内存 OLAP 数据库性能的 CPU 关键硬件影响因素研究

王沁垚[1),2),3)] 韩瑞琛[1),2),3)] 刘佳茹[1),2),3)] 张延松[1),2),3),4)]

[1)](数据库与商务智能教育部工程研究中心 北京 100872)

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

[3)](中国人民大学信息学院 北京 100872)

[4)](中国人民大学中国调查与数据中心 北京 100872)

摘 要 现代 CPU 在核心数量、cache 容量、带宽性能等方面快速增长，不同 CPU 在 chiplet 架构上的不同对数据访问延迟性能产生较大的影响。内存数据库的性能与 CPU 内存访问特性密切相关，本文对不同架构 CPU 平台的内存数据库性能进行了研究，分析数据库基准测试负载中的关键算子，并通过连接算子微基准对当前主流不同架构多核 CPU 平台进行深入的性能对比研究，揭示对内存数据库算子性能影响最大的三个关键硬件因素为有效带宽、有效 L3 cache 容量、核心数量。实验结果表明，优化 CPU 的微架构设计，提高 L3 cache 效率比增加核心对内存数据库性能产生更大的影响。

关键词 数据库基准；连接算子；NUMA 优化

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2026.01197

Research on CPU Key Hardware Influence Factors for In-Memory OLAP Database Performance

WANG Qin-Yao[1),2),3)] HAN Rui-Chen[1),2),3)] LIU Jia-Ru[1),2),3)] ZHANG Yan-Song[1),2),3),4)]

[1)](Engineering Research Center of Database and Business Intelligence (Renmin University of China), Ministry of Education, Beijing 100872)

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

[3)](School of Information, Renmin University of China, Beijing 100872)

[4)](National Survey Research Center at Renmin University of China, Beijing 100872)

Abstract Modern CPUs have seen rapid advancements in core count, cache capacity, and memory bandwidth. Differences in chiplet architectures across CPUs significantly impact data access latency. Since the performance of in-memory databases is closely tied to CPU memory access characteristics, this paper investigates the behavior of in-memory databases on various CPU architectures. We analyze key operators in database benchmark workloads and conduct an in-depth performance comparison of mainstream multi-core CPU platforms using a join-operator micro-benchmark. Our study identifies three critical hardware factors that most significantly affect in-memory database operator performance: effective memory bandwidth, effective L3 cache capacity, and core count. Experimental results demonstrate that optimizing CPU microarchitecture and improving L3 cache efficiency have a greater impact on in-memory database performance than simply in-

收稿日期：2025-08-24；在线发布日期：2026-01-28。本课题得到国家重点研发计划(No. 2023YFB4503600)、企业横向项目(面向新型数据库的特征量化分析及 Benchmark 构建技术合作项目)资助。**王沁垚**，硕士研究生，中国计算机学会(CCF)会员，主要研究领域为新硬件内存数据库技术。E-mail：qywang10@ruc.edu.cn。**韩瑞琛**，博士研究生，主要研究领域为内存数据库和新硬件数据库技术。**刘佳茹**，硕士，主要研究领域为 HTAP 数据库和内存数据库存储。**张延松**(**通信作者**)，博士，副教授，主要研究领域为数据仓库、内存数据库、GPU 数据库和新硬件数据库技术。E-mail：zhangys_ruc@hotmail.com。

creasing the number of cores.

Keywords database benchmark; join operator; NUMA-aware optimization

1 引 言

内存数据库是基于内存计算平台的高性能数据库技术的代表,算法设计多采用 hardware-conscious(硬件敏感)实现技术,数据库性能与硬件性能紧密相关。多核 CPU 技术在近年发展较快,以 Intel 和 AMD CPU 为例,核心数量从几十核心增长至 144 核[1]和 192 核,缓存容量从几十 MB 增长至 500MB,采用 12 通道 DDR5 内存带宽超过 500GB/s,存储容量达到 6TB[2]。硬件技术的发展为内存数据库提供了更加强大的并行计算性能、更低的内存访问延迟和更高的内存吞吐性能,为内存数据库的性能提升提供了强大的硬件支持。

核心数量的增长也增加了 CPU 架构的复杂性,如 chiplet 微架构、cache 共享方式、NUMA 配置的不同等均导致数据访问性能产生较大差异,需要数据库在底层算子实现技术上采用针对性的优化技术。另一方面,数据库负载通常呈现显著的局部性特征,其性能受 CPU 微架构及多种硬件配置参数的共同影响。明确哪些微架构特性与哪些关键硬件参数对数据库性能最为敏感,对于指导 CPU 硬件厂商进行体系结构设计具有重要价值。

数据库在不同架构 CPU 上具有不同的性能特征,NUMA 架构的广泛采用需要数据库在 hardware-conscious 算法基础上从 CPU 硬件级缓存优化扩展到适应 NUMA 内存访问优化的 NUMA-aware 优化技术,chiplet 架构的广泛应用和 chiplet 微架构的差异提供了不同粒度的数据局部性;已有研究证明,分布式数据库通过充分利用数据局部性特征的优化技术获得了较好的性能收益[3],本文将视角聚焦于集中式数据库。从软件维度来看,已有研究较好地指导了在新硬件发展过程中如何通过软硬件一体化设计思想深入挖掘硬件的性能潜力,而且这个过程随着硬件技术的不断迭代更新而持续深入。从硬件维度来看,一方面需要更加深入地研究来揭示相似 CPU 硬件配置下不同硬件微架构对数据库性能的影响,另一方面需要通过简化的微基准测试性能反映数据库的整体性能,从而更好地应用于 CPU 设计阶段不同硬件架构技术路线的性能评估,降低性能测试的复杂度和成本。

本文致力于深度剖析内存数据库的性能特征,主要探讨以下几个科学问题:(1)CPU 的主要硬件参数中对数据库查询性能影响最大的是哪些因素?(2)相互影响的 CPU 硬件因素中,微架构设计的不同如何影响数据库的性能?(3)如何通过轻量的算子微基准映射数据库的查询性能以及在不同 CPU 平台上性能与硬件关键影响因素之间具有哪些相关性?为解决上述科学问题,本文对比了最新的 9 个不同架构的 CPU 平台、不同数据库负载中不同算子的性能特征,提炼出在软件维度对数据库性能影响最大的关键算子,分析其对 CPU 硬件的主要依赖因素,评估当前 CPU 硬件指标对数据库性能的影响。本文主要贡献是:(1)通过数据库负载特征分析抽取具有综合 CPU 硬件关键影响因素分析的轻量化连接算子,设计了 NUMA-aware 连接算子微基准,通过连接微基准对比了当前主流的 9 个不同架构 CPU 平台的性能特征;(2) 通过全面的实验验证提高内存带宽性能是提升内存数据库性能的上限,提高有效 L3 cache 容量能够在相同或较少核心的情况下获得更高的性能;(3)对数据库负载而言,CPU cache 微架构的效率相对于核心数量对性能有更大的影响,为提高核心数量牺牲 L3 cache 容量或访问效率的微架构对数据库性能提升产生不利的影响。

2 相关工作

2.1 硬件感知查询优化技术

OLAP 是以连接为中心的分析处理任务,连接操作是数据库中最复杂、代价最大的算子之一,连接优化也是数据库热点研究问题。CPU 硬件的差异化引出了学者对连接算法软、硬件一体化优化技术的探索,研究主要集中在如何面向 CPU 硬件架构特征优化连接算法性能,即硬件感知的优化技术。不同优化算法与 CPU 硬件架构的相关度差异较大,如内存数据库经典的向量化查询处理技术主要优化 L1 cache 性能[4],radix 分区哈希连接算法主要优化私有 L1/L2 cache 及 TLB cache 的局部性[5],无分区哈希连接算法基于硬件预取及 L3 共享 cache

优化连接性能[6],随着 NUMA 架构的广泛应用,面向 NUMA 存储访问延迟差异的 NUMA-aware 技术[7-9]进一步提高了连接算法的内存访问局部性。数据库硬件优化算法思想的主要目标是提升热数据在多级存储硬件架构中的数据访问层次,减少数据访问延迟,提升各级 cache 的数据访问局部性。硬件感知的连接优化技术主要集中在无分区哈希连接和 radix 分区哈希连接的性能比较上,后者被广泛证明有更高的性能,但主要存在三个不足:(1)无法利用 L3 容量持续增长的性能红利:radix 分区哈希连接主要利用私有 cache 进行分区和 in-cache 连接,持续增长的 L3 cache 缩小了其性能优化窗口[10];(2)分区内存开销较大:分区操作需要额外的内存开销,降低了昂贵内存的利用率,尤其在 GPU 内存连接时其有限高带宽内存利用率的不足极大损害了高性能计算硬件的资源利用率和处理能力[11];(3)基于连接索引的向量连接算法进一步优化了无分区哈希连接算法的存储和计算效率,相对 radix 分区哈希连接算法能够更好地利用 L3 cache 增长的性能红利,在大表连接时接近甚至优于 radix 分区连接算法[10-12],在 NUMA-conscious 优化连接算法中也具有较好的性能-存储效率综合性能[9]。

数据库的硬件优化算法设计通常基于一定的硬件假设,如 cache 容量较小,或 NUMA 延迟较高。近年硬件技术的发展主要体现在核心数量、内存带宽、cache 容量的大幅度增长,如 AMD Zen4[13]、Zen5(https://www.amd.com/en/products/processors/server/epyc/9005-series.html)CPU 最大核心数量达到 192 核,最大 L3 cache 容量为 512MB,Intel 第五代 EMR[14] 和第六代 Granite Rapids CPU L3 cache 容量最大达到 480MB。NUMA 架构在微观视角上将集中式服务器看作是轻量 SN(shared-nothing)和 SE(shared-everything)的融合而应用分布式数据库技术[15-16]。同时,多 chiplet 架构的普及进一步扩展了存储层次,基于核心和 chiplet 粒度的分布式数据库部署策略可以更好地应用现代 CPU 的存储局部性特征[3],但当前研究是在 chiplet 架构上应用分布式数据库技术,牺牲了一定的数据共享访问性能和存储效率。面向扩展存储层次的优化易于在分布式数据库上进行验证,但对集中式内存数据库的硬件优化探索目前尚没有相关研究。

2.2　数据库性能基准与关键算子

基准测试是评估数据库性能的重要手段,其中 TPC-C 用于评估数据库的事务处理性能,TPC-H 和 TPC-DS 用于评估数据库的分析处理性能。在 TPC-H 和 TPC-DS 查询中包含了大量的数据库算子和不同选择率约束,可以评估数据库在不同数据规模、不同查询类型、不同负载特征下的综合性能。数据库由多种不同类型算子组成,不同算子的实现技术在不同架构 CPU 上的性能有较大的差异,一方面体现了不同算子与 CPU 不同硬件配置的相关性的差异,另一方面体现了不同架构 CPU 对数据库性能影响的程度不同。我们在前期工作中面向 OLAP 负载设计了选择、投影、连接、分组、聚集、星型连接、多维计算等算子微基准和基于 SSB 的查询宏基准,对两代、三种架构、五个 CPU 平台进行了全面的性能分析对比研究[10],以连接为基础的算子具有强局部性特征,对 cache 访问效率、核心数量、带宽综合性能较为敏感,与硬件性能关联性最强。为进一步聚焦于数据库性能关键影响因素和简化 CPU 设计阶段模拟测试的复杂度,我们以连接算子微基准作为数据库性能代表性算子,针对现在的 CPU 架构设计模式,提出了 NUMA-conscious 优化的微基准,更好地对比研究了 CPU 硬件因素对于内存数据库的性能影响。从 CPU 硬件厂商的视角来看,CPU 硬件参数之间存在一定的互相制约关系,核心数量的增长带来 cache 容量的增长,也可能在 cache 有效容量和效率方面做出一定的妥协设计,因此需要进一步探索数据库性能与 CPU 硬件架构设计的相关性,确定关键影响因素,通过代表性算子评估 CPU 硬件设计对数据库性能的优化程度。

2.3　主流 CPU 架构

AMD 与 ARM CPU 采用 chiplet-based 架构,由物理分离的计算核心 chiplet 构成 CPU 多核计算资源,chiplet 之间通过内部通道将 L3 cache 互联。以 AMD CPU 为例,一个 chiplet 由 8 个 zen 4 核心构成,共享 32MB L3 cache,AMD_Genoa_96C CPU 由 12 个 chiplet 构成。Chiplet 架构易于扩展核心数量,但在 L3 cache 上产生物理局部性。

Intel 采用 Tile-based 架构,CPU Tile 中集成了计算核心和内存访问控制器,通过逻辑统一 mesh 架构互联各核心的 L3 cache slice,在 cache 有效容量和内存访问延迟性能上相对较好。

3　OLAP 数据库性能评估

为探索数据库性能与 CPU 硬件参数之间的相关

性,本文采用宏基准与微基准两种性能测试评估方法。

(1) 宏基准采用数据库的工业测试基准,其优势在于能够评估数据库在不同 CPU 平台上的综合性能。然而,其不足之处主要体现在三个方面:首先,难以对影响数据库性能的关键因素进行深入剖析;其次,数据规模的离散性导致其难以揭示 CPU 硬件配置参数与核心算子性能之间的完整依赖关系;最后,总体性能高度依赖已有数据库的具体实现技术,难以覆盖并评估更优的算法设计。

(2) 微基准采用连接测试基准,优点是聚焦于对 OLAP 查询性能影响最关键的连接算子,通过优化的算法提高算法性能并减弱非关键影响因素对算法性能的影响,更深入地揭示 CPU 硬件配置参数与连接性能的关键影响因素,基准测试更加轻量,应用场景更加广泛,可以扩展到 CPU 设计阶段;不足之处在于当前的工作聚焦于最具有代表性的连接算子性能,功能覆盖较小,在未来的工作将进一步扩展微基准算子,构建完善的 CPU 性能测试微基准算法集。

3.1 OLAP 数据库基准测试

宏基准测试的目的是分析数据库负载中对性能最关键的影响因素,将研究聚焦于少数关键算子,降低研究的复杂度。本文通过内存数据库 Hyrise[17] 上不同数据量的 TPC-H 和 TPC-DS 基准测试分析对数据库性能最关键的算子,通过算子执行时不同 CPU 硬件配置参数的相关性评估 CPU 对数据库性能影响最大的因素。Hyrise 是一个用于研究目的的开源内存数据库系统(https://hpi.de/plattner/projects/hyrise.html),对 TPC-H 和 TPC-DS 基准测试有良好的支持。

在宏基准测试中,设计了在 Intel、AMD、ARM 三个代表性 CPU 平台上的 Hyrise TPC-H 和 TPC-DS 基准测试,通过火焰图定位数据库在负载执行查询时间较长的算子,发现尽管数据规模不同,但内存数据库中各个算子执行时间的相对占比是相同的。执行时间占比前 10 的算子如表 1、表 2 所示。在 TPC-H 基准测试中,Hyrise 物化中间结果未包含在其他算子调用栈中而单独列出,在 TPC-DS 基准测试中则包含在 JoinHash 算子中,在表 2 中单独列出但并未算在执行时间累计中。

宏基准 Benchmark 按扩展因子生成预设比例大小的表,表 3 显示了 TPC-H、SSB 和 TPC-DS 主键表连接向量在不同数据规模下的大小(宽度为 1B),采用基准测试时大量查询负载仅能覆盖较小的向量规模,对 CPU cache 效率的测试范围较小。

表 1 TPC-H 基准测试算子执行时间占比

算子	执行时间占比(%)		
	Intel 平台	AMD 平台	ARM 平台
TableScan	25.9	23.7	23.8
materialize_input	24.5	22.3	24.4
AggregateHash	20.6	23.3	24.4
JoinHash	18.6	20	18.5
Projection	9.5	9.7	8.0
Sort	0.3	0.3	0.3
Validate	0.3	0.4	0.4
MultiPredicate JoinEvaluator	0.2	0.2	0.2
GetTable	0	0	0
UnionPositions	0	0	0

表 2 TPC-DS 基准测试算子执行时间占比

算子	执行时间占比(%)		
	Intel 平台	AMD 平台	ARM 平台
JoinHash	71.3	71.4	71.1
materialize_input	60.8	59.0	59.2
TableScan	15.6	15.1	14.5
AggregateHash	8.2	8.4	9.6
Projection	2.0	2.5	2.2
Sort	1.5	1.3	1.3
MultiPredicate JoinEvaluator	0.8	0.7	0.8
UnionPositions	0.5	0.4	0.4
Validate	0.1	0.2	0.1
GetTable	0	0	0

表 3 基准测试集连接向量大小

		连接向量大小(MB)			
		SF=100	SF=300	SF=1000	SF=10000
TPC-H	customer	14.31	42.92	143.05	1430.51
	part	19.07	57.22	190.73	1907.35
	supplier	0.95	2.86	9.54	95.37
	nation	0.000024	0.000024	0.000024	0.000024
	region	0.000005	0.000005	0.000005	0.000005
SSB	customer	2.86	8.58	28.61	286.10
	part	1.34	1.72	1.91	2.67
	supplier	0.19	0.57	1.91	19.07
	date	0.0024	0.0024	0.0024	0.0024
TPC-DS	call_center	0.00003	0.00003	0.00004	0.00005
	catalog_page	0.0195	0.0248	0.0286	0.0381
	customer	1.9073	4.7684	11.4441	61.9888
	customer_address	0.9537	2.3842	5.7220	30.9944
	customer_demographics	1.8318	1.8318	1.8318	1.8318
	date_dim	0.0697	0.0697	0.0697	0.0697
	household_demographics	0.0069	0.0069	0.0069	0.0069
	income_band	0.00002	0.00002	0.00002	0.00002
	item	0.195	0.252	0.286	0.383
	promotion	0.00095	0.00124	0.00143	0.00191
	reason	0.00005	0.00006	0.00006	0.00007
	ship_mode	0.00002	0.00002	0.00002	0.00002
	store	0.00038	0.00077	0.00096	0.00143
	time_dim	0.08240	0.08240	0.08240	0.08240
	warehouse	0.000014	0.000016	0.000019	0.000024
	web_page	0.0019	0.0025	0.0029	0.0038
	web_site	0.00003	0.00003	0.00004	0.00005

在 TPC-H 基准测试中,表扫描、中间结果物化、哈希聚合、哈希连接算子占比接近 90%,前两个算子主要涉及内存读、写操作,是内存带宽依赖型算子(总占比接近 50%),后两个算子依赖哈希表的读、写访问性能,是 cache 依赖型算子(总占比接近 40%)。TPC-DS 基准测试中,哈希连接、表扫描、哈希聚合算子执行时间占比接近 95%,cache 依赖型算子总占比接近 80%,带宽依赖型算子占比在 15%左右,三个平台各算子占比相似。

通过宏基准的执行时间分布规律可以归纳出数据库对 CPU 硬件的主要需求:提升内存带宽性能以提升顺序数据读写性能、增加 cache 容量以降低随机读写延迟、增加核心数量以提升并行计算性能。

哈希连接和哈希聚合算子依赖于哈希表的随机读、写访问性能,其性能主要取决于哈希表大小相对于 cache 能否有效缓存以降低内存随机读写延迟。连接算子中哈希表大小取决于基准数据集中维表或主键表大小、选择率高低和扩展因子大小。聚合算子中哈希表大小主要取决于分组大小,OLAP 查询通常为低势集分组查询,如 SSB 基准中分组大小在 1～800,TPC-H 的 Q1、Q4、Q5、Q7、Q8、Q9、Q12、Q21、Q22 分组大小低于 500,Q16 分组低于 30000,Q3、Q10、Q13、Q18 分组大小依赖于主键大小。哈希连接和聚合操作的性能依赖于各级 cache 的容量,当哈希表较小时依赖于核心私有的 L1/L2 cache,当哈希表较大时依赖 L3 cache 及内存访问性能。连接与聚合算子在数据访问特征上是相似的,都是在表扫描基础上的哈希表访问,同时包含了弱局部性数据集和强局部性数据集访问特性,可以代表数据库负载综合数据访问特征,与 CPU 的内存带宽性能、cache 容量与效率、核心数量都有较强的相关性。

3.2　连接微基准测试

在基于宏基准测试的数据访问特性的基础上,本文进一步聚焦于连接算子微基准,通过简化的连接微基准简化 Benchmark 测试中数据库系统的复杂性,降低在多平台测试的成本,相对于数据库系统固定的实现算法可以应用更加灵活和最新的算法设计,扩展对 CPU 硬件的利用率。

3.2.1　向量连接算子

相关研究证明,radix 哈希连接算法性能优于无分区哈希连接及排序-归并连接算法[18]。而与无索引连接算法相对,向量连接(文献[12]中称为 AIR 算法)算法是一种面向主-外键连接的索引连接算法,通过代理键索引将主键映射为内存记录地址,从而使外键成为天然的连接索引,消除了传统哈希索引中的哈希值计算、连接键值匹配和线性探测等计算开销,将复杂的哈希表简化为精简的连接向量,并且进一步通过压缩技术缩小连接向量的大小,提高连接向量探测时的 cache 局部性。

本文采用对 CPU 硬件特性更为敏感的 Vector-Join 向量连接微基准评估 CPU 关键硬件影响因素,向量连接算子执行过程如图 1 所示。R 表 payload 列通过多线程映射为连接向量(图中 vector index),S 表 key 映射为连接向量上的索引,通过值-地址映射直接访问连接向量相应单元,完成 sum(R. payload+S. payload)聚集计算。

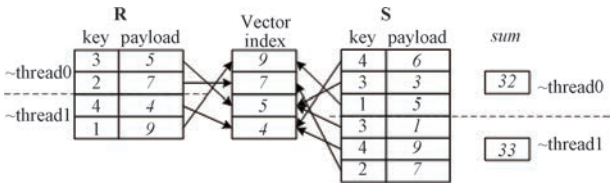

图 1　向量连接算子示例[19]

向量连接算子的内存访问行为包括:内存顺序扫描(S 表扫描)、内存随机访问(连接向量探测)和聚合计算(sum 计算),分别依赖于内存带宽性能、cache 访问效率和多核并行计算能力,这三种依赖刚好代表了 OLAP 数据库负载的关键硬件依赖路径。

向量连接微基准采用连续负载测试方法,固定 S 表大小为 2^{30},相当于 TPC-H 和 SSB 基准中 SF=200 数据集大小,R 表从 2^0,2^5,2^6 按指数连续增长到 2^{30},最大连接向量 2GB,覆盖 OLAP 查询中从小表连接到极大表连接场景,较大范围的连接向量可以测试连接向量在 L1、L2、L3、内存中访问的性能,评估不同 CPU、不同 cache 容量和架构上的随机数据访问性能。

3.2.2　NUMA 架构向量连接策略

在向量连接算子执行过程中主要包含两种内存访问模式,大表 S 上的顺序扫描操作和小表 R 连接向量上的随机访问操作,前者的效率由内存带宽和 NUMA 访问延迟决定,需要保证内存访问的 NUMA 局部性,减少跨 NUMA 访问延迟;后者受内存访问延迟和 cache 效率影响,即向量探测操作中 L3 cache 可以在一定范围内降低跨 NUMA 访问延迟的影响。

NUMA 架构 CPU 上的向量连接算法设计主要考虑以下几种场景:

(1) NUMA-oblivious 向量连接

忽略 NUMA 架构对向量连接算子的影响,对存储和计算没有特殊约束,采用操作系统默认的内存分配模式和多线程任务调度方法,算法不需要了解 CPU 硬件参数特征,实现自适应算法执行。

(2) NUMA-aware partitioned 向量连接

为保证 NUMA 访问局部性,对连接表 R 和 S 按连接键和 NUMA 分区数量进行分区,保证每对 R 和 S 分片存储在相同的 NUMA 节点,并且绑定 NUMA 节点核心上的线程执行连接操作,消除跨 NUMA 访问延迟。NUMA 分区技术需要根据 CPU 的 NUMA 配置参数确定每个 NUMA 节点及与之对应的 CPU,在线程分配上需要将 CPU 与 NUMA 节点绑定,NUMA 分区操作增加了额外的处理代价和内存空间开销,降低了昂贵内存资源的利用率,在 CPU 数据库研究中通常受有限内存容量的约束而较少使用分区优化技术。NUMA 分区连接优化技术需要权衡内存利用率和性能两个维度的综合收益。

(3) NUMA-conscious 向量连接

本文提出的 NUMA-conscious 向量连接算法采用混合 NUMA 存储访问技术,即 R 表采用 NUMA-oblivious 存储与处理技术,S 表采用 NUMA-aware 分片存储和 NUMA-conscious 并行扫描技术。该方法保证了大表扫描时的 NUMA 局部性,R 表连接向量访问通过 L3 cache 进行优化,大表连接时会产生跨 NUMA 访问延迟。

三种方法依赖不同的硬件发展趋势假设:第一种方法依赖于 CPU 的 L3 cache 容量增长能够满足大部分数据库连接需求和跨 NUMA 访问延迟逐渐缩小的硬件假设,通过统一的算法自适应未来的 CPU 平台,简化数据库算法设计;第二种方法依赖于未来跨 NUMA 访问延迟增大,L3 cache 容量有限,内存容量较大,支持以较大的空间代价优化 NUMA 访问局部性;第三种方法依赖于跨 NUMA 访问延迟不可忽视,L3 cache 容量较大,内存资源利用率要求较高,通过核心和内存通道数量增长降低并行内存访问延迟。

基于在存储空间效率及多表连接的流水线处理策略方面的考量,OLAP 数据库系统通常较少采用分区策略[11,20],本文主要对 NUMA-conscious 和 NUMA-oblivious 向量连接算法进行不同架构 CPU 平台的对比测试,分析 CPU 对向量连接算子性能的关键硬件影响因素,评估适应数据库性能需求的未来硬件技术。

前期研究[9]对比了不同的 NUMA 优化算法,基于协同分区的 NUMA 连接算法在性能上与方法二差异较小,但存储空间开销较大,本文中 NUMA-conscious 向量连接算法采用 NUMA-aware 存储策略,大表 S 根据 NUMA 节点数量分片存储在不同 NUMA 节点,R 表采用跨 NUMA 共享存储访问策略。每个 NUMA 节点的 CPU 线程与 NUMA 节点上的 S 表分片绑定,实现 NUMA 并行扫描,并在扫描时共享访问 R 表生成的连接向量。该方法一方面具有较为平衡的性能-存储效率,另一方面更加适合未来 L3 cache 容量增长、NUMA 延迟降低、核心数量增长等硬件发展趋势,达到自适应优化的目标。

4 实验与分析

实验对比了 Intel、AMD、ARM 三种不同类型、9 个不同配置型号的 CPU 平台上向量连接算法的性能,通过向量连接微基准的性能分析现代 CPU 架构中对数据库性能影响的关键因素有哪些。

4.1 硬件配置

连接微基准测试中使用了 AMD、Intel 和 ARM 三种不同架构的 CPU,共计 9 个不同型号,相应的 CPU 配置信息如表 4 所示。

表 4 CPU 和硬件配置

CPU 配置	AMD_Genoa_32C	AMD_Genoa_64C	AMD_Genoa_96C	Intel_EMR_64C	Intel_GR_96C	Intel_GR_128C	Intel_SF_6766e_144C	Intel_SF_6780e_144C	ARM_80C
CPU model	AMD EPYC 9354	AMD EPYC 9534	AMD EPYC 9654	Intel Xeon 8592+	Intel Xeon 6972P	Intel Xeon 6980P	Intel Xeon 6766E	Intel Xeon 6780E	ARM XXX
Cores/Socket	32	64	96	64	96	128	144	144	80
Threads	128	256	384	256	384	512	288	288	160
L1D cache(KB)	64	64	64	64	48	48	32	64	128
L2 cache(MB)	2	2	2	4	2	2	4	2	2.56
L3 cache(MB)	256	256	384	320	480	504	108	108	280
L3 cache/NUMA	128	128	192	160	480	168	108	108	140
NUMA nodes	4	4	4	4	2	6	2	2	4
Bandwidth(GB/s)	460.8*2	460.8*2	460.8*2	358.4*2	600*2	600*2	400*2	409.6*2	—

AMD Genoa 系列 3 款第四代 CPU 除核心数量外其他参数几乎相同,实验主要对比核心数量对性能的影响。Intel 系列 5 款 CPU 包括第五代 EMR、第六代 P 核 Granite Rapids 和 E 核 Sierra Forest CPU,在核心数量、cache 参数、NUMA 数量上有较大的差异,实验主要关注核心数量、NUMA 配置、cache 容量对性能的影响。ARM 最新 80 核 CPU 在主要参数上属于中等配置,主要对比 ARM 和 x86 架构 CPU 之间的性能差异。测试平台的操作系统版本、编译器版本相近,NUMA 配置采用实际配置模式,每个算子连续执行 5 次取稳定的最短时间作为查询时间。

4.2　NUMA-conscious 连接算子性能

实验中首先测试了 9 个 CPU 平台 NUMA-conscious 和 NUMA-oblivious 向量连接算子性能。如图 2 所示,实线代表前者算法,虚线代表后者算法,纵坐标轴代表吞吐性能(GTuple/s)。总体来看 NUMA-conscious 向量连接算法性能有显著的提升,相较于 Intel 平台,AMD 与 ARM CPU 平台上这两种算法的性能表现差异更为悬殊,EMR CPU 平台较 Granite Rapids 和 Sierra Forest CPU 平台性能差距大。连接负载前一阶段 R 表产生的连接向量较小,具有较强的 cache 局部性,向量连接操作主要是表扫描和 in-cache probe 操作,在 AMD 和 ARM CPU 平台这个阶段两个算法性能曲线比较接近,在 Intel 平台有较小的差异,说明单纯的 NUMA-aware 扫描优化对向量连接算子性能影响较小。在负载的中间阶段,由于 R 表产生的连接向量大小逐渐超出 L3 cache 容量而产生 in-memory probe 延迟,两个算法性能曲线之间的差距逐渐加大,在表扫描和连接向量探测两个阶段产生的 NUMA 访问延迟叠加导致连接性能下降。当 R 表继续增大至最大行数时,L3 cache 缓存优化效率降到最低,随机 NUMA 内存访问延迟占主导,两种算法性能逐渐接近。

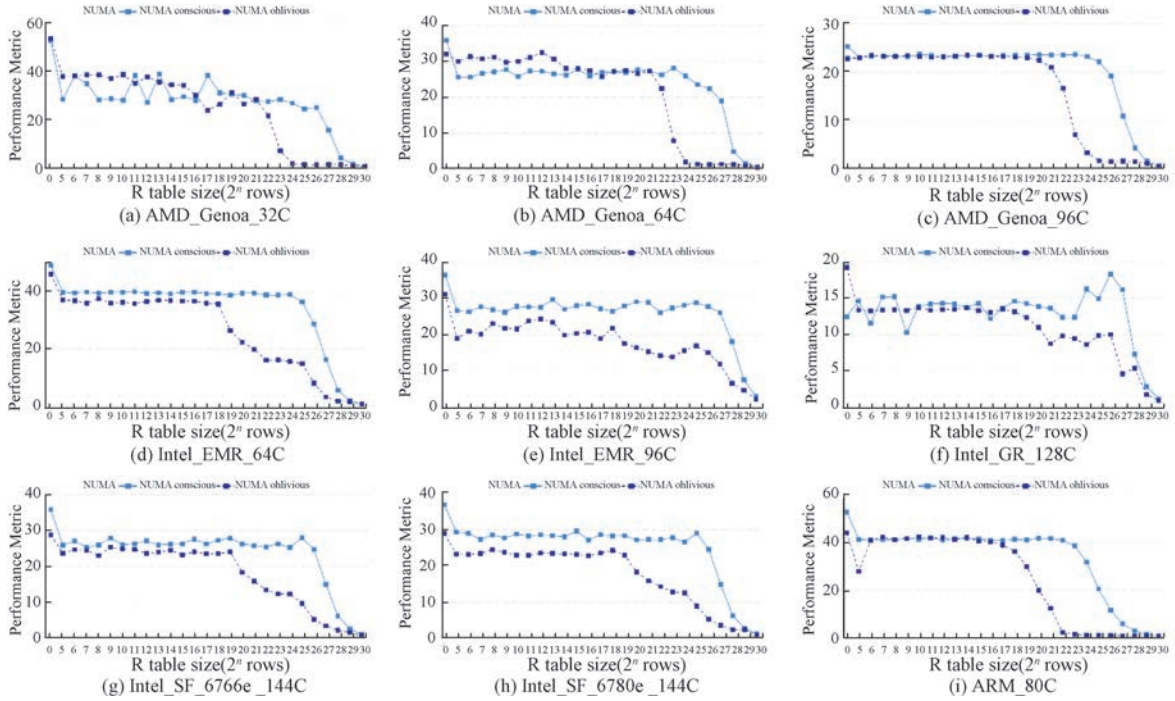

图 2　NUMA-conscious 和 NUMA-oblivious 向量连接算子性能

实验表明,表扫描操作产生的顺序 NUMA 访问延迟对向量连接算法性能的影响低于向量连接阶段的 NUMA 随机访问延迟的影响,优化连接向量探测阶段的 cache 和内存访问局部性是提高向量连接性能的关键影响因素。

总体而言,NUMA-conscious 向量连接方法的性能优于 NUMA-oblivious 向量连接方法,但不同 CPU 架构两条性能曲线的特征有所不同,如图 3 所示。Intel 平台五款 CPU 的 NUMA 优化加速比最大为 5.21,最小为 3.54,加速比曲线较为平缓且滞后,说明其较大容量的 cache 和较高互联性能的结构有效地缓解了跨 NUMA 访问延迟,连接算子性能对 NUMA 优化的敏感度较低,在 NUMA 优化算法设计方面较为放松。AMD 平台三款 CPU 的 NUMA 加速比最小为 13.95,最大为 21,表现为对 NUMA 优化技术的强依赖,虽然 L3 cache 的总容量较大,但对于缓解

NUMA 访问延迟贡献度较低，主要受 chiplet 架构下每 8 核心共享 32MB L3 cache 而导致 L3 cache 有效容量不足的影响。ARM CPU 的加速比曲线最大值为 38.11，远高于 AMD 和 Intel 平台，而且曲线的上升阶段出现较早，说明其有效 L3 cache 容量较 AMD 和 Intel 平台更低，对 NUMA 访问延迟的优化效率较低。

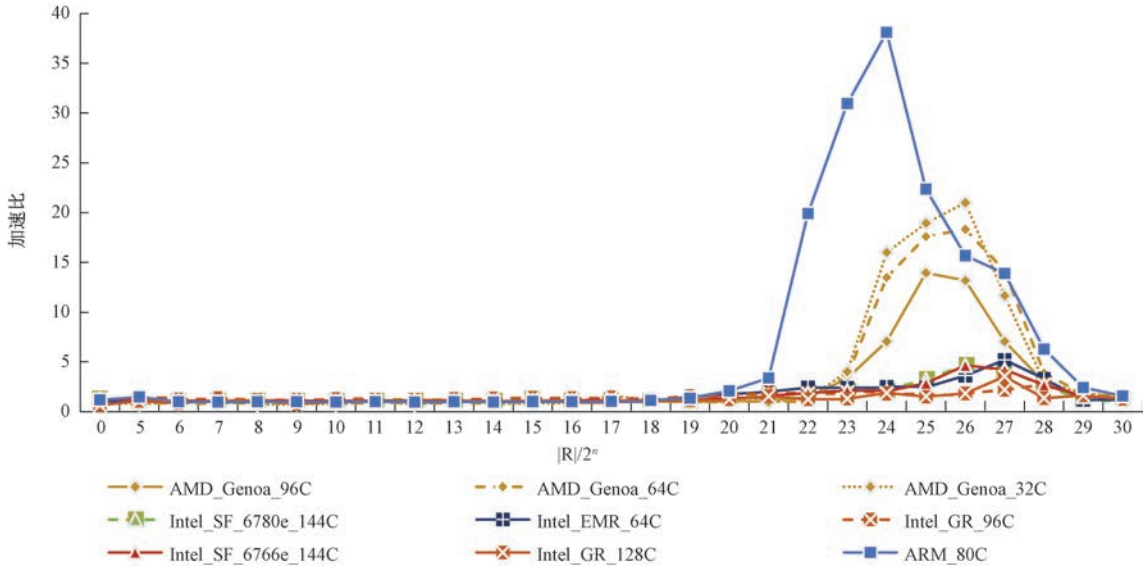

图 3 NUMA-conscious 向量连接算法加速比

小结 Chiplet 架构 CPU 可以集成更多的核心，但同时可能导致 chiplet 内物理 L3 cache 容量的减小、NUMA 节点数量的增加、跨 NUMA 访问延迟增加，从而导致有效 L3 cache 容量和访问效率受到影响，对连接算子性能有较大的影响。优化 CPU cache 微架构设计可以在相近核心数量和 cache 容量的前提下产生更高的 cache 访问效率，提升对 cache 局部性敏感的连接算子性能。

4.3 不同 CPU 平台连接算子性能

实验测试了 9 个不同架构 CPU 平台上的向量连接吞吐性能，如图 4 所示，|R|=0 表示连接向量长度为 0，对应 S 表全表扫描操作，从性能曲线来看，表扫描吞吐性能代表了向量连接性能的上限，内存带宽性能是决定性因素。

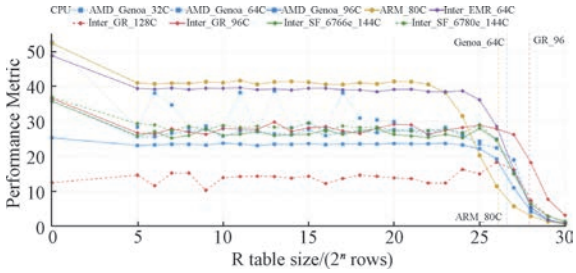

图 4 9 个 CPU 平台向量连接吞吐性能(GT/s)

总体来看，在小表连接负载中吞吐性能曲线前一阶段较为平稳，向量连接主要产生较低的 cache 访问延迟；当 R 表较大时，cache 访问逐渐从 L1 转向 L2、L3 及内存，连接延迟增加，曲线呈现下降趋势，不同曲线下降拐点显示了不同 CPU 平台有效 cache 容量的影响。值得注意的是，性能曲线在连接向量从 L1 向 L2、L3 溢出的过程中并未出现较大的波动，CPU 中不同的 L1 和 L2 容量对向量连接性能影响较小，当 R 表记录增长，向量索引长度超出 L3 cache 时，连接算子执行时间开始发生显著的变化，并且不同 CPU 在大表连接操作中性能差距较为显著，Intel 架构 CPU 的大表连接算子性能显著优于基于 AMD 和 ARM CPU。

4.3.1 Chiplet 架构 CPU 向量连接算子性能

基于 chiplet-based 架构的设计特点，图 5 显示 AMD CPU 向量连接性能在 2^{24} 之前(连接向量小于 32MB)保持平稳，之后开始小幅下降，曲线的拐点产生在 2^{26}(连接向量 128MB)，之后由于 L3 cache 失效(AMD CPU 每 NUMA L3 cache 容量为 128MB/192MB@96C)产生内存访问延迟而开始大幅下降。ARM CPU 在 2^{22} 之前(连接向量小于 8MB)保持平稳，曲线在 2^{24}(连接向量 32MB)时开始快速下降，L3 cache 有效容量相对较小。

4.3.2 Tile 架构 CPU 向量连接算子性能

基于 Tile-based 架构设计特点，分析图 6 显示的 Intel 5 款 CPU 的向量连接算子性能曲线和各 CPU 每 NUMA L3 cache 容量标记线。性能曲线在连接向量大小达到每 NUMA 节点 L3 cache 容量

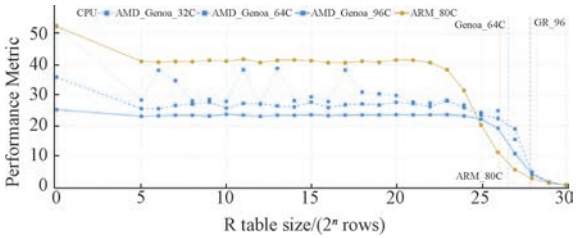

图 5　chiplet 架构 CPU 平台向量连接吞吐性能(GT/s)

之前出现拐点,超出每 NUMA 节点 L3 cache 容量之后大幅下降。在大表连接阶段,性能曲线有较大的差异,GR_96C＞SF_144C＞GR_128C＞EMR_64C,L3 cache 容量大的 CPU 大表连接性能较好,在 L3 cache 容量接近情况下,单 NUMA 结构 CPU(SF_144C)的大表连接性能优于多 NUMA 结构 CPU(GR_128C 和 EMR_64C),核心数量多连接性能更好。

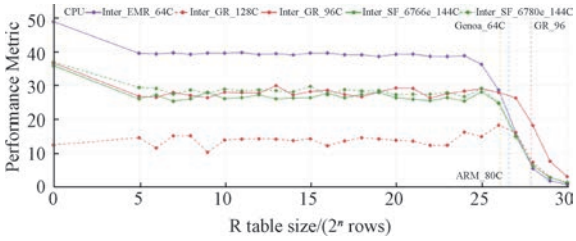

图 6　Tile 架构 CPU 平台向量连接吞吐性能(GT/s)

4.3.3　核心相同不同架构 CPU 向量连接算子性能

Genoa_96C、GR_96C 和 ARM_80C 的总核心数量相近但架构有较大差异,其性能曲线反映了不同 CPU 架构对向量连接算子性能的影响。如图 7 所示,Intel GR_96C 拐点出现最晚,AMD Genoa_96C 次之,ARM_80C 最早,反映了不同 CPU 架构在 L3 cache 有效容量和效率方面的差异。

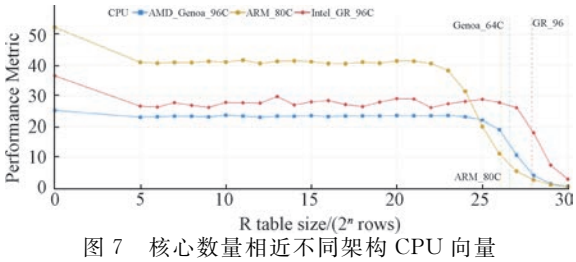

图 7　核心数量相近不同架构 CPU 向量连接吞吐性能(GT/s)

从连接算子执行时间曲线来看,在大表连接时 Genoa_96 与 ARM_80C 性能曲线较为相似,但显著低于 GR_96C 的连接算子性能。主要原因可能在于两个方面:NUMA 节点增多导致跨 NUMA 访问延迟增加;chiplet 架构 CPU 较小的有效 L3 cache 容量导致内存访问平均延迟较高。

小结　向量连接算子基准测试性能曲线分为两个阶段:(1)小表连接 in-cache 向量索引探测,此时性能曲线较为平缓,在 L3 cache 容量内的连接负载性能差异较小,L3 cache 的有效容量成为性能主要的决定因素,统一大容量 L3 cache 相对于 chiplet 架构分离式汇聚大容量 cache 结构具有显著的性能优势;(2)大表连接 in-memory 向量索引探测,主要代价是在 cache 支持下的内存访问延迟,也包含跨 NUMA 访问延迟的影响。整体而言,基于 chiplet 架构的 AMD 与 ARM 处理器在大表连接时的性能显著低于 Intel 架构,一方面是较大的 L3 cache 可以有效地降低内存访问延迟,另一方面是较少的 NUMA 节点降低跨 NUMA 访问延迟。Intel 架构 CPU 的 EMR_64C 和 GR_128C 处理器采用多 NUMA 架构,大表连接性能显著低于单 NUMA 架构的 GR_96C。

4.4　标准化连接算子性能分析

由于测试环境硬件配置无法定制,不同 CPU 平台内存带宽不同,导致连接算子性能曲线难以有效对比。向量连接分为 S 表的顺序扫描和 R 表连接向量上的向量探测两个阶段,前者由实测内存带宽决定,后者由 CPU 架构连接向量访问延迟决定(cache 内、内存、跨 NUMA 内存访问延迟),我们采用模拟方法,对内存带宽做标准化处理,以基础列扫描带宽为基准,将不同带宽标准化为统一带宽,然后比较不同 CPU 平台性能曲线,消除带宽维度的影响,聚焦于核心数量、L3 cache 大小、NUMA 节点数量等硬件参数对连接算子性能的影响。

标准化带宽性能后,如图 8 所示,连接算子性能曲线有较强的规律性:(1)大表连接时,chiplet 架构 CPU 连接算子性能普遍低于 Tile 架构 CPU(EMR_64C 除外),反映了 L3 cache 有效容量相对于核心数量对性能的影响更大;(2)chiplet CPU 核心数量越多,连接性能越好,性能与核心数量正相关,反映了在相同架构下,核心数量对性能的正相关性;(3)Intel 架构连接算子性能对 L3 cache 有效容量依赖度高于核心数量,GR_96C＞GR_128C＞SF_144C＞EMR_64C 的规律与每 NUMA 节点 L3 cache 容量大小的规律相近,EMR_64C 每 NUMA 节点 L3 cache 容量为 160MB,但内部分为两个 Tile,物理 L3 cache 容量为 80MB,符合性能曲线规律。

在大表连接中,Intel Tile-based 架构显著优于 AMD chiplet 架构,主要原因与两个硬件特征相关:(1) L3 cache 架构:Tile-based 相对于 chiplet 架构集成了更多的核心,L3 cache 有效容量更大,cache

局部性更强，支持更大连接表的 cache 优化连接；(2) Intel 架构 CPU 相对于 AMD 的 chiplet 架构 CPU 的跨 NUMA 访问延迟更低，本文采用的 R 表 NUMA-oblivious 存储访问方法受跨 NUMA 访问延迟影响较大，chiplet 架构在大表跨 NUMA 访问时较高的延迟是连接性能下降的主要影响因素。从 CPU 硬件架构来看，Tile-based 和 chiplet 架构 CPU 都在提升 L3 cache 总容量，但微架构的差异在 L3 cache 有效容量及访问效率和跨 NUMA 访问延迟上的差异会对连接性能产生不同的影响。

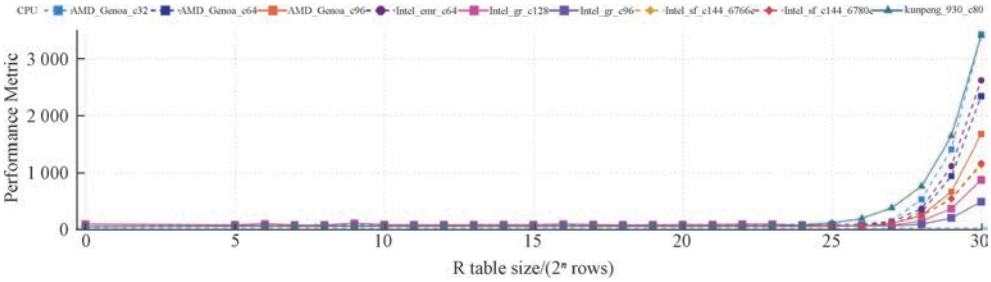

图 8 标准化向量连接执行时间(ms)

向量连接算子在执行时首先扫描 S 表记录，然后根据 S. key 在 R 表生成的向量索引中按地址探测，产生一个顺序扫描延迟和一个随机访问延迟。我们进一步计算了向量连接算子执行时间中平均每记录扫描的时间和探测时间占比，如图 9 所示。

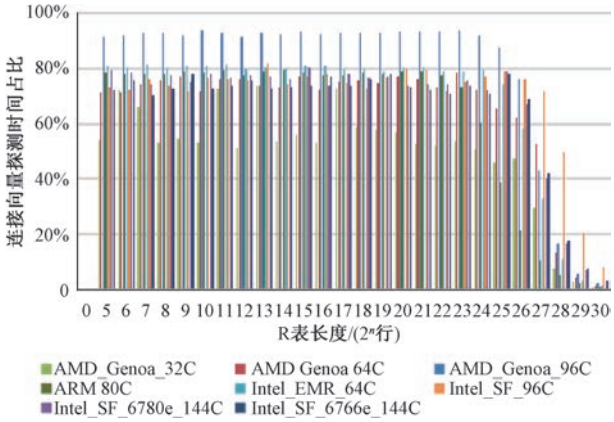

图 9 连接向量探测时间占比

当 cache 效率高时，扫描延迟占比下降较慢，连接算子性能对内存带宽依赖度更高，当 cache 效率低时，向量索引探测延迟占比快速提高，内存带宽性能对整体连接算子性能影响的权重下降。

小结 向量连接算子执行分为两个阶段，S 表记录扫描和 R 表向量索引探测。S 表记录扫描主要由内存带宽性能决定，NUMA-conscious 优化保证每个核心访问本地 NUMA 节点，NUMA-oblivious 会产生跨 NUMA 访问延迟。R 表连接向量探测主要取决于对连接向量的随机访问延迟，执行时间取决于向量索引在哪一级 cache 访问。当连接向量超出 L3 大小，进入内存访问阶段时，由基于 cache 部分缓存和跨 NUMA 访问综合延迟决定。当向量索引超出 L3 cache 大小时，核心数量通过分摊内存访问延迟而对连接算子性能产生正向影响。连接算子性能的 top 3 决定性因素排序为：内存带宽，L3 cache 有效容量，核心数量。

Chiplet 架构具有良好的可扩展性，但在微架构上将统一的 L3 cache 变成物理分离的局部 L3 cache，在总容量相当的情况下在 L3 cache 有效容量的访问效率方面较集中式 L3 cache 有所不足。数据库负载具有较强的局部性特征，其中查询代价最高的连接算子尤其依赖 L3 cache 的有效容量和访问延迟以获得良好的性能。因此，需要 CPU 硬件设计考虑以下问题：增加核心数量还是增加 L3 cache 容量，选择什么样的微架构以提升查询综合性能。本文研究揭示出，数据库负载对有效 L3 cache 容量和访问性能的依赖度高于核心数量，低延迟 L3 cache 整体访问性能至关重要，优化跨 NUMA 访问延迟可以有效降低算法复杂度。

5 结论和下一步工作

本文聚焦于内存 OLAP 数据库性能关键 CPU 硬件影响因素研究：首先，通过代表性内存数据库 Hyrise 的宏基准测试分析数据库负载中性能决定性的算子及其硬件访问特性，确定数据库对内存带宽、核心并行度、cache 效率的关键依赖，然后结合数据库负载数据访问综合特性设计了向量连接微基准，覆盖数据库算子的内存带宽性能、核心并行计算效率和 cache 缓存效率三个关键因素，并进一步结合当前 chiplet-based 和 Tile-based CPU 架构和 NUMA 配置特征设计了 NUMA-conscious 和 NU-

MA-oblivious 向量连接算法用于评估不同 CPU 架构对向量连接算子性能的影响关键因素,并通过 AMD、Intel、ARM 三个代表性平台,9 个最新 CPU 上的综合测试深入分析了向量连接算子在不同 CPU 上的性能特征及其硬件依赖因素。实验结果表明,内存带宽决定了数据库性能的上限,提高 CPU 带宽性能是内存数据库性能提升的基础性影响因素;cache 效率受 CPU 微架构、cache 容量和 NUMA 配置的影响,提高 L3 cache 的有效容量对数据库性能有较大的提升效果,高效的 L3 cache 对数据库性能的优化超过核心数量的影响;核心数量与 L3 cache 容量有一定的正相关性,但受微架构的影响产生一些差异,CPU 微架构对于共享 L3 cache 访问效率有较大的影响,核心数量对于提高内存访问吞吐性能有正向影响,在超过 cache 容量的大表连接应用场景下有较大的性能收益,但总体性能影响权重低于 L3 cache 效率。

本文结合宏基准与微基准提出了一种轻量的 CPU 性能评测方法,该方法可以简化 CPU 硬件设计阶段对数据库性能的评估方法并降低模拟测试代价。与已有研究不同,我们更关注 CPU 硬件发展趋势带来的性能红利,如核心数量和 L3 cache 容量增长,通过高存储效率和计算效率的向量连接算子构建数据库查询性能微基准,可以更好地平衡查询性能与内存存储效率之间的矛盾,自适应地利用 CPU 硬件技术提升带来的性能收益。向量连接算子覆盖了数据库宏基准测试的不同数据访问和局部性特征,具有较好的代表性。未来将进一步扩充微基准算子集,以覆盖数据库更多的综合负载特征,如扫描、投影、聚合融合算子微基准,以提供更加定制化、全面的评测工具。

致谢　感谢 Intel 公司系统与软件优化工程师刘专对本文研究的技术支持。

参　考　文　献

[1] Intel Unveils Future-Generation Xeon with robust performance and efficiency architectures. https://newsroom.intel.com/artificial-intelligence/intel-unveils-future-generation-xeon 2023,8,28

[2] 5th gen AMD EPYC processor architecture. https://www.amd.com/content/dam/amd/en/documents/epyc-business-docs/white-papers/5th-gen-amd-epyc-processor-architecture-white-paper.pdf 2025,3

[3] Alessandro Fogli, Bo Zhao, Peter R Pietzuch, Maximilian Bandle, Jana Giceva. OLAP on modern chiplet-based processors//Proceedings of the VLDB Endowment. 2024,17(11): 3428-3441

[4] Marcin Zukowski, Peter A Boncz, Niels Nes, et, al. MonetDB/X100-A DBMS in the CPU cache. IEEE Data(base) Engineering Bulletin. 2025,28(2): 17-22

[5] Cagri Balkesen, Jens Teubner, Gustavo Alonso, M Tamer Özsu. Main-memory hash joins on multi-core CPUs: Tuning to the underlying hardware//Proceedings of the ICDE. Brisbane, Australia, 2013: 362-373

[6] Spyros Blanas, Yinan Li, Jignesh M Patel. Design and evaluation of main memory hash join algorithms for multi-core CPUs//Proceedings of the SIGMOD Conference. Athens, Greece, 2011: 37-48

[7] Harald Lang, Viktor Leis, Martina-Cezara Albutiu, Thomas Neumann, Alfons Kemper. Massively parallel NUMA-ware hash joins//Proceedings of the IMDM@VLDB. Riva del Garda, Italy, 2013: 1-12

[8] Stefan Schuh, Xiao Chen, Jens Dittrich. An experimental comparison of thirteen relational equi-joins in main memory//Proceedings of th SIGMOD Conference. San Francisco, USA, 2016: 1961-1976

[9] HAN Rui-Chen, ZHANG Yan-Song, LIU Zhuan, et, al. NUMA-conscious foreign key join optimization technique. Journal of Software, 2025, 36(12):5821-5850(in Chinese) (韩瑞琛,张延松,刘专等. NUMA-conscious 外键连接优化技术.软件学报, 2025, 36(12):5821-5850)

[10] Zhuan Liu, Ruichen Han, Yansong Zhang, et, al. Exploring fine-grained in-memory database performance for modern CPUs. IEEE Transactions on Parallel and Distributed Systems, 2023, 34(6): 1757-1772

[11] Anil Shanbhag, Samuel Madden, Xiangyao Yu. A study of the fundamental performance characteristics of GPUs and CPUs for database analytics//Proceedings of the SIGMOD Conference. Online, 2020: 1617-1632

[12] Yansong Zhang, Xuan Zhou, Ying Zhang, et, al. Virtual denormalization via array index reference for main memory OLAP. IEEE Transactions on Knowledge and Data Engineering, 2016, 28(4): 1061-1074

[13] Ravi Bhargava, Kai Troester. AMD next-generation "Zen 4" core and 4th gen AMD EPYC server CPUs. IEEE Micro, 2024, 44(3): 8-17

[14] Ashley O. Munch, Nevine Nassif, Carleton L. Molnar, Jason Crop, et, al. 2.3 emerald rapids: 5th-Generation Intel ® Xeon ® scalable processors//Proceedings of the ISSCC. San Francisco, USA, 2024: 40-42

[15] Danica Porobic, Ippokratis Pandis, Miguel Branco, Pinar Tözün, Anastasia Ailamaki. Characterization of the impact of hardware islands on OLTP. The VLDB Journal, 2016, 25(5): 625-650

[16] Tudor-Ioan Salomie, Ionut Emanuel Subasu, Jana Giceva,

Gustavo Alonso. Database engines on multicores, why parallelize when you can distribute? //Proceedings of the EuroSys. Salzburg, Austria, 2011: 17-30

[17] Martin Grund, Jens Krüger, Hasso Plattner, Alexander Zeier, Philippe Cudré-Mauroux, Samuel Madden. Hyrise-a main memory hybrid storage engine//Proceedings of the VLDB Endowment, 2010, 4(2): 105-116

[18] Cagri Balkesen, Gustavo Alonso, Jens Teubner, M Tamer Özsu. Multi-core, main-memory joins: sort vs. hash revisited// Proceedings of the VLDB Endowment, 2013, 7(1): 85-96

[19] Zhang Yansong, Zhang Yu, Wang Shan. A novel in-memory OLAP star join acceleration technique with vector index. Chinese Journal of Computers, 2019, 42(8):1686-1703(in Chinese)
(张延松, 张宇, 王珊. 一种基于向量索引的内存 OLAP 星型连接加速新技术. 计算机学报,2019, 42(8): 1686-1703)

[20] Maximilian Bandle, Jana Giceva, Thomas Neumann. To partition, or not to partition, that is the join question in a real system//Proceedings of the SIGMOD Conference. Virtual, China, 2021: 168-180

WANG Qin-Yao, master. Her research interests focus on new hardware in-memory database technology.

HAN Rui-Chen, Ph. D.. His research interests include in-memory databases and new hardware database technology.

LIU Jia-Ru, master. Her research interests include HTAP databases and in-memory database storage.

ZHANG Yan-Song, Ph. D., associated processor. His current research interests include data warehouses, in-memory databases, GPU databases, and new hardware database technology.

Background

The underlying hardware-particularly the architectural design of CPUs-has a profound impact on the performance of in-memory databases. Modern systems rely heavily on parallel processing capabilities, efficient utilization of memory hierarchies, and low-latency data access, all of which are shaped by CPU characteristics. Evaluating how different CPU architectures affect database performance is essential for system design, performance tuning, and hardware-software co-optimization.

This study proposes a benchmarking framework that combines macro- and micro-level approaches to evaluate the impact of hardware on database performance. Through controlled experiments, it analyzes how architectural features such as memory bandwidth, core count, and last-level (L3) cache influence the behavior of centralized databases. Macro-benchmarking is employed to capture the performance of representative database workloads on the server side, identifying the key hardware factors and critical operators that affect database performance, while micro-benchmarking isolates individual operators to study their sensitivity to specific CPU characteristics. To enhance the precision of micro-benchmarking, we introduce a NUMA-conscious algorithm that explicitly manages memory placement and thread affinity, and we demonstrate its effectiveness across multiple CPU architectures.

As core counts increase, CPU design has become significantly more complex. Variations in chiplet-based microarchitectures, cache hierarchies, and NUMA configurations result in noticeable differences in data access latency and throughput. These architectural nuances necessitate targeted optimizations of database operators. Meanwhile, database workloads exhibit strong data locality, and their performance is closely tied to multiple hardware parameters. Identifying which microarchitectural features have the greatest performance impact can provide valuable insights for both database engineers and hardware designers.

The rise of NUMA-based servers has shifted the focus of optimization from cache tuning to memory locality management. Hardware-aware scheduling and operator placement strategies can significantly reduce inter-socket communication and improve throughput. Moreover, heterogeneous chiplet-based architectures introduce locality at varying levels, presenting new opportunities for optimization. While distributed systems have long adopted locality-aware designs, this paper specifically focuses on centralized databases, showing that similar performance gains can be achieved through carefully designed execution strategies.

By analyzing operator performance across different CPU platforms and workloads, this study reveals critical performance bottlenecks and architectural dependencies. The findings help explain how CPU metrics-such as memory latency, inter-core communication cost, and cache sharing-impact database behavior. This provides a stronger foundation for architectural design and system-level optimization decisions, ultimately bridging the gap between processor architecture and real-world database performance.