值域(2^{32}–1)能够满足数据仓库维表键值的取值范围.考虑到内存数据库普遍采用数据压缩技术,我们在实验中并未使用 int_64 数据类型.在 CPU 平台和 Phi 平台通过不同线程并行度的测试,确定在 CPU 与 Phi 平台的线程数量设置为与物理线程数相同时获得最佳的连接性能,即:CPU 平台设置并行线程数为 40,Phi 平台并行线程数为 240.

3.1 CPU端连接算法性能对比分析

在实验中,我们重点考查连接算法在数据集相对于各级 Cache 为不同比例时的性能.

图 5 显示了基于代理向量参照访问的 AIR 算法、无分区哈希连接算法 NPO 和 radix 分区哈希连接算法 PRO 在不同连接数据集下的连接性能.其中,AIR 表示代理向量宽度为 8 位,AIR_int_16 和 AIR_int_32 分别表示代理向量宽度为 16 位和 32 位.测试中:较大的 S 表记录长度为 600 000 000,相当于 *SF*=100 时,SSB 和 TPC-H 中事实表长度;较小的 R 表取不同的行数,用于表示相对于各级 Cache 不同大小的连接数据集.纵坐标连接性能的指标为平均每记录连接消耗的 CPU 指令周期数量(cycle/tuple),横坐标表示以 AIR 算法为基准的代理向量大小与 Cache 的比例.如,50%×L2 表示代理向量大小为 50%×256KB(L2 Cache size),长度为 131 072 行.由于 AIR 算法使用代理向量下标作为隐式的键值,因此相同行数的情况下,NPO 和 PRO 的哈希表大小高于 AIR 算法使用的代理向量大小.

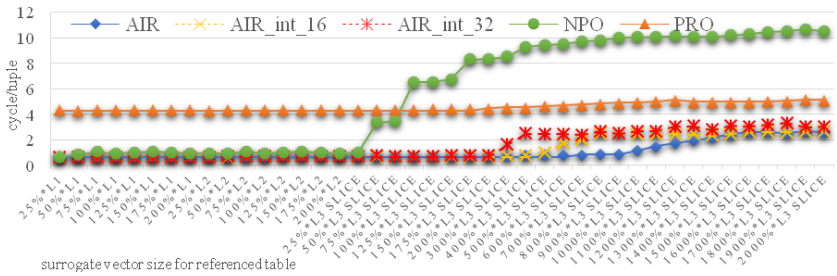

Fig.5 Cache adaptability of different join algorithms on CPU

图 5 CPU 端连接算法的 Cache 适应性

从整体性能曲线来看,Cache-Oblivious 的 NPO 算法是一种对连接数据集大小 Cache 敏感的算法,当连接数据集(哈希表)小于 Cache 时,算法具有较好的性能;而当连接数据集超过 Cache 时,算法执行时间显著增长.Cache-Conscious 的 PRO 算法则显示出连接数据集对 Cache 不敏感的特性,即:不论连接表的大小是否超过 Cache,都产生较为稳定的连接性能.这是因为在哈希连接的分区阶段,较大的数据集已划分为适合 Cache 大小的数据集,其后的哈希连接操作都是 Cache 优化的.

SSB,TPC-H,TPC-DS 中不同的维表与事实表的外键连接操作可以映射为图 5 中不同比例代理向量的连接操作.通过我们对 SSB,TPC-H,TPC-DS 的分析,维表通常较小,对于 OLAP 应用场景,NPO 算法在较大范围内优于 PRO 算法;PRO 算法是一种性能稳定的算法,主要适合大表连接操作,在与较小的表连接时有过度优化的缺点.

与 NPO 和 PRO 算法相比,AIR 算法表现出一种 Cache-Friendly 的平衡性能:当代理向量小于 Cache 时,AIR 算法像 NPO 算法一样自动获得较高的性能;当代理向量超过 Cache 时,AIR 算法像 NPO 一样增加了内存访问代价,但其性能衰减幅度极大地低于 NPO,并逐渐像 PRO 算法一样趋于稳定.

AIR 算法中,每次外键的代理向量参照访问只产生一个 Cache miss,而哈希连接则可能产生多个 Cache miss 以及相应的哈希处理 CPU 代价.因此,AIR 算法执行时间的上升阶段比 NPO 算法更为平缓.不同的代理向量大小决定了 AIR 算法执行时间上升期的起始位置,int_32,int_16 以及 int_8 性能曲线的斜率依次递减,执行时间上升阶段依次延长.

3.2 Phi协处理器端连接算法性能测试

由于 Xeon Phi 协处理器与 Xeon 处理器的兼容性,我们将 CPU 端的 AIR 算法通过 ICC 编译器编译为 Xeon

Phi 端执行的程序,将执行程序复制到 Phi 协处理器内存执行.对应的 NPO 与 PRO 的 Xeon Phi 协处理器端程序使用文献[2]提供的源码编译执行.我们首先执行第 3.1 节中的性能测试,图 6 显示了 AIR,NPO,PRO 分别在多核 CPU,Phi 协处理器上的查询性能曲线.由于 CPU 与 Phi 协处理器主频不同,因此我们使用每记录纳秒值(ns/tuple)作为查询性能指标.

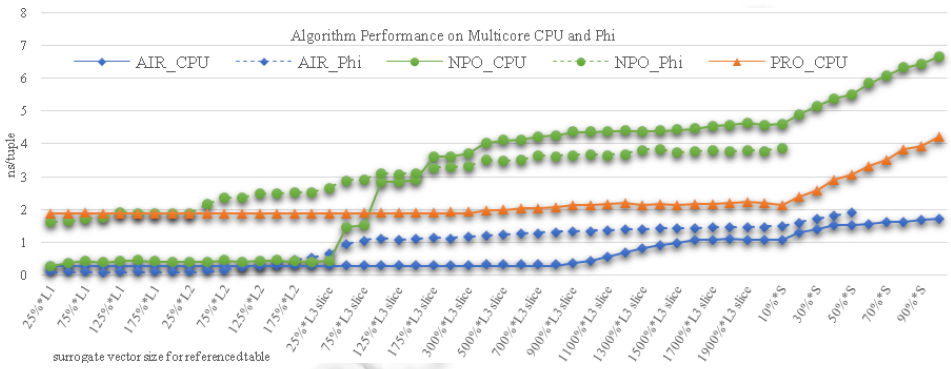

Fig.6 Cache adaptability of different join algorithms on Phi coprocessor

图 6 Phi 协处理器端连接算法的 Cache 适应性

NPO 算法对于低于 Cache 容量的哈希表连接性能较好,当连接表较大时,连接性能下降较快.连接表较小时,PRO 显示了稳定的查询执行性能;但随着连接表大小的增长,radix 分区代价随之增长,查询性能受分区代价的影响而显著下降,性能曲线上升幅度与 NPO 在大表连接时类似.AIR 在构建代理向量时代价较低,在大表连接时仍然保持较高的性能,查询执行时间增长缓慢,查询性能更加稳定.

图 6 中虚线为算法在 Phi 协处理器上的查询性能,当代理向量小于 L2 Cache 时,Phi 协处理器上的 AIR 性能优于 CPU.主要原因是:尽管 Phi 协处理器的主频低于 CPU,但 240 线程并行执行的性能优于 CPU 端 40 线程并行执行性能.Phi 协处理器只有 512 KB 的环形连接 L2 共享 Cache,不像 CPU 处理器拥有较大的 L3 Cache,当代理向量超过 L2 Cache 时,CPU 端的性能优于 Phi 协处理器端,主要原因是 CPU 的 Cache 访问优于 Phi 协处理器的并发多线程内存访问性能.NPO 算法在较小连接表时 CPU 端的性能更优,其主要原因是:Phi 协处理器大量并发的线程,增加了共享哈希表构建时的并发访问代价以及 Phi 协处理器主频与核心性能较低.当连接表较大时, Phi 协处理器的高并发访问表现了较好的内存访问性能,查询性能优于 CPU 端.PRO 算法的 radix 分区操作导致内存需求加倍,*SF*=100 的测试数据集在 Phi 协处理器上 S 表大小为 4.8GB,分区操作需要额外的 4.8GB 内存,直接导致内存空间不足而中断测试.在 Phi 协处理器算法设计中,除性能因素之外,Phi 协处理器有限的内存需要算法具有良好的内存效率,在 CPU 端性能表现较好的 PRO 算法的大数据处理能力显著低于 NPO 和 AIR 算法,其应用受到较大的制约.

3.3 Phi协处理器端连接算法性能对比分析

为进一步完整地对比 AIR,NPO 与 PRO 算法在 Xeon Phi 协处理器上的整体性能,我们缩小 S 表的大小,使 3 种算法能够完成全部的测试任务.经过测试,最终确定,在 Xeon Phi 协处理器 8GB 内存中支持的最大表为 200 000 000 行.我们将 S 表大小固定为 200 000 000 行,测试 AIR 算法代理向量按 L1 Cache 内、L2 Cache 内、L2 共享 Cache 内(注:Phi 协处理器的 L2 Cache 为 512KB,共 60 个)以及与 S 表行数 10%递增的比例设置 R 表行数.图 7 显示了 3 种算法在 Phi 协处理器上整体性能曲线,当 R 表哈希表小于 L2 Cache 时,NPO 算法的性能优于 PRO;当 R 表哈希表超过 L2 cache 时,PRO 性能优于 NPO,均保持比较稳定的性能;当 R 表超过 S 表大小 20%时,PRO 平均查询执行时间增长速度超过 NPO;当 R 表与 S 表大小相同时,PRO 算法与 NPO 算法的性能几乎相同.与 NPO 和 PRO 相对应,AIR 算法对 L1 Cache 大小较为敏感,当代理向量超过 L2 Cache 时,查询执行时间开始增长;代理向量大小在 Phi 协处理器共享 L2 Cache 大小范围内时,连接执行时间呈现非常缓慢的增长趋势;当

代理向量大小超过共享 L2 Cache 时,查询执行时间增速提高,但其增长速度显著低于 NPO 与 PRO.

在最大数据集连接操作中,NPO 算法在 Phi 协处理器上的性能优于 CPU,连接总时间分别为 1 227 785μs 和 1 353 713μs,PRO 算法在 Phi 和 CPU 上的执行时间为 1 188 729μs 和 841 234μs,AIR 算法在 Phi 和 CPU 上的执行时间为 407 087μs 和 327 758μs,PRO 与 AIR 算法在 Phi 上的查询性能略低于 CPU.需要注意的是:实验中使用两块 Intel Xeon E5-2650 v3@2.30GHz 10 核 CPU 和一块 Intel Xeon Phi 5110P@1.053 GHz60 核协处理器,以接近的价格获得了相近的连接操作性能.

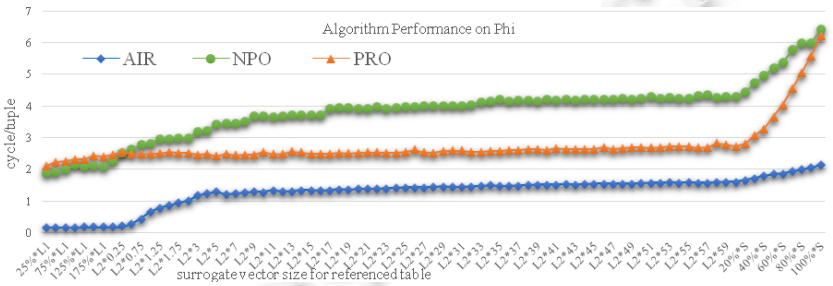

Fig.7 Comparison of different join algorithms on Phi coprocessor

图 7 Phi 协处理器端连接算法性能对比

4 结束语

随着 Xeon Phi 协处理器成为高性能计算的主流处理器,Xeon Phi 协处理器成为连接优化技术研究的新平台.本文的贡献体现在 3 个方面:首先,面向 OLAP 模式特点提出了代理键索引机制,并详细阐述了代理键索引机制的实现及维护技术;然后,通过深入的实验,揭示了基于代理向量参照访问的 AIR 算法的 Cache-Friendly 特征,揭示了 AIR 算法融合了 NPO 小数据和 PRO 大数据时的连接性能特征,验证 AIR 优于 NPO 和 PRO 算法的性能;最后,通过 Phi 协处理器平台上全面的性能对比测试,展现了 3 个算法各自的性能曲线特征,清晰地描述了各个算法在不同连接表大小下的性能特征,为 Phi 协处理器 OLAP 查询处理中连接算法的技术选择提供了详实的实验数据.

从整体实验结果来看,AIR 算法通过 OLAP 模式特点简化连接操作实现技术,通过代理向量参照访问减少内存随机访问数量和 CPU 计算代价,其算法性能与 Cache 大小、并行执行线程数量等硬件参数相关度较高,既能够适应多核 CPU 较大容量 Cache 的优化,也能适应众核协处理器较高并发线程的优化,具有较好的适应性.从总体来说,AIR 算法特点更加适合 Xeon Phi 众核协处理器弱化 Cache 和核心性能、强化并发多线程内存访问的特征,基于代理向量访问的操作简化了算法设计,提高了算法代码执行效率,并且相对于 NPO,PRO 等经典的数据库连接算法更加易于代码改写与平台迁移.

在未来的工作中,我们将继续探索在 Phi 协处理器平台上全面的 OLAP 查询优化技术,尤其是面向 PCI-E 带宽性能瓶颈的 CPU-Phi 协同 OLAP 优化技术.

References:

[1] http://www.expreview.com/44000.html

[2] Jha S, He BS, Lu M, Cheng XT, Huynh HP. Improving main memory Hash joins on Intel Xeon Phi processors: An experimental approach. Proc. of the VLDB Endowment, 2015,8(6):642−653. [doi: 10.14778/2735703.2735704]

[3] Polychroniou O, Raghavan A, Ross KA. Rethinking SIMD vectorization for in-memory databases. In: Proc. of the SIGMOD. New York: ACM Press, 2015. 1493−1508. [doi: 10.1145/2723372.2747645]

[4] Halstead RJ, Absalyamov I, Najjar WA, Tsotras VJ. FPGA-Based multithreading for in-memory Hash joins. In: Proc. of the CIDR. 2015.

[5] He J, Lu M, He B. Revisiting co-processing for hash joins on the coupled CPU-GPU architecture. Proc. of the VLDB Endowment, 2013,6(10):889−900. [doi: 10.14778/2536206.2536216]

[6] Morgan TP. Intel mates FPGA with future Xeon server chip. 2014. http://www.enterprisetech.com/2014/06/18/intel-mates-fpga-future-xeon-server-chip/

[7] Blanas S, Li Y, Patel JM. Design and evaluation of main memory Hash join algorithms for multi-core CPUs. In: Proc. of the SIGMOD. New York: ACM Press, 2011. 37−48. [doi: 10.1145/1989323.1989328]

[8] Balkesen C, Teubner J, Alonso G, Ozsu T. Main-Memory Hash joins on multi-core CPUs: Tuning to the underlying hardware. In: Proc. of the ICDE. Washington: IEEE Computer Society, 2013. 362−373. [doi: 10.1109/ICDE.2013.6544839]

[9] Albutiu MC, Kemper A, Neumann T. Massively parallel sort-merge joins in main memory multi-core data-base systems. Proc. of the VLDB Endowment, 2012,5(10):1064−1075. [doi: 10.14778/2336664.2336678]

[10] He BS, Yang K, Fang R, Lu M, Govindaraju NK, Luo Q, Sander P. Relational joins on graphics processors. In: Proc. of the SIGMOD. New York: ACM Press, 2008. 511−524. [doi: 10.1145/1376616.1376670]

[11] Yuan Y, Lee R, Zhang X. The Yin and Yang of processing data warehousing queries on GPU devices. Proc. of the VLDB Edowment, 2013,6(10):817−828. [doi: 10.14778/2536206.2536210]

[12] Pirk H, Manegold S, Kersten M. Accelerating foreign-key joins using asymmetric memory channels. In: Bordawekar R, Lang CA, eds. Proc. of the Int'l Workshop on Accelerating Data Management Systems Using Modern Processor and Storage Architectures (ADMS). 2011. 27−35.

[13] Zhang Y, Zhou X, Zhang Y, Zhang Y, Su M, Wang S. Virtual denormalization via array index reference for main memory OLAP. IEEE Trans. on Knowledge & Data Engineering, 2016,28(4):1061−1074. [doi: 10.1109/TKDE.2015.2499199]

[14] https://en.wikipedia.org/wiki/Surrogate_key

张宇(1977－),女,黑龙江绥化人,博士,副教授,主要研究领域为 GPU 数据库,数据仓库,OLAP.

陈红(1965－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据仓库,数据挖掘,传感器网络.

张延松(1973－),男,博士,副教授,主要研究领域为内存数据库,数据仓库,OLAP.

王珊(1944－),女,教授,博士生导师,CCF 会士,主要研究领域为高性能数据库,内存数据库,数据仓库.