ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.10, October 2008, pp.2573−2584
DOI: 10.3724/SP.J.1001.2008.02573

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

内存数据库在 TPC-H 负载下的处理器性能*

刘大为[1,2+], 栾 华[1], 王 珊[1], 覃 飙[1]

[1](数据工程与知识工程教育部重点实验室(中国人民大学),北京 100872)

[2](中国石油信息技术服务中心,北京 100724)

Main Memory Database TPC-H Workload Characterization on Modern Processor

LIU Da-Wei[1,2+], LUAN Hua[1], WANG Shan[1], QIN Biao[1]

[1](Key Laboratory of Data Engineering and Knowledge Engineering (Renmin University of China), Ministry of Education, Beijing 100872, China)

[2](China Petroleum Information Technology Service Center, Beijing 100724, China)

+ Corresponding author: E-mail: liudawei@ruc.edu.cn

Liu DW, Luan H, Wang S, Qin B. Main memory database TPC-H workload characterization on modern processor. *Journal of Software*, 2008,19(10):2573−2584. http://www.jos.org.cn/1000-9825/19/2573.htm

Abstract: In 1999, the research of database systems' execution time breakdown on modern computer platforms has been analyzed by Ailamaki, *et al*. The primary motivation of these studies is to improve the performance of Disk Resident Databases (DRDBs), which form the main stream of database systems until now. The typical benchmark used in those studies is TPC-C. However, continuing hardware advancements have "moved-up" on the memory hierarchy, such as the larger and larger on-chip and off-chip caches, the steadily increasing RAM space, and the commercial availability of huge flash memory (solid-state disk) on top of regular disk, etc. To reflect such a trend, the target of workload characterization research along the memory hierarchy is also studied. This paper focuses on Main Memory Databases (MMDBs), and the TPC-H benchmark. Unlike the performance of DRDB which is I/O bound and may be optimized by high-level mechanisms such as indexing, the performance of MMDB is basically CPU and memory bound. In this study, the paper first compares the execution time breakdown of DRDB and MMDB, and the paper proposes an optimize strategy to optimize the memory resident aggregate. Then, the paper explores the difference between column-oriented and row-oriented storage models in CPU and cache utilization. Furthermore, the paper measures performance of MMDBs on different generational CPUs. In addition, the paper analyzes the index influence and gives a strategy for main memory database index optimization. Finally, the paper analyzes each query in the full TPC-H benchmark in detail, and obtains systematic results, which help design micro-benchmarks for further analysis of CPU cache stall. Results of this study are expected to benefit the performance optimization of MMDBs, and the architecture design memory-oriented databases of the next generation.

Key words: MMDB (main memory database); TPC-H workload; processor characterization

摘 要: Ailamaki 等人 1999 年研究了数据库管理系统(database management system,简称 DBMS)在处理器上的时

* Supported by the National Natural Science Foundation of China under Grant Nos.60496325, 60473069, 60503038 (国家自然科学基金); the Grant from HP Labs. China (国际合作(HP Lab.)项目)

Received 2007-07-20; Accepted 2008-01-29

间开销分解.此后,相关研究集中在分析 DBMS 在处理器上的瓶颈.但这些研究工作均是在磁盘数据库 DRDBs(disk resident databases)上开展的,而且都是分析DBMS上的TPC-C类负载.然而,随着硬件技术的进步,现代计算机的多级缓存结构(memory hierarchy)在逐渐地"上移".例如,容量越来越大的芯片内缓存(on-chip caches)和芯片外缓存(off-chip caches),容量越来越大的RAM,Flash Memory 等等.为此,处理器负载分析的研究工作也应随之"上移".研究内存数据 MMDBs(main memory resident databases)在计算密集型负载下的处理器行为特性.由于磁盘数据库的主要性能瓶颈是磁盘 I/O,因而可以用索引、压缩等技术进行优化;然而,内存数据库的性能瓶颈却在于处理器和内存之间的数据交换.针对这一问题,首先分析了磁盘数据库和内存数据库在 TPC-H 负载下处理器性能瓶颈的差异,并给出了一些优化建议,提出了通过预取的优化方法.其次,通过实验比较了不同存储体系结构(行存储与列存储)对处理器利用率的差异,并探索了下一代内存数据库体系结构方面的解决方案.此外,还研究了索引结构对处理器多级缓存的影响,并给出了索引的优化建议.最后,提出一个微测试集用于评估内存数据库在 DSS(decision support system)负载下处理器的性能及行为特性.研究结果会对运行于下一代处理器上的内存数据库体系结构设计和性能优化提供一定的实验依据.

关键词: 内存数据库;TPC-H 负载;处理器特性

中图法分类号: TP311 **文献标识码**: A

数据库管理系统(database management system,简称 DBMS)在现代处理器上通常表现出较低的IPC(instrucions-per-cycle),尤其是在 DSS 这种计算密集型的负载下,如 OLAP(online analytical processing)、多媒体检索、数据挖掘等应用[1].这表明,在这类应用下,DBMS 对处理器的利用率较低.目前,从事计算机体系结构和数据库研究的人员对此问题都给予了重视.从已发表的文献来看,多数研究集中在 DBMS 在各种负载下对处理器利用情况的分析,如文献[1–7].其中,文献[2]的研究结果得到了广泛的认可,Ailamaki 等人在 1999 年分析了几个商用数据库管理系统在处理器上的时间开销,并从处理器的角度将查询执行时间分解为:处理器的有效计算时间、处理器各级缓存产生的延迟(stall)时间、分支指令预测失误而产生的延迟时间以及因资源不可用而产生的延迟时间等几部分.其研究结果表明,处理器在 DBMS 的应用负载下,大部分时间(近 50%)处于停滞(stall)状态.其中,处理器的第1级指令缓存缺失(L_1 instruction cache miss)和第2级数据缓存缺失(L_2 data cache miss)是引起停滞的最主要的原因.其他文献,如文献[3–7]也进行了相关的研究工作.但这些研究均是针对当时的主流数据库,即磁盘数据库 DRDB(disk resident databases)开展的,其目标是研究 DRDB 在处理器上的性能瓶颈并探索相关的优化策略.在研究方法上,主要是通过统计处理器各级缓存上发生的缺失次数,从而得以精确地分析 DRDB 在多级缓存上的性能瓶颈;实验平台则是基于当时的处理器,例如 Intel Pentium,Pentium Pro 等.然而,随着处理器技术的不断进步,如现代处理器的乱序执行(out-of-order execution)、多线程(multi-threading)、多级缓存(multi-level memory hierarchies)、多核(multiple cores)等等,处理器的计算能力也在不断增强.那么,DBMS 上的计算密集型应用,如数据挖掘、多媒体信息检索、OLAP 等能否从处理器的进步中获益呢?这类负载在处理器上的性能瓶颈又是什么?DBMS 应该如何从体系结构及核心算法方面进行再设计以便充分利用现代处理器的高计算能力?这些都是值得研究的问题.

基于以上研究动机,本文研究了内存数据库 MMDB(main memory resident databases)在 TPC-H 类负载下,现代处理器的特性,如在安腾®2(Itanium® 2)上的行为特性以及性能瓶颈.之所以针对内存数据库和 TPC-H 负载进行研究,是基于如下考虑:随着内存价格的下降以及容量的不断增大,将所有的数据放入内存进行处理将成为现实;而且,64 位处理器逐渐成为主流,这极大程度地增加了计算机内存的可用容量,32 位处理器最多可用 4GB 的内存容量,而 64 位处理器的最高内存容量理论上为 17 179 869 184GB.此外,多处理器的出现使得并行计算成本也越来越低.因此,内存容量可以容纳当前绝大多数 OLAP 应用的基础数据,在不久的将来,基于内存数据库的 OLAP 将逐渐走向实际.由于数据可以直接在内存中访问,因而系统可以具有更好的响应时间和吞吐量.与磁盘数据库的 I/O 瓶颈不同,内存数据库的瓶颈在于处理器和内存之间的数据交换代价.因此,需要重新分析这类负载在处理器上的行为特性.

总之,现代计算机的多级缓存体系结构已经“上移”,表现为容量越来越大的芯片内缓存(on-chip caches)和芯片外缓存(off-chip caches)、容量越来越大的 RAM(random access memory)等,因此,负载分析的研究工作也应随之“上移”.为此,我们通过一系列实验研究分析了 TPC-H 负载下现代处理器的行为特性.在处理器方面,我们选择了安腾®2 和至强两个典型的现代处理器.就目前已发表的文献来看,对于内存数据库在处理器上的性能瓶颈研究尚未有相关的研究工作.

本文的主要贡献简要总结如下:首先,分析了磁盘数据库和内存数据库在 TPC-H 负载下的处理器性能瓶颈差异,并针对内存数据库的处理器瓶颈提出了一种基于预取的优化策略.其次,通过实验比较了不同存储体系结构(行存储与列存储)对处理器利用率的差异.此外,研究了索引结构对处理器多级缓存的影响.最后,通过实验,我们提出一个微测试集用于评价内存数据库在 DSS 负载下处理器的性能及行为特性.本文的研究结果希望能够对下一代处理器上的内存数据库体系结构设计以及性能优化提供一定的实验依据.

本文第 1 节介绍当前国际上的相关研究和进展情况.第 2 节介绍数据库查询在处理器上的执行过程以及 TPC-H 负载.第 3 节给出实验环境.第 4 节详细分析实验结果以及我们提出的优化策略.第 5 节对全文总结并讨论未来的工作方向.

1 相关工作

Shreekant 等人在文献[3]中首次研究了 DBMS 的性能与硬件的关系,该文献研究了 DBMS 在 OLTP(online transaction processing)负载下如何将多个进程分配至不同的处理器上进而提高数据库系统的性能的方法.此外,Maynard 等人在文献[4]中分析了关系数据库在 TPC-A 和 TPC-C 负载下对处理器的影响,其研究结果表明,增大处理器第 1 级缓存(level 1 cache)的容量可以提高处理 TPC-C 类的负载能力,而科学计算类的负载(TPC-A)并不能直接获得收益.Rosenblum 等人在文献[5]中的研究结果表明,虽然磁盘 I/O 是数据库的主要瓶颈,但处理器在处理 OLTP 类负载时近 50%的时间是处于停滞状态的,产生停滞的原因主要是由于处理器的缓存缺失(cache miss)造成的.在分析数据库负载对处理器利用情况的研究中,文献[2]分析了 4 种商用数据库并指出 DBMS 在 OLTP 和 OLAP 两类负载下,造成处理器“停滞”的主要原因是由于处理器的第 1 级的指令缓存的缺失(L_1 instruction cache miss)和第 2 级的数据缓存的缺失(L_2 data cache miss)而引起的.其他文献[5−7]则在多处理器平台分析了 DBMS 的系统瓶颈.其中,有些研究只针对 OLAP 类负载的特性分析,如文献[7];而有些文献[8,9]则对 OLTP 和 OLAP 两类负载都进行了研究.以上的研究表明,在数据库的两种负载中,OLAP 类负载相对于 OLTP 类负载对处理器利用率更好.此外,得出的另一结论是,由各级缓存引起的处理器“停滞”是造成处理器利用率低下的主要原因.就目前已发表的文献来看,尚未有针对内存数据库在现代处理器上的行为进行分析的文献.

2 查询处理的时间分解模型与 TPC-H 负载

2.1 查询执行在处理器上的时间分解

现代处理器的指令采用流水线方式执行.一条指令在处理器上通常划分为几个不同阶段来执行.每个阶段执行的操作之间会产生重叠,但当一个阶段的操作不能立即结束时,就将在流水线中产生延迟(stall).现代处理器为了克服这种延迟,通常采用以下几种技术:非阻塞缓冲(non-blocking caches)、乱序执行(out-of-order execution)、预测执行(speculative execution)和分支预测(branch prediction),但延迟仍不能完全被避免.因此,从处理器的角度来看,一个数据库查询的时间(T_Q)可以分解为以下几个部分:CPU 有效计算时间(T_C)、各级缓存产生的延迟时间(T_M)、分支预测失误产生的延迟时间(T_B)、资源缺失产生的延迟时间(T_R),另外以上各个部分会有重叠(T_{OVL}),因此有下面的公式成立[2].不同的处理器平台上,T_C 的具体计算方法可能会有差别,如表 1 给出了公式(1)在安腾®2 处理器平台上各个时间的详细分解:

$$T_Q = T_C + T_M + T_B + T_R - T_{OVL} \tag{1}$$

2.2 TPC-H基准测试

TPC-H 是 TPC 组织制定的用来模拟决策支持类应用的一个测试集.目前,在学术界和工业界普遍采用它来评价决策支持技术方面应用的性能.TPC-H 基准测试是由 TPC-D(由 TPC 组织于 1994 年指定的标准,用于决策支持系统方面的测试基准)发展而来的.TPC-H 用 3NF 实现了一个数据仓库,共包含 8 个基本关系,其数据量可以设定从 1G~3T 不等.TPC-H 基准测试包括 22 个查询(Q_1~Q_{22}),其主要评价指标是各个查询的响应时间,即从提交查询到结果返回所需时间.TPC-H 基准测试的度量单位是 Qph*H*@*size*,其中 *H* 表示每小时系统执行复杂查询的平均次数,*size* 表示数据库规模的大小,它能够反映出系统在处理查询时的能力.TPC-H 是根据真实的生产运行环境来建模的,这使得它可以评估一些其他测试所不能评估的关键性能参数.总而言之,TPC 组织颁布的 TPC-H 标准满足了数据仓库领域的测试需求,并且促使各个厂商以及研究机构将该项技术推向极限.

Table 1 Execution time components on Itanium® 2

表 1 安腾®2 平台上执行时间的各个组成部分

T_C			Computation time of CPU
T_M			Stall time related to memory hierarchy
	T_{L1}	T_{L1D}	Stall time due to L_1 *D*-cache misses
		T_{L1I}	Stall time due to L_1 *I*-cache misses
	T_{L2}	T_{L2D}	Stall time due to L_2 *D*-cache misses
		T_{L2I}	Stall time due to L_2 *I*-cache misses
	T_{L3}	T_{L3D}	Stall time due to L_3 *D*-cache misses
		T_{L3I}	Stall time due to L_3 *I*-cache misses
	T_{DTLB}		Stall time due to DTLB misses
	T_{ITLB}		Stall time due to ITLB misses
T_B			Branch misprediction penalty
T_R			Resource stall time

按照 TPC-H 标准,我们自行设计程序实现了 TPC-H 基准测试,用于模拟 DSS 类负载.本文的研究关注于该类负载,因为基于内存的 OLAP 将会是未来内存数据库的一种重要的应用模式.内存的 OLAP 是指这样一种技术:从基于磁盘的永久性数据存储中将数据加载到内存,在内存中对数据进行各种各样的 OLAP 分析.相对于传统的基于磁盘的 OLAP 而言,内存 OLAP 的性能特征、优化策略、性能瓶颈主要受半导体 RAM(内存,而不是磁盘)的影响.基于内存 OLAP 的市场非常广阔,可以用来支持大部分要求苛刻的业务系统,如股票交易、通信设备、金融分析、航线调度以及网上书店等.内存 OLAP 所带来的好处是,可以以更低的成本完成以前相同的工作;可以更快、更好地完成以前相同的工作;可以完成那些以前在技术上或经济上不可行的工作.目前已有少数几个厂商,如 Applix,QlikTech 和 Panoratio Database Image 开始研究内存 OLAP 产品.

3 实验环境与研究方法

3.1 硬件环境

实验的硬件服务器:其中一台是基于安腾®2 的 HP Integrity rx2620-2 服务器,处理器的主频为 1.6 GHz,共有 3 级缓存,其中第 1 级缓存的指令缓存和数据缓存集成为一体,系统内存容量为 4G;另外一台是基于至强(AMD Opteron™)的服务器,内存容量为 2GB.表 2、表 3 分别给出了这两台服务器各级缓存的详细信息.

Table 2 Itanium® 2 cache characteristics

表 2 安腾® 2 各级缓存参数

Characteristic (unit)	L_1 (split)	L_2	L_3
Cache size (KB)	16 data 16 inst.	256	3 072
Cache line size (bytes)	64	128	128
Associativity	4-way	8-way	6-way
Miss penalty (cycles)	7	16	182

Table 3 AMD Opteron™ cache characteristics

表 3 至强处理器各级缓存参数

Characteristic (unit)	L_1 (split)	L_2
Cache size (KB)	16 data 16 inst.	1 024
Cache line size (bytes)	64	64
Associativity	2-way	16-way
Miss penalty (cycles)	7	565

3.2 度量工具与方法

我们用 Calibrator[10]来测量安腾®2 和至强处理器各级缓存每次缺失产生的延迟时间以及各级

TLB(translation lookaside buffer)上每次缺失引起的延迟时间.例如,我们使用下面的命令来测试 HP Integrity rx2620-2 服务器的各级缓存及 TLB 的延迟信息.

• calibrator 1600 500M Itanium2

安腾®2 和至强处理器上都设有性能监测单元(performance monitor unit),可以用来记录处理器各类事件的发生次数.我们用 PerfSuite[11]捕获处理器的各类事件并完成统计功能.其中,PerfSuite 工具包中的 psrun 是一个命令行工具,可以收集处理器的各个性能参数,并通过一个 XML 文件指定所要收集的性能参数.例如,下面的命令可以监测应用程序 postmaster.exe 运行期间 60 余个处理器的性能参数,具体参数在文件 Itanium2.xml 中指定.其中使用"–f"参数可以将其子进程一起监测.

• srun -c Itanium_2.xml -f postmaster.exe

psrun 统计的处理器性能参数记录了不同类型处理器事件发生的次数,例如第 1 级指令缓存上发生的缺失次数(L_1 instruction cache miss).将测得的缺失次数乘以一次缺失的延迟时间即可以计算出在该级缓存上总的延迟时间.例如:

L_1 上的指令缺失产生的延迟时间=L_1 指令缓存缺失次数×单次 L_1 缺失延迟时间 (2)

根据以上计算方法,可以得到安腾®2 平台和至强平台上查询时间各组成部分的计算公式,见表 4 和表 5.

Table 4 Method of measuring each of the stall time components on Itanium® 2 platform

表 4 安腾®2 平台上各部分延迟时间计算方法

Stall time			Description	Measurement method
T_C			Computation time	Computation time of CPU
T_M	T_{L1}	T_{L1D}	Stall time due to L_1 *D*-cache misses	(misses)×7 cycles
		T_{L1I}	Stall time due to L_1 *I*-cache misses	(misses)×7 cycles
	T_{L2}	T_{L2D}	Stall time due to L_2 *D*-cache misses	(misses)×16 cycles
		T_{L2I}	Stall time due to L_2 *I*-cache misses	(misses)×16 cycles
	T_{L3}	T_{L3D}	Stall time due to L_3 *D*-cache misses	(misses)×182 cycles
		T_{L3I}	Stall time due to L_3 *I*-cache misses	(misses)×182 cycles
	T_{DTLB}		Stall time due to D_{TLB} misses	(misses)×16 cycles
	T_{ITLB}		Stall time due to I_{TLB} misses	(misses)×8 cycles
T_B			Branch misprediction penalty	(branch mispredictions retired)×17 cycles
T_R	T_{FU}		Functional unit stalls	Actual stall time
	T_{DEP}		Dependency stalls	Actual stall time
	T_{ILD}		Instruction-Length decoder stalls	Actual stall time
T_{OVL}			Overlap time	Actual stall time

Table 5 Method of measuring each of the stall time components on AMD platform

表 5 至强处理器平台上各部分延迟时间计算方法

Stall time			Description	Measurement method
T_C			Computation time	Computation time of CPU
T_M	T_{L1}	T_{L1D}	Stall time due to L_1 *D*-cache misses	(misses)×7 cycles
		T_{L1I}	Stall time due to L_1 *I*-cache misses	(misses)×7 cycles
	T_{L2}	T_{L2D}	Stall time due to L_2 *D*-cache misses	(misses)×565 cycles
		T_{L2I}	Stall time due to L_2 *I*-cache misses	(misses)×565 cycles
	T_{DTLB}		Stall time due to D_{TLB} misses	(misses)×36 cycles
	T_{ITLB}		Stall time due to I_{TLB} misses	(misses)×36 cycles
T_B			Branch misprediction penalty	(branch mispredictions retired)×17 cycles
T_R	T_{FU}		Functional unit stalls	Actual stall time
	T_{DEP}		Dependency stalls	Actual stall time
	T_{ILD}		Instruction-Length decoder stalls	Actual stall time
T_{OVL}			Overlap time	Actual stall time

4 结果分析

4.1 内存数据库的处理器特性分析

4.1.1 结果与分析

本节实验分析内存数据库在 TPC-H 负载下处理器的行为特性,选取了两种不同类型的内存数据库系统进

行分析,分别是 MonetDB[12]和 System A(一个商业化内存数据库产品,隐去实际名称).实验基于安腾®2 处理器平台进行 TPC-H 基准测试,包括 Power Test 和 Throughput Test.其中,Power Test 是由一系列查询串行构成的负载流,Throughput Test 是以并发方式执行多个查询的负载流.

根据第 2.1 节中描述的分析方法,我们分别计算出各个系统在运行 TPC-H 负载时处理器各方面时间开销所占的比例,结果如图 1 所示.为了对比内存数据库和磁盘数据库在处理器上的不同行为特性,图中还给出了磁盘数据库的实验结果(这里,磁盘数据库为 Postgre SQL[13]).

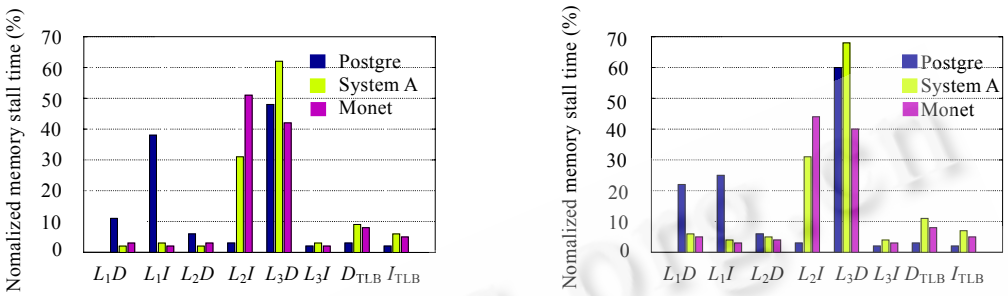


Fig.1 Different deep memory behavior of DRDB and MMDB: Power test (left), throughput test (right)

图 1 磁盘数据库与内存数据库在处理器各级缓存延迟时间比例对比:压力测试(左图)、吞吐量测试(右图)

从图 1 可以看出,两个内存数据库系统在基于安腾®2 处理器的平台上运行 TPC-H 负载时,处理器的延迟时间主要是由第 2 级指令缓存缺失(L_2 *I*-cache miss)和第 3 级数据缓存缺失(L_3 *D*-cache miss)造成的.在 Power test 下,第 2 级指令缓存缺失造成的延迟时间超过总的内存缺失延迟时间的 30%,第 3 级数据缓存缺失造成的处理器延迟时间超过 40%;在 Throughput test 下,第 2 级指令缓存缺失产生的延迟时间超过总延迟的 30%,而第 3 级数据缓存缺失产生的延迟时间超过 40%.

从这组实验结果同时可以看出,磁盘数据库在 TPC-H 负载下,处理器的主要延迟时间产生在第 1 级指令缓存缺失和第 3 级数据缓存缺失.以 PostgreSQL 为例,在 Power test 下,第 1 级指令缓存缺失产生延迟约占总延迟的 38%,第 3 级数据缓存缺失产生的延迟占到 48%;在 Throughput test 情况下,第 1 级指令缓存缺失产生的延迟约占总延迟的 25%,第 3 级指令缓存缺失产生的延迟占近 60%.磁盘数据库在基于安腾®2 处理器平台上的实验结果和 Ailamaki 等人在 1999 年在奔腾处理器(Pentium)上的实验结果是一致的,同样表现出处理器时间开销主要是由最外层(距内存最近)的数据缓存缺失和最内层(距离处理器最近)的指令缓存造成的.

以上实验结果反映了磁盘数据库和内存数据库在运行 TPC-H 这类负载时行为特性以及性能瓶颈存在较大的差异.内存数据库在基于安腾®2 处理器的系统上运行 TPC-H 负载时,由第 2 级指令缓存缺失和第 3 级数据缓存缺失所造成的延迟是构成总延迟的主要部分;而磁盘数据库在 TPC-H 负载下,处理器的主要延迟时间产生在第 1 级指令缓存缺失和第 3 级数据缓存缺失.同时,也反映出在磁盘数据库上运行 TPC-H 负载时,处理器的利用率仍然是一个瓶颈,系统未能从处理器的进步中获得很好的收益.

实验结果的启示是,在 TPC-H 这类负载下,内存数据库的优化策略设计应该考虑如何减少第 2 级指令缓存的缺失和第 3 级数据缓存的缺失.例如,在存储体系结构设计、索引结构设计以及查询优化算法设计等方面应该考虑如何有效地减少最外层数据缓存的缺失、提高缓存的性能进而提高处理器利用率.因此,通过降低缓存缺失提高处理器的利用率将是优化的目标之一.

4.1.2 优化策略研究

针对以上实验发现的内存数据库在现代处理器上的性能瓶颈,我们在优化策略方面也进行了探索.针对如何减少第 3 级的数据缓存缺失问题,我们提出了一种带有预取的聚集计算.如图 2 所示,算法的基本思想是通过预取弥补处理器计算性能和数据访问之间的差异,通过预取将要访问的数据在被访问之前提前送入缓存,从而减少因缓存访问缺失而造成的时间延迟.具体实现时,我们在聚集计算进行扫描元组的过程中插入预取代码,在

每次访问元组之前“预取”内存内容将其放入数据缓存中,从而有效地缩短了内存延迟,隐藏了内存子系统因访问缺失而造成的处理器延迟.具体实现的算法如图 3 所示.

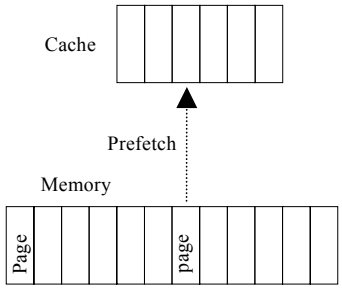


Fig.2 Prefetch memory to data cach

图 2 预取内存内容至缓存

Algorithm 1. Scan with prefetch.

```
Start
scan table;
Visit Tuple(i);
Computation;
Prefetch Tuple(i+1);
End;
```

Fig.3 Aggregate with prefetch

图 3 结合预取核心算法

我们对上述算法进行了实验评估,实验是在基于安腾®2 处理器的平台上进行的.实验中的聚集计算是 TPC-H 负载中的 Q_6,这是一个以扫描(scan)为主要基础运算的查询.实验的数据按 TPC-H 标准实现的数据产生器生成,数据量分别为 100M,200M 至 500M,而且在测试缓存性能前全部预先装入内存,消除 I/O 影响.预取指令用 Prefetch 0,可以实现将数据预取至第一、二和三级缓存.实验中主要比较了预取对第 3 级数据缓存缺失率、第 3 级缓存系统带宽以及处理器的利用率.实验结果如图 4 所示,可以看出,预取对缓存子系统性能起到了一定的优化作用,实验中第 3 级数据缓存的缺失率下降了近 3%,由于第 3 级缓存的单次缺失延迟时间和第一、第二级缓存相比非常大,因此,降低第 3 级数据缓存缺失次数对降低处理器总的延迟时间非常有效,实验中预取优化算法可以将理器延迟时间(stall time)减少近 27%.

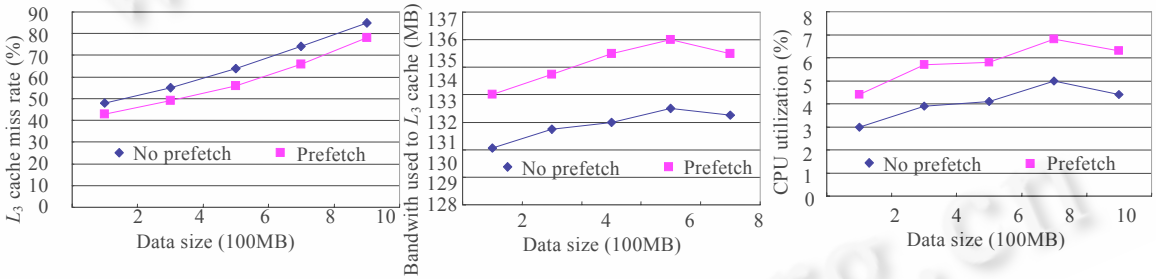


Fig.4 L_3 cache performance: L_3 D-cache miss ratio(left), L_3 cache bandwidth (middle), CPU utilization(right)

图 4 第 3 级缓存缓存性能:第 3 级数据缓存缺失率(左图)、第 3 级缓存带宽(中图)、处理器利用率(右图)

4.1.3 动态预取

基于简单访问模式的聚集计算(如顺序扫描)可以通过预取起到一定优化作用,原因是预取可以有效地减少缓存访问缺失次数,从而减少处理器因等待数据从内存到达缓存造成的延迟时间.但是,简单预取方法也存在一定的局限性.例如,由于程序具有动态性,即使是同一段代码在相同的环境下多次运行也可能产生不同的行为特性,从而导致缓存性能不稳定,所以这种简单的静态预取在效果上往往不够理想,在实验中往往表现出对缓存优化的效果不是很稳定,为此,我们考虑利用动态预取来改进优化算法.动态预取的目标是通过调整预取的时机达到对缓存优化的最佳效果.之所以进行动态调整预取时机是因为预取的时机对缓存的性能至关重要.举例来说,如果预取过早,可能会把即将被访问的数据淘汰出缓存,因此,当处理器对这部分数据进行请求时导致了额外的缓存缺失.反之,如果预取过迟,当处理器对数据缓存进行数据请求时,因为数据尚未到达缓存,同样也会发生缓存缺失的现象,而达不到优化效果.总之,适当的预取时机才会对缓存性能起到优化作用.预取时机可以通过调整预取距离(prefetch distance)来进行控制,预取距离具体是指两次迭代之间预取的时间间隔,通常可以用消耗的处理器指令数作为度量单位.

基于静态预取的局限性,我们对基于简单静态预取的聚集计算算法进行了改进.提出了一种动态调整预取距离的预取算法.算法描述如图 5 所示,其基本思想是在进行下一次预取之前,先获取数据缓存的性能统计信息(具体实现中,我们用 Papi[14]实现对 hardware counters 的访问,获取近 60 个处理器事件统计信息);再根据当前的统计信息计算出缓存的命中情况,并对下一次的预取距离进行调整.我们将预取距离的单次调整步长设定为处理器访问一次数据缓存线所需的指令周期数.预取距离的初始值按以下公式进行估算:

$$\text{预取距离}=\text{内存延迟}\times\text{跨距}/128/\text{每个叠代周期数} \tag{3}$$

其中,跨距等于每迭代消耗的字节数量,内存延迟与具体的处理器相关,在安腾®2 处理器上约为 300 个指令周期.

我们对动态调整算法也进行了实验评估,实验是在基于安腾®2 处理器的平台上进行的.实验中的聚集计算是 TPC-H 负载中的 Q_{10},数据量为 500M 且全部装入内存,实验中的预取指令是 Prefetch 0,把内存中的数据预取至各级缓存中,我们将动态预取算法和前述的静态预取算法进行了对比,并对两种算法的缓存缺失率进行了抽样统计,统计间隔设定为 36 000 指令周期.实验结果如图 6 所示,从实验结果可以看出,与简单的预取算法相比,动态预取算法的平均缓存缺失率要优于静态预取算法,对缓存的优化作用相对更稳定.

Algorithm 2. Dynamic prefetch.

Start

Access hardware counters;

Get L_3 *D*-cache statistics;

Adjust prefetch distance;

Prefetch according new distance;

End

Fig.5 Dynamic prefetch

图 5 动态预取算法

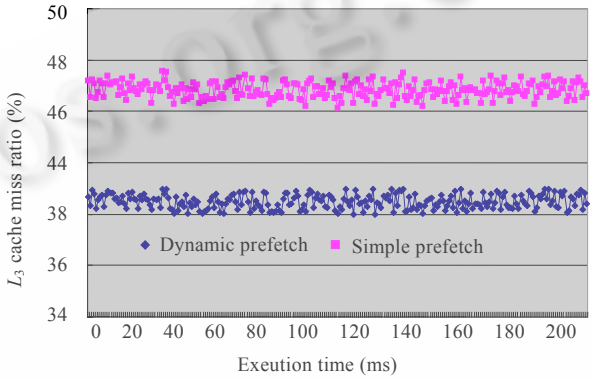


Fig.6 Dynamic prefetch cahce miss

图 6 动态预取算法缓存缺失

4.1.4 进一步讨论

对于基于简单访问模式的聚集计算,特别是以扫描为基础运算的聚集计算,如 TPC-H 中的 Q_6,Q_{10} 这样的负载,使用预取能够起到一定的优化作用.那么,如何对具有复杂访问模式的聚集计算,如 TPC-H 负载中的其他各个复杂查询进行优化是我们下一步要开展的工作.

4.2 存储体系结构对处理器行为的影响

目前,多数 DBMS 采用的是行存储模型.在行存储模型下,元组的属性按照建表时定义的先后次序连续存放,一次写磁盘操作可以把一个记录的所有属性写到磁盘上,从而写操作具有较高的性能.因此,按行存储的数据库系统又称为“写优化(write optimized)”系统.这类系统特别适合处理 OLTP 类事务,如 TPC-C 类的负载.然而,在某些应用中数据库系统应该被设计成“读优化(read optimized)”.例如,数据仓库中会定期地把数据批量装载至仓库中,然后处理耗时较长的查询;又如客户关系管理(customer relationship management,简称 CRM)系统、数字图书馆卡片目录系统和其他更新修改操作少而查询多的系统等等.在这种应用环境下,按列存储意味着所有属于同一列的数据项连续存放,因此数据库系统只需要读取与即席查询相关的列即可,从而避免了处理与查询无关的那些属性列.简而言之,两种不同存储体系结构的差异是:前者按行将某些记录的所有属性读入内存中;后者则按列将某些属性的全部值读入内存中.那么两者对处理器的利用情况如何呢?

下面首先通过实验比较不同的存储体系结构对处理器的利用情况以及差异,并分析产生差异的原因;其次,在存储体系结构设计方面我们给出一些建议;实验主要评价两种存储模型:行存储(row-based)和列存储(column-based)在现代处理器上的不同行为特性.

4.2.1 实验与结果分析

为了比较不同的存储结构对处理各级缓存的影响,我们对 MonetDB 和 System A 进行了实验分析.其中,MonetDB 的存储体系结构采用的是列存储的策略,而 System A 采用的是行存储体系结构.实验分别在基于安腾®2(Itanium®2)处理器和至强(Opteron (TM))处理器的平台上进行测试,实验的结果如图 7 所示.图中比较了两种不同存储模型的内存数据库在不同实验平台上运行 TPC-H 负载时处理器的时间开销比例,各个比例的计算采用的是第 2.1 节中所描述的方法.从图 7 中我们可以看出,二者的差异主要在第 3 级数据缓存所产生的时间延迟上;基于安腾®2 处理器的实验平台上,MonetDB 在第 3 级数据缓存缺失产生的延迟比 System A 要少近 20%.

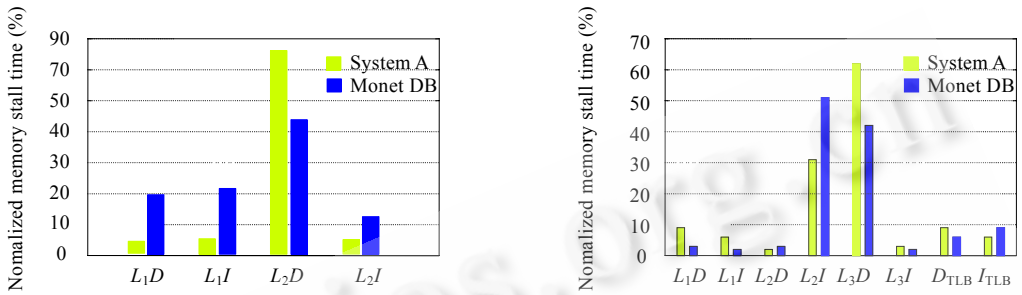


Fig.7 Storage architecture influence on deep memory: AMD Opteron (TM)(left), Itanium®2 (right)

图 7 不同存储体系结构对处理器各级缓存的影响:AMD 至强平台(左图)、安腾®2 平台(右图)

造成这两个系统在第 3 级数据缓存上的缺失存在较大差异的主要原因是:TPC-H 负载中的查询多是针对表中部分属性列的聚集计算.在 22 个查询中,有些查询需要大量的全表扫描,如 Q_1,Q_6 和 Q_{10};有些需要大量的连接操作,如 Q_4,Q_{13} 等.当元组从内存被调入最近的数据缓存(在安腾®2 处理器上是第 3 级数据缓存;在至强处理器上是第 2 级数据缓存)时,如果数据库系统是以行存储方式存储元组的,则每次读入数据缓存中的元组会有一部分不必被访问,这部分属性本来无须装入数据缓存.这种存储方式使得数据缓存中存放的元组数量相对要少,造成数据缓存频繁进行数据交换,进而导致处理器空闲时间延长而降低了系统的性能.采用按列存储的 MonetDB 系统在执行查询时,按列将元组从内存读入数据缓存,在相同缓存大小的条件下,每次可以读入更多数量的元组,因此数据缓存和内存数据交换的时间将会减少,从而性能优于行存储系统的性能.我们在基于至强处理器的实验平台上进行了同样的测试,实验结果和在基于安腾®2 处理器的实验平台上的结果相一致;在 TPC-H 这类负载下,列存储模型的性能优于行存储模型.

实验结果表明,不同存储体系结构对处理器的利用存在较大的差异,数据库系统的设计者应该针对具体的应用背景考虑数据库的存储体系结构.特别是随着内存容量的不断增加,当全部的工作集(working set)可以放在内存时,如何有效地减少处理器和内存之间的数据交换代价,从而提高处理器的利用率将变得更为重要.根据以上实验结果,在下一代的 DBMS 体系结构设计方面,我们也开展了一些研究工作:我们设计开发了 PMDB(parallel main memory database)原型系统.PMDB 是一个列存储的、并行的、内存数据库系统.PMDB 的设计目标是希望能够充分利用现代计算机硬件发展的优势(如处理器、内存).

4.3 索引对处理器行为的影响

磁盘数据库优化技术的主要目标是减少磁盘 I/O,因此,索引、数据压缩等技术可以通过减少磁盘 I/O 来优化性能.然而,内存数据库中所有的数据都驻留在内存,这种情况下磁盘 I/O 不再是系统的主要瓶颈,而在于处理器内存之间的数据交换代价.因此,研究索引对处理器各级缓存的利用情况以及索引在处理器上的行为特性可以对性能评价以及优化策略的设计提供依据.本节先以实验分析了索引对处理器上各级缓存的影响以及使用索引时瓶颈的所在;然后,对实验结果进行分析并给出索引设计方面的优化建议.

本节的实验是在 MonetDB 和 System A 两个系统上进行的.实验中,我们在 TPC-H 基准测试中的数据库模式上创建了索引,然后分别在内存数据库 MonetDB 和 System A 上进行 TPC-H 测试.实验在基于至强处理器的

系统平台上进行,实验结果如图 8 所示.从图中可以看出,两个内存数据库在没有索引的情况下,由第 1 级数据缓存造成的缺失和第 1 级指令缓存以及第 2 级指令缓存引起的处理器延迟时间要比有索引的情况下对应的缺失多一些;另一方面,从实验结果来看,当使用索引时,第 2 级数据缓存造成的缺失比例在增长.

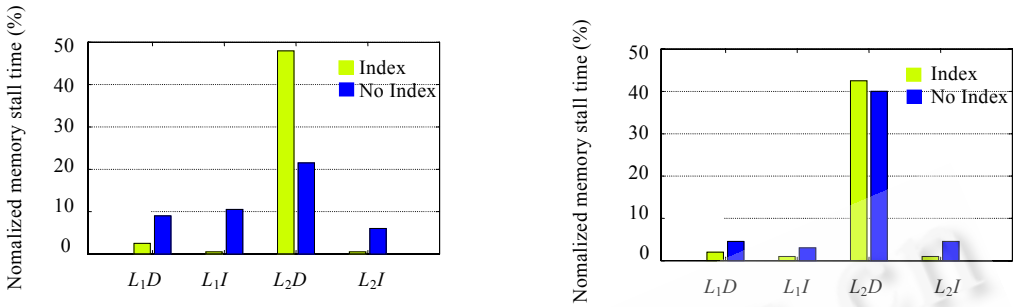


Fig.8 Index influence on deep memory: MonetDB on AMD platform (left), System A on AMD platform (right)
图 8 AMD 至强平台上索引对处理器各级缓存的影响情况:MonetDB(左图)、系统 A(右图)

索引的使用会对处理器各级缓存的行为特性产生一定的影响,其主要原因是:索引扫描的空间局部性与无索引时相比会好一些,因此,索引操作符访问和预取的元组个数相对较少,从而数据缓存中可以保留更多的有效数据.所以,当数据访问变少时,将会有更多的空间用于存放指令,使得每次取指令可以获取更多的指令数,并使得指令缓存的缺失率响应也下降.针对这一实验现象,在索引的设计和使用上应该如何才能提高缓存子系统的性能以减少时间延迟、提高性能呢?我们给出的一些优化策略和建议如下:在索引的数据结构方面,其结构应该具有更好的"缓存"特性.例如,如果使用 B^+树,那么应该尽可能减少树的"高度"、尽量多地去除节点内的指针,同时将节点的大小控制在一个缓存线(cache line)大小左右;此外,还可以使用压缩方法使单个节点存放更多的索引项等等.通过这些方法,可以增加节点的扇出度、降低树的深度,从而改善树的缓存性能.

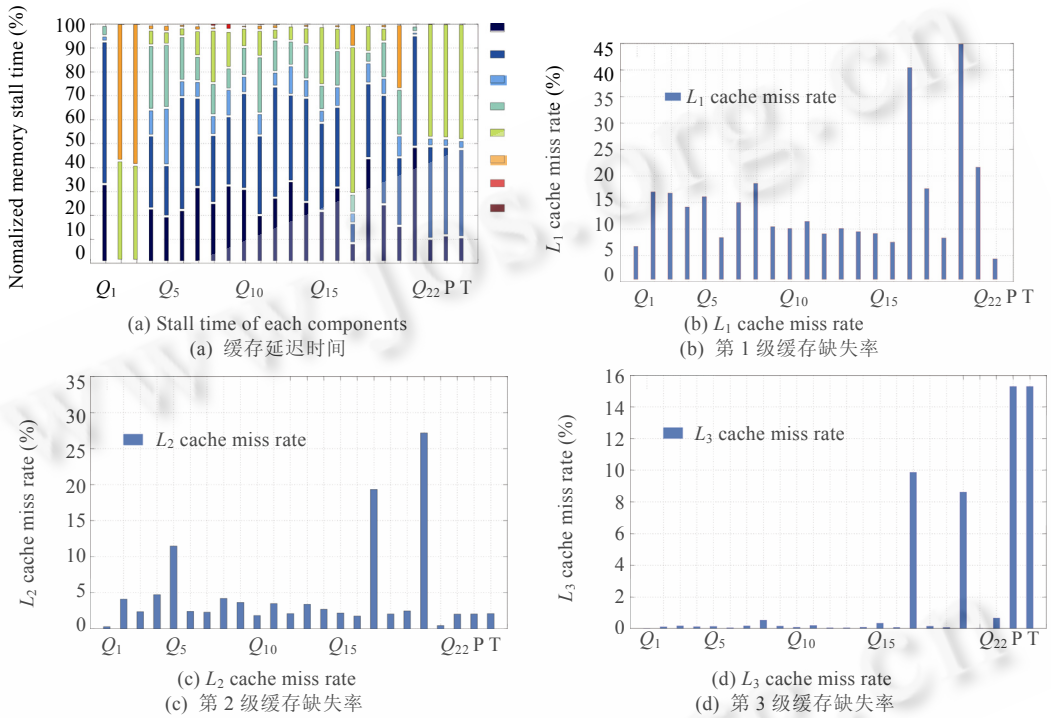
在改进索引的缓存性能设计方面,我们研究组的 Luan 等人提出了一种针对内存数据库优化的索引结构——J+Tree,该索引结构与 B 树和 T 树相比具有更好的适应缓存性能.J+–Tree 主要由两部分组成,上层是一个 Judy[15]数据结构,下层是叶子节点.所有的关键字都放到叶子节点中,每个叶子节点的最小值作为这个叶子节点的引用值同时存储在上层的 Judy 结构中.逻辑上讲,Judy 是一个 256 路的数字树,关键字分层存储在树中.由于 Judy 的特点,J+–Tree 的高度与关键字的数目没有直接关系,而是由关键字的大小来决定,对于 32 位的关键字,无论有多少值需要存储,J+–Tree 只用 5 层即可满足需要.使用分层存储的方式,J+–Tree 降低了树的高度,同时内部节点和叶子节点的大小设为缓存线的倍数,仅需几个缓存读取即可找到需要的数据.关于该索引的详细设计以及优化方法可以参见文献[16].

4.4 微测试集(micro benchmark)

使用 TPC-H 基准测试可以准确地分析 MMDB 在 DSS 类负载上的处理器行为特性,但是还存在着一定的不足.首先,建立实验环境以及对负载的调整过程比较复杂.例如,调配磁盘的数量和速度、I/O 控制器、磁盘 I/O 总线和处理器 I/O 总线等过程都比较复杂.其次,建立标准所需环境的硬件配置也比较高.比如,物理内存的大小会直接影响内存的带宽.此外,在软件方面,商用数据库手册中指明的初始化参数多达 75~200 个,它们相互配合用以控制和管理运行时的各种问题,如缓冲区大小和管理策略、多线程或多进程的程度、日志以及与 DSS 应用相关的内存管理方案的诸多选择等;操作系统软件也有大量配置参数,诸如同步或异步 I/O 的选择参数、I/O 操作的缓冲区管理参数、文件在不同磁盘间条带状分布参数、时间片选取以及网络传输和缓冲区参数等.这些复杂的参数进一步增大了实验的复杂性.因此,用一个简单的负载来模拟复杂的工业标准测试,即对微测试集的研究是值得研究的一个问题.

目前针对 TPC-C 以及 TPC-H 的微测试集方面已有部分研究工作,例如,文献[7]比较了 OLTP 负载和 OLAP 负载,作者指出,OLTP 对磁盘的访问多为随机读写操作,而 OLAP 多为顺序大量读操作.针对这一特点,作者仅用

单表和若干简单查询来模拟磁盘数据库运行完整的 OLTP 和 OLAP 负载,并取得了较好的效果.本文用 TPC-H 标准负载来研究 MMDB 在处理器上的行为特性,但是测试过程相对复杂而且比较耗时.因此,我们希望能够构建一个微测试集,使得测试的规模尽量缩小,同时,在精度方面能够和使用 TPC-H 标准负载测试的结果相当.为此,我们首先对 TPC-H 负载的 22 个查询进行了详细的分析,得到了单个查询在处理器上的统计信息.如图 9 所示.由实验结果我们发现:在 TPC-H 的所有查询中,第 8 个查询和第 17 个查询(Q_8,Q_{17})在处理器各级缓存上的时间分布与整个负载的时间分布基本接近.我们还对每个查询在各级缓存上的命中率进行了比较,结果如图 9 所示.从图中可以看出,这两个查询(Q_8 和 Q_{17})同样具有典型的代表性.通过本节的实验,可以得到一个方法上的结论:可以用简单的查询代替整个测试集来研究 MMDB 在 TPC-H 负载上的处理器特性.关于微测试集的构建与精度的评价是我们正在开展的研究工作.



(a) Stall time of each components
(a) 缓存延迟时间

(b) L_1 cache miss rate
(b) 第 1 级缓存缺失率

(c) L_2 cache miss rate
(c) 第 2 级缓存缺失率

(d) L_3 cache miss rate
(d) 第 3 级缓存缺失率

Fig.9 Cache performance statistics on Itanium®2-Based platform

图 9 安腾®2 平台 TPC-H 负载下各级缓存性能统计信息

5 结论与未来工作

本文研究了内存数据库在承受 TPC-H 负载时对现代处理器的利用情况以及瓶颈所在,并分析了导致瓶颈所在的原因以及优化策略.首先,我们通过实验分析了磁盘数据库和内存数据库在 TPC-H 负载下处理器的性能瓶颈差异及其原因,并针对内存数据库的瓶颈提出了优化方法;其次,分析了不同存储体系结构(行存储与列存储)对处理器利用率的差异,并在下一代基于海量内存的数据库体系结构方面进行了研究;此外,还通过实验分析了索引结构对处理器多级缓存的影响,并针对这种现象给出了优化索引策略;最后,提出了一个微测试集用于评价内存数据库在 DSS 负载下处理器的性能及行为特性.我们希望该文的研究结果能对基于下一代处理器的内存数据库体系结构设计以及性能优化提供一定的研究依据.

基于目前的研究工作,我们下一步的工作重点首先是完善现有的优化方法,设计更具一般性的优化策略;其次,研究如何在指令级设计高度并行算法以提高各级缓存命中率,从而充分利用处理器的高计算能力来优化数据库的性能.

References:

[1] Boncz P, Zukowski M, Nes N. MonetDB/X100: Hyper-Pipelining query execution. In: Proc. of the CIDR. 2005.

[2] Ailamaki A, DeWitt DJ, Hill MD, Wood DA. DBMSs on a modern processor: Where does time go? In: Proc. of the 25th Int'l Conf. on Very Large Data Bases. 1999. 266−277.

[3] Thakkar SS, Sweiger M. Performance of an OLTP application on symmetry multiprocessor system. In: Proc. of the 17th Annual Int'l Symp. on Computer Architecture. Washington, 1990. 228−238.

[4] Maynard AMG, Donnelly CM, Olszewski BR. Contrasting characteristics and cache performance of technical and multi-user commercial workloads. In: Proc. of the 6th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems. San Jose, 1994. 145−156.

[5] Rosenblum M, Bugnion E, Herrod SA, Witchel E, Gupta A. The impact of architectural trends on operating system performance. In: Proc. of the 15th ACM Symp. on Operating Systems Principles. Copper Mountain, 1995. 285−298.

[6] Eickemeyer RJ, Johnson RE, Kunkel SR, Squillante MS, Liu S. Evaluation of multithreaded uniprocessors for commercial application environments. In: Proc. of the 23rd Annual Int'l Symp. on Computer Architecture. Philadelphia: 1996. 203−212.

[7] Keeton K, Patterson DA, He YQ, Raphael RC, Baker WE. Performance characterization of a quad Pentium Pro SMP using OLTP workloads. In: Proc. of the 25th Annual Int'l Symp. on Computer Architecture. Barcelona, 1998. 15−26.

[8] Trancoso P, Larriba-Pey JL, Zhang Z, Torrellas J. The memory performance of DSS commercial workloads in shared-memory multiprocessors. In: Proc. of the 3rd IEEE Symp. on High-Performance Computer Architecture. 1997. 250.

[9] Barroso LA, Gharachorloo K, Bugnion E. Memory system characterization of commercial workloads. In: Proc. of the 25th Annual Int'l Symp. on Computer Aarchitecture. Barcelona, 1998. 3−14.

[10] Manegold S. The calibrator (v0.9e), a cache-memory and TLB calibration tool. 2004. http://monetdb.cwi.nl/Calibrator/

[11] Ahn D, Kufrin R, Raghuraman A, Seo JH. perfsuite 2007. 2007. http://perfsuite.ncsa.uiuc.edu/

[12] Boncz PA. Monet: A next-generation dbms kernel for query-intensive applications [Ph.D. Thesis]. Amsterdam: Universiteit van Amsterdam, 2002.

[13] PostgreSQL Organization. PostgreSQL 6.3 User Manual. 1998. http://www.postgresql.org/

[14] Performance Application Programming Interface (PAPI). Innovative Computing Library. 2008. http://icl.cs.utk.edu/papi/overview/index. html

[15] Baskins D. Judy functions—C libraries for creating and accessing dynamic arrays [EB/OL]. 2006. http://judy.sourceforge.net

[16] Luan H, Du XY, Wang S, Ni YZ, Chen QM: J+-Tree: A new index structure in main memory. In: Proc. of the 12th Int'l Conf. on Database Systems for Advanced Applications (DASFAA 2007). LNCS 4443, Bangkok, 2007. 386−397.

刘大为(1974－),男,内蒙古突泉人,硕士,CCF 学生会员,主要研究领域为高性能数据库新技术,内存数据库.

王珊(1944－),女,教授,博士生导师,CCF 高级会员,主要研究领域为高性能数据库新技术,数据库信息检索,数据仓库,BI 技术.

栾华(1980－),女,硕士,CCF 学生会员,主要研究领域为高性能数据库新技术,数据仓库,BI 技术.

覃飙(1972－),男,博士,副教授,主要研究领域为数据库,数据挖掘,P2P 数据管理.