dynamic dictionary [J]. Journal Algorithms, 2004, 52(2).

[31] YOTOV K, ROEDER T, PINGALI K, et al. An experimental comparison of cache-oblivious and cache-conscious programs[C]//Proceedings of SPAA'2007.

[32] AILAMAKI A. Database architectures for new hardware[C]//Proceedings of VLDB'2004.

[33] HE B, LI Y, LUO Q, et al. EaseDB: A cache-oblivious in-memory query processor[C]//Proceedings of SIGMOD'2007.

[34] BONCZ P, GRUST T, van KEULEN M. et al. MonetDB/XQuery: a fast XQuery processor powered by a relational engine[C]//Proceedings of SIGMOD'2006.

[35] FASTDB. 2002. http://www.ispras.ru/ knizhnik/fastdb.html.

[36] TIMESTEN. 2006. http://www.oracle.com/timesten/index.html.

[37] KALLMAN R, KIMURA H, NATKINS J, et al. H-store: a high-performance, distributed main memory transaction processing system[C]//Proceedings of VLDB'2008.

[38] VoltDB，2009，http：// voltdb. com.

[39] HANA，2012，http：//www. sap. com/solutions/technology/in-memory-computing-platform/hana/overview/index. epx.

[40] Memcached-a distributed memory object caching system，http：//memcached. org，2003.

[41] Membase，http：//www. couchbase. com/membase.

[42] OUSTERHOUT J，AGRAWAL P，ERICKSON D. et al. The case for RAMClouds：scalable high-performance storage entirely in DRAM[J]. ACM SIGOPS operation systems review，2010，43(4).

[43] ROSENBLUM M，OUSTERHOUT J K. The design and implementation of a log-structured file system[J]. ACM transactions on computer systems，1992，10(1).

[44] ONGARO D，RUMBLE S M，STUTSMAN R. et al. Fast crash recovery in RAMCloud[C]//Proceedings of SOSP'2011.

[45] Resilient distributed datasets：A fault-tolerant abstraction for in-memory cluster computing，Resilient distributed datasets：A fault-tolerant abstraction for in-memory cluster computing，Technical Report UCB/EECS-2011-82，EECS Department，University of California，Berkeley，2011.

[46] POWER R，LI J. Piccolo：building fast，distributed programs with partitioned tables[C]//Proc. OSDI 2010，2010.

[47] YU Y，ISARD M，FETTERLY D，et al. Currey. DryadLINQ：A system for general-purpose distributed data-parallel computing using a high-level language[C]//OSDI'08，2008.

[48] OLSTON C，REED B，SRIVASTAVA U. et al. Tomkins. Pig latin：a not-so-foreign language for data processing[C]//Proceedings of SIGMOD '08，1099-1110.

[49] CHAMBERS C，RANIWALA A，PERRY F. et al. Flumejava：easy，ef? cient data-parallel pipelines[C]//Proceedings of the 2010 ACM SIGPLAN conference on Programming language design and implementation，PLDI'10. ACM，2010.

[50] Storm，https：//github. com/nathanmarz/storm.

[51] MURRAY D G，SCHWARZKOPF M，SMOWTON C，et al. Hand. Ciel：a universal execution engine for distributed data-? ow computing. In NSDI，2011.

[52] EKANAYAKE J，LI H，ZHANG B，et al. Twister：a runtime for iterative mapreduce[C]//HPDC'10，2010.

[53] BU Y，HOWE B，BALAZINSKA M，et al. Ernst. HaLoop：efficient iterative data processing on large clusters. Proc. VLDB Endow.，3：285-296，September 2010.

[54] MALEWICZ G，AUSTERN M H，BIK A J C，et al. Pregel：a system for large-scale graph processing[C]//Proceedings of SIGMOD'2010.

[55] SolidDB，http：//www-01. ibm. com/software/data/soliddb/

[56] ZAHARIA M，CHOWDHURY M，FRANKLIN M J，et al. Spark：cluster computing with working sets[C]//Proceedings of the 2nd USENIX conference on Hot topics in cloud computing，2010.

[57] Garter. Gartner Taps Predictive Analytics as Next Big Business Intelligence Trend，http：//www. enterpriseappstoday. com/business-intelligence/gartner-taps-predictive-analytics-as-next-big-business-intelligence-trend. html，2012. 4.

[58] SCHAPRANOW M P，KOHNE R. ZEIER A. Enabling real-time charging for smart grid infrastructures using in-memory databases[C]//1st IEEE LCN Workshop on Smart Grid Networking Infrastructure，2010.

[59] SCHAPRANOW M P，ZEIER A，Plattner H. A formal model for enabling RFID in pharmaceutical supply chains [C]//44th Hawaii International Conference on System Sciences，2011.

（责任编辑　林磊）