软件学报 ISSN 1000-9825, CODEN RUXUEW
2025,36(1):424−445 [doi: 10.13328/j.cnki.jos.007225] [CSTR: 32375.14.jos.007225]

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

HTAP 评测基准的评测能力综述*

翁思扬[1], 俞 融[1], 王清帅[1], 胡梓锐[1], 倪 葳[1], 张 蓉[1], 周 烜[1], 周傲英[1], 徐泉清[2], 杨传辉[2], 刘 维[3], 杨攀飞[3]

[1](华东师范大学 数据科学与工程学院, 上海 200062)

[2](蚂蚁集团 OceanBase, 北京 100015)

[3](工业和信息化部电子第五研究所, 广东 广州 511300)

通信作者: 倪葳, E-mail: lni@dase.ecnu.edu.cn

摘 要: 对数据库系统即时修改数据的高效实时分析需求推动了数据库系统向同时支持 OLTP 业务和 OLAP 业务两种场景的 HTAP 数据库系统的快速发展. 面对众多的 HTAP 数据库系统, 为了推动 HTAP 数据库系统的公平比较和健康发展, 定义和实现相应的评测基准来评估 HTAP 数据库系统的新特性至关重要. 首先, 分析 HTAP 数据库系统的关键特征并抽象总结 HTAP 数据库系统实现的关键技术. 然后, 提炼出 HTAP 数据库系统的设计难点和构建 HTAP 评测基准的挑战, 并基于此提出 HTAP 评测基准应考虑的设计维度, 包括数据生成、负载生成、评价指标和一致性模型支持性. 对比现有 HTAP 评测基准在设计维度和实现技术上的差异, 总结评测基准在不同设计维度上的优劣. 此外, 运行已公开的典型评测基准, 展示并分析它们对 HTAP 数据库系统关键特征的评测能力以及对不同 HTAP 数据库系统的横向对比的支持能力. 最后, 总结对 HTAP 评测基准的能力需求和未来的一些研究方向, 指出语义一致的负载控制和新鲜数据访问度量是 HTAP 数据库系统评测基准定义的关键问题.

关键词: HTAP 评测基准; HTAP 数据库系统; 性能分析; 新鲜度

中图法分类号: TP311

中文引用格式: 翁思扬, 俞融, 王清帅, 胡梓锐, 倪葳, 张蓉, 周烜, 周傲英, 徐泉清, 杨传辉, 刘维, 杨攀飞. HTAP 评测基准的评测能力综述. 软件学报, 2025, 36(1): 424–445. http://www.jos.org.cn/1000-9825/7225.htm

英文引用格式: Weng SY, Yu R, Wang QS, Hu ZR, Ni L, Zhang R, Zhou X, Zhou AY, Xu QQ, Yang CH, Liu W, Yang PF. Survey on Benchmarking Ability of HTAP Benchmarks. Ruan Jian Xue Bao/Journal of Software, 2025, 36(1): 424–445 (in Chinese). http://www.jos.org.cn/1000-9825/7225.htm

Survey on Benchmarking Ability of HTAP Benchmarks

WENG Si-Yang[1], YU Rong[1], WANG Qing-Shuai[1], HU Zi-Rui[1], NI Lü[1], ZHANG Rong[1], ZHOU Xuan[1], ZHOU Ao-Ying[1], XU Quan-Qing[2], YANG Chuan-Hui[2], LIU Wei[3], YANG Pan-Fei[3]

[1](School of Data Science and Engineering, East China Normal University, Shanghai 200062, China)

[2](OceanBase, Ant Group, Beijing 100015, China)

[3](The 5th Electronics Research Institute of MIIT, Guangzhou 511300, China)

Abstract: Requirements for the effective real-time analysis of instant data modification of database systems have driven the rapid development of Hybrid Transactional/Analytical Processing (HTAP) database systems, which support to process both OLTP and OLAP workloads. To realize fair comparisons and healthy development, it is crucial to define and implement new benchmarks to evaluate new features of HTAP database systems. Firstly, this study analyzes the key characteristics of HTAP database systems and summarizes the

* 基金项目: 国家自然科学基金 (62072179, 62307014, 92270202); 基础软硬件性能与可靠性测评工业与信息化部重点实验室开放课题; 上海市青年科技英才扬帆计划 (22YF1411300); OceanBase 联合实验室项目
收稿时间: 2023-09-13; 修改时间: 2023-11-20; 采用时间: 2024-05-13; jos 在线出版时间: 2024-07-03
CNKI 网络首发时间: 2024-07-05

distinct technologies in their implementations. Secondly, the difficulties of designing HTAP database systems and the challenges of constructing HTAP benchmarks are extracted. Based on these, the design dimensions of HTAP benchmarks are proposed, including data generation, workload generation, evaluation metrics, and consistency model supportability. This study compares differences between existing HTAP benchmarks in terms of design dimensions and implementation technologies and sums up their merits and defects in different dimensions. Additionally, the published benchmarks are demonstrated and their abilities of evaluating key features and supporting horizontal comparisons among HTAP database systems are analyzed. Finally, this study concludes the requirements for HTAP benchmarks and some future research directions, pointing out that semantically consistent workload control and fresh data access metrics are the key issue in defining benchmarks for HTAP database systems.

Key words: HTAP benchmark; HTAP database system; performance analysis; freshness

1 引 言

1.1 HTAP 数据库系统的背景

HTAP (hybrid transaction and analytical process) 数据库系统一般指支持同时进行实时事务处理 OLTP (online transaction processing) 和分析查询处理 OLAP (online analysis processing) 的数据库系统, 旨在实时分析事务更新的数据, 也即支持处理 TP/AP 混合负载. 这一概念由 Gartner 于 2014 年提出[1,2], 并迅速得到 Oracle[3] 和 SQL Server[4] 等传统数据库厂商的响应. 他们随即扩展了原有的架构设计来支持 HTAP 业务. 除了传统的单机数据库, 具有大规模数据、业务处理能力的分布式系统的发展也催生了一批具有混合负载处理能力的分布式数据库系统, 越来越多的数据库厂商开始重视支持 HTAP 业务这一特性, 如 Greenplum[5]、SingleStore[6]、Apple[7], MySQL [8], PingCAP [9]、Google[10]、阿里巴巴[11] 和 Amazon [12,13] 等. 进而涌现出众多针对 HTAP 数据库系统架构和实现技术的研究. 研究工作大多聚焦于 HTAP 数据库系统实现与性能优化关键技术. 例如, 在索引更新、版本链追踪等方面, Abebe 等人[14] 提出根据混合负载的变化自适应地选择合适的副本数量、混合行列组织的存储格式; 沈斯杰等人[15] 提出利用高可用副本降低 TP 和 AP 之间的资源竞争, 并且通过修改数据同步的粒度和索引更新算法提升 HTAP 性能; Sirin 等人[16] 提出通过对 last-level cache (LLC) 进行分区, 将不同分区服务于不同工作负载, 实现 TP 和 AP 共享数据的访问隔离, 提高负载的缓存命中率的同时减少缓存争用. 此外, 也有大量工作对现有 HTAP 数据库系统架构进行分类和综述. Özcan 等人[17] 从负载的处理引擎以及数据的组织方式分类并分析了现有的 HTAP 数据库系统; 张超等人[18] 从架构组成维度思考并提出 HTAP 数据库系统 4 个关键技术挑战; 胡梓锐等人[19] 总结了 HTAP 数据库系统中数据的隔离与共享技术, 结合现有 HTAP 数据库系统的实现方式提炼出 HTAP 数据库系统数据共享的一致性模型分类及优化策略.

HTAP 数据库系统多样化的系统架构、实现技术以及数据共享模型服务于不同的应用需求, TP 和 AP 负载之间的数据共享压力的变化会引起系统内部资源竞争的变化, 进而产生不同的性能表现. 因此, 与传统的处理单一负载类型的数据库系统相比, 除了要抽象或定义一个复杂的语义一致性 HTAP 业务场景之外 (即 TP 和 AP 负载在相同的表模式下访问相同数据且负载的语义逻辑正确), 混合负载交互的不确定性给评测此类数据库系统造成了较大挑战.

1.2 评测基准基本要素

数据库管理系统的不断发展, 推动数据库系统评测工具的革新和进步; 数据库系统评测基准面向用户需求, 不断推陈出新, 对数据库系统本身产生引领性的影响, 进而推动数据库技术的不断进步, 两者相辅相成. 一款优秀的评测基准往往需要满足代表性、关联性、透明度、公平性、可重复性、可伸缩性等特点[20,21]. 此外, 对于新型数据库系统来说, 评测基准需要能评估用户关注的维度, 支持公平、公正的横向数据库系统比较, 从而引导数据库系统健康、良性的成长.

数据库系统评测基准, 如 TPC-C[22], TPC-H[23], HATtrick[24] 等, 都是从数据、负载、度量体系[25] 这 3 个基本维度进行设计. 评估 HTAP 数据库系统性能时, 在 3 个维度上的设计与面向单类业务场景的数据库系统评测会具有不同的要求[26], 具体表现为:

1) 数据方面: 不同数据类型、数据结构和数据行列组织方式[17]对查询和事务处理的影响不容小觑. 例如, 一个大型电子商务网站在处理短时间内产生的数百万订单、客户信息和产品数据的实时分析[17]任务时, 数据类型的丰富性、数据关系的复杂性和数据分布特殊性[27]等都将对 TP 事务处理和 AP 查询处理产生影响.

2) 负载方面: 不同类型的负载会对数据库系统性能产生不同的影响. 例如, 社交媒体网站支持数百万用户同时进行浏览、搜索、评论和点赞操作时, 这些操作可能会在不同时间发生, 也可能涉及不同类型的数据查询和修改. 负载操作的多样性、随时间的动态变化情况[28]、数据访问模式与访问分布、关联关系[29]等都是影响数据库系统性能的关键负载特征. HTAP 评测基准负载需要展现 HTAP 业务的混合负载交互的特征, 且负载性能可复现.

3) 性能指标方面: 数据库性能通过性能指标反应, 如事务吞吐量 (throughput, TPS)、查询时延 (latency)、新鲜度 (freshness)[9]等. 统一、公平、有效的性能指标体系对于数据库系统自身问题的发现和不同数据库之间的横向比较来说有着重要作用. 但是 HTAP 数据库系统独特的对混合负载的处理能力以及负载之间基于数据的复杂关联性[19], 使得传统的针对单一任务 (AP 或 TP) 的评测指标[22,23]以及简单混合这些指标无法公平地度量、比较 HTAP 数据库系统的能力, 亟待新评测指标的出现.

本文旨在探究现有 HTAP 评测基准对 HTAP 数据库系统关键技术的评测能力, 进而回答如下 4 个核心问题.

Q1: 现有评测基准是否具有评测基于不同一致性模型的 HTAP 数据库系统的能力?

现代 HTAP 数据库系统基于多种一致性模型设计与实现, 包括线性/顺序/会话一致性[19]. 其中, 线性一致性 HTAP 数据库要求 AP 能实时访问 TP 最新数据修改, 而后两类则允许 AP 访问一定时间范围内的旧数据而非与 TP 强一致的实时数据, 以降低对 TP 性能的影响. 架构上的差异使得基于传统指标的评价方式已经无法满足 HTAP 数据库系统横向比较的目的. 例如, 部分数据库快速响应的代价是返回低新鲜度数据, 即“老”数据, 那么单一的吞吐 (TPS) 或者延时 (Latency) 指标的可用性降低. 因此, 新基准定义需要考虑不同架构设计带来的评测结果可参考性与可用性问题.

Q2: 现有评测基准面向 HTAP 场景的设计是否具有代表性?

HTAP 业务同时具有 TP 业务的高并发和 AP 任务的大批查询特征. 如在金融场景中, 一方面数据库需要高效支持实时的增、删、改、查等交易操作, 另一方面也需要及时应对大批量的历史报表指标分析或针对用户个人收支交易情况的实时决策分析. 而评测基准往往需要从实际应用中抽象出一套符合 HTAP 业务特征、语义一致的评测场景, 即针对统一应用场景定义 TP 负载与 AP 负载, 而不是简单地对已有 TP/AP 基准进行场景缝合. 具体而言, 语义一致指评测基准应当在同一套表模式下运行事务负载和查询负载, 且保证混合负载能基于一致且正确的语义访问相同的数据. 所以基准定义代表性需要重点关注于语义一致性、TP/AP 负载的代表性以及不同负载的交互 3 个维度.

Q3: 现有评测基准面向 HTAP 数据库系统的评测是否具有可重复性?

评测基准要求评测结果可复现. 在 HTAP 场景下, TP 负载频繁的修改操作, 使得数据库规模或数据分布都会产生变化, 进而产生 AP 端数据的变化, 比如不同的数据库系统由于 TP 能力的差异很可能造成 AP 访问数据量的不同, 即计算复杂度不同, 影响执行性能, 造成数据库系统间的横向比较的不公平性. 因此需要新评测基准具有在动态 TP 负载下, 控制 AP 查询复杂度, 达到性能稳定性的能力.

Q4: 现有评测基准在评测 HTAP 数据库系统核心能力, 如负载隔离能力或实时分析能力上是否充分?

由于 HTAP 数据库系统的核心目标是兼备 TP 处理高吞吐及 AP 分析实时性[30], 所以 HTAP 数据库系统往往会着力解决混合负载同时运行时的性能降级以及为获取实时数据造成同步压力过大的问题[15]. HTAP 数据库系统需要具备负载隔离能力, 以保证混合负载执行时性能的稳定性; 具有实时分析能力以提供用户新鲜的分析结果. 此类问题的解决方案也是 HTAP 数据库系统的核心竞争力所在. 因此, 评测基准在数据、负载及指标设计上应能直观地揭示或者呈现该类数据库在核心能力上的优劣, 例如新鲜度[9,15,31]、负载隔离能力[32]等.

1.3 本文结构

本文第 2 节从 HTAP 数据库系统区别于其他数据库系统的关键特征出发, 陈述 HTAP 数据库系统中的关键

技术, 从中提炼出 HTAP 数据库系统实现的难点和构建 HTAP 评测基准的挑战. 第 3 节总结 HTAP 评测基准的设计维度, 并分析了各维度的关键技术点. 第 4 节概述了现有 HTAP 评测基准的实现技术, 并做详细对比. 第 5 节通过运行已有评测基准, 展示它们对 HTAP 数据库系统关键特征的测试覆盖能力以及不同数据库横向对比的支持能力. 第 6 节总结 HTAP 评测基准, 并展望未来的研究方向.

2 HTAP 数据库系统及评测基准研究进展

HTAP 数据库系统及其评测基准的发展时间线如图 1 所示. 近 10 年来 HTAP 系统迅速发展壮大. 最先实现 HTAP 功能的是 SAP HANA 内存数据库. 在 2014 年, Gartner 首次定义了 HTAP [1] 概念. 同年, 出现了支持 HTAP 功能的内存数据库 MemSQL. 此后, 也有传统数据库, 如 Oracle、SQL Server, 通过优化自身的架构或者实现技术, 支持 HTAP 业务. 在 2020 年, TiDB、F1 Lighting 等 HTAP 数据库系统的相继出现又掀起了 HTAP 数据库系统研发热潮. 随后, OceanBase[33,34]、PolarDB、Greenplum 等一系列数据库系统也崭露头角.

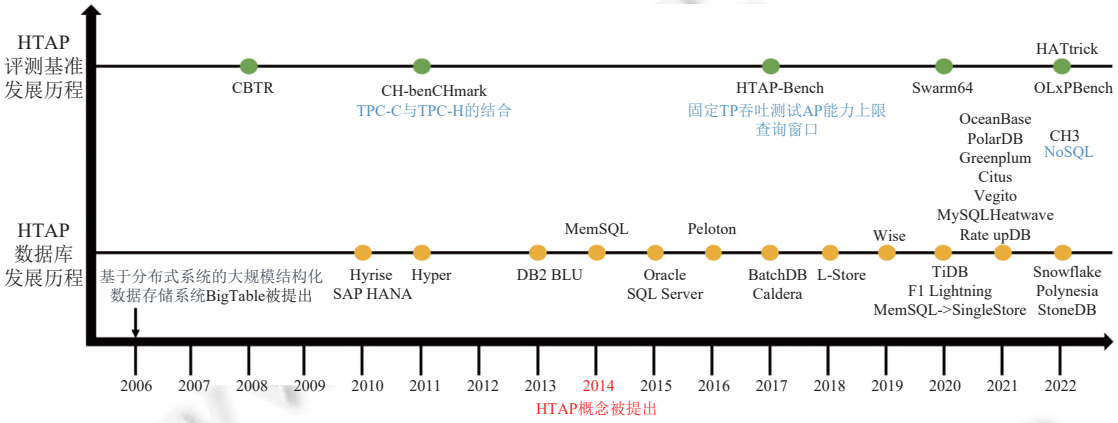

图 1 HTAP 数据库系统和 HTAP benchmark 发展历程

相应地, 评测基准的研究工作也不断推陈出新. 在 Gartner 定义 HTAP 之前, 业界已经开始期待具有混合负载处理能力的数据库系统的出现, 相应地开始定义评测混合负载处理能力的基准. 第一款面向混合负载处理能力的评测基准是 CBTR[35], CH-benCHmark[36], 而在这两款基准也并非针对后续的 HTAP 定义设计, 使用“HTAP 关键字”描述其评测对象及目标. 从 2016 年开始, HTAP 数据库系统在架构和实现技术上发展的越来越快, 相应地刺激了评测基准的发展, 可以看到 HTAPBench[37]、Swarm64[38]、OLxPBench[31]、HATtrick[23] 评测工具的推出.

2.1 HTAP 数据库系统关键技术

HTAP 数据库系统需要同时处理 TP 和 AP 负载, 使得 AP 查询能在一定的延迟容忍度下访问 TP 事务产生的新数据, 且保持混合负载相互之间影响可控[23]. 为满足上述要求, HTAP 数据库系统需要提供 3 种关键技术, 具体介绍如下.

2.1.1 处理 TP 和 AP 负载的混合架构

HTAP 数据库系统需要在一个系统中, 同时支持两种类型负载的运行, 其中 TP 负载是短时延、高并发、小查询为主的在线事务处理业务, 一般倾向于行式存储; AP 负载是高延迟、带宽密集、复杂查询的在线分析处理业务, 一般使用列式存储和向量化引擎. HTAP 数据库系统用于处理两种负载的存储格式实现可分为两类, 如图 2 所示. 一种是基于原有存储格式扩充实现另一类存储格式的原生 HTAP 数据库系统, 如基于行式存储, 在内存中扩展列式存储的数据库 DB2[39] 和 Oracle[3] (图 2(a)); 通过 Lightning 部件实现列式存储的 F1 Lightning[10], 基于列式存储, 扩展到行式存储的 Greenplum[5] (图 2(b)); 另外一种是集成 Spark、Flink 等第三方计算引擎[40] 以用于处理 AP 负载的集成 HTAP 数据库, 如 TiDB 通过 TiSpark[41] 将 TP 和 AP 功能结合, Wildfire[42] 通过可扩展 Spark API 快速

处理 AP 任务, CBase[43] 通过 JDBC 连接 Spark 等 (图 2(c)). 针对这两类不同的负载, HTAP 数据库系统架构期待能够提供自适应多样化的访问模式, 如读写平衡场景、读密集场景或大批量读取场景等.

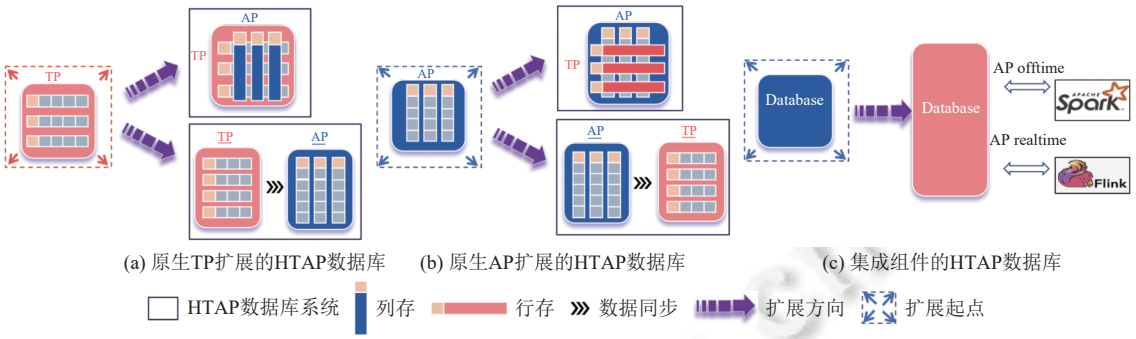

图 2 基于不同扩展方式的 HTAP 数据库系统分类

2.1.2 保持数据版本一致的同步技术

在 HTAP 数据库系统中, TP 和 AP 负载会访问相同的数据, 因此 HTAP 数据库需要实现能容忍一定延迟的数据共享, 即 TP 负载修改的数据在一定的时间范围内能对 AP 负载可见. 其延迟一方面取决于 HTAP 数据库系统所基于的一致性模型, 即容忍 AP 端访问历史数据的落后时间范围; 另一方面取决于数据库对同步流程的优化. 数据同步的方式依据数据库系统隔离方式进行设计, 如图 3 所示. 若数据库采用一体式架构, 在原有数据版本的存储格式上直接扩展另一类存储格式 (图 3(a)), 使两种负载直接访问相同数据版本, 则两种负载之间存在逻辑的同步, 无同步带来的版本差, 无新鲜度上的差异, 有较好的实时分析能力, 如 OceanBase; 若数据库采用缓存隔离的分离式架构 (图 3(b)), 在处理 TP 负载和 AP 负载时分别使用独立的缓存, 仅共享最终存储的数据版本, 则 TP 负载更新的数据会同步到 AP 端[44], 从而保证操作读取一致的数据版本; 若数据库采用存储隔离的分离式架构 (图 3(c)), 提供不同的存储副本以处理两种负载, 存储副本间的物理同步往往导致两种负载访问同一版本时存在时间差, 也就是所谓的新鲜度差异. 无论采用内存拷贝数据[27] 还是传输日志[10] 的方式同步数据版本, 都需要在设计相关存储结构时考虑如何兼备行存与列存数据存储的优势以及如何应对数据格式转换等问题.

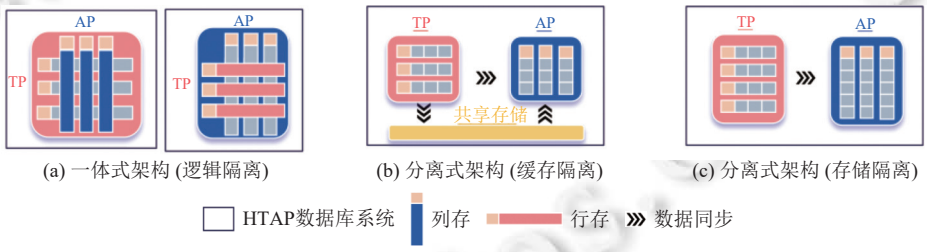

图 3 基于隔离方式的 HTAP 数据库系统分类[18]

2.1.3 降低访问冲突的负载隔离方式

混合执行 TP 与 AP 负载可能造成资源竞争压力. 这会同时影响二者的性能[36]. 不同负载使用资源的规模和方式存在差异. 短时高并发的 TP 负载往往需要 Cache 和 Memory 资源[45], 数据库通过缓存被频繁访问的热点数据加速访问效率. 而 AP 负载可能会访问大量历史数据, 若缓存大批历史数据则可能导致后续执行 TP 负载时缓存频繁与磁盘的交互, 使系统面临性能降级风险. 数据库可以采取不同的负载隔离方案, 以牺牲新鲜度、实时性为代价来弱化资源冲突, 保障性能. 例如, 在共享存储的数据库架构中, 通过创建独立的快照[46] 或以租户为单位细粒度划分资源单元[27] 等方式隔离不同负载所使用的资源; 也可以不共享底层存储, 在存储层采用物理隔离[9], 从而避免负载之间数据访问带来的干扰. 同时, HTAP 数据库系统也需要根据负载实际运行情况[16] 或是性能指标的实时

反馈 (如新鲜度阈值[44]) 动态调度 CPU/内存等系统资源, 以更好地满足运行混合负载时的性能要求.

根据以上分析, HTAP 数据库主要包括 TP 端、AP 端和 TP 到 AP 的数据同步模块. 根据这 3 个模块成为性能瓶颈的场景, 我们划分出 3 种负载访问模式. 针对不同的访问模式, 总结了负载压力和相应的数据库一致性模型以及隔离程度和使用的同步技术 (见表 1).

表 1　面向不同负载访问模式的 HTAP 数据库系统关键技术

访问模式	性能瓶颈	TP压力	AP压力	一致性模型	隔离情况	同步技术
写密集	TP端	高	中	顺序一致性	分离式架构存储隔离	批同步写日志
读密集	数据同步机制	中	高	线性一致性	分离式架构存储隔离 一体式架构逻辑隔离	以最小粒度同步写操作 内存拷贝数据
大批量读	AP端	低	高	会话一致性	分离式架构缓存隔离	根据请求同步数据

2.2　HTAP 数据库系统评测基准设计的挑战

HTAP 数据库系统实现的关键技术与 HTAP 数据库系统所包含的混合架构、数据同步任务和负载隔离要求相对应, 在实现的过程中主要解决 3 个难点问题 (如图 4 所示).

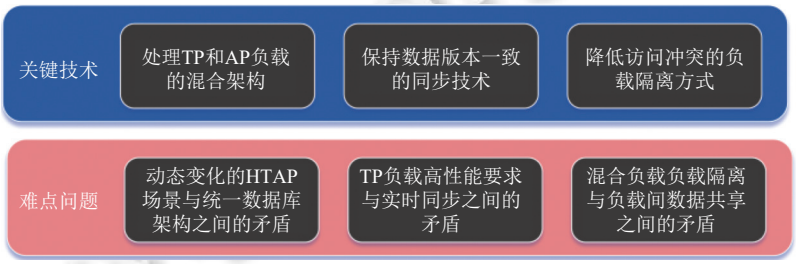

图 4　HTAP 数据库系统关键技术与难点问题的对应关系

1) 动态变化的 HTAP 场景与统一数据库架构之间的矛盾: TP 和 AP 负载具有不同特征, 优化负载执行的架构存在差异, HTAP 数据库系统在统一结构下兼容这些差异往往会造成性能损失.

2) TP 负载高性能要求与实时同步之间的矛盾: TP 负载频繁修改数据, 产生大量数据版本, 同步的时间开销高, 造成资源的恶性竞争, 高性能 TP 处理下满足 HTAP 数据库系统实时同步的要求是一个难点问题.

3) 混合负载负载隔离与负载间数据共享之间的矛盾: TP 和 AP 负载执行过程中由于数据共享要求可能造成资源抢占, 隔离负载会保证稳定吞吐但是也会阻碍数据同步, 降低数据共享能力.

为了解决这些难点问题, 相关技术不断更迭. 为了判断对应技术的优劣、区分不同系统性能高低, 面向 HTAP 数据库系统评测的 HTAP 评测基准的定义需要考虑以下几个方面.

1) 混合负载交互模式的代表性: 混合负载体现 HTAP 负载的代表性场景, 反映 TP 和 AP 负载的互相影响程度.

2) 评测工具测试指标的合理性: 评测工具能够准确公平测量 OLTP 和 OLAP 负载下的数据库性能.

3) 评测维度的支持性和丰富性: 评测工具能够评测 HTAP 系统在数据一致性、实时性、负载隔离能力、新鲜度各维度的支持度.

4) 评测场景的公平性: 在评测不同数据库时, 需要保证相同的运行环境资源配置, 以及一致的 TP 负载压力和 AP 查询复杂度.

3　HTAP 评测基准设计维度

评测 HTAP 数据库系统时, 除了需要评测 TP 和 AP 性能以外, 也需要度量两种负载之间的相互影响, 在同一套评测基准中公平地度量这些影响. 因此, HTAP 评测基准需要从两类负载以及相互间的影响设计 3 个基本要素,

即数据、负载和度量指标, 如图 5 所示. 其中, 数据生成需要保证 TP 和 AP 语义一致性[31], 即 TP 和 AP 负载涉及的表模式在相同场景下, 且数据具有相同的特征. 其次, 负载[21]需要保证 TP 和 AP 负载语义逻辑正确且一致, 并能够访问相同数据, 同时控制混合负载的访问模式、两种负载的混合方式以及查询的复杂程度. 最后, 评价指标除了传统指标外, 还需要考虑数据同步过程中的耗时和两种负载相互干扰的影响. 此外, HTAP 评测基准还需要具备对基于不同一致性模型的 HTAP 数据库系统评测能力.

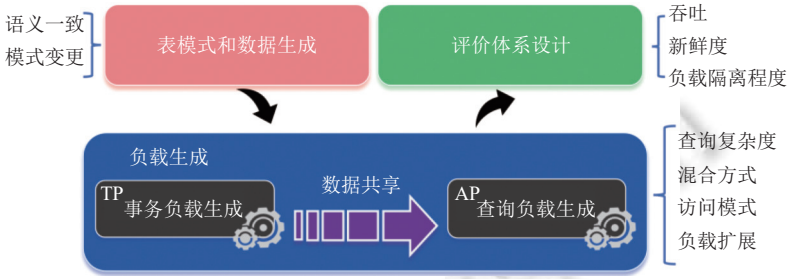

图 5 HTAP 评测基准构成元素

3.1 HTAP 数据生成

3.1.1 语义一致的表模式和数据

HTAP 数据库评测基准的表模式需要考虑语义一致的特征, 即支持在同一套表模式下运行事务负载和查询负载, 且保证混合负载访问相同的数据. 一般而言, 承担事务负载的表模式较为简单, 尽可能减少冗余设计, 以降低每个查询访问的数据, 提升处理性能[47]; 承担查询负载的表模式更为复杂, 往往包含多组动态变化的事实表和相对应的静态维度表. 在 HTAP 数据库评测基准中, 表模式需要同时满足两种负载的需要, 实现手段包括以下两类.

基于表模式扩展: 基于成熟的评测基准设计语义一致的表模式. 目前, 传统的 TP 和 AP 评测基准均有成熟的表模式, 如 TP 评测基准中的 TPC-C, AP 评测基准中的 TPC-H. HTAP 数据库评测基准可以在某一类表模式的基础上进行修改和扩展, 使之符合另一类负载的要求. CH-benCHmark[36]是第一个结合 TP 和 AP 负载, 提出评测 HTAP 负载处理能力的评测基准. 它以 TPC-C 的表模式为基础, 增加一系列符合 AP 负载要求的表, 以支持指定的 AP 查询, 即基于 TP 表模式扩展 AP 表模式, 并以最小化事务模式侵入代价的方式, 修改 TPC-H 的 22 个查询, 使之能够正确运行在扩展出来的 AP 表模式上. 而 HATtrick[24]则是以 SSB 的表模式为基础, 扩展生成符合 TP 要求的表模式.

自定义表模式: 根据 HTAP 负载的要求, 自行设计完整的表模式. 这种方式设计得到的表模式不会受到原有场景的限制, 更易于在同一个场景中满足两种负载要求. 且这种表模式可以更为平等地考察两种负载, 不会偏重于某个负载. OLxPBench[31]采用自定义表模式的方式, 其中 TP 与 AP 负载涉及的表既有重合的内容, 也有独立存在的部分. OLxPBench 基于这一表模式自定义 TP 和 AP 负载, 设计了 3 个场景 (零售、银行、电信), 其中只有部分 TP 负载访问的数据项会被 AP 负载查询, 两者在语义上一致, 保证了两种负载能够正常运行.

基于语义一致的表模式, 数据库的初始数据同样需要具有一定的一致性. 数据特征, 如数据分布, 会影响负载的执行性能. 因此两种负载要求数据满足特定的分布特征. 一方面, 在语义一致的表模式下, 两种负载需要访问相同的数据, 因此这部分数据需要同时满足两种负载的数据特征, 使两种数据特征达成一致; 另一方面, 所有表中的初始数据从语义上属于相同场景, 应当与场景所代表的数据特征保持一致.

3.1.2 模式变更

HTAP 评测基准的表模式除了需要保证语义一致, 还需要考虑模式变更的问题. 表模式变更往往服务于 AP 负载, 包括逻辑上修改表模式, 如增加、删除逻辑视图, 以及物理上修改数据库表结构, 如对表进行列更改、索引更改、创建或删除物化视图等. 无论是何种类型的变更, 表模式变更都涉及修改数据库元数据. 传统数据库需要在

模式发生变更时停止数据库服务, 进而影响 TP 负载的执行性能. 实时模式变更[48]指数据库系统在不停止服务的情况下修改数据库表和元数据, 目标是维持混合负载执行过程中的性能稳定. 由于改动表结构 (物理变更) 是常见的业务需求, HTAP 数据库系统[9,13,34]也在逐步增强对实时逻辑以及物理模式变更的支持能力. 为了评测 HTAP 数据库系统该特性, 评测基准的表模式设计也需要考虑对模式变更的支持.

两种类型表模式变更开销不同, 对表模式维护提出了不同程度的要求. 表模式逻辑变更只需要修改数据库元数据, 执行开销较小; 表模式物理变更则需要修改数据库的物理结构, 开销较大, 并且需要表模式满足一定的约束, 如用于构建索引的列非空等. 目前的评测基准主要关注逻辑变更, 对物理变更探索缺乏.

3.2 HTAP 负载生成

由于 TP 负载和 AP 负载对数据库系统施加的负载压力和需求的资源不同, 传统的数据库系统无法在一个实例中同时满足其性能需求且不互相干扰. HTAP 数据库则是通过隔离 TP 和 AP 负载, 以确保两种类型的处理可以同时运行而不互相干扰, 从而保障处理性能. HTAP 评测基准需要对混合负载性能和混合时的相互影响进行评估, 以求同时评测 HTAP 系统各维度性能[21].

与传统数据库的评测基准类似, HTAP 数据库评测基准也需要考虑 TP 和 AP 负载的生成. 为了更好模拟真实场景中不同的业务需求, TP 负载需要支持在事务中执行多种类型的操作, 如查询、插入、更新、删除等. 与此同时, 随着分布式数据库的广泛应用, TP 负载还需要支持对分布式事务的控制. 为了评测数据库在执行分析查询时的执行性能、优化能力以及系统的负载隔离能力, AP 负载需要包含足够复杂的查询, 对查询中涉及的子查询数量和连接规模提出了要求. 由于 TP 负载操作较为简单, 现有评测基准在操作和负载种类的覆盖上已较完善. 因此, 在 HTAP 评测基准中 TP 负载的生成将不再赘述, 仅简要分析对 AP 负载中复杂查询的支持能力.

HTAP 评测基准需要关注负载混合执行时的相互影响以及执行逻辑的正确性, 对负载的定义提出新的要求. 首先, 两种负载需要竞争资源, HTAP 数据库系统需要根据负载需求的资源数量调度和隔离资源. 负载的混合方式, 包括两种负载的划分方式和比例, 会影响数据库的负载的隔离效果和资源调度策略, 进而影响执行性能. 其次, TP 负载会修改和更新数据, 对数据库中的数据规模和分布造成影响, 从而间接影响 AP 负载的查询性能, 可能导致查询的性能波动. HTAP 评测基准需要控制 AP 负载所访问的数据在 TP 负载修改过程中规模和分布的稳定性. 同时, 基于其表模式和数据, HTAP 评测基准同时需要保证负载的语义一致性, 即 TP 和 AP 负载以正确且一致的负载语义逻辑访问相同的数据. 最后, 执行混合负载涉及 HTAP 数据库的多个模块, 包括 TP 执行引擎、AP 执行引擎和数据同步机制, 特定的负载访问模式对各个模块造成的压力不同, 通过控制访问模式, 评测基准可以更好地评测各个模块的性能.

3.2.1 查询复杂度

在 AP 负载中, 查询模板的复杂度取决于算子的复杂性、连接的规模和子查询的数量. 基础算子依据计算复杂性可以分为两种, 其中投影、选择、聚合、过滤算子只需要遍历一次数据集, 对数据进行简单计算, 执行代价较低, 仅在处理大规模数据情况下会占用较多的计算资源和内存空间. 排序算子需要对数据进行多次遍历并排序, 连接算子需要传输、遍历和比较两组数据, 执行代价较高. 连接算子除了选择不同的物理执行方法之外, 不同的连接顺序也会对查询的执行性能造成巨大的影响. 优化连接的顺序是数据库查询优化的重要方法. 因此连接的规模越大, 涉及的表数量越多, 查询越复杂, 对数据库查询优化的评测能力越强.

除了基础算子外, 嵌套子查询也可以根据查询代价分为两种, 即相关子查询和非相关子查询. 相关子查询中, 内部查询包含对外部查询结果的引用, 需要多次执行并访问外部查询的结果集, 因此它的执行代价相较非相关子查询更高. 非相关子查询和外部查询之间没有关联, 通常用于计算统计数据、检查数据一致性等查询需求, 执行代价较低. 因此在评测 HTAP 数据库系统的分析处理能力时, 包含不同复杂度算子的测试基准必不可少.

3.2.2 混合方式

HTAP 评测基准在语义一致的表模式下, 基于一定的混合方式同时执行 TP 和 AP 两类负载. 负载的混合方式包括是否划分独立的 TP 和 AP 负载, 以及两种负载的执行比例. 一般来说, 在现有的 HTAP 评测基准中, TP 和

AP 负载相对独立. 一方面, 两种负载所使用的模板独立, 在 TP 负载的模板中不存在复杂的分析查询, 反之亦然; 另一方面, 两种负载的执行流程互不干扰, 通常分别以线程为粒度独立执行, 一个线程内只执行一类负载. 这便利了对两种负载比例的控制, 评测基准可以通过调整对应线程的数量直观地控制负载的运行比例, 进而测试不同的负载混合比例. 除了采用这类较为独立的混合方式, 部分 Micro-Benchmark[29,49] 采用更细粒度的负载混合方式, 只给定不同的操作模板, 在进行评测时自由组合, 不明确区分两种负载的模板. 这种混合方式可以模拟 TP 和 AP 负载混合交错的场景, 但难以控制两种负载的比例.

3.2.3 访问模式

访问模式刻画了基于给定表模式的访问场景, 用于评测数据库面临不同负载时的执行性能. 根据对数据库各模块施加的压力不同, 我们基于正常业务下的访问模式, 以及造成性能瓶颈的访问模式, 将访问模式划分为 4 种, 分别是读写平衡场景、写密集场景、读密集场景和大批量读取场景, 如表 2 所示. 在读写平衡场景中, TP 和 AP 负载的执行强度低, 对数据库中各个模块的性能压力小, 描述了业务场景正常运行的情况. 写密集场景刻画了 TP 负载高强度执行, 且 AP 负载执行强度低的情况, 对数据库中 TP 执行引擎的性能压力较大. 在读密集场景中, TP 负载以一定强度执行的同时, AP 负载进行频繁的查询, 这对数据库数据同步机制造成较大的性能压力. 大批量读取场景刻画了 AP 负载执行大批量复杂查询, TP 负载执行强度低的情况, 对数据库的 AP 执行引擎造成较大的压力. 4 种访问模式分别产生了对数据库 3 个主要模块造成不同压力的场景, 用于评测数据库应对不同场景时的处理能力. 目前, OLxPBench 评测基准对访问模式做出了定义, 可以通过改变 TP 和 AP 负载的混合比例以控制访问模式, 但大部分 HTAP 评测基准尚未对访问模式进行明确的定义和刻画.

表 2 访问模式分类

访问模式	负载强度		数据库模块承担的压力		
	TP	AP	TP执行引擎	AP执行引擎	数据同步机制
读写平衡	低	低	低	低	低
写密集	高	低	高	低	低
读密集	中	高	中	中	高
大批量读	低	高	低	高	低

3.2.4 负载扩展

执行混合负载会改变数据库系统中的数据, 进而影响分析查询计算复杂度变化, 影响分析性能. 为了满足性能稳定性需求以实现公平比较的目标, 评测基准需要控制分析查询涉及的数据规模随 TP 负载的执行稳定扩展. 与此同时, 为了保证负载的语义一致性, 评测基准需要确保混合负载以一致且正确的语义访问相同的数据, 所以 AP 查询访问的数据除规模随 TP 负载的执行变化, 其内容也需要与 TP 负载访问的数据相关.

TP 负载通常包含大量访问和修改少量数据的事务. 在负载执行过程中, 数据库中的数据量很可能不断变化, 往往遵循连续伸缩的数据扩展模型 [24]; 而 AP 负载一般用于分析处理大规模数据, 通常要求数据规模稳定或仅有小范围数据更新. HTAP 评测基准需要同时保证 TP 负载和 AP 负载的要求, 即 TP 负载可以新增大量数据, 同时 AP 负载在访问新鲜数据的同时保持稳定的性能, 即大规模数据的计算要保持稳定的复杂度. 因此, TP 负载新增的数据需要以可控的方式扩展至 AP 端, 使 AP 端读取的数据数量变化趋势和数据分布稳定, 以保证计算的稳定性. 目前有两种负载数据对齐的方式, 一种是 AP 负载向 TP 负载的对齐, 在 TP 负载执行过程中将以行为单位连续增加的新数据映射到以基于扩展因子 (scale factor, SF) 比例扩展的 AP 表模式中, 使得新增数据均匀落在 AP 负载查询的范围上, 如 CH-benCHmark; 另一种是 TP 负载与 AP 负载相对独立, AP 的查询涉及维度表和事实表, 而 TP 负载只在一个事实表上新增数据, 对 AP 负载影响较小, 如 HATtrick. 现有评测基准多采用前一种方式, 此时评测基准需要考虑 TP 负载所影响的数据量, 通过 AP 查询的参数控制读取的数据规模. 但大部分评测基准对 AP 查询参数控制还沿袭了传统 AP 基准的控制方法, 即没有定量控制 TP 端数据增长对 AP 端数据访问的影响; 少数评测基准考虑了控制过滤条件, 如 HTAPBench 控制时间相关的过滤条件, Swarm64 控制了多种过滤条件的参数. 评测基

准也通过控制过滤条件以实现负载的语义一致, 即保证 AP 查询能根据正确的语义访问 TP 负载修改的数据内容. CH-benCHmark 未考虑过滤条件, 因此无法控制 AP 查询访问最新的数据; HTAPBench 和 Swarm64 则控制过滤条件, 使得 AP 查询能够访问 TP 负载修改的数据. 但由于它们基于已有表模式扩展, TP 和 AP 负载所访问的数据大部分不相交, 语义一致的设计仍有欠缺.

3.3 HTAP 评价指标

评测基准应该使用具备可靠性、有效性和敏感性的评测指标, 为数据库使用者和开发者提供决策依据和改进方向. 数据库系统常用的评测指标包括响应时间、吞吐量、并发度、资源利用率等[21]. 结合 HTAP 数据库的特征, HTAP 评测基准的评测指标应该关注 3 个维度, 即吞吐、新鲜度和负载隔离能力.

3.3.1 吞　吐

HTAP 评测基准需要评测并量化 TP 和 AP 负载极限吞吐性能, 常用的指标包括用于评价 TP 负载的吞吐量 (TPS) 和评测 AP 负载的查询响应时间 (Latency). 同时, 由于 TP 负载不断修改或增减数据, 数据规模和分布动态变化, 吞吐指标需要保证在多次执行时评测结果稳定, 以使不同数据规模下的结果可比较. Funke 等人[50] 提出应当考虑 HTAP 负载执行时数据规模不断上升对查询性能的影响, 捕获查询响应时间与数据规模的关系.

HTAP 评测面向的对象是整个 HTAP 数据库系统, 不是单个独立的 TP 数据库或是 AP 数据库, 而且两种负载存在数据共享和负载隔离的矛盾, 统一的评测指标难以设计. 现有的工作或是继续沿用原有的吞吐指标, 或是对常用吞吐指标进行组合, 如 CH-benCHmark 提到使用 $\frac{tpmC}{QphH}@tpmC$ 和 $\frac{tpmC}{QphH}@QphH$ 来进行单个数据库 TP/AP 性能的展示, HTAPBench 使用 $\frac{QphH}{\#OLAPworkers}@tpmC$ 进行单个 worker 性能之间的比较.

3.3.2 新鲜度

在 HTAP 数据库系统中, TP 负载修改的数据可能需要一定的延迟才能被 AP 查询读取. HTAP 评测基准采用新鲜度描述这一延迟. 一个数据项的新鲜度越高, 表示相同时刻下, AP 端读所获取的版本越接近于 TP 端修改的版本, 反之则表示 AP 端读取的版本相对较老. 数据同步机制需要使 AP 端尽快读取到 TP 负载最新修改数据版本. 数据同步机制的设计决定了数据同步的时延, 进而影响了数据的同步耗时和新鲜度. 对于 HTAP 数据库系统的实时数据处理能力而言, 数据的时效性是保证分析结果价值的关键因素, 因此, 评测 HTAP 数据库系统的分析能力时, 新鲜度指标非常重要.

现有工作对新鲜度的定义也存在一定的差异. 在信息系统中, 新鲜度基于数据源抽取数据并发送给用户这一过程产生, 根据度量维度和指标分为两类 4 种[51], 如表 3 所示. 一类是根据数据抽取过程定义新鲜度, 比如, 数据抽取阶段的耗时、数据抽取过程中数据源变更频度、所抽取数据中数据更新率等, 另外一类是基于数据的实时性定义新鲜度, 如当前读取数据版本和最新数据版本生成的时间差.

表 3　信息系统中新鲜度主要因素和指标分类[51]

度量维度	度量指标	描述
数据抽取过程	数据流通耗时	数据抽取阶段的耗时
	数据变更频度	数据抽取过程中数据源的修改次数
	数据更新率	所抽取数据中新鲜数据的比例
数据实时性	时间差距	当前读取数据版本和最新数据版本生成的时间差

现有的 HTAP 数据库评测基准计算新鲜度的方式大多扩展自信息系统对新鲜度的定义, 如表 4 所示. Adaptive HTAP 论文[44] 考量的是 TP/AP 端数据版本重叠的比例; Hyper 认为新鲜度与快照建立的频率相关, 将新鲜度作为一种调控快照建立速度的参数. Vegito[15] 和 ByteHTAP[52] 定义 TP 负载最新产生数据令 AP 端可见的时间为新鲜度, 而 HATtrick 定义新鲜度为全局最新快照点落后的时间. 在这 5 种定义中, 除 Adaptive HTAP 论文所定义的新鲜度是扩展自信息系统中的数据更新率指标外, 其余 4 种定义均基于时间差距指标. 因此当前数据库系统和评测基准主要基于混合负载在访问数据的版本上的时间差距指标定义新鲜度. 其中, 大多对新鲜度的定义由数据库系

统给出, 评测基准中仅 HATtrick 定义了新鲜度, 其余评测基准并无提出或定义新鲜度指标以评测 HTAP 数据库系统, 不能区分基于线性一致性和非线性一致性模型的数据库系统.

表 4 HTAP 数据库系统新鲜度计算方式总结

相关工作名称	度量指标的扩展来源	计算方式
Adaptive HTAP[44]	数据更新率	TP/AP端数据版本重叠比例
Hyper[46]	时间差距	快照建立的频率
Vegito[15]	时间差距	TP端最新产生数据在AP端可见的时间差
ByteHTAP[52]	时间差距	
HATtrick[24]	时间差距	AP全局快照落后TP端全局快照的时间差

3.3.3 负载隔离能力

对于 HTAP 数据库而言, 不同的负载隔离方式对应不同的数据同步方式[19], TP 和 AP 负载之间的相互干扰程度也不同. 目前大部分 HTAP 评测基准或是不控制负载, 或是通过控制一种负载的线程数或吞吐不变, 观察另一种负载的最大吞吐能力以评测 HTAP 数据库系统的负载隔离能力. 除了这两种方式, HATtrick 提出采用二维图刻画吞吐边界的方式评测负载隔离能力. 它将不同线程下的 TP/AP 吞吐作为坐标, 绘制在二维图表中, 通过吞吐边界和线条的平行程度反映负载隔离能力的强弱, 如图 6 所示. 图 6(a) 中蓝色虚线表示 TP 负载吞吐, 对应 x 轴, 红色虚线表示 AP 负载吞吐, 对应 y 轴. 虚线与对应的坐标轴交角越小, 与坐标轴越平行, 表示增加对应负载线程对另一种负载的吞吐能力影响越小, 负载隔离能力越高. 图 6(b)、(c)、(d) 中的实线由整个图形的最外围边界线连接而形成, 它越靠近对角线, 在特定负载吞吐下, 另一种负载性能的预测性越好; 越靠近红色虚线, 负载隔离能力越高.

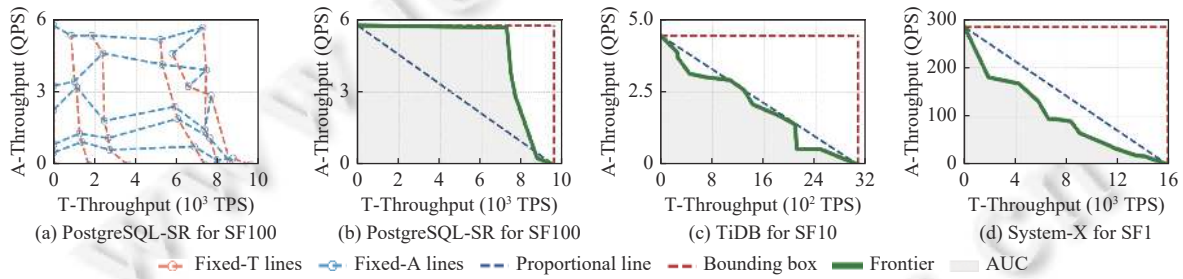

图 6 HATtrick 通过吞吐边界及线条平行程度反映负载隔离能力强弱的示意图[24]

3.4 一致性模型支持性

HTAP 数据库系统采用 3 种一致性模型来满足不同的数据同步需求, 如表 5 所示. 基于不同一致性模型的数据库系统, 存在读写数据版本的差异性. 若评测的数据库采用间歇同步的顺序一致性和按需同步的会话一致性模型, HTAP 评测基准在计算新鲜度时需要区分所读数据的版本与系统中全局最新写版本. 不同的 HTAP 评测基准对不同一致性模型的测评能力有所不同. 评测基准的一致性模型支持性旨在描述该能力, 体现为对读写数据版本差异的评测能力. CBTR 主要针对采用 ETL 方式同步的数据库设计, 在 TP 和 AP 负载之间存在一次显式的模式转换, 与现在的 HTAP 在数据同步方面都有较大的差异, 不适合对现在的 HTAP 数据库系统进行评测比较. CH-benCHmark、HTAPBench 等只考虑了线性一致性的同步模型, 不区分读写数据版本的差异, 因此它们无法有效评测基于后两种一致性模型的 HTAP 数据库系统的数据同步.

表 5 3 种一致性模型对比[19]

一致性模型	同一时刻读取的数据版本特征	同步方式	适用场景
线性一致性	各AP端一致且最新	数据实时同步	查询实时数据
顺序一致性	各AP端数据一致, 但不保证是最新的	数据间歇同步	周期性分析存在有限延迟的数据
会话一致性	各AP 端数据既非最新的, 也非一致的	数据按需同步	频繁分析可能存在延迟的数据

4 已有 HTAP 评测基准实现技术总结

综上所述, 在 HTAP 评测基准的设计中, 数据、负载、评测指标这 3 个维度上的设计和考量不同于传统的面向 OLTP/OLAP 数据库系统的评测基准, 且需要考虑评测基于不同一致性模型的 HTAP 数据库系统. 在表 6 中, 根据前文对评测基准设计维度的分析, 本文对现有 HTAP 评测基准进行分类和技术总结, 包括 7 款面向应用的基准 (Macro Benchmark) 和两款面向技术点的评测基准 (Micro Benchmark). 虽然面向应用和面向具体技术点的评测基准在评测对象和复杂程度上有所差异, 但它们的设计维度一致, 所以表 6 统一展示了两类评测基准在 4 个评测维度上的设计. 其中, “ — ”表示没有明确定义.

表 6 HTAP 评测基准能力总结

评测基准名称	表模式和数据生成		负载生成						评价指标			一致性模型支持性
	语义一致性	模式变换	访问模式	负载扩展	查询复杂度		混合方式		吞吐指标	新鲜度指标	负载隔离能力	
					子查询数量	表数量	划分负载	负载比例				
CBTR	自定义表模式	—	—	向TP负载对齐	—	—	是	不固定	TPS QPS	—	—	线性一致性
CH-benCHmark	从TP向AP扩展	逻辑变更	—		3	8			$\frac{tpmC}{QphH}@tpmC$ $\frac{tpmC}{QphH}@QphH$		固定线程	
HTAPBench					3	8			$\frac{QphH}{\#OLAPworkers}@tpmC$		固定吞吐	
Swarm64					3	8			TPS QPS		固定线程	
CH3		—			3	8			TPS QPS			
HATtrick	从AP向TP扩展				0	5		固定	$\max(0, t_{Aq}^{s} - t_{Aq}^{f_{ns}})$	有	刻画吞吐边界	线性一致性 顺序一致性 会话一致性
OLxPBench	自定义表模式		3种模式	相对独立	2	4		不固定	TPS QPS	—	固定线程	线性一致性
ADAPT			—		0	1	否		TPS QPS		—	
HAP					0	1			TPS QPS		—	

4.1 面向应用的评测基准

大部分 HTAP 评测基准扩展自己有 TP 和 AP 评测基准, 也有少数基准自行设计表模式和负载. 除了设计混合负载的生成方式, 部分 HTAP 评测基准也提出了新的评价指标, 用于展示 HTAP 数据库系统的整体能力. 我们总结了 7 款主流的 Macro HTAP 评测基准, 并分别介绍它们在各设计维度上的考量和设计亮点.

4.1.1 CBTR

CBTR[35] 在 2008 年提出, 是由事务处理和运维报告组成的复合基准. 它基于一个全球运营企业的原始数据集合构建负载, 抽象出企业内部电商业务场景, 重点考虑事务处理和运维报告之间的干扰. CBTR 基于业务场景自定义表模式, 以部门数为扩展因子, 即通过增加部门进行数据扩展. CBTR 运行时向 TP 负载数据对齐. CBTR 的事务处理负载 (OLTP) 包含读写事务、只读事务的事务模板; 由于提出时间较早, CBTR 中的运维报告与 HTAP 数据库系统中的 OLAP 负载不同, 它不考虑实时查询. 运维报告与事务处理负载以线程为粒度进行划分, 可以设置两种负载的比例, 但没有定义特定的访问模式. 同时, CBTR 仍然采用传统的评价指标, 即 TPS 和 QPS, 未考虑新鲜度和负载隔离能力的影响, 只支持测试线性一致性下的模型, 无法区分不同数据同步模型的差异. 虽然它是第一个提

出同时评测 TP 和 AP 能力的评测基准, 但是它并不要求 AP 分析 TP 实时修改的数据, 并不满足现代 HTAP 数据库系统的评测要求.

4.1.2 CH-benCHmark

CH-benCHmark[36] 于 2011 年被提出, 是第 1 个通过结合 TPC-C 和 TPC-H 实现的 HTAP 评测基准, 并提出综合考虑 TP 和 AP 性能的评价指标. 它基于标准的 TPC-C 表模式, 扩展了修改后的 TPC-H 表模式, 并在特定查询中创建视图, 支持表模式的逻辑变更. 在 CH-benCHmark 中, TP 负载使用 TPC-C 的 5 个事务模板; AP 负载使用 TPC-H 的 22 个查询, 如图 7 所示. 在 AP 负载中, 一个查询包含至多 3 个子查询, 连接至多 8 张表, 包含了不同复杂度的算子. 该设计被后续的评测基准广泛采用, 包括 HTAPBench 以及 Swarm64, 因此这 3 个评测基准的表模式和数据生成、以及负载生成部分一致, 主要区别在于其评价指标的设计和实现. 在运行过程中, AP 负载的查询参数向 TP 负载对齐, 以仓库为扩展因子, 即通过增加仓库的方式扩展数据. 与 CBTR 类似, 两种负载运行在不同线程上, 负载运行的线程数和比例不固定, 未定义访问模式. CH-benCHmark 首次采用 $\frac{tpmC}{QphH}@tpmC$ 和 $\frac{tpmC}{QphH}@QphH$ 作为评测指标, 分别表示在给定 TP 负载吞吐和 AP 负载吞吐下两种负载吞吐的比值, 可以同时描述了两种负载的执行性能. CH-benCHmark 通过固定执行线程的方式评测 HTAP 数据库系统的负载隔离能力. 虽然评测指标可以客观描述混合负载的执行性能, 但是由于不同数据库的吞吐能力不同, 无法直接横向比较数据库系统的性能.

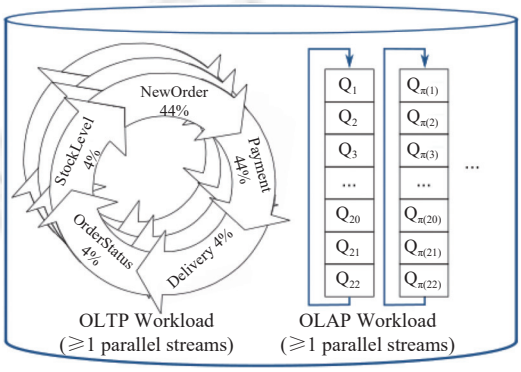

图 7 CH-benCHmark 的 TP 负载和 AP 负载运行模式[36]

4.1.3 HTAPBench

HTAPBench[37] 在 2017 年被提出, 首次采用固定 TP 负载吞吐的方式测试数据库系统的负载隔离能力, 并引入了时间窗口 (time window) 以控制 AP 查询的数据访问边界. 它的负载和表模式同 CH-benchmark 一致, 也是基于 TPC-C 的表模式扩展 AP 的表模式, 并在独立的线程中运行对应评测基准的负载, 可自定义负载的混合比例. HTAPBench 在测试过程中, 在固定 TP 吞吐下不断增加 AP 线程, 直到吞吐低于特定阈值, 以此来测试在固定吞吐下数据库的最大 AP 性能, 以及在 HTAP 场景下两种负载的干扰程度, 以此显示数据库系统的 HTAP 能力. 为了进一步提高评测基准对查询性能的比较能力, HTAPBench 引入了时间窗口. 当分析查询遍历数据时, 根据事务负载更新的数据改变查询范围, 进而控制查询数据的规模. HTAPBench 通过将查询范围限定在特定时间窗口内, 缓解了这个问题. 另一方面, 基于固定的 TP 吞吐, HTAPBench 对 CH-benCHmark 的评测指标进行修改, 使用 $\frac{QphH}{\#OLAPworkers}@tpmC$ 表示在特定 tpmC 下 AP 线程的平均吞吐. 由于 HTAPBench 固定了 TP 吞吐, 各 HTAP 数据库系统的这一指标可以横向比较.

4.1.4 Swarm64

Swarm64[38] 与 HTAPBench 类似, 严格控制了查询的数据规模, 并且保证 AP 负载查询涉及最新数据. 它的负载和表模式同 CH-benCHmark 一致. Swarm64 同样将两种负载划分到不同线程中执行. 在固定 TP 线程的情况下,

它通过控制 TP 负载执行事务的间隔保证了 TP 吞吐拥有稳定的下界, 并获得每个事务的执行时间. 相比 HTAPBench, Swarm64 只固定 TP 负载的线程数而不固定 TP 负载的吞吐. 在此基础上, Swarm64 维护最新的事务执行时间, 并将这一时间同步至 AP 线程. AP 线程根据最新的执行时间设定查询参数. Swarm64 的查询参数使查询访问最新更新的数据, 保证与 TP 负载访问数据之间存在重叠, 从而测试 HTAP 数据库系统的同步能力. Swarm64 虽然相较于 HTAPBench, 保证了 AP 负载能访问 TP 负载最新修改的数据, 但只使用传统的 TPS 和 QPS 分别描述两种负载的吞吐, 没有统一的指标来评测 HTAP 数据库系统的能力, 并且与 HTAPBench 一样, 只能测试基于线性一致性的 HTAP 数据库系统.

4.1.5 CH3

CH3[53] 于 2022 年被提出, 是一款用于评测 NoSQL 数据库 HOAP (Hybrid Operational/Analytical Processing) 能力的评测基准. CH3 沿袭 CH2[54] 的思路, 将 NoSQL 数据库的 Operation 操作与关系型数据库的事务对标, 在 CH-benCHmark 的基础上适配了 NoSQL 的使用场景, 并增加了符合 NoSQL 的特殊负载. CH3 的表模式和负载与 CH-benCHmark 相似, 但与之不同的是, 由于 NoSQL 不需要遵循严格的 1NF 范式, CH3 中将 Order-Line 表与 Order 表合并, 得到具有嵌套内容的表 Orders. 同时, 因为 NoSQL 对表模式无严格限制, 在 CH3 中也没有对模式变化的支持. 除基本的 TP/AP 负载, 基于 NoSQL 的应用场景, CH3 中增加了对数据库的全文扫描 (FTS) 查询, 并为此增强了数据库各字段的语义. 它将 3 种负载划分到不同线程中执行, 线程的数量和比例不固定, 没有设计特定的访问模式. CH3 采用 TPS 和 QPS 作为吞吐指标, 以固定线程的方式测试数据库系统的负载隔离能力, 只能评测基于线性一致性的 HTAP 数据库系统.

4.1.6 HATtrick

HATtrick[24] 在 2022 年被推出, 在评测中考虑了新鲜度的计算方式, 并创新性地使用吞吐边界刻画两种负载之间的干扰. 它基于评测 AP 数据库的 SSB 评测基准构建表模式, 将 AP 的表模式扩展到 TP 负载的表模式, 共包含 5 张维度表和 1 张事实表. TP 负载只在事实表上增加数据, 和 AP 负载相对独立. 基于这一表模式, HATtrick 自定义了 3 个事务模板, 仅包含 insert、select 和 update 操作, 不支持表模式变更. AP 负载使用 SSB 基准原本的 4 类 13 个查询, 不包含子查询, 负载较为简单. HATtrick 在评价指标上进行了创新. 它将新鲜度定义为查询发起版本与第 1 个不可见的 TP 版本之间的时间差, 支持量化和比较各 HTAP 数据库系统的新鲜度. 它提出使用吞吐边界 (throughput frontier) 测量负载隔离能力, 如 3.3.3 节及图 6 所示. 同时, 它对新鲜度的定义考虑了事务和查询访问数据项版本差, 可以评测基于线性、顺序、会话一致性的数据库系统.

4.1.7 OLxPBench

OLxPBench[31] 于 2022 年被推出, 率先提出 HTAP 的基准评测需要具有语义一致的表模式, 实时查询以及不同应用领域的负载. OLxPBench 指出, 基于 TPC-C 与 TPC-H 结合的表模式语义不一致, TP 写入的数据不能被 AP 负载完全读取. 这降低了混合负载间的竞争冲突, 难以深入地评测 HTAP 数据库系统. 它基于 TPC-C 的单一表模式和事务, 自定义一套 AP 负载使得负载之间的访问交织更加充分, 并且定义了 3 种访问模式, 具备分别评测数据库系统各模块的能力. 但 OLxPBench 的 AP 负载包含的查询较 TPC-H 更为简单, 不支持创建视图、对表模式进行逻辑变更, 且其参数相对独立, 不考虑对齐 TP 负载所更新数据. 其 TP 和 AP 负载同样划分到不同线程上, 不固定线程比例. 除此之外, OLxPBench 在 TP 事务执行之前添加一次实时单点查询, 以模拟实际业务场景中的数据统计与决策行为, 提高了两种负载的混合程度. 最后, 其引入了 Fibenchmark 和 Tabenchmark 分别源于 Small Bank 和 TATP 的面向银行和通信业务的特殊基准评测, 以更好地迎合特定领域的 HTAP 场景评测需求.

4.2 Micro-Benchmark

Micro-Benchmark 是用于测试和评估单个小功能或模块性能的基准测试, 它更关注系统的某个局部性能, 如计算、内存读写等. Micro-Benchmark 针对系统的局部设计具有更高的评测精度, 可以帮助开发人员更好地理解和优化系统性能的瓶颈/局限性.

4.2.1 ADAPT

ADAPT[29] 是 2016 年被提出, 用于评测 HTAP 数据库在不同数据存储格式 (行存 NSM、列存 DSM、混合存

FSM) 下性能表现. 不同于普通的 HTAP 数据库评测基准, 它在表模式设计上侧重于测试系统不同存储格式对于查询性能的影响, 共包含两张表, 一张包含 50 个属性的窄表和一张包含 500 个属性的宽表, 宽表能够明显放大查询在列存和行存执行上的性能差异.

该评测基准未定义 TP 负载和 AP 负载的模板, 使用者可根据需要从备选查询中挑选后进行组合. 备选查询共 5 个, 包含插入、扫描、聚合、查询求和以及连接, 如图 8 所示. 其中的 k 和 δ 控制查询投影和数据范围, 用于影响负载中查询的执行. 负载参数在执行过程中可由用户动态改变. 首先, 加载初始负载进行预热, 让数据库识别到系统使用的存储方式后, 再修改负载中的参数, 通过比较不同存储模型在宽表和窄表上执行不同类型查询时的性能, 表现评测不同存储模型. 该评测基准无法区分对数据项的修改和该数据项为查询可见的版本差距, 仅适合评测基于线性一致性的 HTAP 数据库, 且缺少 delete 和 update 操作, 缺乏对 TP 负载处理能力的测试能力.

Q_1: **INSERT INTO** R **VALUES** $(a_0, a_1, \ldots, a_p)$
Q_2: **SELECT** $a_1, a_2, \ldots, a_k$ **FROM** R **WHERE** $a_0 < \delta$
Q_3: **SELECT** MAX(a_1),...,MAX(a_k) **FROM** R **WHERE** $a_0 < \delta$
Q_4: **SELECT** $a_1 + a_2 + \ldots + a_k$ **FROM** R **WHERE** $a_0 < \delta$
Q_5: **SELECT** X.a_1,...,X.a_k,Y.a_1,...,Y.a_k
FROM R **AS** X, R **AS** Y **WHERE** X.a_i < Y.a_j

图 8 ADAPT 评测基准查询负载[29]

4.2.2 HAP

HAP (hybrid access patterns)[49] 评测基准基于 ADAPT 设计. 它自定义表模式, 其中包含两张表, 一张 16 属性的窄表和一张 160 属性的宽表, 并基于表模式设计了 6 个查询, 包含点查询、范围查询、增删改等操作, 主要测试数据库系统的访问路径和更新性能, 如图 9 所示.

Q_1: **SELECT** $a_1, a_2, \ldots, a_k$ **FROM** R **WHERE** $a_0 = v$
Q_2: **SELECT** $count(*)$ **FROM** R **WHERE** $a_0 \in [v_s, v_e)$
Q_3: **SELECT** $a_1 + a_2 + \cdots + a_k$ **FROM** R **WHERE** $a_0 \in [v_s, v_e)$
Q_4: **INSERT INTO** R **VALUES** $(a_0, a_1, a_2, \ldots, a_p)$
Q_5: **DELETE FROM** R **WHERE** $a_0 = v$
Q_6: **UPDATE** R **SET** $a_0 = v_{new}$ **WHERE** $a_0 = v$

图 9 HAP 评测基准查询负载[49]

HAP 由查询组合形成 3 类工作负载, 分别是混合负载、只读负载和只更新负载. 混合工作负载中包含单点读和范围读的情况, 只读和只更新的工作负载考虑了均匀访问和偏斜访问两种数据访问分布. 为了模拟混合负载中的频繁更新, 每个工作负载均存在少量的更新操作. 与 ADAPT 不同的是, HAP 中提供了更多的查询参数, 包括 k、v、v_s 和 v_e, 可用于调整负载的投影、数据范围、访问数据的重叠比例和冷热数据分布, 可以模拟更复杂的业务场景. 但它与 ADAPT 相似, 并不能区分读取版本与修改版本之间的版本差距, 无法评测基于顺序一致性和会话一致性的 HTAP 数据库系统.

5 实验分析

本节选择代表性评测基准, 通过实验验证它们在评测基准的可重复性, 负载隔离能力以及数据新鲜度的评测能力.

5.1 实验环境

实验使用 4 台服务器, 每台服务器拥有 8 个 Intel(R) Core(TM) i5-8500 CPU @ 3.00 GHz, 32 GB 内存和 100 GB SSD 磁盘, 系统平台为 CentOS 7.9. HTAP 数据库系统选用 PostgreSQL Streaming (v14.5)、OceanBase (v4.2) 以及 TiDB (v6.1), 其中, PostgreSQL Streaming 是各对比 HTAP 评测基准都进行验证的一款 HTAP 数据库系统, 其采用顺序一致性模型, 所以数据同步具有非实时性的特征; OceanBase 作为一款 HTAP 数据库系统, 提供了弱一致性读的特性, 支持基于负载压力自适应调整一致性模型. TiDB 是典型的基于线性一致性进行数据同步的 HTAP 数据

库系统, 所以具有读取实时数据的特征. 在 4 台服务器中, 其中一台服务器用于运行 HTAP 评测基准, 3 台服务器上部署 3 节点的 PostgreSQL Streaming、OceanBase、TiDB 集群. PostgreSQL Streaming 集群具有一个主节点、一个从节点和一个读节点, 并且它提供多种数据同步方式, 当从它的读节点读取数据时, 满足顺序一致性的同步要求; 而读取它的主节点时, 可以获取当前最新的数据, 满足线性一致性的要求. 因此, 通过选择 AP 查询访问的节点, 可以改变所读取数据在同步时所遵循的一致性模型. 在实验中, 分别在主节点和读节点处理 TP 和 AP 负载, 默认 AP 查询读取 PostgreSQL Streaming 的读节点, 也即采用顺序一致性读取数据. 同时, 设置 OceanBase 开启其弱一致性读特性, 以自适应调整一致性模型. 因此我们采用 PostgreSQL Streaming 和 OceanBase 比较不同评测基准评测不同数据同步模型的效果. 同时, 在比较评测基准对新鲜度的评测能力时, 实验中增加了基于线性一致性的 TiDB 作为对比对象.

实时分析是 HTAP 数据库系统的核心功能, 也是 HTAP 评测基准关注的重点. 本节考虑评测基准评测实时查询的能力, 所以实验中评估了 4 款开源的具有较为复杂实时分析查询要求的评测基准, 包括最新的 HATtrick、CH-benCHmark、HTAPBench 和 Swarm64. 由于 AP 负载性能与数据规模紧密相关, 实验中控制 4 款评测基准运行的数据规模均为 12 GB.

5.2 评测过程的可重复性

评测过程的可重复性是对评测基准的基本要求. 为了测试现有评测基准的可重复性, 实验固定 CH-benCHmark、HTAPBench 和 Swarm64 的 TP 线程数和 AP 线程数均为 8, 分别运行 4 款评测基准 3 次.

其中, CH-benCHmark、HTAPBench 和 Swarm64 具有相同的 AP 查询模板. 实验选择了 3 种具有不同复杂度的查询即 Q6、Q10、Q13, 其中 Q6 是仅涉及单表的简单查询, Q10 为根据时间条件过滤的多表连接查询, Q13 为含有子查询的复杂查询. 图 10、图 11 展示它们在混合负载运行过程中的延迟变化. 当查询结构简单, 不包含连接和子查询时, 如 Q6, 各评测基准均能得到稳定结果. 但对于含连接或子查询的复杂查询, TP 负载的运行会影响查询性能, 导致相同查询的延迟波动. 因此若不对查询的数据规模进行控制, 如 CH-benCHmark, 评测结果将存在明显波动, 难以重复. HTAPBench 基于时间窗口, 通过时间条件控制查询的数据规模, 因此在 Q10 上的执行延迟趋于稳定. Swarm64 进一步控制了其他过滤条件, 当查询不含时间条件时, HTAPBench 无法像 Swarm64 一样控制查询的数据规模, 所以在 Q13 上表现不如 Swarm64 稳定. 综上, 现有的评测基准无法量化控制参数使得 AP 查询性能稳定, 导致大部分评测基准的评测结果不可重现.

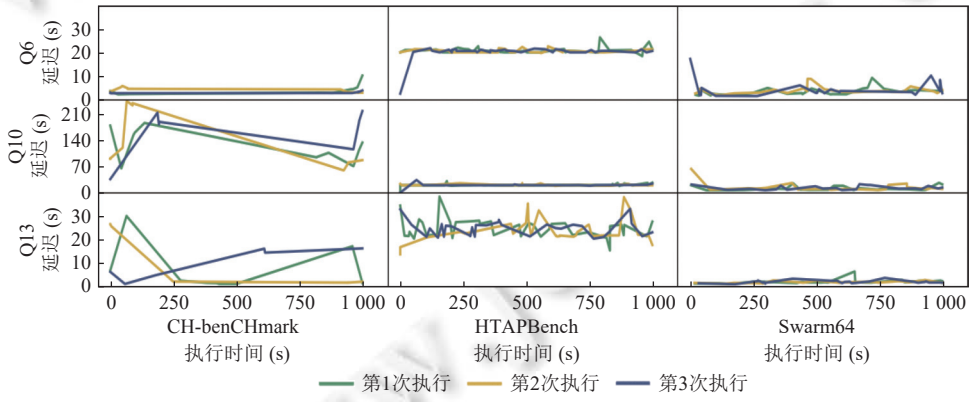

图 10 PostgreSQL Streaming 执行不同类型查询的延迟变化

HATtrick 使用吞吐边界刻画评测的结果, 在 PostgreSQL Streaming 和 OceanBase 上分别运行 3 次获得的吞吐边界如图 12 所示. 多次运行 HATtrick 获得的吞吐边界较为稳定, 这一方面是由于 HATtrick 评测基准中 AP 查询简单, 吞吐对查询数据规模不敏感; 另一方面是由于 HATtrick 在一次执行中, 基于不同 TP/AP 负载比例下执行结果的最高吞吐量画出吞吐边界, 对于给定的数据库, 其性能固定, 因而获得的吞吐边界稳定.

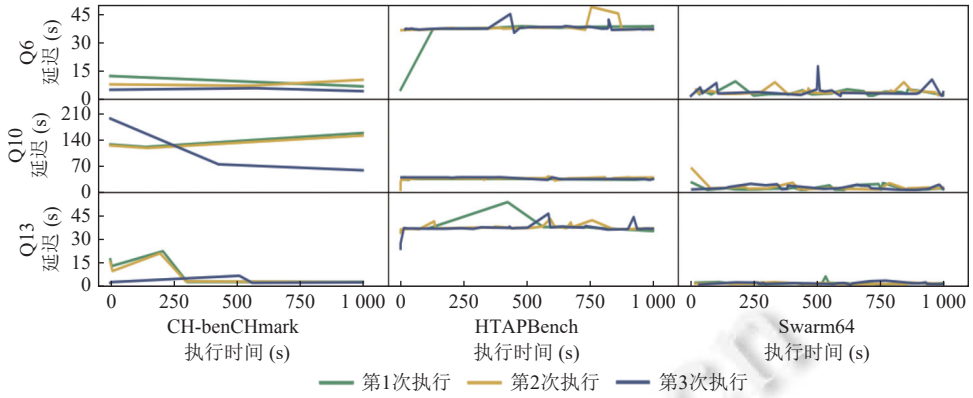

图 11 OceanBase 执行不同类型查询的延迟变化

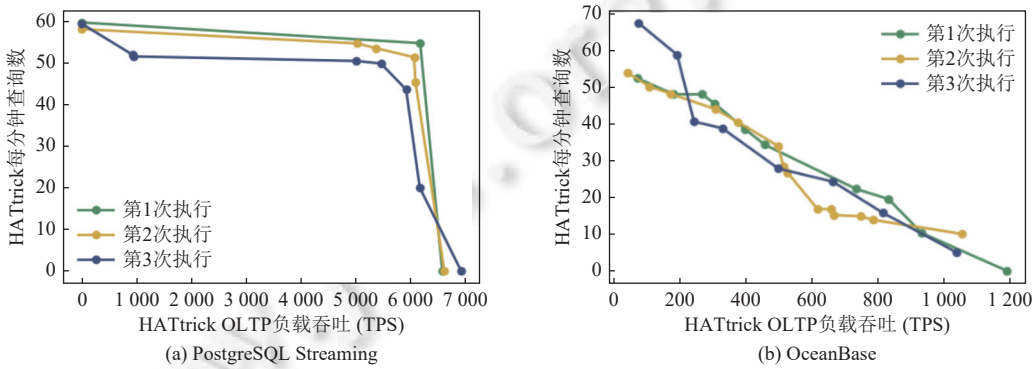

图 12 HATtrick 多次执行的吞吐量边界

5.3 评测负载隔离能力

本节关注现有评测基准是否足以充分评测负载隔离能力. 对于 HTAPBench、Swarm64 和 CH-benCHmark, 实验调整线程数和负载比例, 并根据调整过程中 AP 负载的吞吐变化刻画负载隔离能力. 对于 HATtrick, 实验则通过其吞吐边界判断它对负载隔离能力的刻画能力. 各评测基准的结果如图 13、图 14 所示.

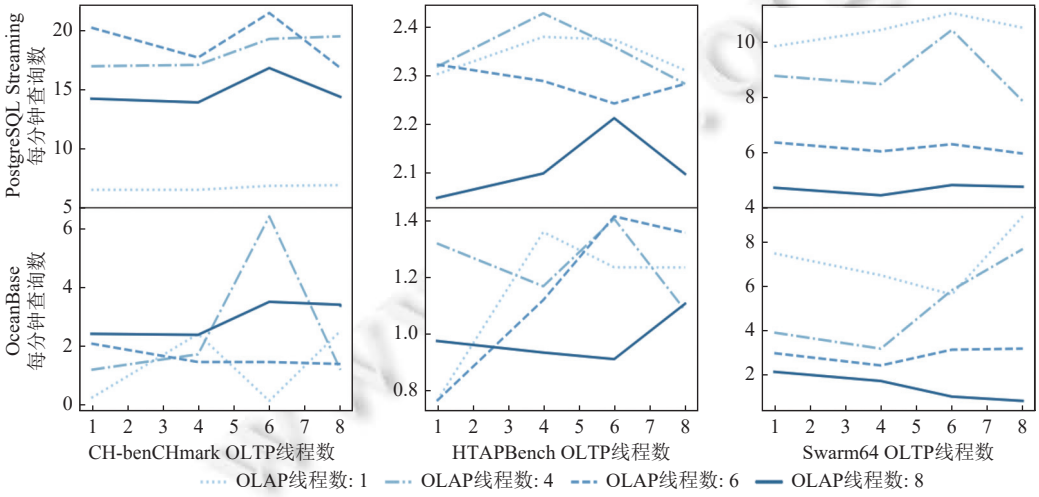

图 13 评测基准在不同线程数下的 OLAP 吞吐

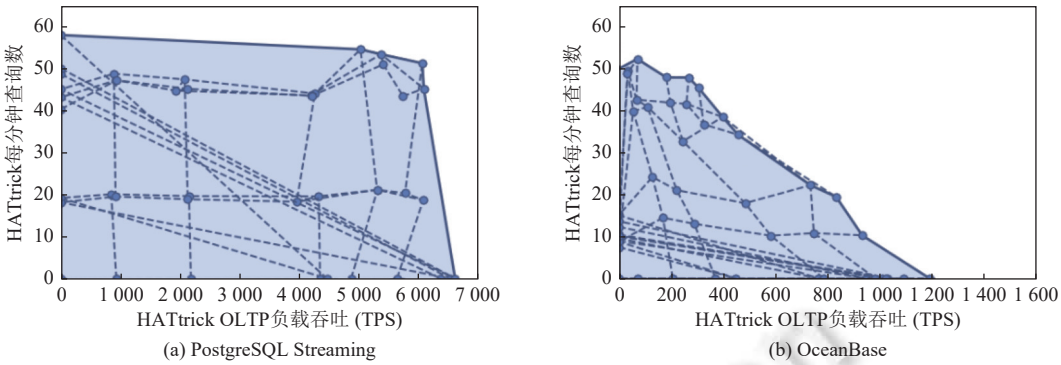

图 14 HAtrick 的吞吐边界波动情况

HTAPBench、CH-benCHmark 和 Swarm64 仅提供了特定线程数下的负载吞吐, 数据库使用者难以直接与实际业务吞吐对应. 在图 13 中, 各评测基准的负载具有不同的压力, 导致 AP 负载吞吐变化趋势存在差异, 无法判断 TP 负载对 AP 查询的影响程度. 而图 14 给出了两个数据库系统的吞吐边界, PostgreSQL Streaming 的吞吐边界呈矩形, 表示 TP/AP 负载的吞吐彼此互不影响, 负载隔离能力强; OceanBase 的吞吐边界倾斜, 负载隔离能力较弱. 相比之下, HATtrick 可以直接从吞吐边界中获取数据库系统的负载隔离能力和两种负载的相互作用关系, 具有更强的评测能力.

5.4 评测新鲜度

本节关注现有评测基准是否足以充分评测实时分析能力. 目前, 仅 HATtrick 提出新鲜度的量化指标, 所以实验主要挖掘 HATtrick 量化新鲜度的细节. 为此, 实验中比较了不同一致性模型的数据库系统 TiDB、PostgreSQL Streaming 和 OceanBase 所测得新鲜度的累积概率分布, 即新鲜度低于给定值 (实验中设置为 0) 的查询出现概率之和. 其中我们分别设置 PostgreSQL Streaming 使用顺序一致性 (记为 PostgreSQL Streaming) 和线性一致性 (记为 PostgreSQL) 的数据同步方式, 如图 15 所示.

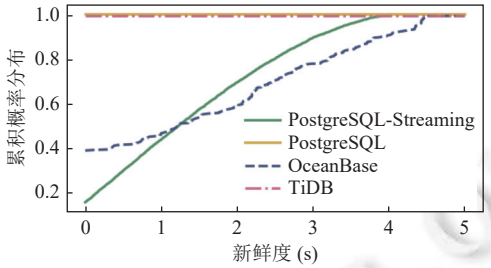

图 15 各数据库所执行查询的新鲜度分布

图 15 中, TiDB 累积概率分布恒为 1, 表示测得的新鲜度均为 0, 这是由于 TiDB 采用线性一致性模型, 其查询可以获取实时修改的数据; 而 PostgreSQL Streaming 采用顺序一致性模型, 查询只能获取落后一定时间的数据, 因而读取的数据存在延迟, 大部分查询对应的新鲜度不为 0. 相比之下, OceanBase 由于可以自适应调整一致性模型, 当负载压力较小时可以获得比 PostgreSQL Streaming 更高的新鲜度, 因此新鲜度小于 1 s 的查询比例更高. HATtrick 的新鲜度可以判断读取数据与 TP 负载更新数据间的延迟, 进而区分不同一致性模型下的数据同步机制以及非线性一致性模型下数据库系统的新鲜度. 虽然基于线性一致性的数据库系统, 如 TiDB 和 PostgreSQL, 测得的新鲜度均为 0, 但数据同步耗时不同, TiDB 由于采用异步的方式同步数据, 当发起 AP 查询时需要等待数据同步, 导致需要比 PostgreSQL 更高的耗时来完成新鲜数据的同步, 不同的数据同步耗时会影响数据库客户端获取新鲜数据的时延, 进而影响客户端吞吐. 但 HATtrick 无法展示同步新数据的耗时, 即同步数据的代价, 因此 HATtrick 的新鲜度指标不能完全反映数据库使用者对数据时效性的度量需求.

6 总结与展望

本文整理和总结了近年来出现的 HTAP 评测基准, 从评测基准的基本要素出发, 结合 HTAP 数据库系统的特征、关键实现技术和评测基准的设计挑战, 提出包含数据生成、负载生成、评价体系设计和一致性模型支持性的评测基准设计维度. 基于设计维度, 本文分析了现有 HTAP 评测基准的实现技术. 同时, 本文提供实验展示主流 HTAP 数据库评测基准的能力, 发现已有基准评测工具在评测结果的可重复性以及负载隔离能力和新鲜度的测量上存在差异, 对于各评测基准设计维度的设计存在欠缺, 如负载的访问方式单一、新鲜度无法完全反映实时分析能力等. 根据上述对现有 HTAP 评测基准的分析和实验, 本文总结并回答 4 个核心问题.

Q1: 现有评测基准是否具有评测基于不同一致性模型的 HTAP 数据库系统的能力?

一致性模型决定了 HTAP 数据库系统的数据同步机制, 对数据获取的时延、时效性造成影响. 目前大部分 HTAP 评测基准仅考虑评测基于线性一致性的 HTAP 数据库系统. HATtrick 可以区分一致性模型并评估非线性一致性模型下数据库系统的新鲜度, 但无法量化数据同步过程中的时间开销, 评测能力有限.

Q2: 现有评测基准面向 HTAP 场景的设计是否完备?

HTAP 评测基准需要从实际应用中提取符合 HTAP 实际业务特征、语义一致的评测场景, 从而更好地刻画混合负载的相互影响. 但目前只有少数评测基准自定义语义一致的表模式, 它们所构建的负载较为简单, 难以深入评测复杂查询的处理能力. 而基于已有表模式扩展的评测基准, 两种负载访问的大部分数据不相交, 不满足语义一致性中混合负载访问相同数据的要求, 对相互影响的刻画能力较弱, 对语义一致的设计仍有欠缺.

Q3: 现有评测基准面向 HTAP 数据库系统的评测是否具有可重复性?

HTAP 评测基准使用混合负载评测数据库系统性能, 其中 TP/AP 负载运行过程中互相干扰, 易导致执行结果难以重复, 影响评测结果的可信度. 大部分评测基准未充分考虑 TP 负载运行对数据量的影响, 对 AP 查询参数的控制存在不足, 导致评测结果不稳定. 其余评测基准以不同力度控制负载参数, 保证了部分评测结果的可重复性.

Q4: 现有评测基准在评测 HTAP 数据库系统核心能力, 如负载隔离能力或实时分析能力上是否充分?

现有评测基准在评测 HTAP 数据库时, 对 HTAP 数据库系统典型的核心能力的评测较弱. 大部分评测基准缺乏对负载隔离和实时分析能力的精确刻画, 评测结果不易比较. HATtrick 通过吞吐边界和新鲜度指标展现数据库系统的负载隔离和实时分析能力, 但仍无法满足评测需求.

基于 HTAP 评测基准的设计维度考虑和对评测基准的总结, 现有工作对语义一致性的测试场景定义这一要求已经有比较明确的共识, 也已经提出了新鲜度指标的度量方案, 但是评测技术在通用性和可重复性上的能力还有所欠缺, 这也是后续工作需要重点关注的维度.

References:

[1] Hybrid transaction/analytical processing will foster opportunities for dramatic business innovation. Gartner. 2022. https://www.gartner.com/en/documents/2657815

[2] What is hybrid transaction/analytical processing (HTAP)? 2014. https://www.zdnet.com/paid-content/article/what-is-hybrid-transactionanalytical-processing-htap/

[3] Lahiri T, Chavan S, Colgan M, Das D, Ganesh A, Gleeson M, Hase S, Holloway A, Kamp J, Lee TH, Loaiza J, Macnaughton N, Marwah V, Mukherjee N, Mullick A, Muthulingam S, Raja V, Roth M, Soylemez E, Zait M. Oracle database in-memory: A dual format in-memory database. In: Proc. of the 31st IEEE Int'l Conf. on Data Engineering. Seoul: IEEE, 2015. 1253–1258. [doi: 10.1109/ICDE.2015.7113373]

[4] Larson PÅ, Birka A, Hanson EN, Huang WY, Nowakiewicz M, Papadimos V. Real-time analytical processing with SQL server. Proc. of the VLDB Endowment, 2015, 8(12): 1740–1751. [doi: 10.14778/2824032.2824071]

[5] Lyu Z, Zhang HH, Xiong G, Guo G, Wang HZ, Chen JB, Praveen A, Yang Y, Gao XM, Wang A, Lin W, Agrawal A, Yang JF, Wu H, Li XL, Guo F, Wu J, Zhang J, Raghavan V. Greenplum: A hybrid database for transactional and analytical workloads. In: Proc. of the 2021 Int'l Conf. on Management of Data. Virtual Event: ACM, 2021. 2530–2542. [doi: 10.1145/3448016.3457562]

[6] Nugroho DPA, Ismail HA. In-memory database and MemSQL. 2019. https://cs.ulb.ac.be/public/_media/teaching/infoh415/student_projects/2019/memsql.pdf

[7] Zhou JY, Xu M, Shraer A, Namasivayam B, Miller A, Tschannen E, Atherton S, Beamon AJ, Sears R, Leach J, Rosenthal D, Dong X, Wilson W, Collins B, Scherer D, Grieser A, Liu YN, Moore A, Muppana B, Su XG, Yadav V. FoundationDB: A distributed unbundled

transactional key value store. In: Proc. of the 2021 Int'l Conf. on Management of Data. ACM, 2021. 2653–2666. [doi: 10.1145/3448016.3457559]

[8] MySQL Heatwave. Real-time Analytics for MySQL Database Service. 2021. https://www.mysql.com/products/mysqlheatwave/

[9] Huang DX, Liu Q, Cui Q, Fang ZH, Ma XY, Xu F, Shen L, Tang L, Zhou YX, Huang ML, Wei W, Liu C, Zhang J, Li JJ, Wu XL, Song LY, Sun RX, Yu SP, Zhao L, Cameron N, Pei LQ, Tang X. TiDB: A raft-based HTAP database. Proc. of the VLDB Endowment, 2020, 13(12): 3072–3084. [doi: 10.14778/3415478.3415535]

[10] Yang JC, Rae I, Xu J, Shute J, Yuan Z, Lau K, Zeng Q, Zhao X, Ma J, Chen ZY, Gao Y, Dong QL, Zhou JX, Wood J, Graefe G, Naughton J, Cieslewicz J. F1 lightning: HTAP as a service. Proc. of the VLDB Endowment, 2020, 13(12): 3313–3325. [doi: 10.14778/3415478.3415553]

[11] Cao W, Liu ZJ, Wang P, Chen S, Zhu CF, Zheng S, Wang YH, Ma GQ. PolarFS: An ultra-low latency and failure resilient distributed file system for shared storage cloud database. Proc. of the VLDB Endowment, 2018, 11(12): 1849–1862. [doi: 10.14778/3229863.3229872]

[12] Verbitski A, Gupta A, Saha D, Corey J, Gupta K, Brahmadesam M, Mittal R, Krishnamurthy S, Maurice S, Kharatishvilli T, Bao XF. Amazon aurora: On avoiding distributed consensus for I/Os, commits, and membership changes. In: Proc. of the 2018 Int'l Conf. on Management of Data. Houston: ACM, 2018. 789–796. [doi: 10.1145/3183713.3196937]

[13] Verbitski A, Gupta A, Saha D, Brahmadesam M, Gupta K, Mittal R, Krishnamurthy S, Maurice S, Kharatishvili T, Bao XF. Amazon aurora: Design considerations for high throughput cloud-native relational databases. In: Proc. of the 2017 ACM Int'l Conf. on Management of Data. Chicago: ACM, 2017. 1041–1052. [doi: 10.1145/3035918.3056101]

[14] Abebe M, Lazu H, Daudjee K. Proteus: Autonomous adaptive storage for mixed workloads. In: Proc. of the 2022 Int'l Conf. on Management of Data. Philadelphia: ACM, 2022. 700–714. [doi: 10.1145/3514221.3517834]

[15] Shen SJ, Chen R, Chen HB, Zang BY. Retrofitting high availability mechanism to tame hybrid transaction/analytical processing. In: Proc. of the 15th USENIX Symp. on Operating Systems Design and Implementation. Virtual Event: USENIX Association, 2021. 219–238.

[16] Sirin U, Dwarkadas S, Ailamaki A. Performance characterization of HTAP workloads. In: Proc. of the 37th Int'l Conf. on Data Engineering (ICDE). Chania: IEEE, 2021. 1829–1834. [doi: 10.1109/ICDE51399.2021.00162]

[17] Özcan F, Tian YY, Tözün P. Hybrid transactional/analytical processing: A survey. In: Proc. of the 2017 ACM Int'l Conf. on Management of Data. Chicago: ACM, 2017. 1771–1775. [doi: 10.1145/3035918.3054784]

[18] Zhang C, Li GL, Feng JH, Zhang JT. Survey of key techniques of HTAP databases. Ruan Jian Xue Bao/Journal of Software, 2023, 34(2): 761–785 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6713.htm [doi: 10.13328/j.cnki.jos.006713]

[19] Hu ZR, Weng SY, Wang QS, Yu R, Xu JK, Zhang R, Zhou X. Data sharing model and optimization strategies in HTAP database systems. Ruan Jian Xue Bao/Journal of Software, 2024, 35(6): 2951–2973 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6901.htm [doi: 10.13328/j.cnki.jos.006901]

[20] Dai W, Berleant D. Benchmarking contemporary deep learning hardware and frameworks: A survey of qualitative metrics. In: Proc. of the 1st Int'l Conf. on Cognitive Machine Intelligence (CogMI). Los Angeles: IEEE, 2019. 148–155. [doi: 10.1109/CogMI48466.2019.00029]

[21] Gray J. Benchmark Handbook: For Database and Transaction Processing Systems. San Francisco: Morgan Kaufmann Publishers Inc., 1992.

[22] TPC-C. https://www.tpc.org/tpcc/

[23] TPC-H. https://www.tpc.org/tpch/

[24] Milkai E, Chronis Y, Gaffney KP, Guo ZH, Patel JM, Yu XY. How good is my HTAP system? In: Proc. of the 2022 Int'l Conf. on Management of Data. Philadelphia: ACM, 2022. 1810–1824. [doi: 10.1145/3514221.3526148]

[25] Jin CQ, Qian WN, Zhou MQ, Zhou AY. Benchmarking data management systems: From traditional database to emergent big data. Chinese Journal of Computers, 2015, 38(1): 18–34 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2015.00018]

[26] Bonnet P, Shasha D. Database benchmarks. In: Liu L and Özsu MT, eds., Encyclopedia of Database Systems. New York: Springer, 2018. 936–938. [doi: 10.1007/978-1-4614-8265-9_80797]

[27] Boissier M, Schlosser R, Uflacker M. Hybrid data layouts for tiered HTAP databases with pareto-optimal data placements. In: Proc. of the 34th Int'l Conf. on Data Engineering (ICDE). Paris: IEEE, 2018. 209–220. [doi: 10.1109/ICDE.2018.00028]

[28] Perera RM, Oetomo B, Rubinstein BIP, Borovica-Gajic R. No DBA? No regret! Multi-armed bandits for index tuning of analytical and HTAP workloads with provable guarantees. arXiv:2108.10130, 2021.

[29] Arulraj J, Pavlo A, Menon P. Bridging the archipelago between row-stores and column-stores for hybrid workloads. In: Proc. of the 2016 Int'l Conf. on Management of Data. San Francisco: ACM, 2016. 583–598. [doi: 10.1145/2882903.2915231]

[30] Makreshanski D, Giceva J, Barthels C, Alonso G. BatchDB: Efficient isolated execution of hybrid OLTP+OLAP workloads for interactive applications. In: Proc. of the 2017 ACM SIGMOD Int'l Conf. on Management of Data. Chicago: ACM, 2017. 37–50. [doi: 10.1145/3035918.3035959]

[31] Kang GX, Wang L, Gao WL, Tang F, Zhan JF. OLxPBench: Real-time, semantically consistent, and domain-specific are essential in benchmarking, designing, and implementing HTAP systems. In: Proc. of the 38th Int'l Conf. on Data Engineering (ICDE). Kuala Lumpur: IEEE, 2022. 1822–1834. [doi: 10.1109/ICDE53745.2022.00182]

[32] Lee R, Zhou MH, Li C, Hu SG, Teng JP, Li DY, Zhang XD. The art of balance: A RateupDB™ experience of building a CPU/GPU hybrid database product. Proc. of the VLDB Endowment, 2021, 14(12): 2999–3013. [doi: 10.14778/3476311.3476378]

[33] Yang ZK, Yang CH, Han FS, Zhuang MQ, Yang B, Yang ZF, Cheng XJ, Zhao YZ, Shi WH, Xi HF, Yu H, Liu B, Pan Y, Yin BX, Chen JQ, Xu QQ. OceanBase: A 707 million tpmC distributed relational database system. Proc. of the VLDB Endowment, 2022, 15(12): 3385–3397. [doi: 10.14778/3554821.3554830]

[34] Yang ZF, Xu QQ, Gao SY, Yang CH, Wang GP, Zhao YZ, Kong FY, Liu H, Wang WH, Xiao JL. OceanBase paetica: A hybrid shared-nothing/shared-everything database for supporting single machine and distributed cluster. Proc. of the VLDB Endowment, 2023, 16(12): 3728–3740. [doi: 10.14778/3611540.3611560]

[35] Bog A, Kruger J, Schaffner J. A composite benchmark for online transaction processing and operational reporting. In: Proc. of the 2008 IEEE Symp. on Advanced Management of Information for Globalized Enterprises (AMIGE). Tianjin: IEEE, 2008. 1–5. [doi: 10.1109/AMIGE.2008.ECP.30]

[36] Cole R, Funke F, Giakoumakis L, Guy W, Kemper A, Krompass S, Kuno H, Nambiar R, Neumann T, Poess M, Sattler KU, Seibold M, Simon E, Waas F. The mixed workload CH-benCHmark. In: Proc. of the 4th Int'l Workshop on Testing Database Systems. Athens: ACM, 2011. 8. [doi: 10.1145/1988842.1988850]

[37] Coelho F, Paulo J, Vilaça R, Pereira J, Oliveira R. HTAPBench: Hybrid transactional and analytical processing benchmark. In: Proc. of the 8th ACM/SPEC on Int'l Conf. on Performance Engineering. L'Aquila: ACM, 2017. 293–304. [doi: 10.1145/3030207.3030228]

[38] Swarm64. https://github.com/swarm64/tpc-toolkit

[39] Raman V, Attaluri G, Barber R, Chainani N, Kalmuk D, KulandaiSamy V, Leenstra J, Lightstone S, Liu SR, Lohman GM, Malkemus T, Mueller R, Pandis I, Schiefer B, Sharpe D, Sidle R, Storm A, Zhang LP. DB2 with BLU acceleration: So much more than just a column store. Proc. of the VLDB Endowment, 2013, 6(11): 1080–1091. [doi: 10.14778/2536222.2536233]

[40] Carbone P, Katsifodimos A, Ewen S, Markl V, Haridi S, Tzoumas K. Apache flink™: Stream and batch processing in a single engine. IEEE Data Engineering Bulletin, 2015, 38(4): 28–38.

[41] Giavaresi G, Fini M, Chiesa R, Giordano C, Sandrini E, Bianchi AE, Ceribelli P, Giardino R. A novel multiphase anodic spark deposition coating for the improvement of orthopedic implant osseointegration: An experimental study in cortical bone of sheep. Journal of Biomedical Materials Research Part A, 2008, 85A(4): 1022–1031. [doi: 10.1002/jbm.a.31566]

[42] Barber R, Raman V, Sidle R, Tian Y, Tözün P. Wildfire: HTAP for big data. In: Zomaya A, Taheri J and Sakr S, eds. Encyclopedia of Big Data Technologies. Cham: Springer, 2019. 1–7. [doi: 10.1007/978-3-319-63962-8_257-1]

[43] Liu WJ, Li JB, Li ZH, Zhang LJ. A massive distributed relational database for financial applications. Journal of Huazhong University of Science and Technology (Natural Science Edition), 2019, 47(2): 121–126 (in Chinese with English abstract). [doi: 10.13245/j.hust.190222]

[44] Raza A, Chrysogelos P, Anadiotis AC, Ailamaki A. Adaptive HTAP through elastic resource scheduling. In: Proc. of the 2020 ACM SIGMOD Int'l Conf. on Management of Data. Portland: ACM, 2020. 2043–2054. [doi: 10.1145/3318464.3389783]

[45] Grund M, Krüger J, Plattner H, Zeier A, Cudre-Mauroux P, Madden S. HYRISE: A main memory hybrid storage engine. Proc. of the VLDB Endowment, 2010, 4(2): 105–116. [doi: 10.14778/1921071.1921077]

[46] Kemper A, Neumann T. HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. In: the 27th Int'l Conf. on Data Engineering. Hannover: IEEE, 2011. 195–206. [doi: 10.1109/ICDE.2011.5767867]

[47] Diederich J, Milton J. New methods and fast algorithms for database normalization. ACM Trans. on Database Systems, 1988, 13(3): 339–365. [doi: 10.1145/44498.44499]

[48] Huang CH, Cahill MJ, Fekete AD, Röhm U. Decongestant: A breath of fresh air for MongoDB through freshness-aware reads. In: Proc. of the 24th Int'l Conf. on Extending Database Technology. 2021. 535–546. [doi: 10.5441/002/edbt.2021.64]

[49] Athanassoulis M, Bøgh KS, Idreos S. Optimal column layout for hybrid workloads. Proc. of the VLDB Endowment, 2019, 12(13): 2393–2407. [doi: 10.14778/3358701.3358707]

[50] Funke F, Kemper A, Krompass S, Kuno H, Nambiar R, Neumann T, Nica A, Poess M, Seibold M. Metrics for measuring the performance of the mixed workload CH-benCHmark. In: Proc. of the Third TPC Technology Conf. on Topics in Performance Evaluation, Measurement and Characterization. Seattle: Springer, 2011. 10–30. [doi: 10.1007/978-3-642-32627-1_2]

[51] Bouzeghoub M. A framework for analysis of data freshness. In: Proc. of the 2004 Int'l Workshop on Information Quality in Information Systems. Paris: ACM, 2004. 59–67. [doi: 10.1145/1012453.1012464]

[52] Chen JJ, Ding YH, Liu Y, Li FS, Zhang L, Zhang MY, Wei K, Cao LX, Zou D, Liu Y, Zhang L, Shi R, Ding W, Wu K, Luo SY, Sun J, Liang YM. ByteHTAP: Bytedance's HTAP system with high data freshness and strong data consistency. Proc. of the VLDB Endowment, 2022, 15(12): 3411–3424. [doi: 10.14778/3554821.3554832]

[53] Mahin MT, Wang BC, Jagtiani K, Carey M, Murthy K. CH3: A mixed workload benchmark for scalable NoSQL. In: Proc. of the 2022 IEEE Int'l Conf. on Big Data (Big Data). Osaka: IEEE, 2022. 3780–3789. [doi: 10.1109/BigData55660.2022.10021092]

[54] Carey M, Lychagin D, Muralikrishna M, Sarathy V, Westmann T. CH2: A hybrid operational/analytical processing benchmark for NoSQL. In: Proc. of the 13th TPC Technology Conf. on Performance Evaluation and Benchmarking. Copenhagen: Springer, 2022. 62–80. [doi: 10.1007/978-3-030-94437-7_5]

附中文参考文献:

[18] 张超, 李国良, 冯建华, 张金涛. HTAP 数据库关键技术综述. 软件学报, 2023, 34(2): 761–785. http://www.jos.org.cn/1000-9825/6713.htm [doi: 10.13328/j.cnki.jos.006713]

[19] 胡梓锐, 翁思扬, 王清帅, 俞融, 徐金凯, 张蓉, 周烜. HTAP 数据库系统数据共享模型和优化策略. 软件学报, 2024, 35(6): 2951–2973. http://www.jos.org.cn/1000-9825/6901.htm [doi: 10.13328/j.cnki.jos.006901]

[25] 金澈清, 钱卫宁, 周敏奇, 周傲英. 数据管理系统评测基准: 从传统数据库到新兴大数据. 计算机学报, 2015, 38(1): 18–34. [doi: 10.3724/SP.J.1016.2015.00018]

[43] 刘文洁, 李戬勃, 李战怀, 张利军. 一种面向金融应用的海量分布式关系数据库. 华中科技大学学报 (自然科学版), 2019, 47(2): 121–126. [doi: 10.13245/j.hust.190222]

翁思扬(2000－), 男, 博士生, CCF 学生会员, 主要研究领域为数据库基准评测, 数据库事务处理, 数据库负载仿真.

俞融(1999－), 女, 硕士生, 主要研究领域为 HTAP 数据库基准评测, 数据库系统.

王清帅(1997－), 男, 博士生, 主要研究领域为面向应用的数据库负载仿真, 新型数据库基准评测.

胡梓锐(2001－), 男, 博士生, CCF 学生会员, 主要研究领域为 HTAP 数据库基准评测, 数据库智能化, 查询基数预估.

倪葳(1991－), 女, 博士, 副教授, CCF 专业会员, 主要研究领域为大数据分析与应用, 数据科学与工程.

张蓉(1978－), 女, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为分布式数据管理, 数据库基准评测, 数据流管理.

周烜(1979－), 男, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为数据库系统, 大数据处理技术.

周傲英(1965－), 男, 博士, 教授, 博士生导师, CCF 会士, 主要研究领域为数据库, 数据管理, 数据驱动的教授教育学, 教育科技、物流科技等基于数据的应用.

徐泉清(1980－), 男, 博士, 正高级工程师, CCF 杰出会员, 主要研究领域为数据库系统, 分布式系统.

杨传辉(1985－), 男, 硕士, CCF 专业会员, 主要研究领域为分布式系统, 数据库系统.

刘维(1986－), 女, 高级工程师, 主要研究领域为质量可靠性, 基础软硬件测评.

杨攀飞(1989－), 男, 硕士, 主要研究领域为软件性能与可靠性测试, 基准工具研发.