浙江大學
ZHEJIANG UNIVERSITY

《大数据存储与处理》
云原生数据库系统综述

姓　名　刘京宗

学　号　22451040

学　院　软件学院

实验日期　2024 年 10 月 11 日

授课教师　贝毅君

云原生数据库系统综述

刘京宗 22451040

摘 要 随着云计算技术的普及，云原生数据库已成为现代企业应用中必不可少的核心组件。本文综述了两种先进的云原生数据库系统，包括 PolarDB-X 和 PolarDB-IMCI，重点探讨了其在多主架构、分布式弹性和混合事务分析处理（HTAP）方面的技术创新。通过分析这些系统的设计原理和实际应用案例，本文揭示了它们在应对大规模并发写操作、跨数据中心事务协调以及弹性扩展方面的优势与挑战。

关键词 云原生数据库，多主架构，分布式弹性，混合事务分析处理，HTAP，跨数据中心事务，一致性

The Review of Cloud-Native Database Systems: Technological Evolution and Innovation Challenges

Abstract With the widespread adoption of cloud computing technologies, cloud-native databases have become an indispensable core component in modern enterprise applications. This paper reviews two advanced cloud-native database systems, PolarDB-X and PolarDB-IMCI, with a focus on their technical innovations in multi-master architecture, distributed elasticity, and Hybrid Transactional/Analytical Processing (HTAP). By analyzing the design principles and real-world use cases of these systems, the paper reveals their advantages and challenges in handling large-scale concurrent write operations, cross-data-center transaction coordination, and elastic scaling.

Keywords Cloud-native database, multi-primary architecture, distributed elasticity, hybrid transactional and analytical processing, HTAP, cross-data center transaction, consistency

一 引言

随着信息技术的快速发展和数据量的爆炸式增长，传统数据库系统面临着处理高并发、大规模数据访问的挑战。同时，随着云计算技术的成熟，企业对数据库的高可用性、弹性和性能提出了更高的要求。在此背景下，云原生数据库系统逐渐崭露头角，成为现代信息技术架构中的重要组成部分。

云原生数据库与传统数据库的显著区别在于其架构的解耦性和弹性。通过将计算和存储资源分离，这类系统能够在应对突发流量时快速扩展计算能力，并提供按需付费的资源管理模式。此外，随着事务型和分析型工作负载日益融合，支持 HTAP 的云原生数据库系统成为企业优化数据处理流程、提升业务决策效率的关键技术之一。

本文综述了当前两种具有代表性的云原生数据库系统，包括 PolarDB-X 和 PolarDB-IMCI，探讨其在架构设计、性能优化和技术创新方面的特点与差异。通过对比分析这些系统在多主架构、跨数据中心事务处理和 HTAP 支持等方面的设计思路，本文旨在揭示当前云原生数据库技术的发展现状，并为未来的研究方向提出建议。

二 PolarDB-X：面向云原生应用的弹性分布式关系数据库[1]

1 背景

随着云计算的广泛应用，企业数据库系统逐步迁移到云端，促使新一代云原生数据库系统的诞生。这些系统面临的三大核心挑战包括：跨数据中心的高可用性和容灾能力：需要支持多数据中心部署，确保在任意单点失效时能够维持系统的运行。资源弹性和可扩展性：云应用的需求可能在短时间内急剧变化，系统需要能够迅速扩展计算和存储资源以适应突发流量。HTAP（Hybrid Transactional and Analytical Processing）工作负载的支持：需要在同一系统中同时高效地处理事务性和分析性工作负载，避免传统分离 OLTP 与 OLAP 系统所带来的数据冗余和复杂性。

PolarDB-X 是在这种需求下提出的一个分布式关系型数据库系统，旨在满足云原生应用对弹性、扩展性和混合工作负载支持的需求。PolarDB-X 基于 PolarDB 的云原生架构，提供了跨数据中心事务支持、快速弹性伸缩以及高效的 HTAP 处理能力。

2 系统架构设计

PolarDB-X 采用了云原生分离计算和存储的架构设计，将系统划分为三层：计算节点（CN）、数据库节点（DN）、存储节点（SN）。计算节点（CN）负责处理分布式事务和查询，包括事务协调、查询优化和执行。数据库节点（DN）负责处理单分片事务和跨数据中心的复制，并与存储节点进行交互。存储节点（SN）通过 PolarFS（PolarDB 的文件系统）持久化存储数据，提供数据存储和复制功能。这种设计允许每一层独立扩展，从而使系统能够应对大规模的读写请求并且保证系统的可扩展性和弹性。

图 1 展示了 PolarDB-X 的系统架构，这种架构设计极大增强了系统的弹性和可扩展性。例如，当负载增加时，计算节点和数据库节点可以分别扩展，而无需移动数据，大幅减少了系统扩展的时间和成本。

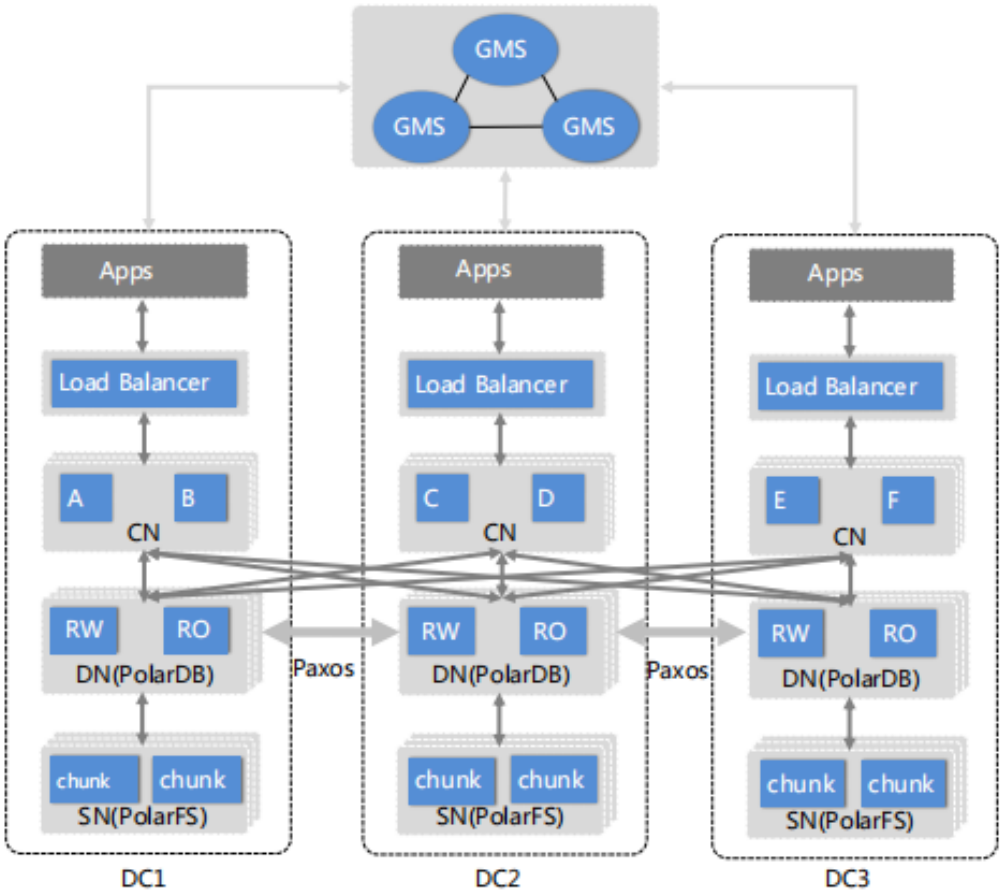


Fig. 1 PolarDB-X 的系统架构示意图

3 关键技术

3.1 HLC-SI 两阶段提交事务机制

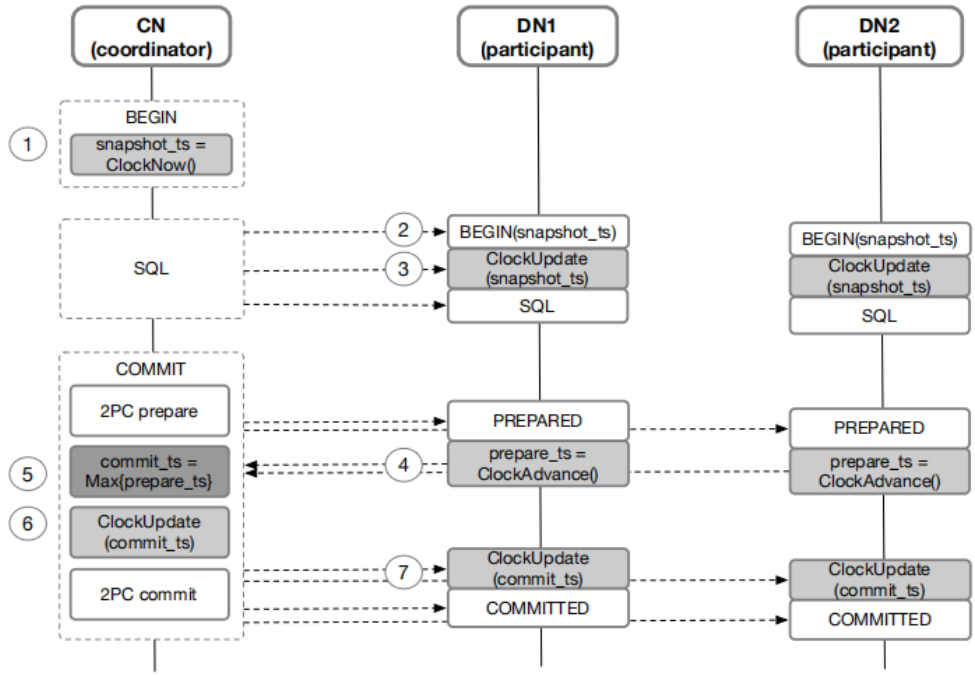


Fig. 2 HLC-SI 和两阶段提交

图 2展示了 HLC-SI（混合逻辑时钟快照隔离）在分布式数据库中的两阶段提交事务机制。HLC-SI 结合物理时钟和逻辑时钟，确保事务的顺序性和一致性，避免了传统时间戳服务的性能瓶颈。

事务开始：当事务开始时，协调者（CN）调用 ClockNow() 获取事务的快照时间戳（snapshot_ts），确定读取数据的版本，并将该时间戳发送给参与的数据库节点（DN）。

时钟同步：参与节点在接收 snapshot_ts 后，调用 ClockUpdate(snapshot_ts) 同步本地时钟，确保所有节点看到一致的数据版本。

预提交阶段：在两阶段提交的第一阶段，参与节点对事务的写集进行验证后，调用 ClockAdvance() 获取预提交时间戳（prepare_ts），并返回给协调者。

生成提交时间戳：协调者根据参与节点返回的 prepare_ts，选择最大值作为全局的提交时间戳（commit_ts），并将该时间戳发送给所有参与节点。

完成提交：参与节点收到 commit_ts 后，调用 ClockUpdate(commit_ts) 更新本地时钟，完成事务提交。

HLC-SI 具有如下优势：去中心化：通过分布式的时钟管理，避免了集中式时间戳服务的瓶颈和单点故障。减少延迟：无需频繁访问中心化的时钟服务，降低了跨数据中心事务的延迟。高可扩展性：每个节点独立管理时钟，同时通过轻量级同步确保全局一致性。

3.2 多租户架构

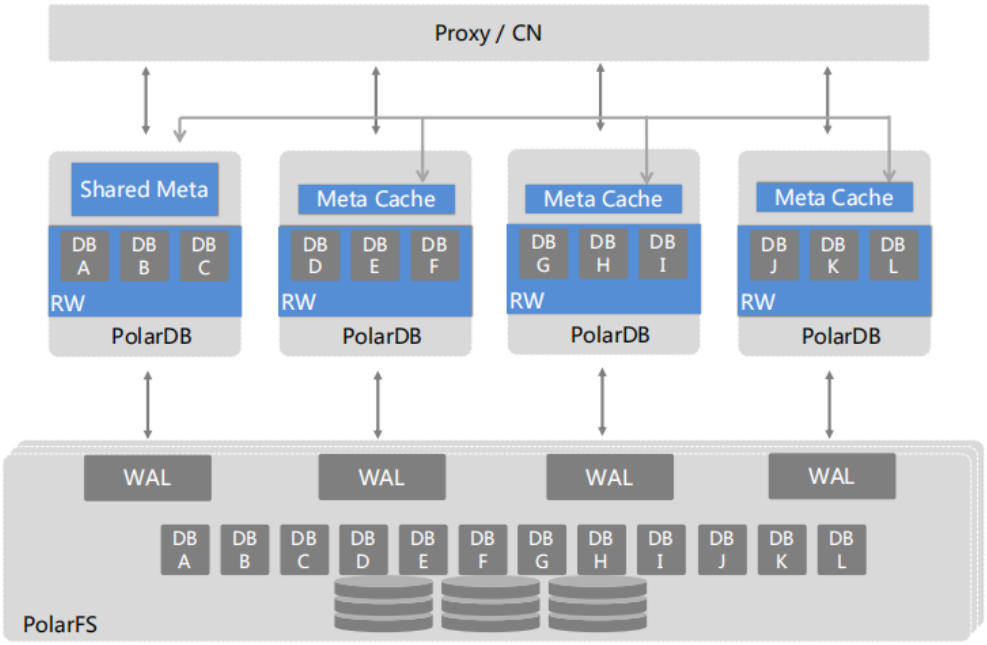


Fig. 3 PolarDB-X 多租户架构

PolarDB-X 将每个租户视为一个独立的逻辑实体，通常对应于一组数据库或表。为了提高资源利用效率，多个租户可以共存在一个数据库实例中，但这些租户在逻辑上是隔离的，避免了跨租户的事务冲突。

图 3展示了在 PolarDB-MT（多租户架构的 PolarDB）中，可以为不同的租户分配多个读写节点 (RW nodes)，以扩展系统的写入能力。这种设计打破了传统数据库单点读写的限制，使得租户之间的写操作不会相互干扰。每个读写节点 (RW node) 可以处理不同租户的数据，且各节点的写操作相互独立。例如，在图 3中，多个 RW 节点分别绑定不同的数据库（DB A-F、DB G-L），因此每个节点都专注于特定租户的数据写入。所有 RW 节点共享底层的存储系统（如 PolarFS），使得不同节点之间的数据保持一致。这种设计提高了存储的弹性和可扩展性。

每个 RW 节点有其私有的重做日志（Redo Log），记录该节点的写操作。这些重做日志是相互独立的，且日志中的修改仅对应各自的租户数据。这一设计避免了写入日志时的冲突，并允许不同 RW 节点并行处理事务。在 RW 节点发生故障时，其他 RW 节点可以通过重做日志来恢复故障节点的事务状态。重做日志可以并行回放，因此即使一个节点失败，系统中的其他节点也能快速接管其事务处理，确保系统的高可用性。

所有 RW 节点共享一个全局数据字典（Global Data Dictionary），这意味着所有节点对于数据库的元数据（如表结构、索引信息等）是一致的。数据字典由一个特定的 RW 节点（主节点）管理，负责对数据字典的修改操作，而其他 RW 节点只保持只读缓存。当需要进行 DDL（数据定义语言）操作时，相关 RW 节点需要获取独占的元数据锁（MDL），阻止其他节点对该表进行修改，确保数据一致性。

为了实现系统的弹性扩展，租户可以在不同的 RW 节点之间迁移。当某个租户的资源需求激增时，可以将该租户迁移到另一个较空闲的 RW 节点，以缓解资源瓶颈。这一过程被称为租户迁移 (Tenant Transfer)。

迁移过程具体为首先，代理服务或计算节点（Proxy/CN）会暂停新事务请求，并等待当前 RW 节点完成所有正在进行的事务。然后，源 RW 节点会将租户的脏页刷新到共享存储，并更新系统表中的租户绑定信息。最后，目标 RW 节点接管租户的数据处理。整个迁移过程对应用程序透明，用户不会察觉到数据库实例的改变，仅会经历短暂的阻塞。

多租户架构使系统具有如下优势：

1. 高弹性：通过支持多个 RW 节点和租户动态迁移，PolarDB-X 可以快速扩展写入能力，确保系统能够应对突发的流量增长。例如，当某个租户的访问量突然增加时，可以将其迁移到一个新的 RW 节点来平衡负载。

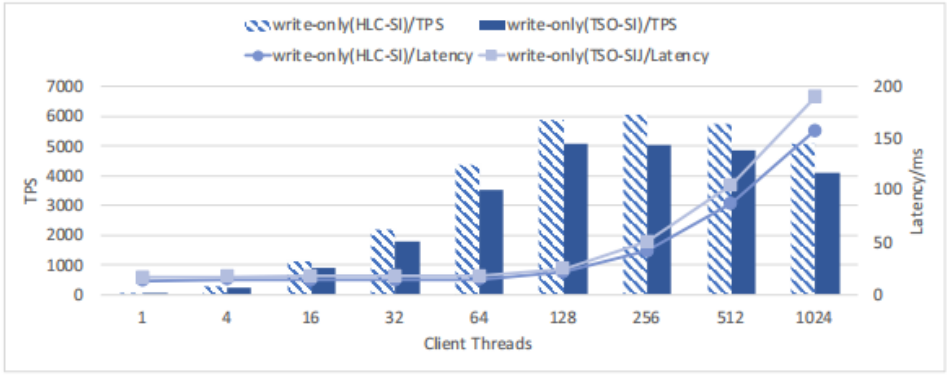
2. 写操作并行处理：由于每个租户都绑定到独立的 RW 节点，多个 RW 节点可以并行处理不同租户的写操作，从而提高系统的整体吞吐量。

3. 快速恢复和高可用性：通过共享的重做日志和存储系统，任何 RW 节点的故障都能被其他节点快速恢复，从而保持系统的高可用性和数据一致性。

4. 租户隔离：每个租户的数据完全隔离，不会发生跨租户的事务冲突，确保数据的安全性和一致性。

4 系统评估

4.1 跨数据中心事务性能评估



(a) Sysbench Write-Only Transactions

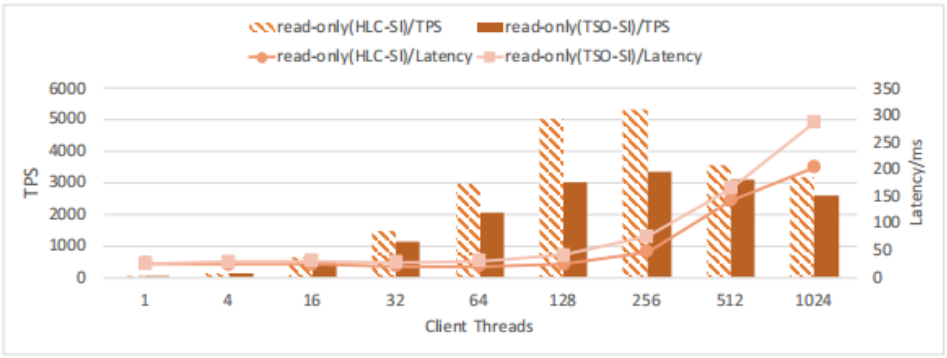


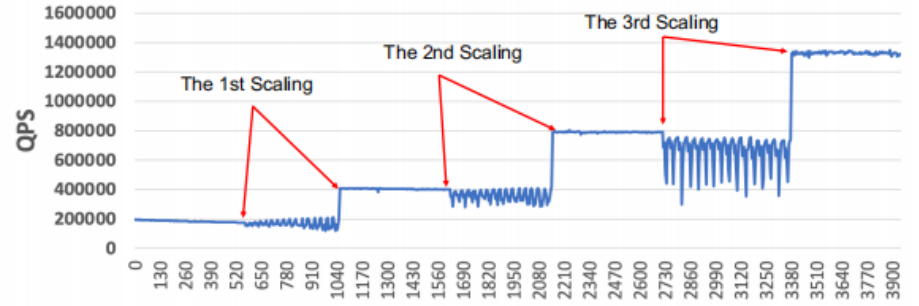
Fig. 4 跨数据中心事务性能评估

该实验比较了 HLC-SI 和传统 TSO-SI 在跨数据中心（DC）环境中的性能表现。实验使用 Sysbench 的读写事务负载，通过部署在三个数据中心的 PolarDB-X 集群，测量读写事务的吞吐量和延迟。

图 4(a) 展示了写事务中 HLC-SI 和 TSO-SI 的吞吐量和延迟对比。HLC-SI 的事务延迟更低，且在高并发情况下，吞吐量提升了 19%。图 4(b) 展示了读事务的对比，HLC-SI 同样表现出更好的事务延迟，随着并发线程的增加，吞吐量表现明显优于 TSO-SI。HLC-SI 在跨数据中心的分布式环境下，通过减少时钟同步带来的延迟，显著提高了系统的事务性能。

4.2 弹性扩展能力测试

(a) Scale Using Fast Tenant Migration Mechanism Of PolarDB-MT



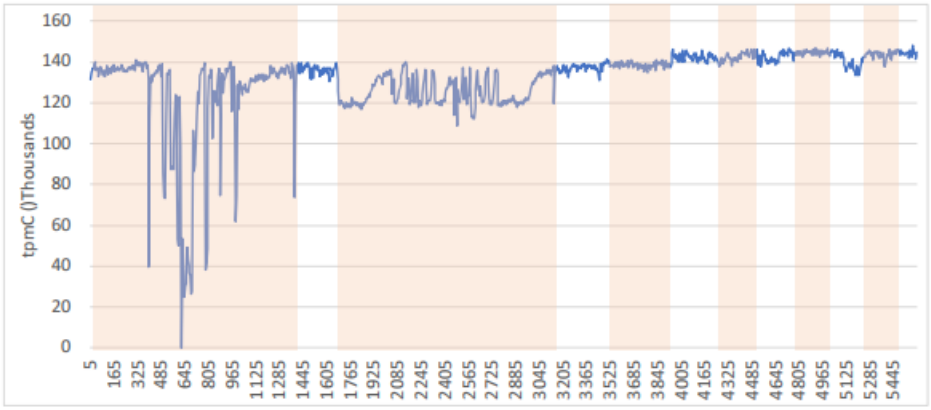
(b) Scale By Copying Data Between Source And Target

Fig. 5 弹性扩展能力测试

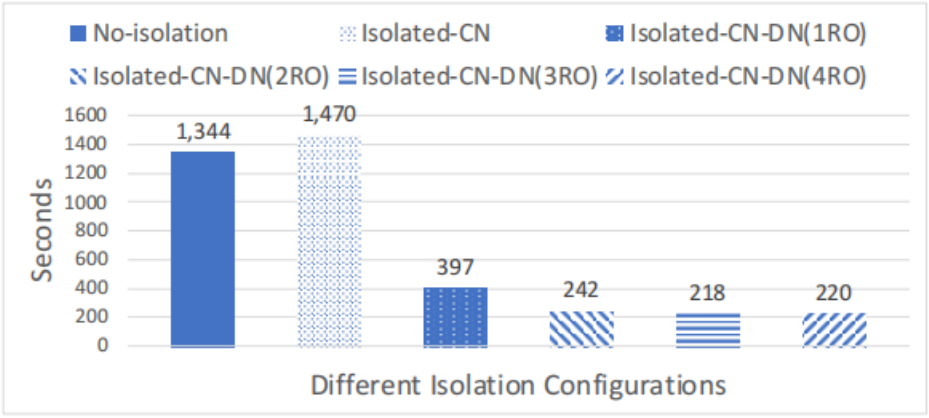
通过对 PolarDB-X 集群的动态扩展能力进行测试，实验比较了两种扩展方法的性能：快速租户迁移和传统数据传输方法。扩展期间运行 Sysbench 的读写事务负载，并测量扩展过程的耗时和吞吐量提升。

图 5(a) 显示，使用快速租户迁移机制时，三次扩展的时间分别为 4.2 秒、4.5 秒和 4.6 秒，吞吐量提升了 113%、94% 和 68%。图 5(b) 表明，使用传统的数据传输方法扩展集群耗时更长，每次扩展耗时分别为 489 秒、527 秒和 660 秒，比快速租户迁移慢了 116-143 倍。快速租户迁移机制大幅缩短了扩展时间，在应对流量激增的场景下具有明显优势。

4.3 HTAP 混合负载测试



(a) Performance variation of TPC-C while TPC-H runs six times



(b) Latency of each run of TPC-H

Fig. 6 HTAP 混合负载测试

混合工作负载（OLTP 和 OLAP），同时运行 TPC-C（事务处理）和 TPC-H（分析查询）负载，评估 PolarDB-X 在 HTAP 模式下的表现。重点测试了资源隔离、只读节点（RO Node）的使用对 OLTP 和 OLAP 的影响。

图 6(a) 显示，在没有资源隔离的情况下，TPC-C 的事务处理性能有显著抖动（最低跌至 57K tpmC），OLTP 任务受到 OLAP 的强烈干扰。当开启资源隔离后，OLTP 性能明显稳定，抖动减少。图 6(b) 显示，通过增加 RO 节点处理 OLAP 查询，TPC-H 的查询延迟逐渐下降。例如，使用两个 RO 节点时，OLAP 查询延迟下降了 39%，使用三个 RO 节点时进一步下降了 10%。资源隔离和通过增加 RO 节点处理 OLAP 查询，有效减轻了 OLTP 工作负载的干扰，提高了混合负载环境下系统的整体性能。

4.4 MPP 和内存列存索引测试

5 小结

PolarDB-X 在架构设计上延续了共享存储数据库系统（如 Amazon Aurora）的思路，但通过引入三层架构和多租户设计，解决了 Aurora 等系统在跨数据中心事务和弹性扩展方面的局限性。

相比 TiDB 等分布式数据库系统，PolarDB-X 采用去中心化的时间戳服务（HLC-SI），有效避免了 TSO

Fig. 7 MPP 和内存列存索引测试

的瓶颈问题。同时，PolarDB-X 在 HTAP 支持上通过引入 MPP 执行引擎和内存列存储索引，显著提升了混合工作负载的性能，而其他系统（如 Spanner）则未充分优化这一点。

未来，PolarDB-X 可以进一步优化租户迁移的技术，特别是进一步减少事务暂停时间。此外，可以借鉴其他数据库系统（如 Zephyr 和 Albatross）的在线迁移技术，提升数据库在高负载下的可用性和性能表现。

三 PolarDB-IMCI：阿里巴巴的云原生 HTAP 数据库[2]

1 背景

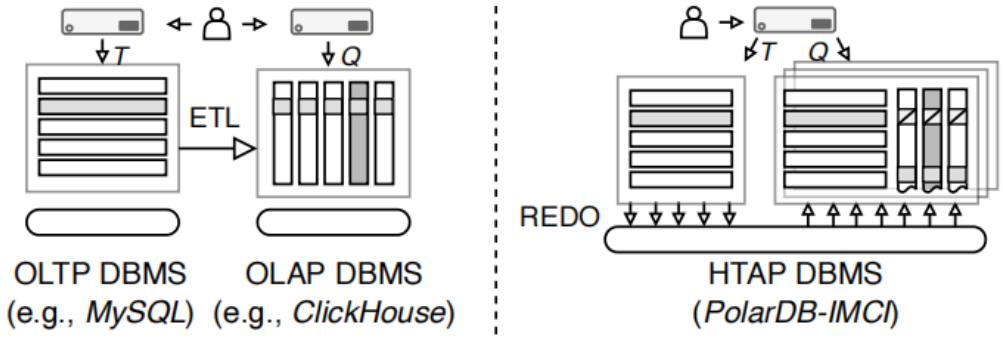


Fig. 8 ETL 与 PolarDB-IMCI 的比较

随着云原生数据库的兴起，企业逐渐向云迁移以获取更高的资源弹性和成本效益。然而，传统的 OLTP（在线事务处理）和 OLAP（在线分析处理）数据库通常被独立部署，各自专注于事务处理或分析处理。这样做的一个显著问题是需要额外的数据同步（如 ETL 流程，图 6 所示），不仅导致数据新鲜度降低，还增加了复杂性和成本。文章提到，在阿里巴巴的实际客户中，近 30% 的 PolarDB 用户需要将数据同步到独立的数据仓库进行分析。这一现象反映了市场上对 HTAP（混合事务和分析处理）数据库的强烈需求。

图 6 中显示了传统方案如何通过 ETL 将 OLTP 数据同步至 OLAP 数据库，以及 PolarDB-IMCI 如何通过集成的 HTAP 架构实现数据同步的消除。这解决了许多企业面临的复杂同步问题，为构建统一的数据处理平台提供了强大支持。

2 系统架构

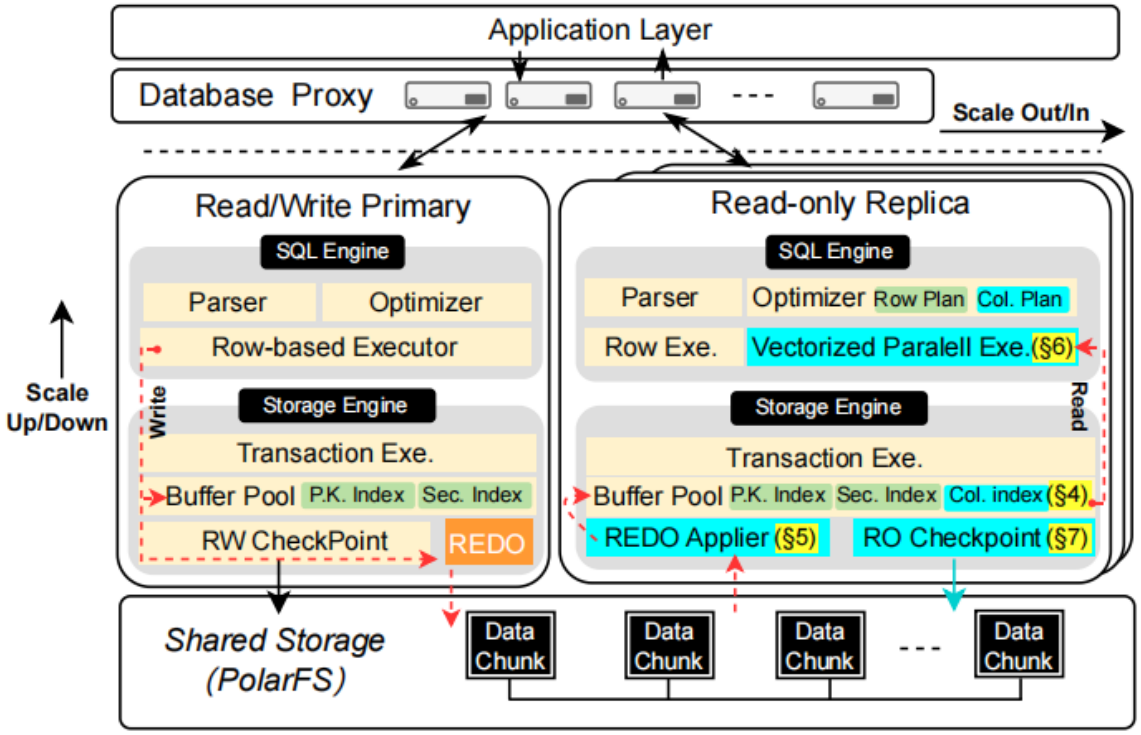


Fig. 9 PolarDB-IMCI 的云原生架构

PolarDB-IMCI 的系统架构。该系统采用了典型的计算与存储分离设计，这种设计提供了高度的资源弹性，允许计算节点根据负载变化快速扩展或缩减，而无需移动数据。图 9 展示了 PolarDB-IMCI 的云原生架构，包括计算层的主节点（处理读写请求）和多个只读副本节点（处理分析查询），以及底层的 PolarFS 共享存储系统。

在这个架构下，读写节点（RW）处理 OLTP 事务，而只读节点（RO）主要负责处理 OLAP 查询。这种设计通过将分析查询负载转移至 RO 节点，确保了 OLTP 查询的响应速度不会受到影响。同时，RO 节点上的列存储设计进一步提高了 OLAP 查询性能，存储和查询效率得到显著提升。

3 关键技术

3.1 列索引存储与优化

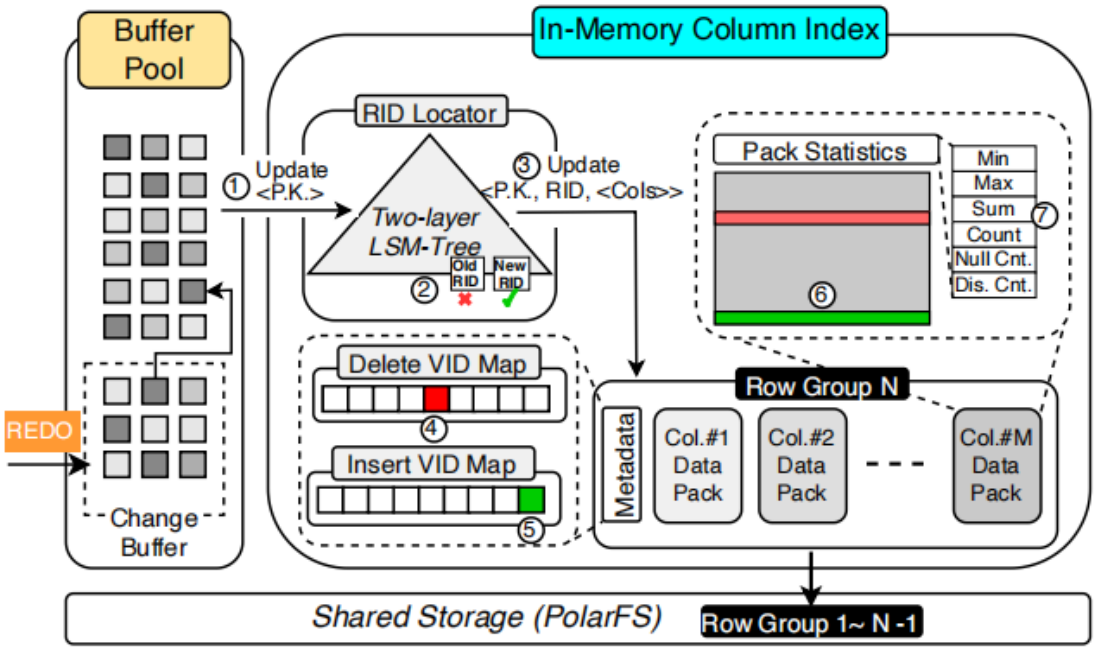


Fig. 10 IMCI 存储的数据更新流程

图 10展示了 PolarDB-IMCI 如何在 IMCI（内存列索引）存储中进行数据更新。该图通过编号 1 到 7 的步骤，说明了数据更新的全过程：

插入数据：首先，系统会为插入的行分配一个空的 RID（行 ID）。更新定位器：接着，利用主键将新分配的 RID 更新到定位器中（即两层 LSM 树中）。写入数据：系统将数据写入列数据包（Data Pack）中的空槽。记录版本 ID（VID）：插入的 VID 记录了该事务提交的时间戳。删除操作：删除时通过定位器找到记录的物理位置，并设置相应的删除 VID。更新操作：更新是通过先删除再插入实现的，即新版本的记录会被追加到部分包中，而旧版本将被逻辑删除。压缩与重排：数据包在达到最大容量后会被压缩，而删除后的“空洞”会被系统定期重排以提升扫描性能。

3.2 更新传播机制与性能优化

更新传播是 HTAP 系统中非常重要的功能，它影响到 OLAP 查询的数据新鲜度。PolarDB-IMCI 通过提前提交日志发送（CALS）和两阶段无冲突并行重放（2P-COFFER）机制，解决了传统 HTAP 系统中更新延迟高和对 OLTP 负载影响大的问题。

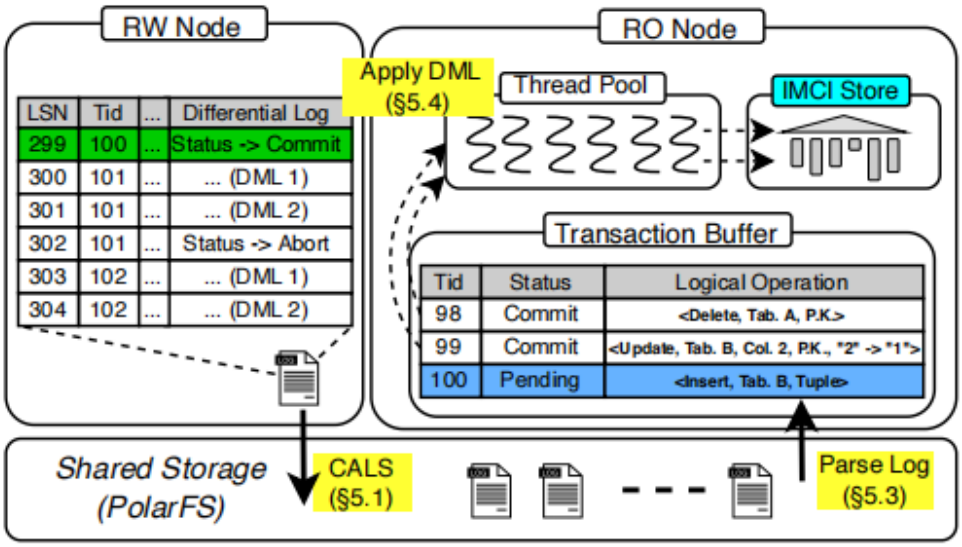


Fig. 11 REDO 日志传输的概览

图 11 展示了 PolarDB-IMCI 如何通过 REDO 日志进行更新传播。其关键步骤如下：

写入日志：在主节点（RW 节点）执行事务时，每个日志条目都会被分配一个日志序列号（LSN），这些日志条目包括 DML 操作（如插入、删除、更新）和事务的提交或回滚信息。广播 LSN：主节点将最新的 LSN 广播给只读节点（RO 节点）。RO 节点通过读取这些日志来执行相应的操作。CALS 机制：PolarDB-IMCI 采用了提前日志传输机制（CALS），可以在事务提交之前就将日志传输给 RO 节点，从而减少数据更新的延迟。事务缓冲区：DML 操作被解析为逻辑操作后，存储在事务缓冲区中，并等待 RO 节点进一步处理。

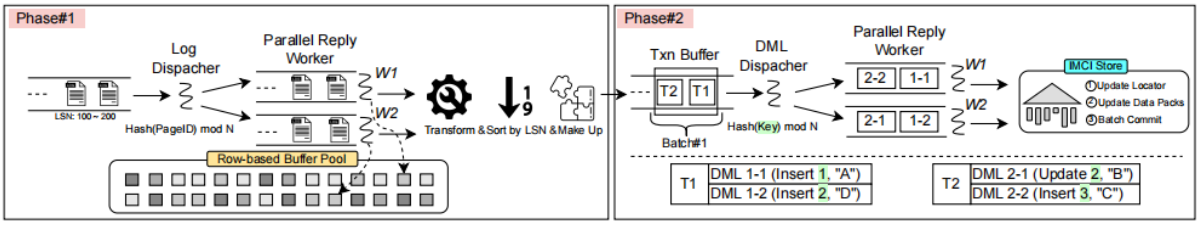


Fig. 12 PolarDB-IMCI 的两阶段无冲突并行重放（2P-COFFER）机制

2P-COFFER 机制（图 12）将日志解析和 DML 应用分为两个阶段：首先将 REDO 日志解析为物理操作，再将其转换为逻辑 DML 语句。这种机制允许多线程并行处理不同的数据页和行，实现了无冲突的并行重放，显著提高了数据同步效率。

3.3 分析查询处理与优化

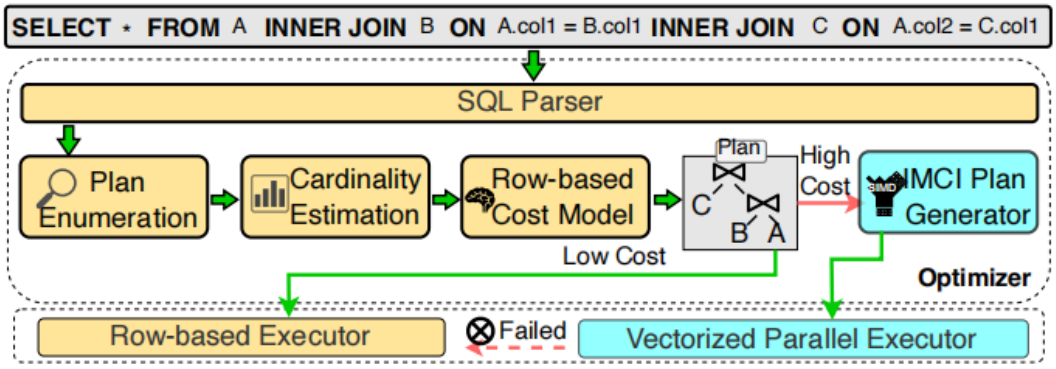


Fig. 13 PolarDB-IMCI 优化器的工作流程

图 8 展示了 PolarDB-IMCI 系统中优化器的工作流程，具体描述了查询如何在系统中进行路由和执行引擎的选择。PolarDB-IMCI 的优化器采用了双层策略，分别是在节点之间和节点内部进行路由。

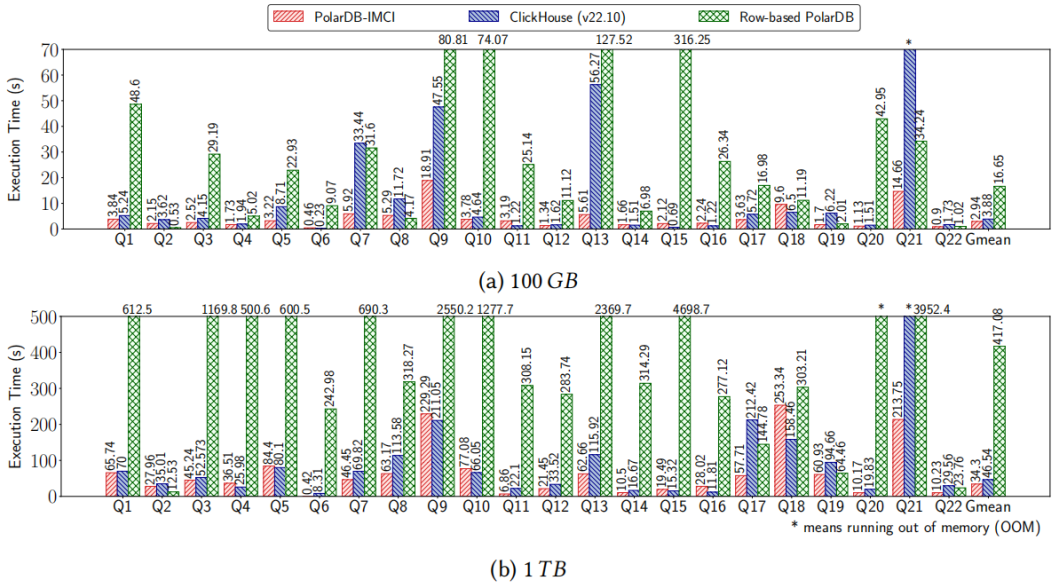
PolarDB-IMCI 的代理层提供了统一的 SQL 接口，所有应用请求（包括事务性和分析性查询）都可以通过该接口提交。代理层根据请求的类型，将读写请求（如事务处理）路由到读写（RW）节点，而只读查询（如分析查询）则路由到只读（RO）节点。对于多个 RO 节点，代理层会根据当前会话数量平衡流量。

如图 11 所示，PolarDB-IMCI 在每个 RO 节点内部实现了两个执行引擎：一个用于点查询的行存储执行引擎，另一个用于分析查询的列存储执行引擎。PolarDB-IMCI 的优化器通过基于行存储的成本估计来选择合适的执行引擎。优化器首先生成一个行存储执行计划（通常成本较低）。如果行存储计划的成本超出某个阈值（即高成本），则生成列存储计划，并在列存储引擎上执行。这种优化策略可以充分利用 PolarDB-IMCI 的行列混合存储架构，确保每个查询都选择最合适的执行引擎，从而在性能和资源利用之间取得平衡。

IMCI 计划生成：与直接构建列存储执行计划不同，PolarDB-IMCI 通过从行存储执行计划转换生成列存储执行计划。这一转换过程如图 8 所示。通过这种方式，列存储计划可以保留原有查询的所有行为特性，并确保在表达式计算时能充分利用 SIMD（单指令多数据）优化。此外，列存储计划还能够重用行存储计划中的错误检测机制，确保在不同执行引擎间的一致性。图 8 中的整个流程展示了 PolarDB-IMCI 系统中查询优化器如何根据查询的类型和成本动态选择合适的执行引擎，实现高效的查询执行。

4 系统评估

4.1 TPC-H 基准测试评估 OLAP 性能



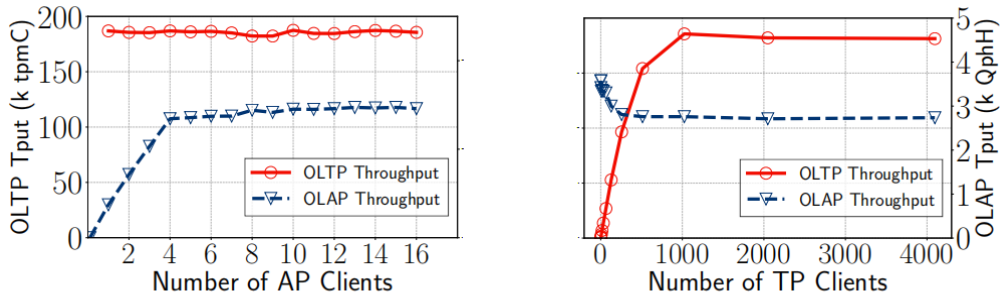
(a) 100 *GB*

(b) 1 *TB*

Fig. 14 Enter Caption

通过 TPC-H 基准测试，评估了 PolarDB-IMCI 在 OLAP（在线分析处理）查询中的性能。图 14展示了 PolarDB-IMCI 与行存储 PolarDB 以及 ClickHouse 的对比。在 100GB 数据集下，PolarDB-IMCI 在几何平均值上比行存储 PolarDB 的性能提升了 5.56 倍，而在 1TB 数据集下，性能提升达到了 12.15 倍。其中，针对扫描密集型查询的加速效果最为显著（例如 Q10 和 Q15）。与 ClickHouse 相比，PolarDB-IMCI 在大多数查询中也表现出色，证明了其在处理大规模 OLAP 工作负载时的优越性。

4.2 混合负载测试



(a) Perf. isolation of OLTP

(b) Perf. isolation of OLAP

Fig. 15 Enter Caption

图 15评估了 PolarDB-IMCI 在混合负载（HTAP）场景下的表现。通过 CH-benCHmark 进行的实验显示，PolarDB-IMCI 可以同时处理高并发的 OLTP（在线事务处理）和 OLAP 查询。在 512 个 OLTP 客户端饱和系统的条件下，OLTP 吞吐量达到 186,890 tpmC（新订单事务/分钟），同时 OLAP 查询也能以

2916 QphH（每小时 TPC-H 查询）运行。此实验验证了 PolarDB-IMCI 在 OLTP 和 OLAP 之间的资源隔离效果，即便在 OLTP 负载增加的情况下，OLAP 性能仅下降了不到 20%。

4.3 更新传播和 OLTP 性能影响

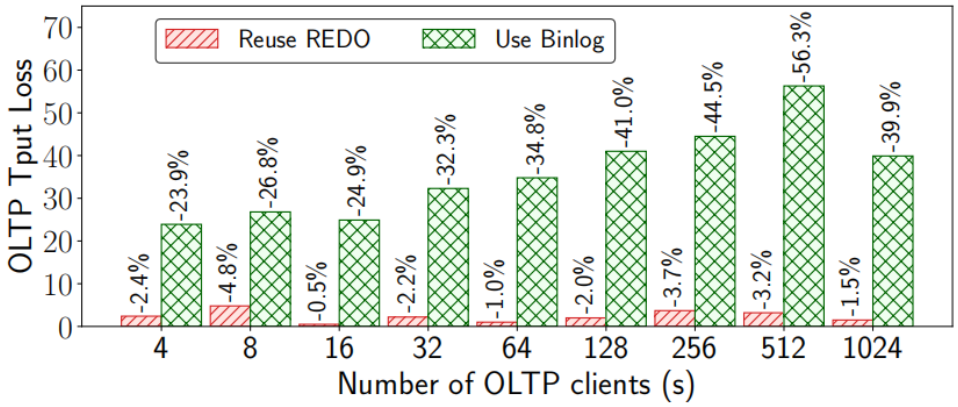


Fig. 16 Enter Caption

图 16展示了 PolarDB-IMCI 在处理 OLTP 负载时对系统性能的影响。实验采用 sysbench 写操作负载进行，结果显示，与使用 Binlog 相比，IMCI 通过重用 REDO 日志进行更新传播，对 OLTP 性能的影响极小。启用 IMCI 的情况下，系统吞吐量仅下降了不到 5%，而使用 Binlog 的系统性能下降则更为明显。

4.4 资源弹性评估

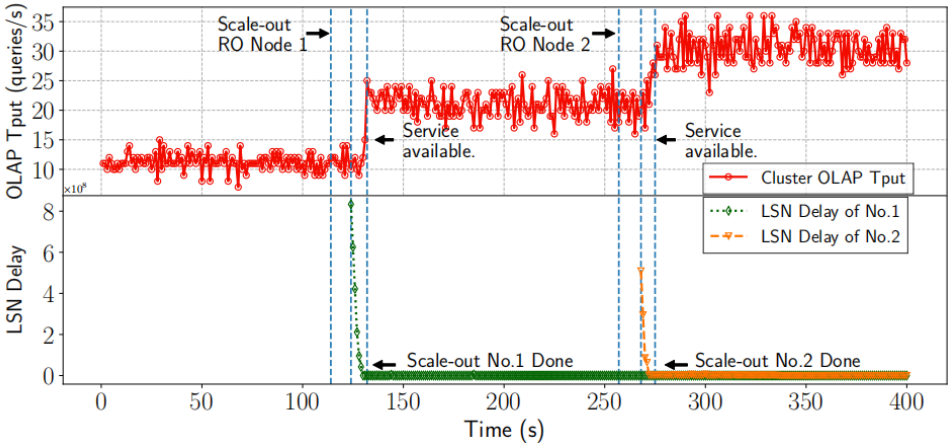


Fig. 17 Enter Caption

为了测试 PolarDB-IMCI 的资源弹性，实验通过 sysbench 插入负载和 TPC-H Q6 查询来评估系统的横向扩展能力。图 17显示，在加入新的只读（RO）节点后，系统在数秒内能够完成节点扩展，并迅速恢复 OLAP 服务。新节点在加入后 9 秒内就能赶上最新的数据更新，展示了 PolarDB-IMCI 快速扩展的能力和弹性。

4.5 生产环境中的性能评估

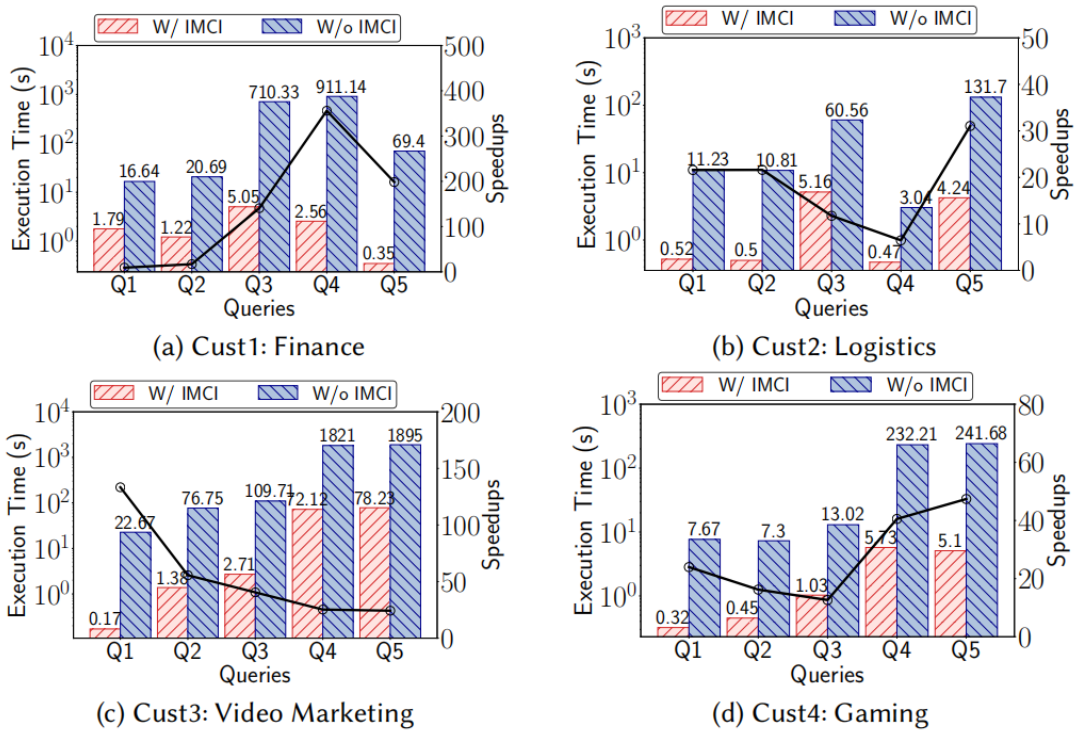


Fig. 18 Enter Caption

在图 18中，实验展示了 PolarDB-IMCI 在金融、物流、视频营销和在线游戏等实际生产环境中的表现。针对复杂的查询，尤其是涉及多表连接的查询，PolarDB-IMCI 显示了显著的加速效果。通过对比行存储 PolarDB，实验结果表明，PolarDB-IMCI 可以为这些复杂查询带来数量级的性能提升。例如，对于物流和游戏行业的查询，IMCI 在某些 SQL 查询上的加速比高达 100 倍以上。

5 小结

PolarDB-IMCI 通过创新的设计，成功实现了云原生 HTAP 系统的多个关键目标。其内存列索引和日志重放机制为 OLAP 查询提供了显著的性能提升，同时对 OLTP 负载影响最小。该系统的资源弹性和数据新鲜度进一步验证了其在实际生产环境中的可行性和优势。

四 总结与展望

文献题目	背景	核心贡献
PolarDB-X: An Elastic Distributed Relational Database for Cloud-Native Applications	PolarDB-X 设计于满足多数据中心部署的需求，同时通过分离计算和存储提升弹性，支持 OLTP 和 OLAP 的混合工作负载	采用三层架构（计算节点、数据库节点、存储节点），支持 HLC-SI（混合逻辑时钟快照隔离）实现跨数据中心事务一致性
PolarDB-IMCI: A Cloud-Native HTAP Database System at Alibaba	PolarDB-IMCI 旨在提供同时支持 OLTP 和 OLAP 的高效解决方案，减少数据同步开销，提升数据新鲜度和资源弹性	通过列存储和透明查询路由机制实现高效的分析性能和低延迟的事务处理。设计了基于存储与计算分离的双格式存储架构，实现了 OLTP 和 OLAP 请求的资源隔离。

Table 1 两篇云原生数据库系统文献的简单对比

云原生数据库系统的发展伴随着云计算技术的进步，正在不断改变数据存储与处理的传统模式。通过对 PolarDB-X 和 PolarDB-IMCI 的综述可以看出，这些系统在应对高并发写操作、跨数据中心事务一致性和混合事务分析处理等关键问题上各有创新和突破。然而，这些系统在实际应用中也面临诸多挑战，如如何进一步降低跨数据中心的网络延迟、提高数据新鲜度，以及优化资源调度和弹性扩展的速度等。

未来的云原生数据库系统需要在架构设计上更加灵活，以适应多样化的业务需求。同时，随着新技术（如 RDMA、分布式存储）的不断发展，这些系统有望在性能和可扩展性上取得更大突破。通过持续的技术创新和实践验证，云原生数据库将在大数据时代发挥更加重要的作用。

参 考 文 献

[1] CAO W, LI F, HUANG G, et al. Polardb-x: An elastic distributed relational database for cloud-native applications [C]//2022 IEEE 38th International Conference on Data Engineering (ICDE). IEEE, 2022: 2859-2872.

[2] WANG J, LI T, SONG H, et al. Polardb-imci: A cloud-native htap database system at alibaba[J]. Proceedings of the ACM on Management of Data, 2023, 1(2): 1-25.