计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2016.20150613
53(12):2836-2846,2016

并发内存 OLAP 查询优化技术研究

张延松[1,2,3] 焦 敏[1,2] 张 宇[4] 王 珊[1,2]
[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](中国人民大学信息学院 北京 100872)
[3](中国调查与数据中心(中国人民大学) 北京 100872)
[4](中国气象局国家卫星气象中心 北京 100081)
(zhangys_ruc@hotmail.com)

Concurrent In-Memory OLAP Query Optimization Techniques

Zhang Yansong[1,2,3], Jiao Min[1,2], Zhang Yu[4], and Wang Shan[1,2]
[1](*Key Laboratory of Data Engineering and Knowledge Engineering* (*Renmin University of China*), *Ministry of Education*, *Beijing* 100872)
[2](*School of Information*, *Renmin University of China*, *Beijing* 100872)
[3](*National Survey Research Center* (*Renmin University of China*), *Beijing* 100872)
[4](*National Satellite Meteorological Center*, *China Meteorological Administration*, *Beijing* 100081)

Abstract Recent researches not only focused on query-at-a-time query optimizations but also focused on group-at-a-time query optimizations due to the multicore hardware architecture support and highly concurrent workload requirements. By grouping concurrent queries into shared workload, some high latency operations, e.g., disk I/O, cache line access, can be shared for multiple queries. The existing approaches commonly lie in sharing query operators such as scan, join or predicate processing, and try to generate an optimized global executing plan for all the queries. For complex analytical workloads, how to generate an optimized shared execution plan is a challenging issue. In this paper, we present a template OLAP execution plan for widely adopted star schema to simplify execution plan for maximizing operator utilization. Firstly, we present a surrogate key oriented join index to transform traditional key probing based join operation to array index referencing (AIR) lookup to make join CPU efficient and support a lazy aggregation. Secondly, the predicate processing of concurrent queries is simplified as cache line conscious predicate vector to maximize concurrent predicate processing within single cache line access. Finally, we evaluate the concurrent template OLAP (on-line analytical processing) processing with multicore parallel implementation under the star schema benchmark (SSB), and the results prove that the shared scan and predicate processing can double the concurrent OLAP query performance.

Key words concurrent OLAP query processing; array index referencing (AIR); template OLAP query processing; join index; filtering vector

收稿日期:2015-06-30;**修回日期**:2015-10-13

基金项目:国家"八六三"高技术研究发展计划基金项目(2015AA015307);中国人民大学科学研究基金(中央高校基本科研业务费专项资金资助)项目(16XNLQ02)

This work was supported by the National High Technology Research and Development Program of China (863 Program) (2015AA015307) and the Research Funds of Renmin University of China (the Fundamental Research Funds for the Central Universities) (16XNLQ02).

通信作者:焦敏(shingle@ruc.edu.cn)

摘 要 基于多核处理器硬件技术和高并发查询负载需求,近年来的研究不仅关注于一次一查询模式的查询优化技术,而且也关注于一次一组模式的查询优化技术.通过将并发查询转换为共享负载,一些低访问延迟的操作,如磁盘 I/O、cache 访问,可以被多个并发的查询所共享.当前的研究通常基于共享查询操作符,如扫描、连接、谓词处理等,通过生成全局执行计划优化并发查询.对于复杂的分析型负载,如何创建优化的执行计划是一个具有挑战性的问题.在广泛使用的星形模型的基础上提出一种模板 OLAP 查询执行计划来简化查询执行计划,以达到最大化查询操作符利用率的目标.1)提出了基于代理键的连接索引技术,将传统的基于值探测的连接操作转化为内存数组索引引用(AIR),使连接操作的 CPU 效率更高并且支持聚集计算的后物化;2)并发查询的谓词处理简化为 cache line 敏感的谓词向量,在单次 cache line 访问中最大化并发查询谓词计算性能;3)通过多核并行实现技术在 SSB 基准上进行测试.实验结果表明:共享扫描和共享谓词处理能够将并发 OLAP 查询处理性能提升 1 倍.

关键词 并发 OLAP 查询处理;数组索引引用;模板 OLAP 查询处理;连接索引;过滤向量

中图法分类号 TP311.5

大内存、多核处理器已经成为高性能计算平台的基本特征,基于内存的分析处理系统,如 MonetDB[1],Vectorwise[2],SAP HANA[3],Oracle Exadata X4[4],Hyper[5],IWA[6] 等逐渐成为高性能分析型数据库和新一代数据库一体机技术的代表.随着互联网用户和互联网业务的飞速发展,高负载的实时分析处理需求是数据库需要面对的新技术挑战,不仅要求数据库引擎具有良好的实时查询响应性能,还要求数据库引擎具有强大的查询吞吐性能,能够同时满足成百上千的并发分析处理任务.

分析型内存数据库技术面向低延迟的实时查询处理负载,查询处理技术的核心是提高单个查询的执行性能,以 MonetDB 的二元关联表(binary associated table, BAT)处理和 Vectorwise 的向量处理技术为代表,通过列存储访问、优化内存 Hash 表设计、以向量为单位处理来优化数据的 cache 访问局部性,提高查询处理性能.在并发查询处理时,每个查询处理线程需要并发地访问数据,维护独立的线程私有化数据(如 Hash 表、向量等),从而造成线程间数据共享度低、内存数据在 cache 中频繁换入换出,降低了并发查询处理效率.

并发查询优化技术主要包括查询间结果集共享访问、查询间数据共享扫描和查询间操作符共享等方式.MonetDB 采用缓存中间结果技术[7],并通过对并发查询执行顺序进行优化以使缓存的中间结果能够被后续的查询利用.Database Cracking[8] 通过查询时的动态数据划分技术加速后续查询的谓词处理性能.共享结果集并发优化技术的基础是查询的相关性,通过相近查询结果集复用减少冗余的计算代价.Blink 系统[9-10] 通过 Denormalization 技术将模式简化为单一的表,并发查询转换为表扫描和谓词操作,通过共享扫描技术提高并发查询的数据访问效率.Crescando[11] 在存储访问层采用共享扫描技术,并通过连接数据与查询集合的方式实现并发查询处理.共享扫描是并发查询处理广泛采用的技术,主要优化访问延迟较大的数据集上的全表扫描操作,通常采用循环扫描方式支持“always-on”模式的查询处理.QPipe[12],DataPath[13],SharedDB[14],CJoin[15] 等研究面向复杂的分析型查询操作符,通过共享执行代价大的操作符减少并发查询处理时的冗余计算代价.共享操作符的并发查询优化技术需要面向并发查询构造全局性的查询执行计划,调整查询执行顺序以实现高代价操作符的共享.

在内存数据仓库应用场景中,OLAP 查询是一个在多维数据空间中定位多维数据子集并对其进行聚集计算的过程.在关系 OLAP(relational OLAP, ROLAP)处理模式中,多维查询处理表现为事实表通过维表外键与各维表生成的 Hash 表进行连接并对连接结果进行分组聚集的计算过程,是一个 SPJGA(选择、投影、连接、分组、聚集)关系操作,查询性能的关键是星形连接(star join)的效率,Hash 连接操作的性能取决于 Hash 表相对于 cache 的大小以及连接选择率.当构建并发查询共享 Hash 表时,由于查询中需要携带不同查询的维表分组属性,共享 Hash 表中的记录可能为维表记录宽度,Hash 表存储空间增大;星形 Hash 连接在选择率较低时产生较大的无效 Hash 探测代价,在并发查询处理时消耗了过多的 CPU cycle.

本文以数据仓库中通用的代理键索引为基础,通过代理键连接索引机制实现基于数组索引引用

(array index referencing, AIR)的连接技术,将传统的 Hash 连接操作简化为内存数组索引直接引用,消除了 Hash 表.进一步地,通过维表位图压缩维表记录,实现连接过滤,支持后物化的维表分组属性访问.在 AIR OLAP 多维查询处理技术的支持下,并发 OLAP 查询处理被划分为 3 个阶段:1)并发查询分组并创建维表位图向量,记录并发查询在每一个维表记录上的谓词过滤结果;2)并发星形过滤,通过事实表记录外键定位各维表位图向量,通过按位与操作计算出当前事实表记录在各个维表上每个并发查询的过滤结果;3)基于代理键连接索引的分组聚集计算,事实表记录按映射的数组内存索引地址直接访问维表分组属性并进行聚集计算.

与已有的并发查询技术相比,AIR OLAP 并发查询处理直接在原始星形模式上执行,不需要物化表;星形 OLAP 查询处理通过代理键连接索引构造了一个模板化的 OLAP 查询执行计划,不同的 OLAP 查询对应结构相同、数据不同的维过滤位图,相同的查询执行计划能够最大化并发查询的共享访问和计算性能;后物化的聚集计算则在低选择率的 OLAP 查询中提高了数据访问效率.

1 相关工作

1.1 内存 OLAP 查询优化技术

列存储已经成为内存分析型数据库普遍采用的存储模型,面向事务处理优化的内存数据库通常支持列存储、行存储以及混合存储模型以优化不同的查询负载. MonetDB 采用一次一列(column-at-a-time)的处理技术,在低选择率时具有良好的性能,但选择率较高时需要物化较大的中间结果,增加了存储及计算代价. Vectorwise 采用一次一向量(vector-at-a-time)的查询处理技术,以 L1 cache 敏感的较小向量为处理单位,实现 L1 cache 内的物化数据存储访问,消除了内存物化数据存储和访问代价.传统的行存储模型执行的是一次一记录(tuple-at-a-time)执行方式,存在数据访问效率低、查询解析代价高的问题.但在并发查询处理负载中,不同的查询访问不同的列,导致内存带宽访问竞争加剧;同时,并发查询产生较多的中间结果数据,增加了 cache 及内存访问代价.

Hash 连接是内存数据库优化技术的核心问题.文献[16-17]通过实验验证了在多核处理器平台上基于 radix 分区的并行 Hash 连接算法优于基于共享 Hash 表的并行 Hash 连接算法和排序连接算法,通过分区提高 Hash 探测阶段的 cache 数据局部性,提升整体 Hash 连接操作性能.但在并发查询处理时,radix 分区操作会产生较大的内存消耗,而共享 Hash 表连接方法则可以通过建立并发查询共享 Hash 表的方式支持批量 Hash 连接操作.因此,并发 OLAP 查询优化技术与占用全部资源的单查询优化技术在设计思想上有较大的区别,前者强调全局共享效率,后者强调局部性能.

内存列存储支持基于偏移地址计算的直接内存访问,文献[18]实现了数据仓库中将事实表外键映射为维表内存偏移地址的 DDTA-JOIN(directly dimensional tuple accessing-join)算法,支持在原始数据上的内存直接关联记录访问,简化了连接操作并消除了传统的 Hash 表或排序操作.

1.2 并发 OLAP 查询优化技术

在数据仓库负载中,表扫描操作通常代价较高,基于共享扫描的并发查询技术能够更高效地利用顺序访问的内存带宽,同时服务于多个 OLAP 查询处理任务. Blink 采用非规范化设计,将 OLAP 查询转换为单一的表扫描操作,实现了常量时间的查询处理性能,同时支持基于共享扫描的并发查询处理. Crescando 通过索引运行的查询和共享扫描来支持可预期性能的查询处理,MonetDB/X100 也提供了协同扫描机制以优化动态的内存带宽性能.

代表性的关系操作符并发优化技术包括 Datapath,CJoin,SharedDB,QPipe,COD[19]等,基本思想是为并发查询创建全局操作符. CJoin 是面向数据仓库星形模型的一种并发 OLAP 查询优化技术,它为并发查询创建全局 Hash 表,事实表循环扫描为每个事实表记录依次通过每个 Hash 过滤器传递事实表记录在每个过滤器上对每个并发查询的过滤结果,实现一次 Hash 探测操作服务于多个并发查询任务. QPipe 作为执行代价较大的操作创建一直在线运行的操作符,查询进入系统后通过代价估算连接上活动的共享操作符,从操作符产生的数据流中获取当前查询需要的结果. COD 通过列存储、水平分片策略,基于 NUMA 架构在每个核心的数据分片上执行 Clock-Scan,连续扫描每个核心上数据集的水平分片.查询首先进入执行队列,按谓词索引,扫描线程执行数据分片上的顺序扫描,完成查询队列的查询任务,然后通过归并和聚集线程将查询结果推送到输出队列.文献[20]实现了磁盘存储事实表共享扫描、内存 DDTA-JOIN 连接模式的并发

OLAP 查询处理,在一个 I/O 时间片内通过多核内存并发 DDTA-JOIN 算法最大化共享 I/O 访问服务的并发查询数量.

在共享操作符的并发 OLAP 查询处理技术中,并发查询处理的性能通常优于一次一查询执行方式的性能,尤其在存在慢速磁盘 I/O 的 OLAP 应用场景中.内存 OLAP 并发查询处理时,不仅需要通过共享扫描减少表扫描的内存访问延迟,更重要的是通过并发查询优化技术减少星形连接过程的 cache miss,即通过算法数据结构的优化在一个 cache line miss 中尽可能完成并发查询的计算任务.

2 并发 OLAP 查询处理

2.1 AIR OLAP 多维查询处理

数据仓库不同于关系数据库之处在于采用的是多维数据模型,即每一个事实数据 F 由多个维度 $(D_1, D_2, \cdots, D_n)$ 映射 (Map),记作 $F \leftarrow Map(D_1, D_2, \cdots, D_n)$.

图 1 显示了 OLAP 应用中最具有代表性的模式:TPC-H[21],SSB[22],TPC-DS[23].在数据仓库中分别对应雪花模型(snow-flake schema)、星形模型(star schema)和暴风雪(snow storm schema)模型.

TPC-H 是一个满足 3NF 的数据库,存在共享多级维表,可以看作是一种雪花模型,也可以看作是 3 个子星形模式:(ORDERS,CUSTOMER→NATION→REGION),(LINEITEM,PART,SUPPLIER→NATION→REGION),(PARTSUPP,PART,SUPPLIER→NATION→REGION).

SSB 是对 TPC-H 非规范化处理后的星形模型,使用单一的星形模式:(LIENORDER,PART,SUPPLIER,CUSTOMER,DATE).

TPC-DS 可以看作是 2 个子星形模式:(Store_Sales,Date_Dim,Promotion,Customer→Customer_Address,Customer→Customer_Demographics,Customer→Household_Demographics→Income_Band,Item,Store,Time_Dim),(Store_Returns,Reason,Date_Dim,Customer→Customer_Address,Customer→Customer_Demographics,Customer→Household_Demographics→Income_Band,Item,Store,Time_Dim).

3 个模式普遍遵循数据仓库的代理键约束,除

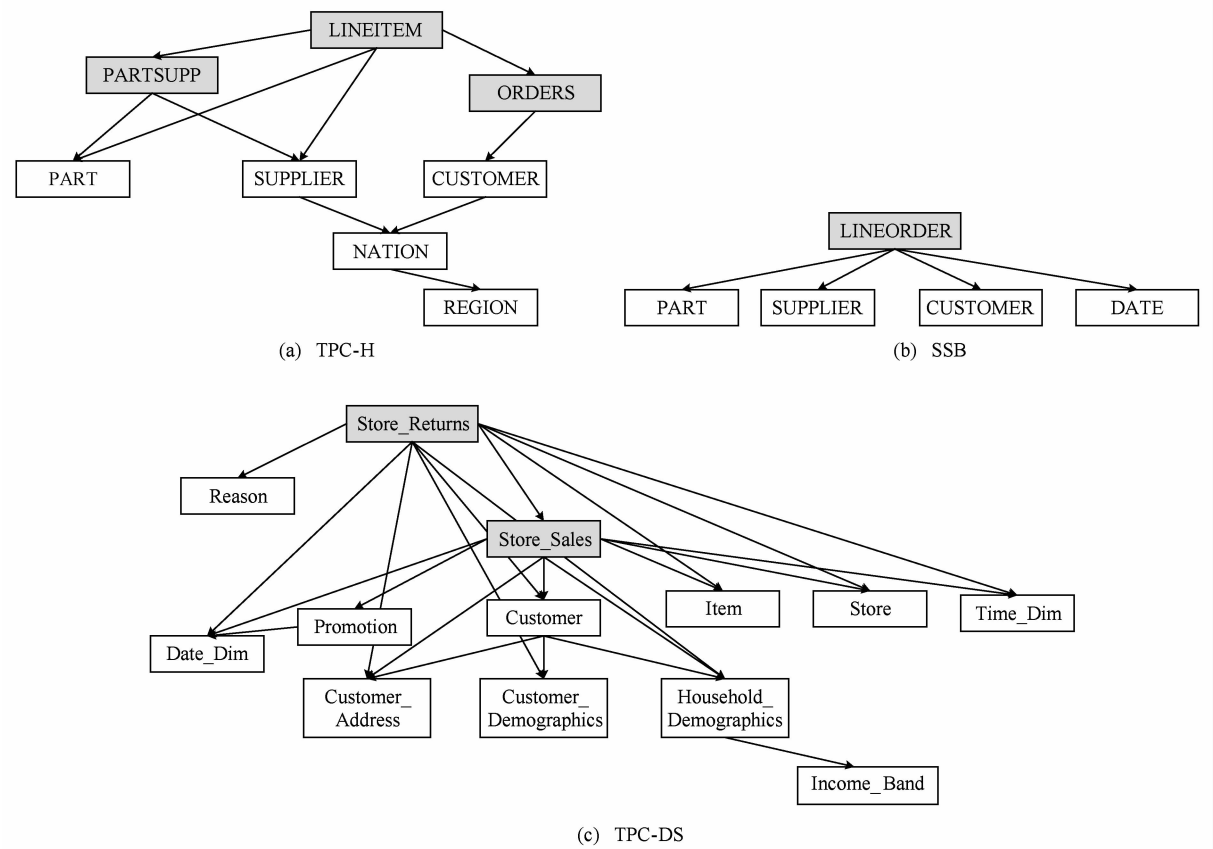

Fig. 1 Multidimensional models of typical benchmarks.

图 1 典型多维数据模型

SSB的维表DATE主键为19920101,19920102,…,格式的连续日期外,其余所有维表主键均为从0或1开始的连续序列,当采用内存列存储时,维表主键可以直接映射为维表属性列的数组下标地址,事实表记录可以通过外键数组下标引用方式直接访问内存维属性列对应的记录.

在图2所示的AIR OLAP多维查询处理中,多维分析处理分为3个阶段:1)通过SQL命令改写在维表上创建过滤位图;2)事实表外键通过代理键连接索引映射到维表过滤位图完成星形连接过滤;3)满足连接过滤条件的事实表记录通过内存索引引用分组属性并进行聚集计算.在AIR OLAP查询处理过程中,计算代价最大的多维连接操作简化为从事实表外键向维表位图的位置映射,聚集计算采用后物化维分组属性直接内存访问方式,简化了查询处理算法,同时,维表构成了查询共享访问数据集,减少查询执行时私有连接Hash表的开销,提高了维表列的数据局部性.

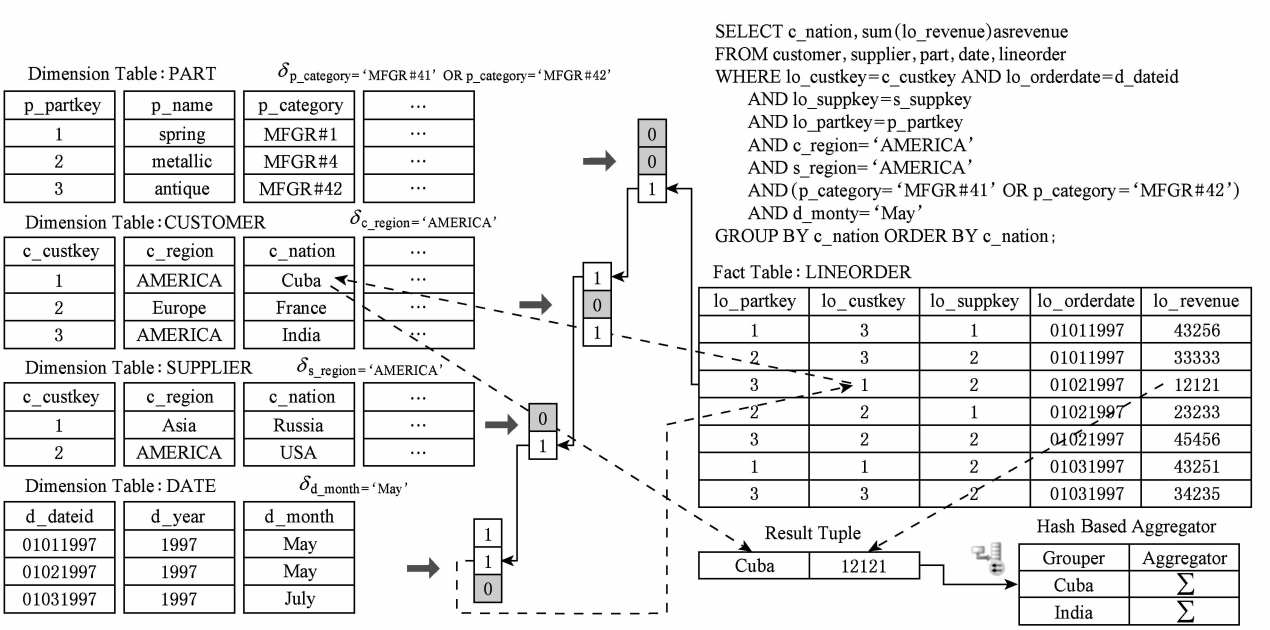

Fig. 2 Dimensional bitmap filtering oriented AIR OLAP multidimensional query processing.

图2 基于维表位图过滤的AIR OLAP多维查询处理

2.2 并发AIR OLAP多维查询处理

在AIR OLAP算法中,多维过滤需要访问各个维表位图对应位置并进行按位与运算.每次位图访问对应1个cache line访问,但仅使用512 b中的1 b.如图3所示,我们以5个并发查询为例,在并发AIR OLAP查询处理时,维表过滤位图转换为维表过滤向量,5 b向量表示5个并发查询在当前维表记录上的谓词处理结果,并发查询Q_0,Q_1,Q_2,Q_3,Q_4生成宽度为5 b的维表过滤向量.

事实表对维表位向量的每次访问都可以获得512个并发查询过滤位图,多个维表位向量的按位与操作可以利用SIMD机制并行计算,从而提高维表位图访问和计算的效率.

2.3 并发AIR OLAP多维查询处理实现技术

2.3.1 并发查询预处理

基于维过滤位向量的并发OLAP查询处理需要对并发查询分组,并通过查询预处理过程为并发查询创建维过滤位向量.如图4所示,OLAP查询被分解为维表上的谓词处理和分组聚集表达式,每个维表谓词表达式生成一个维过滤位图,存储在行存储的维过滤向量中查询序号对应的列.并发查询存储在查询队列(query queue)中,分别记录了查询在各个维表上的分组属性,度量属性通过事实表属性位向量来记录哪些度量属性需要进行聚集计算.

在并发查询分组过程中创建了维过滤位向量,加速维过滤位图的位与计算效率,同时创建并发查询分组聚集表,用于对满足连接过滤条件事实表记录以后物化方式进行的直接维表分组属性访问操作.

2.3.2 并发连接过滤操作

在图5的并发OLAP查询处理过程中,事实表上执行共享的顺序扫描操作,每个维表对应一个位向量过滤器(***BVecFilter***),事实表记录外键分别映射到各个维表位向量过滤器中抽取对应的位向量并进行如图3所示的按位与操作.按位与操作结果产

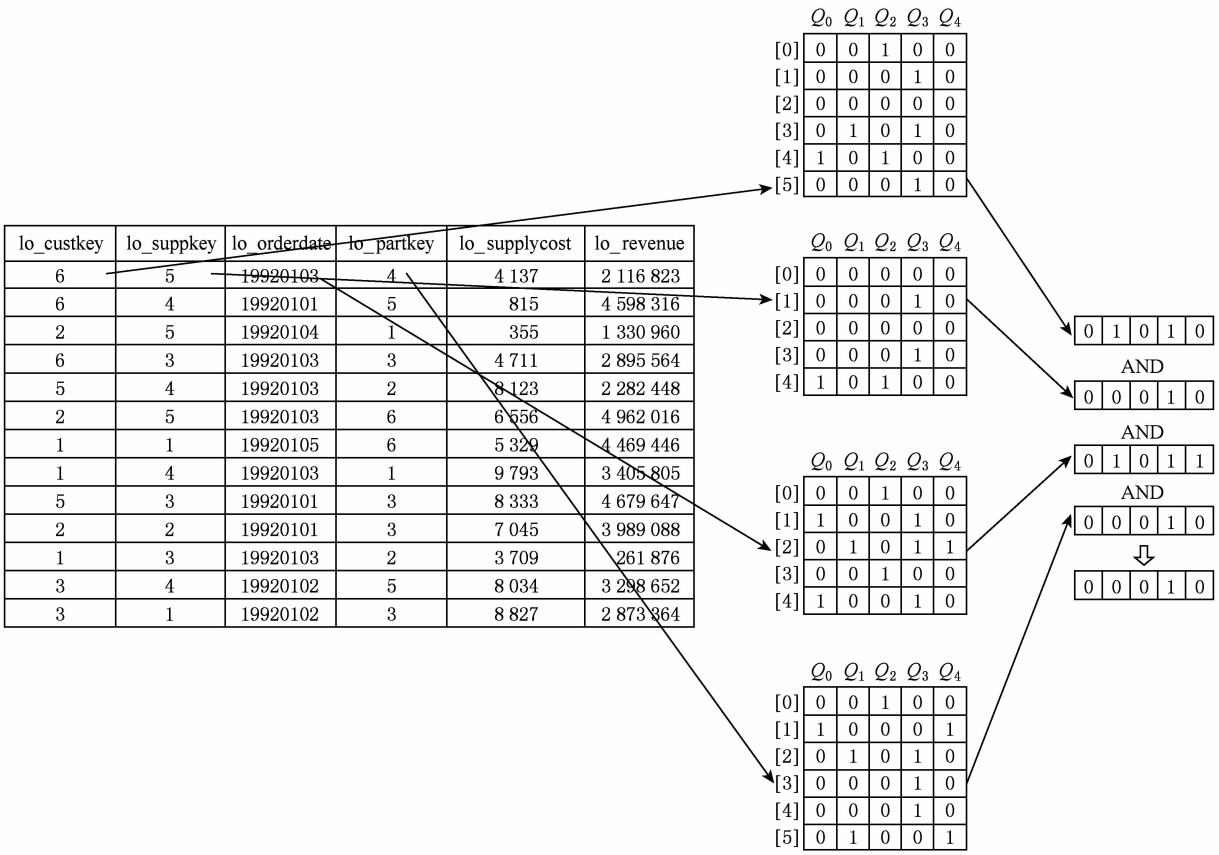

Fig. 3 Concurrent AIR OLAP query processing.

图 3 并发 AIR OLAP 多维查询处理

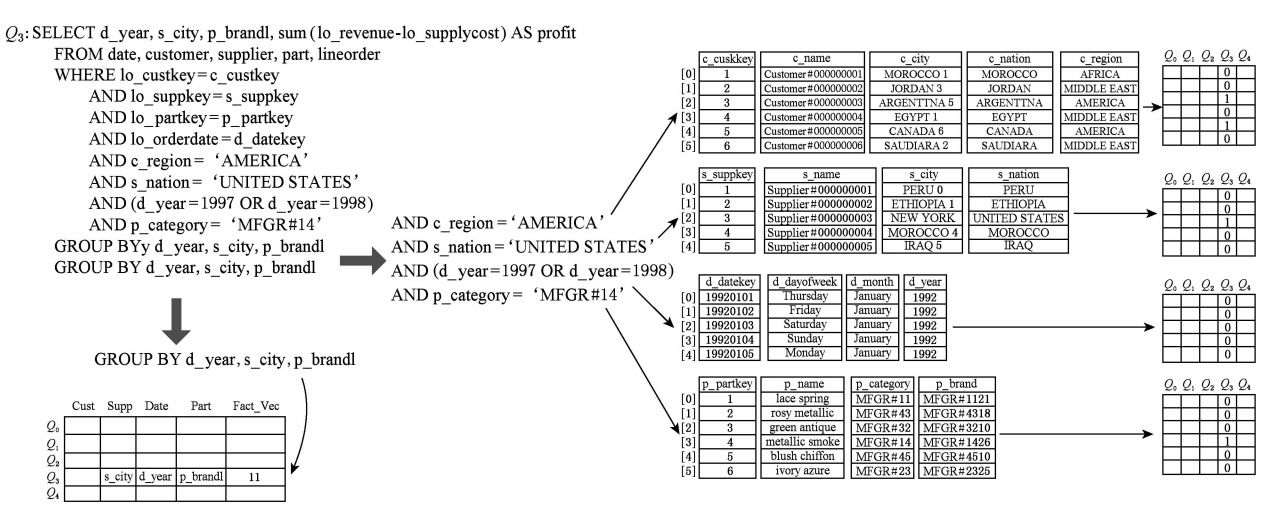

Fig. 4 Concurrent query grouping and predicate pre-computing.

图 4 并发查询分组和谓词预计算

生的位向量中的"1"表示该位置 i 对应的查询 Q_i 在当前事实表记录上满足所有维表上的连接过滤条件,需要进行后续的分组聚集计算.向量的按位与操作结果传递给查询分发器(distributor),根据位向量中"1"的位置调用各个对应查询预设的聚集计算函数(aggr operator),完成并发查询对当前事实表记录的聚集计算.

2.3.3 并发分组聚集计算操作

查询分发器的分组聚集计算需要经过查询映射和分组映射 2 个阶段.如图 6 所示,首先通过生成的

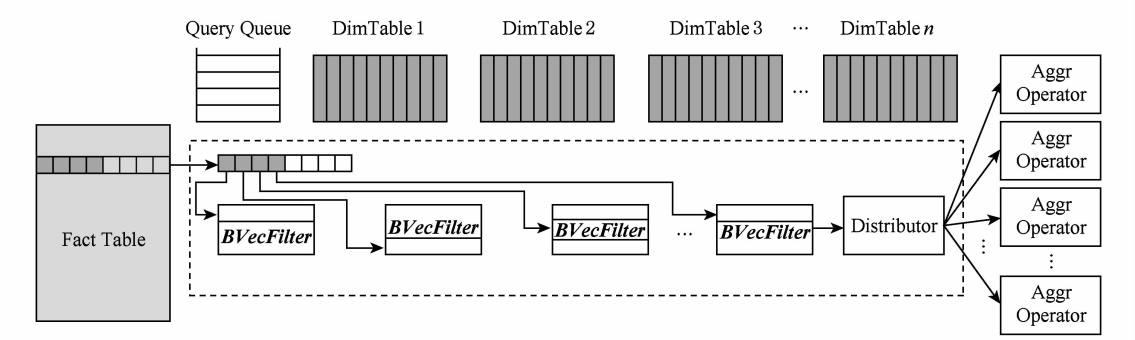

Fig. 5 dimensional filtering bit vector oriented concurrent OLAP.

图 5 基于维过滤位向量的并发 OLAP 查询处理

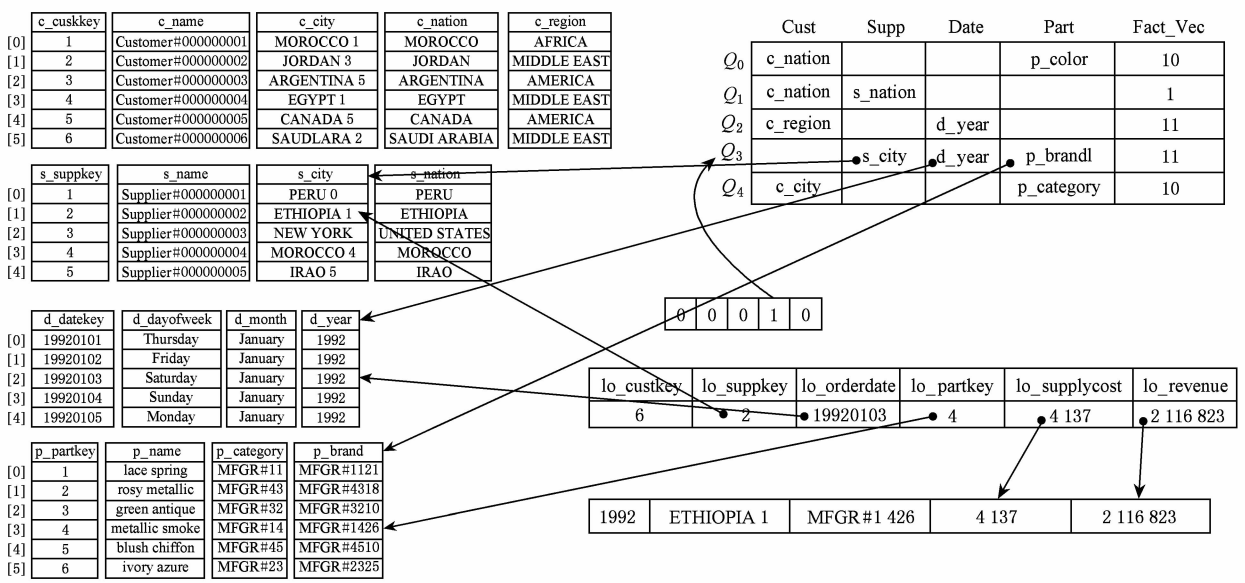

Fig. 6 Bit vector oriented grouping and aggregate computing.

图 6 基于位向量的分组聚集计算

位向量中“1”的位置映射查询队列中的查询记录,然后获得查询队列中记录的分组属性名称通过事实表记录中外键的值映射到对应的维表分组属性列中获取分组属性值,并根据事实向量访问对应的度量属性值,生成若干个连接输出记录,通过聚集计算中的 Hash 表进行分组聚集计算.

AIR 的维表记录内存映射访问机制实现了后物化的分组聚集计算,不需要在共享 Hash 表中存储较大的并发查询分组属性记录,实现了对共享事实表记录按并发查询位过滤向量结果的“按需访问”.

2.3.4 多线程并发 OLAP 操作

在基于维过滤向量的并发 OLAP 查询处理中,事实表扫描和多维过滤操作转化为串行处理过程,查询分发器对应不确定数量的并发聚集计算,若只并行化聚集计算则串行负载比例较高,内存带宽不能得到充分利用,查询的加速性能较低. 在多核处理器平台,如图 7 所示,我们将事实表按线程数量进行按位置分区(即按记录下标范围)逻辑分片,在每一个事实表分片上执行基于维过滤向量的并发查询处理,其中 n 个维表对应 n 个维过滤位向量 ***BVecFilter***,维过滤向量被各个线程共享访问,其大小为 $\frac{n}{8}\times(|D_1|+|D_2|+|D_3|+\cdots+|D_n|)$ 字节,当 n 个 ***BVecFilter*** 小于多核处理器 LLC(通常 20~45 MB)大小时,多线程并发 OLAP 查询处理在进行多维过滤时具有较好的 cache 局部性. 每个分区上的处理线程负责事实表分片上的并发查询处理任务,在完成全部事实表记录的分区聚集操作(partition aggr operator)后,通过全局聚集操作(global aggr operator)生成最终的并发查询结果.

2.3.5 AIR 算法对不同模式的适应性

星形模型是数据仓库的基础,也构成了多维数据

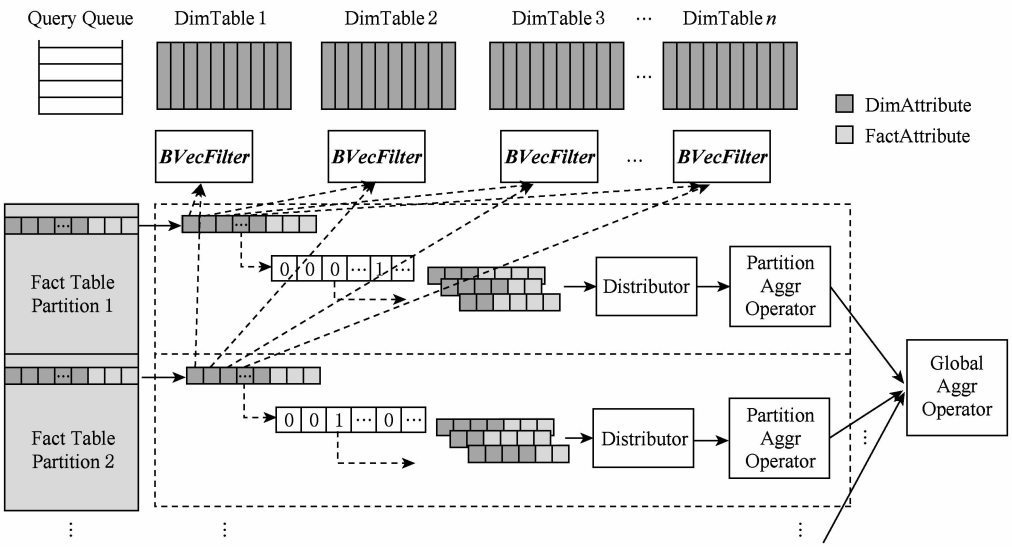

Fig. 7 Bit vector oriented multi-thread concurrent OLAP.

图 7 基于位向量的多线程并发 OLAP 查询处理

集中事实表与多个维表之间的映射关系.内存列存储和数据仓库代理键机制支持了事实表外键可以直接映射为维记录内存偏移地址的特性,即事实表外键在维表列上的 array index referencing,从而将传统 Pipeline 连接分解为简单的多维连接过滤和后物化模式的分组聚集计算,多维连接过滤操作转换为位向量计算,适合当前 CPU 的单指令多数据流(single instruction multiple data, SIMD)计算特性.雪花型模式的维表构成主-外键参照引用关系的维层次,可以通过 AIR 算法将底端层次维表上的谓词条件映射到顶端层次维表记录上,规范化为星形过滤向量结构,满足模板化 OLAP 查询处理的基础条件.

星形模型上的并发查询基于共享事实表扫描和共享多维过滤计算,复杂的数据仓库模式,如雪花模型 TPC-H、暴风雪模型 TPC-DS 等可以看作是多个星形模式上的协同计算,各事实表之间同样存在主-外键参照引用关系.在实践中,涉及多事实表的查询可以采用基于事实表间主-外键的 AIR 协同扫描技术完成并发 OLAP 查询时多事实表间的共享扫描;当查询包含多个独立星形结构之间的协同查询时,可以将一个星形模型并发 OLAP 计算操作符的输出流作为另一个星形模式并发 OLAP 计算操作符的输入,在星形子模式之间共享并发 OLAP 计算.

3 实验结果与分析

3.1 实验环境配置

实验硬件平台为一台 HP 服务器,配置有 4 块 Intel® Xeon® E5-4640@2.40 GHz(8-core,20 MB Cache)处理器、256 GB 内存、操作系统为 Windows 2012 64 位版,查询数据集为 Star Schema Benchmark[22].在 SSB 测试集中,扩展因子(scale factor, SF)为数据量单位,$SF=1$ 对应事实表记录数量为 6 000 000.实验数据集设置多组,大小分别取 SF 为 1,10,100、并发查询数量分别为 32,64,128,256,最大测试并发数量为 1 024.

在并发 OLAP 查询性能测试中,我们选择了 6 个对比对象:AIR OLAP 单线程查询处理、多线程并发执行的 AIR OLAP 查询处理、基于维过滤位向量的 AIR OLAP 并发查询处理、基于 MySQL 内存表的 OLAP 查询处理、开源列存储数据库 MonetDB、基于向量处理的内存数据库 Vectorwise.参考文献所述并发查询研究主要为原型系统,难于进行有效的性能对比.Vectorwise,MonetDB 等内存数据库在学术界与产业界可以作为性能标杆,我们以它们作为基础性能参照对象能够为后续研究提供可类比的性能.

实验中主要的测试指标是并发查询执行总时间、并发查询平均查询执行时间、查询吞吐性能(每小时执行查询的数量).实验使用服务器为对称多处理结构(symmetric multi-processing, SMP)多核 CPU 架构,使用 pthread 库函数实现并发查询的多线程并行处理,测试数据库的并发查询通过多线程 JDBC 访问实现,在测试时间中去掉 JDBC 初始化时间,只计算并发查询执行时间.

3.2 实验测试结果和分析

表1、表2给出了在 $SF=100$ 数据集下,并发查询数量为32,64,128,256时的并发OLAP查询执行总时间和平均查询执行时间对比.

AIR OLAP采用行式扫描,通过外键值-地址映射维表谓词处理机制,查询时间比较稳定,查询处理的主要代价是表扫描代价,串行执行时查询平均执行时间性能与MySQL内存表查询处理性能相当.当通过多线程并发执行AIR OLAP查询处理时,查询任务被并发地执行,但由于并发查询线程超过物理核心数量时产生线程切换代价以及并发查询处理时在LLC所产生的cache争用,并发AIR OLAP查询的加速比低于物理核心数量32,在256并发线程时达到最大加速比22.基于MySQL内存表的并发OLAP查询平均执行时间达到22 s,而列存储内存数据库MonetDB并发查询最小平均查询执行时间为4.63 s,体现了列存储相对于行存储的性能优势.基于向量处理的Vectorwise是当前性能最好的产品级内存数据库,当前为TPC-H测试中 $SF=100$ 和 $SF=300$ 数据集上的性能最佳的数据库系统[24],在并发OLAP查询时的最小平均查询执行时间为0.89 s,与并发AIR OLAP的平均查询执行时间相当.基于维过滤位向量技术的并发AIR OLAP的最小平均查询执行时间最短,仅为0.48 s,通过事实表共享扫描和并发维过滤技术进一步提高了并发查询的CPU处理效率.

Table 1 Comparison of Total OLAP Query Execution Time ($SF=100$)

表1 总OLAP查询执行时间对比($SF=100$)

Concurrent Query	Total Execution Time/ms					
	Serial AIR OLAP	Concurrent AIR OLAP	Bit Vector Oriented Concurrent AIR OLAP	MySQL Memory Table	MonetDB	Vectorwise
32	635 597	59 236	34 105.3	1 258 510	170 692	51 785
64	1 701 430	85 297.6	49 119	1 450 580	334 730	108 930
128	2 585 990	121 676	70 734	2 845 570	701 767	157 715
256	5 134 570	233 205	123 371	5 642 210	1 186 446	227 800

Table 2 Comparison of Average OLAP Query Execution Time ($SF=100$)

表2 平均OLAP查询执行时间对比($SF=100$)

Concurrent Query	Average Execution Time/ms					
	Serial AIR OLAP	Concurrent AIR OLAP	Bit Vector Oriented Concurrent AIR OLAP	MySQL Memory Table	MonetDB	Vectorwise
32	19 862.41	1 851.13	1 065.79	39 328.44	5 334.13	1 618.28
64	26 584.84	1 332.78	767.48	22 665.31	5 230.16	1 702.03
128	20 203.05	950.59	552.61	22 231.02	5 482.55	1 232.15
256	20 056.91	910.96	481.92	22 039.88	4 634.55	889.84

图8显示了不同并发度时的查询吞吐性能(每小时查询执行数量),其中MonetDB和MySQL内存表的查询吞吐性能趋于稳定,而并发AIR OLAP、基于维过滤位向量的AIR OLAP和Vectorwise的查询吞吐性能随并发度的增加而增加,当并发查询数量增加时能够相应地提高数据访问及计算的效率,能够较好地利用现代多核处理器的并行计算资源.

图9显示了基于维过滤位向量的AIR OLAP在数据集大小为 $SF=1$,$SF=10$,$SF=100$ 时,不同并发线程数量时执行的平均查询时间.在实验中,线程内并发查询数量与线程数量相同,线程数量增长时,并发查询数量也随之增长,维过滤位图宽度增长.图9所示的查询平均执行时间中,当线程(并发查询)数量为512时具有最少的平均查询执行时间,主要原因是512个并发查询对应的维过滤位向量长度为512 b(64 B),对应一个cache line访问,能够最大化cache line数据的利用率,内存访问数据利用率最高.

综上所述,AIR OLAP与当前主流的内存分析型数据库在OLAP算法实现上最大的不同是充分利用了数据仓库代理键的地址映射(array index referencing)特点,实现了基于事实表外键映射的连接索引,支持基于维表过滤位图的连接过滤和后物化的分组聚集计算,将OLAP查询处理过程中的查询

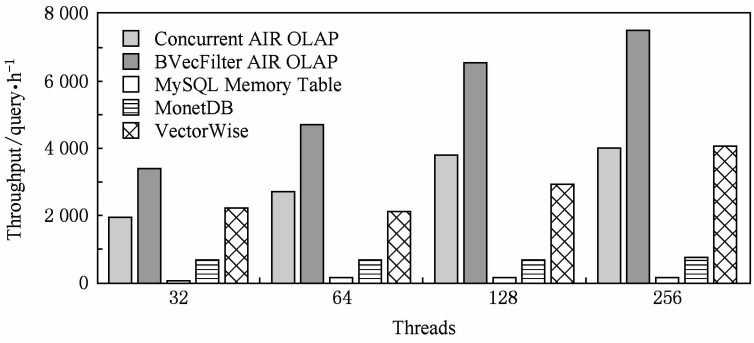

Fig. 8 Query throughput of concurrent query workload(SF=100).

图 8 并发查询吞吐性能(SF=100)

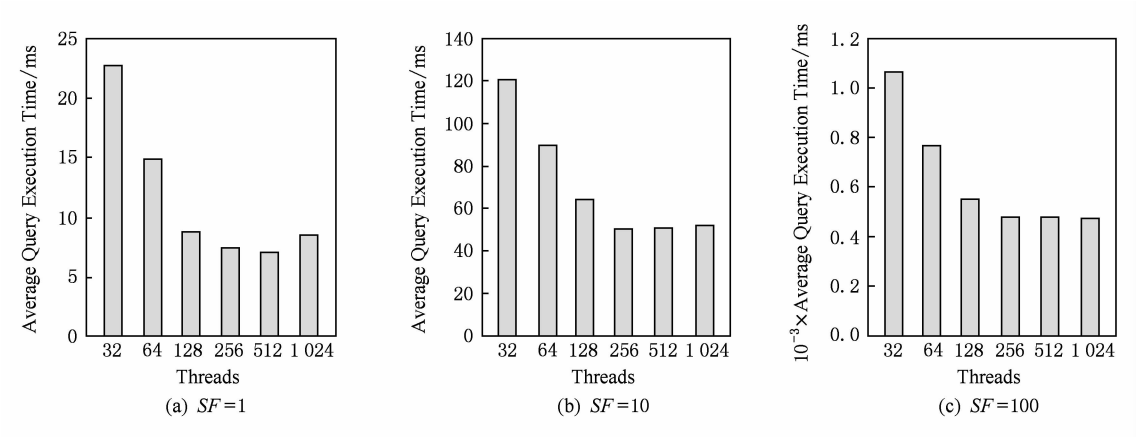

Fig. 9 Average concurrent query execution time.

图 9 并发查询平均执行时间

私有数据集转换为并发查询共享访问的较小的维过滤位向量和原始维表列,在并发查询处理时提高了数据访问的局部性,因此在并发 AIR OLAP 查询处理时有较好的数据局部性和吞吐性能.

4 结束语

AIR OLAP 查询处理中与并发查询优化相关的操作是并发数据访问和并发连接过滤计算:并发数据访问体现在并发查询共享事实表扫描操作,并发连接过滤计算体现在维表过滤位图映射由一位扩展到多位,充分利用 cache line 访问和计算更多的并发查询维过滤位,提高并发 OLAP 查询的整体性能.

本文研究范围定位于可以作为基础操作符的星形模式下的共享 OLAP 查询处理技术,可以作为通用 OLAP 查询处理的基础并发操作符. 与 Datapath, sharedDB, SWO[25] 等技术相比,AIR OLAP 算法首先基于数据仓库模式分解为星形模式集合,在星形模式上应用维过滤位向量技术优化共享扫描与连接过滤操作,达到局部最佳共享 OLAP 处理性能,在多个星形操作符之间再进行数据流共享优化,简化复杂模式下的并发查询共享优化技术. 在未来的工作中,我们将针对 TPC-H 和 TPC-DS 模式特点,研究多个共享操作符之间的协同并发查询优化技术.

参 考 文 献

[1] Boncz P A, Kersten M L, Manegold S. Breaking the memory wall in MonetDB [J]. Communications of the ACM, 2008, 51(12): 77-85

[2] Zukowski M, Boncz P A. Vectorwise: Beyond column stores [J]. IEEE Data Engineering Bulletin, 2012, 35(1): 21-27

[3] Sikka V, Färber F, Lehner W, et al. Efficient transaction processing in SAP HANA database: The end of a column store myth [C] //Proc of ACM SIGMOD 2012. New York: ACM, 2012: 731-742

[4] Oracle. Oracle exadata database machine [EB/OL]. [2015-03-11]. http://www.oracle.com/technetwork/database/exadata/overview/index.html

[5] Kemper A, Neumann T. HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots [C] //Proc of ICDE 2011. Piscataway, NJ: IEEE, 2011: 195-206

[6] IBM. Informix warehouse accelerator-performance is everything [EB/OL]. (2013-03-21) [2015-01-14]. http://www.iiug.org/library/ids_12/IWA%20White%20Paper-2013-03-21.pdf

[7] Zukowski M, Hèman S, Nes N, et al. Cooperative scans: Dynamic bandwidth sharing in a DBMS [C] //Proc of VLDB 2007. New York: ACM, 2007: 723-734

[8] Pirk H, Petraki E, Idreos S, et al. Database cracking: Fancy scan, not poor man's sort! [C] //Proc of the 10th Int Workshop on Data Management on New Hardware. New York: ACM, 2014: 1-8

[9] Qiao L, Raman V, Reiss F, et al. Main-memory scan sharing for multi-core CPUs [C] //Proc of VLDB 2008. New York: ACM, 2008: 610-621

[10] Raman V, Swart G, Qiao L, et al. Constant-time query processing [C] //Proc of ICDE 2008. Piscataway, NJ: IEEE, 2008: 60-69

[11] Unterbrunner P, Giannikis G, Alonso G, et al. Predictable performance for unpredictable workloads [C] //Proc of VLDB 2009. New York: ACM, 2009: 706-717

[12] Harizopoulos S, Shkapenyuk V, Ailamaki A. QPipe: A simultaneously pipelined relational query engine [C] //Proc of ACM SIGMOD 2005. New York: ACM, 2005: 383-394

[13] Kossmann D, Konrad S. Iterative dynamic programming: A new class of query optimization algorithms [J]. ACM Trans on Database Systems, 2000, 25(3): 43-82

[14] Giannikis G, Alonso G, Kossmann D. SharedDB: Killing one thousand queries with one stone [J]. Proceedings of the VLDB Endowment, 2012, 5(6): 526-537

[15] Candea G, Polyzotis N, Vingralek R. A scalable, predictable join operator for highly concurrent data warehouses [C] // Proc of VLDB 2009. New York: ACM, 2009: 277-288

[16] Balkesen C, Teubner J, Alonso G, et al. Main-memory Hash joins on multi-core CPUs: Tuning to the underlying hardware [C] //Proc of ICDE 2013. Piscataway, NJ: IEEE, 2013: 362-373

[17] Balkesen C, Alonso G, Teubner J, et al. Multi-core, main-memory joins: Sort vs. hash revisited [J]. Proceedings of the VLDB Endowment, 2013, 7(1): 85-96

[18] Zhang Yansong, Jiao Min, Wang Zhanwei, et al. One-size-fits-all OLAP technique for big data analysis [J]. Chinese Journal of Computers, 2011, 34(10): 1936-1947 (in Chinese)
(张延松,焦敏,王占伟,等. 海量数据分析的 One-size-fits-all OLAP 技术[J]. 计算机学报, 2011, 34(10): 1936-1947)

[19] Giceva J, Salomie T I, Schüpbach A, et al. COD: Database/operating system co-design [C/OL] //Proc of CIDR 2013. 2013 [2014-11-10]. http://people.inf.ethz.ch/troscoe/pubs/cidr13-cod.pdf

[20] Zhang Y S, Jiao M, Wang Z W, et al. Multi-core vs I/O wall: The approaches to conquer and cooperate [C] //Proc of WAIM 2011. Berlin: Springer, 2011: 467-479

[21] TPC-H Homepage [EB/OL]. [2015-03-01]. http://www.tpc.org/tpch/default.asp

[22] O'Neil P, O'Neil B, Chen X D. Star schema benchmark [EB/OL]. (2009-06-05) [2014-12-23]. http://www.cs.umb.edu/~poneil/StarSchemaB.PDF

[23] TPC. TPC-DS homepage [EB/OL]. [2015-03-01]. http://www.tpc.org/tpcds/default.asp

[24] TPC. TPC-H top ten performance [EB/OL]. [2015-01-23]. http://www.tpc.org/tpch/results/tpch_perf_results.asp?resulttype=noncluster

[25] Giannikis G, Makreshanski D, Alonso G, et al. Shared workload optimization [J]. Proceedings of the VLDB Endowment, 2014, 7(6): 429-440

Zhang Yansong, born in 1973. Associate professor in Renmin University of China. His main research interests include main memory database, OLAP and cloud computing.

Jiao Min, born in 1975. Senior engineer in Renmin University of China. Her main research interests include main memory database, OLAP and cloud computing.

Zhang Yu, born in 1977. PhD, associate professor in National Satellite Meteorological China Meteorological Administration. Her main research interests include data warehouse, GPU database and OLAP (zyszy@ruc.edu.cn).

Wang Shan, born in 1944. Professor and PhD supervisor in Renmin University of China. Senior member of China Computer Federation. Her main research interests include high performance database, main memory database and data warehouse.