面向HTAP的内存数据库

汇报人：任勇闯

DSC
分布式存储与计算实验室

2022.07.15

目录

01 HTAP概述

01 OLTP&OLAP概述

1.1 OLTP

什么是OLTP

- On-Line Transaction Processing
- 在线事务处理/在线事务交易
- 使得大量的用户通过 Internet 实时执行大量数据库事务
- OLTP最基本的要求：原子性




01 OLTP&OLAP概述

分布式存储与计算实验室

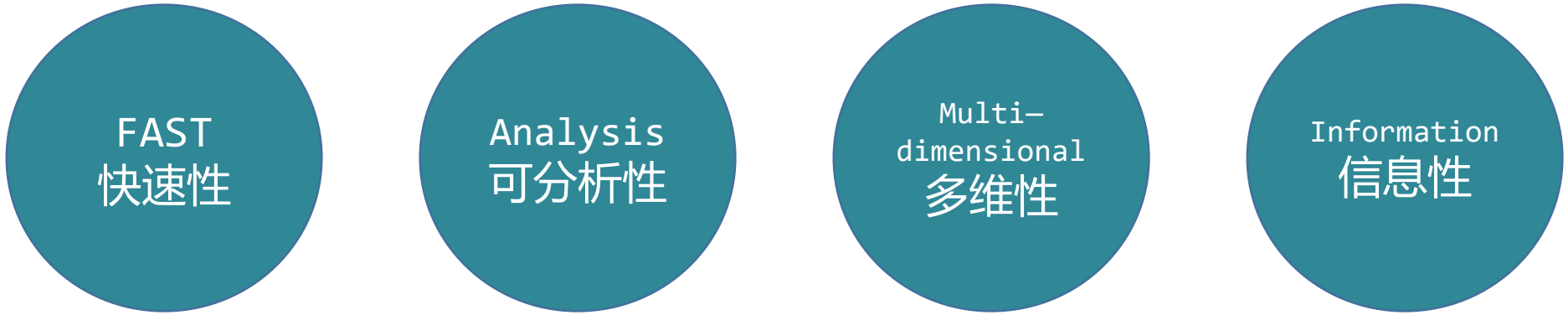
1.1 OLAP

On-Line Analytical Processing

单个数据没有价值，但聚在一起的数据就会蕴含巨大的财富

OLAP（在线分析处理）在 1993 年提出

主要能力就是计算分析决策



01 OLTP&OLAP概述

分布式存储与计算实验室

OLTP技术需求：

1．行存

2．MVCC

3．事务

4．多索引数据结构

5．WAL数据恢复技术

6．Raft数据复制技术

7．预编译存储过程

OLAP技术需求：

1．列存

2．SIMD向量化处理

3．多类型压缩

4．轻量级索引构建

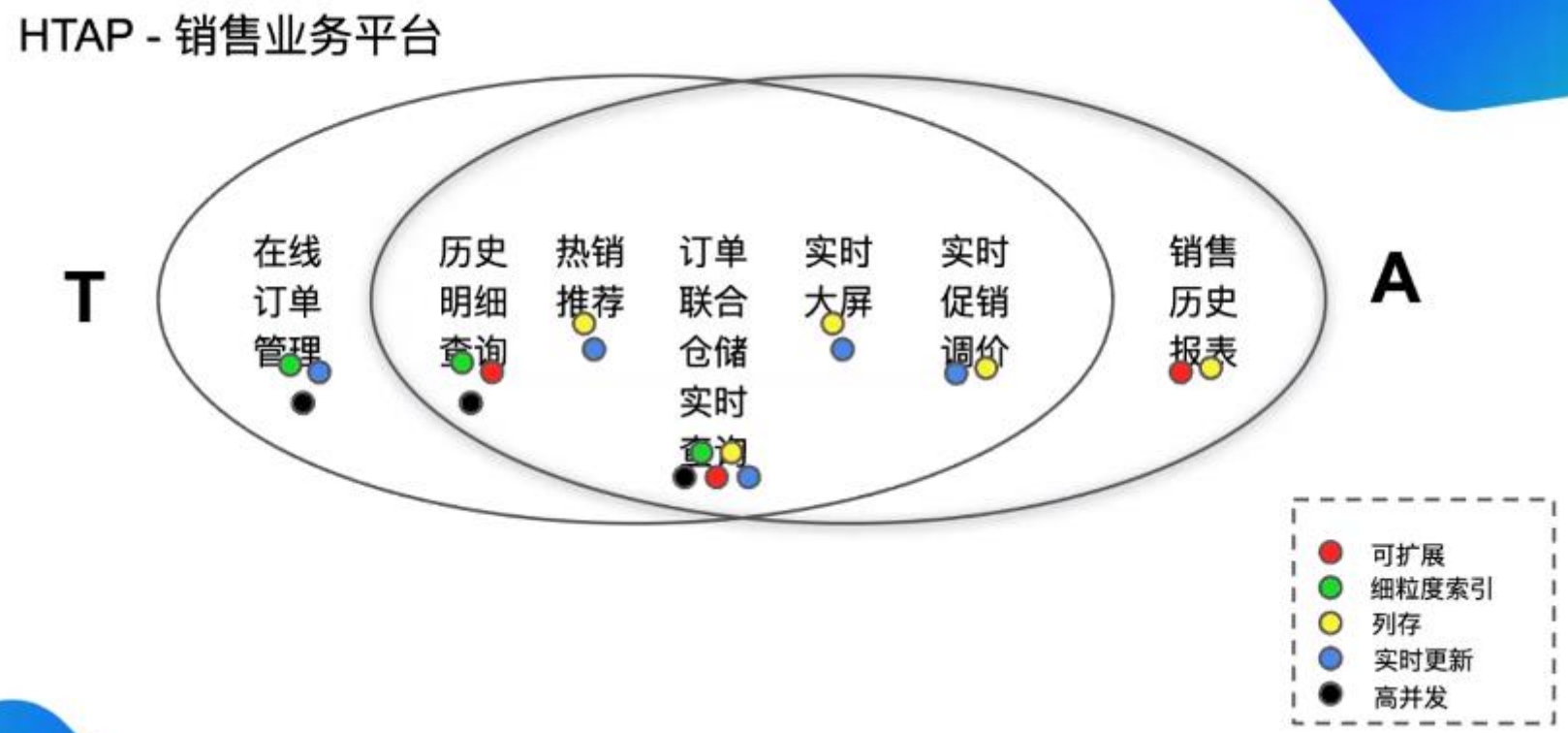
5．MPP大规模并发计算

6．MapReduce批量计算

7．GPU并发计算

01 HTAP概述

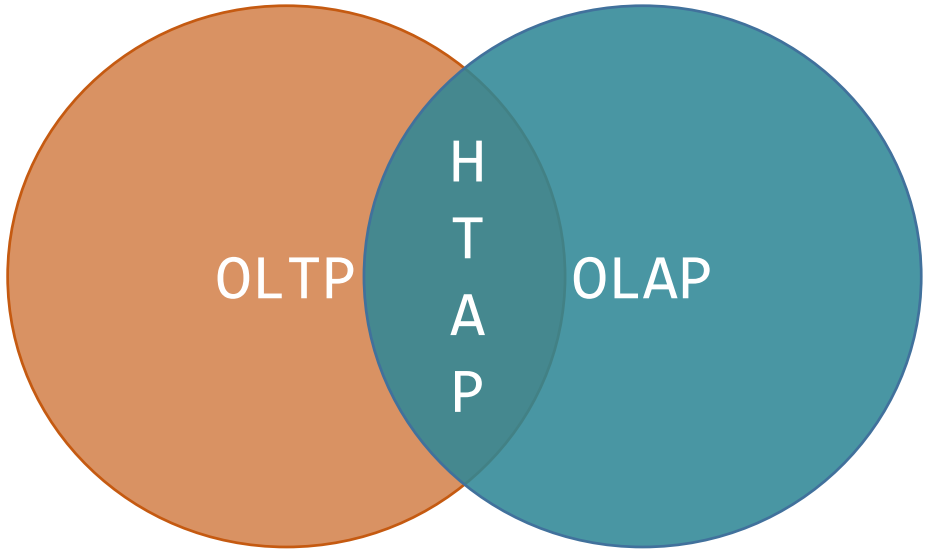
分布式存储与计算实验室



01 HTAP概述

2.1 HTAP

- Hybrid transactional/analytical processing
- 实时分析、事务处理
- 简化架构
- 降低运维成本
- 使用便捷
- 应用场景
  - 实时推荐
  - 物流跟踪
  - 风控



01 HTAP概述

分布式存储与计算实验室

HTAP目标：

1．HTAP在不影响事务吞吐量情况和响应时间下对最新的数据进行分析处理
2．同时分析处理的响应时间不受最新数据新鲜度的影响

HTAP的关键技术：

1．隔离负载：并发OLAP查询不能影响OLTP事务处理的响应时间和吞吐量
    1．内存存储数据资源的隔离
    2．CPU计算单元资源的隔离

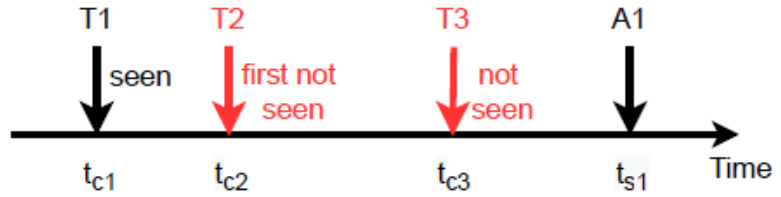


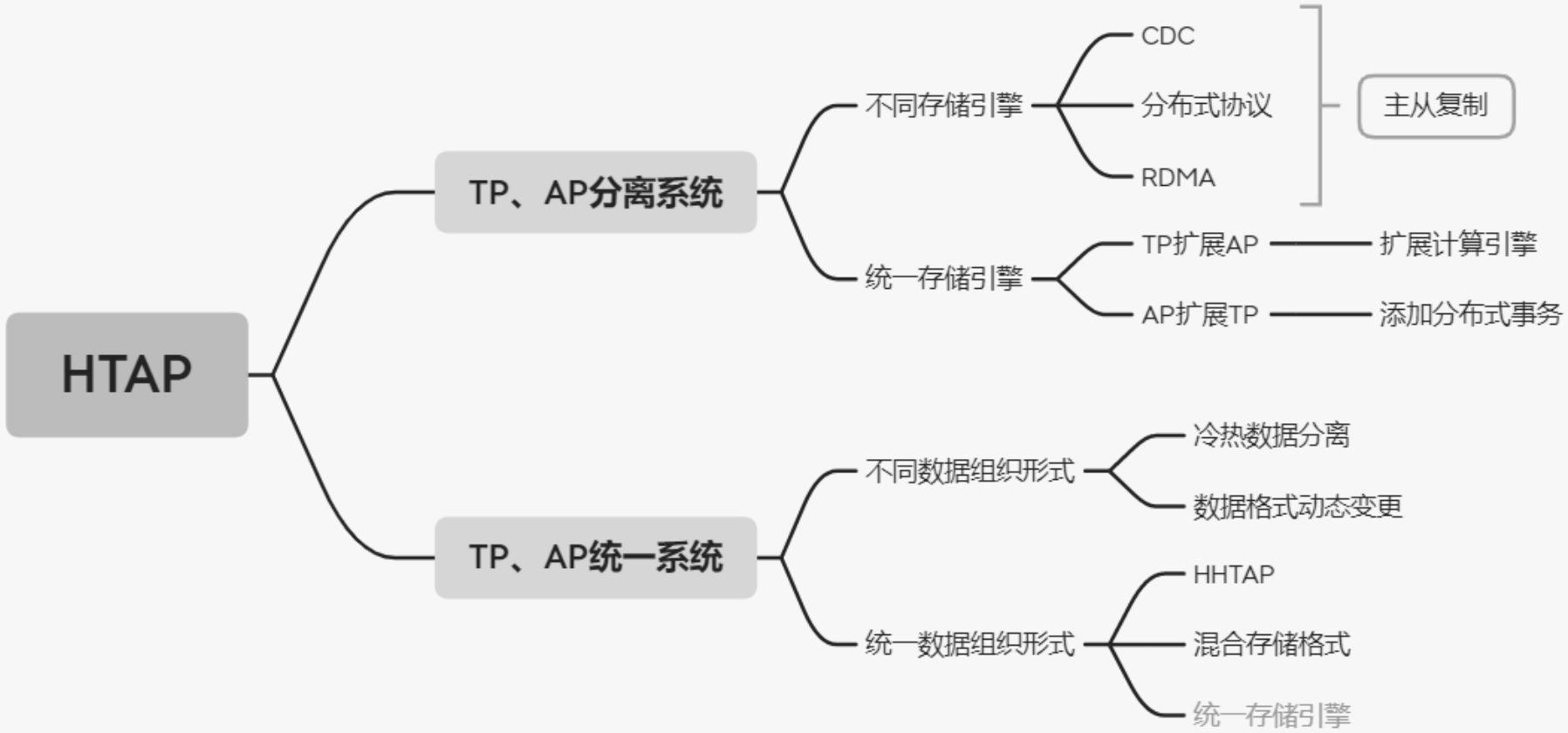
Figure 3: Illustration of freshness for analytical queries.

2．数据新鲜度：OLAP查询应在最新的数据版本上
    1．数据新鲜度率：OLAP中数据与OLTP数据相同的元组数 / OLTP元组总数
    2．数据新鲜时间间隔：ts1 – tc2

3．一致性保证：OLAP查询应确保是正确一致的数据视图

HTAP概述

2.3 htap实现方案总结



2.1 面向HTAP的内存数据库 -共享设计架构

02 共享设计架构

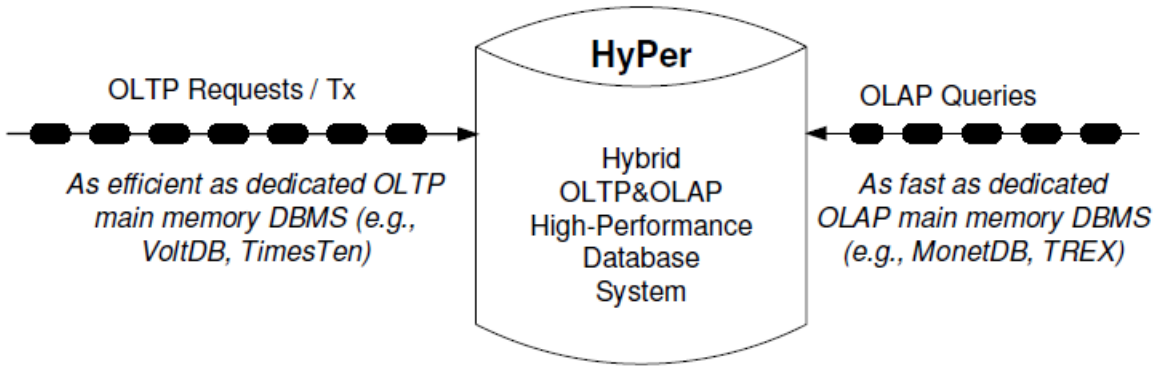


Fig. 1. Hybrid OLTP&OLAP Database Architecture

Hyper：

1．从零开发的主存数据库

2．同一数据库上支持HTAP能力，又不会相互干扰

3．保证TP事务的ACID同时支持OLAP查询

4．串行执行，无需交叉使用CPU

5．充分利用了虚拟内存的管理机制，依赖MMU内存管理单元支持的page-shadowing

6．硬件辅助的复制机制维护事务数据一致性快照

7．通过一致性快照支持OLAP多会话查询

02 共享设计架构

TP设计：

1．所有OLTP事务按序执行（单线程），无需锁机制的开销

2．主存架构吞吐量较大

AP设计：

1．处理AP工作负载不影响TP业务

2．获取在最新的更新数据

3．OS具备为新线程创建虚拟内存快照的能力，Linux中的fork()系统调用创建OLTP进程的子进程

4．为确保事务一致性，fork()应确保在两个TP事务之间执行

5．Fork()子进程可以获取父进程地址空间的副本

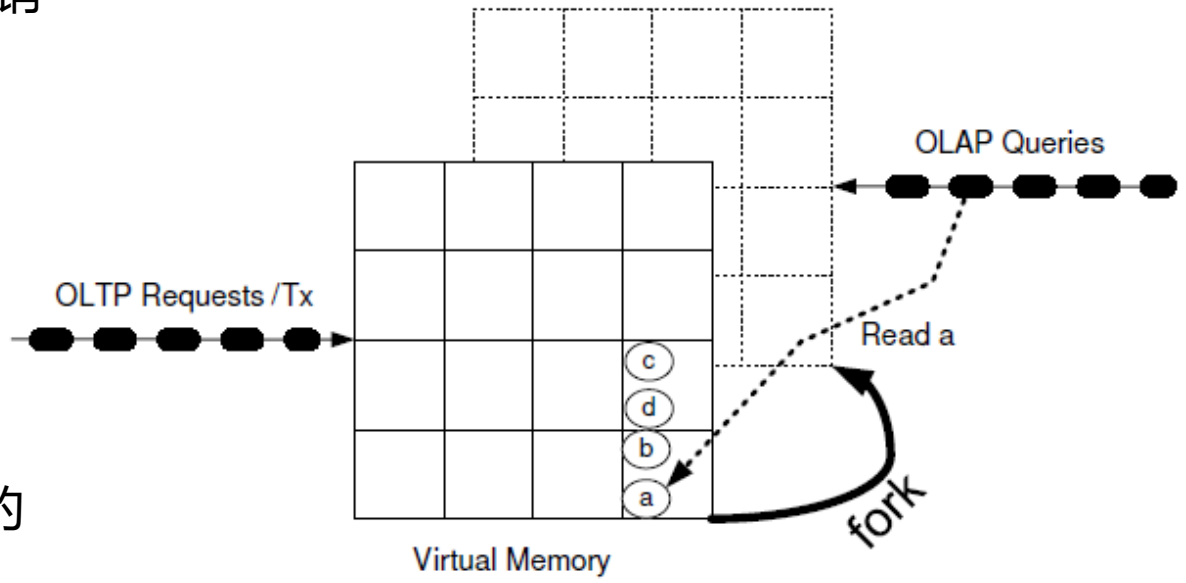


Fig. 2. Forking a New Snapshot

02 共享设计架构

分布式存储与计算实验室

1. Lazy copy-on-update
   1. 父进程与子进程通过虚拟地址指向相同物理内存段
   2. 父进程数据更新后，OS复制对象到子进程虚拟内存页中

2. 当page中页面变更后，OLAP快照指向旧页面。这种更新方式使得支持OLAP多快照

3. OS支持虚拟内存fork可在亚秒级完成，非常高效

4. OLAP不会更改共享page数据

5. 复制的page只会持续到OLAP会话终止，就会删除

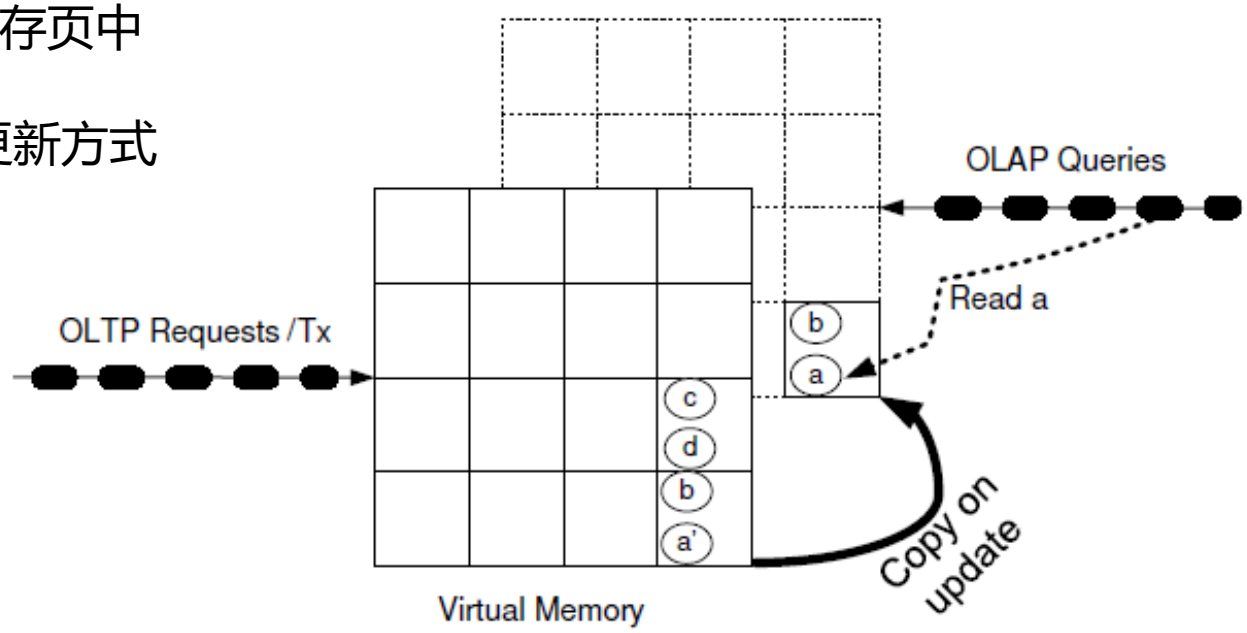


Fig. 3. Copy on Update to Preserve Consistent Snapshot

缺点:

1. page通常4KB，及时其中一个数据该更改也要复制整个page

2. 快照产生的存储开销与父进程（OLTP）请求更新page数量成正比

02 共享设计架构

分布式存储与计算实验室

1．OLAP是只读的，很容易在共享相同地址空间中多线程并行执行，可以显著提高查询速度

2．图中，对数据项连续的更改，大多数数据不会发生变化，但为最新的查询创建快照，可以及时处理实时数据

3．线程快照数可能超过CPU核数，但快照会在OLAP查询后及时删除，且无需按照创建顺序删除快照

4．OS通过物理页的引用计数可以自动检测对应的共享快照

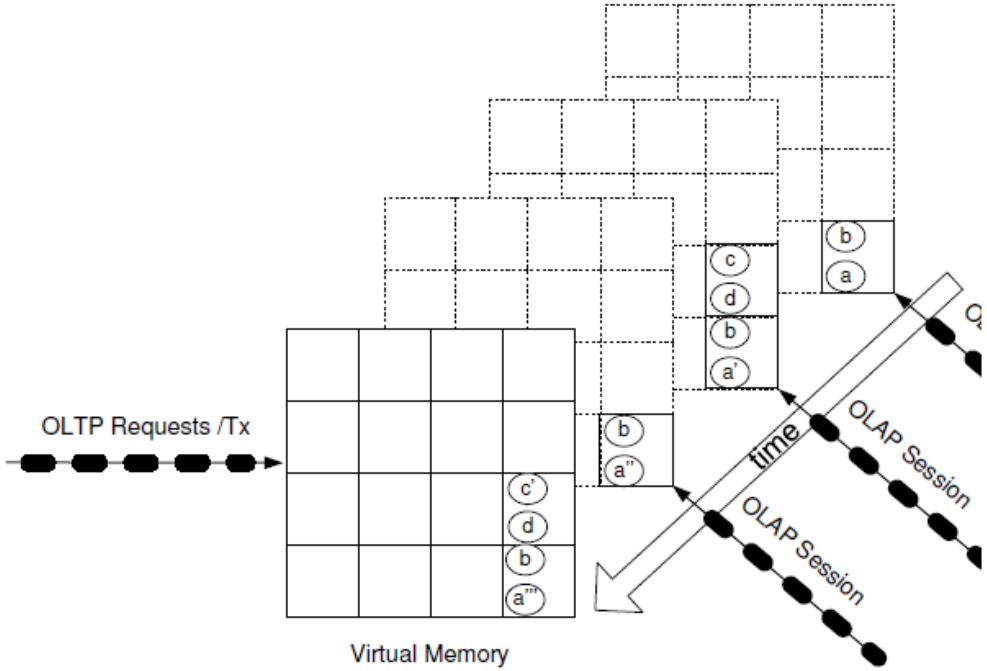


Fig. 4. Multiple OLAP Sessions at Different Points in Time

02 共享设计架构

关键问题解决方法：

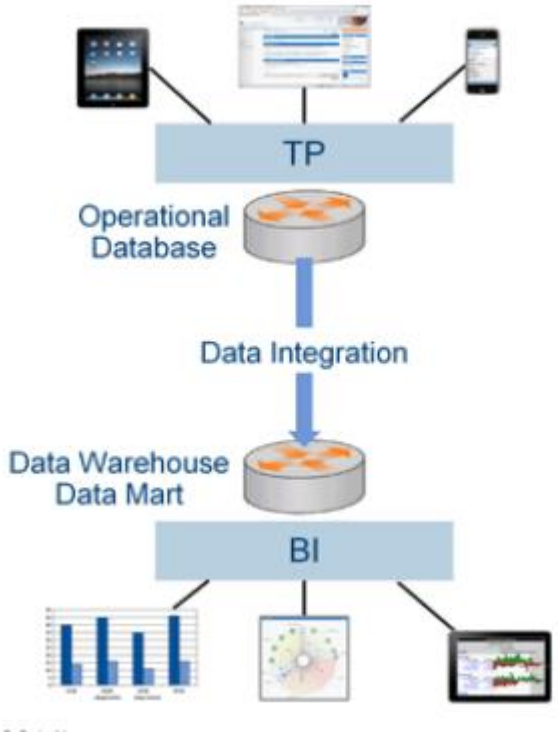
1．隔离性：通过线程的虚拟快照达到隔离性
2．数据新鲜度：数据更改后快速形成最新快照
3．事务一致性：查询的数据总是在最新数据上查询

优点：新鲜度最高

缺点：
1．不同工作负载性能影响较大，并不能为特定工作负载做优化
2．交叉工作负载对硬件资源干扰较强
3．依赖OS的内存数据管理
4．不易扩展

2.2 面向HTAP的内存数据库
-隔离设计架构

02 隔离架构设计



- ETL
  - 离线
  - 设计复杂性
  - 数据验证困难
- T+1
- 不同系统之间传播变化所花费的时间通常以分钟甚至小时来衡量。当数据输入到数据库中时，这种数据传输限制了应用程序对数据立即采取行动的能力。其次，部署和维护两个不同dbms的管理开销并不小

02 隔离架构设计

分布式存储与计算实验室

BatchDB

1．主从复制实现

2．支持TP和AP分别实现各自的算法优化和数据结构优化

3．牺牲空间实现性能隔离

4．通过分批调度来均衡开销（查询成批、数据传播成批）

5．OLAP查询等待成批以支持查询数据共享

6．使用RDMA提高传输效率，不断的将副本数据推送到AP侧

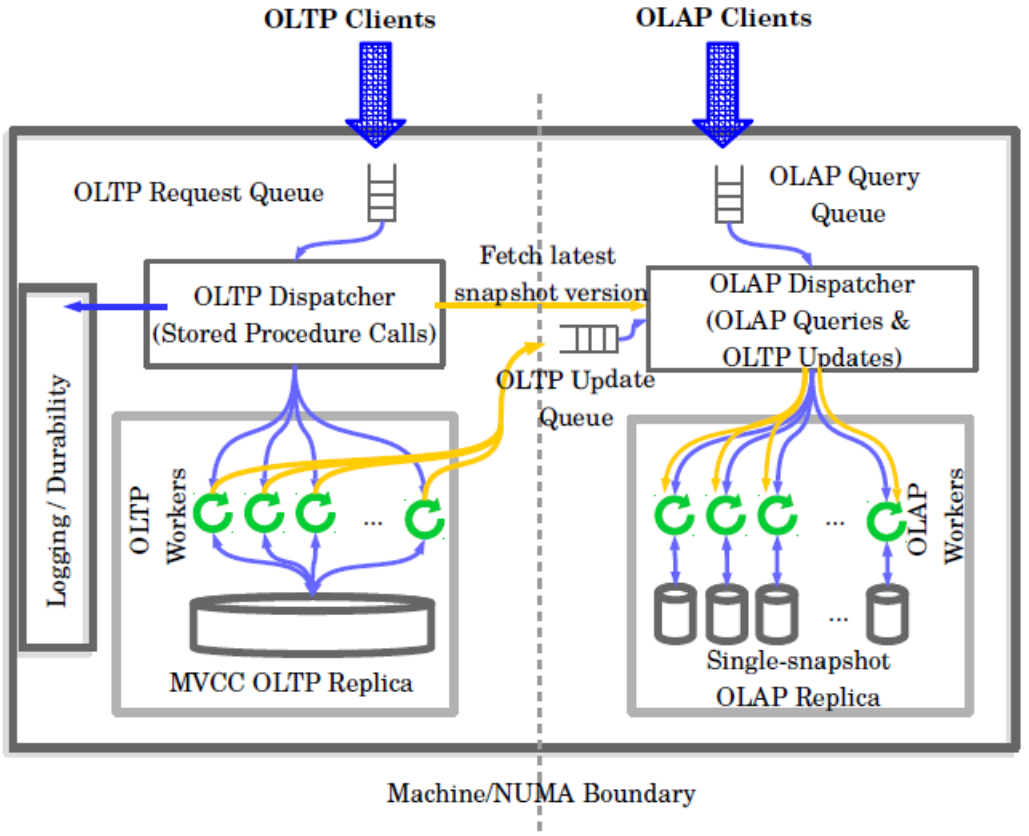


Figure 1: System architecture

02 隔离架构设计

OLTP设计：

1．数据库主副本

2．处理更新事务，提供短延迟请求响应

3．行式存储布局

4．索引数据结构，为特定键建立索引，支持快速查找

5．分批调度，OLTP请求批量处理，共享内存带宽

6．持久化日志，并将每个记录的更新日志作为传播日志保存

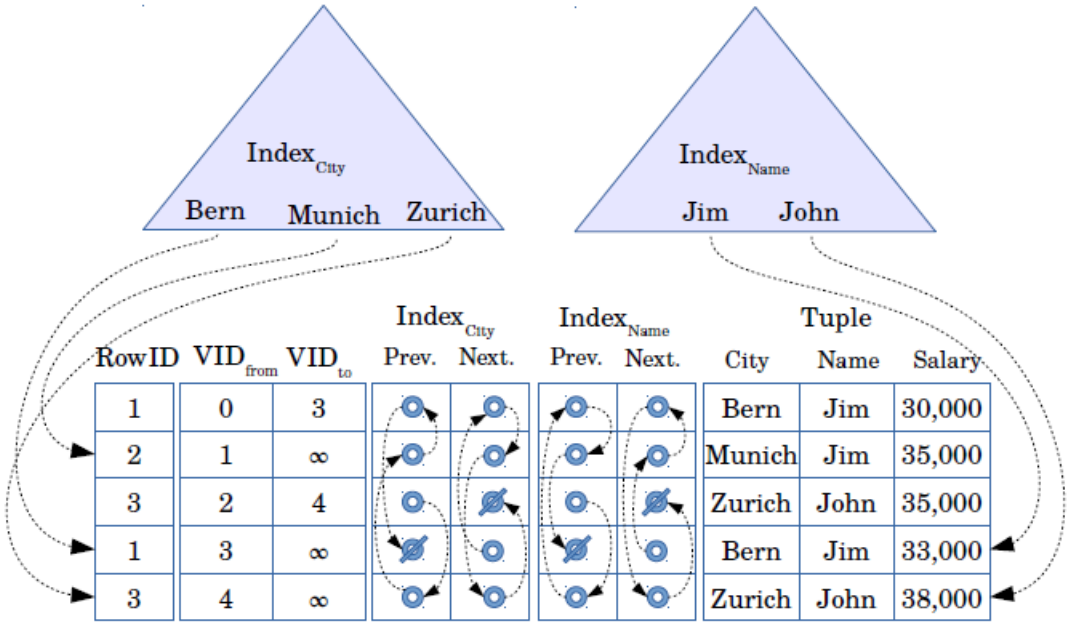


Figure 2: OLTP Record Storage Format and Index Layout Example

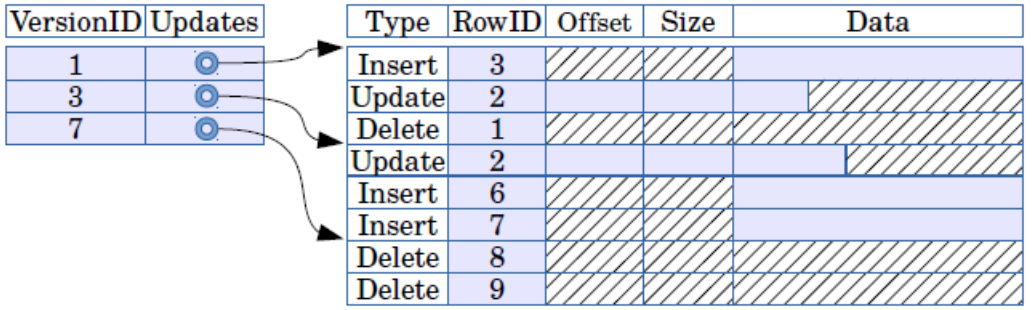


Figure 3: Propagated update format for a specific table from a single OLTP worker thread

02 隔离架构设计

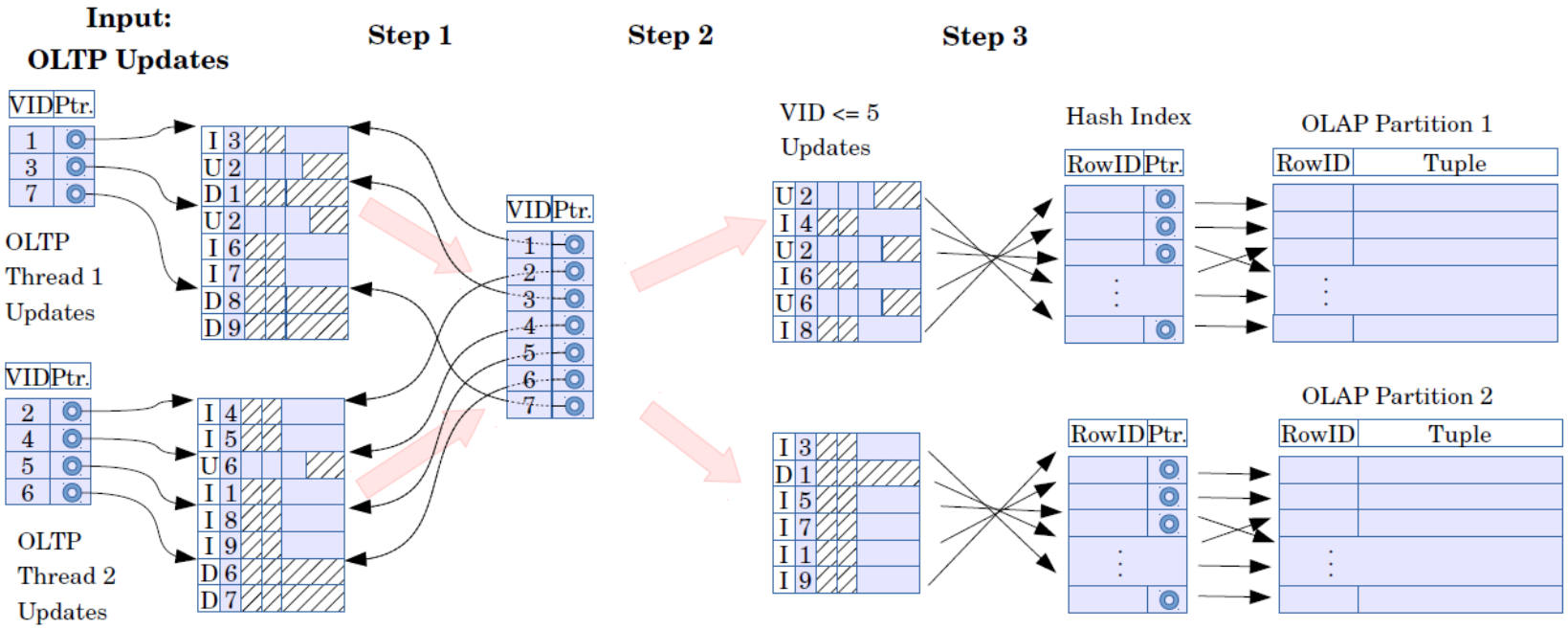


Figure 4: Process of applying propagated updates from OLTP replica into OLAP replica

OLAP设计：

1．OLAP分批查询
   1．共享内存带宽
   2．共享网络带宽

2．查询前获取最新事务数据

3．OLAP无需维护版本数据，只维护最新数据

4．更新传播
   1．按照快照版本ID进行排序
   2．根据查询的版本快照进行分区传播处理
   3．对每一个分区进行更新，查找对应元组位置，进行更新、删除、等操作，也可以支持存储布局转换

02 隔离架构设计

分布式存储与计算实验室

Janus

1．日志形式的主从复制

2．支持不同请求负载不同存储格式，提交后的事务数据转变存储格式

3．支持多分区、分布式事务和不同的分区策略

4．执行引擎做分区元数据信息管理

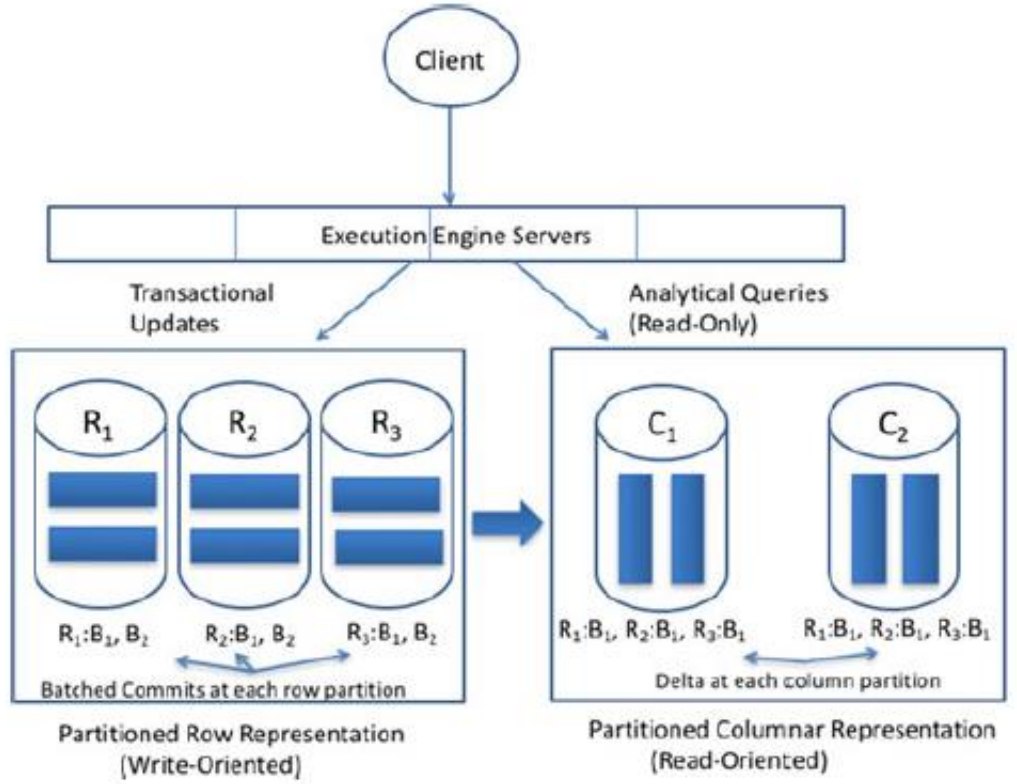


Fig. 1. Janus design.

02 隔离架构设计

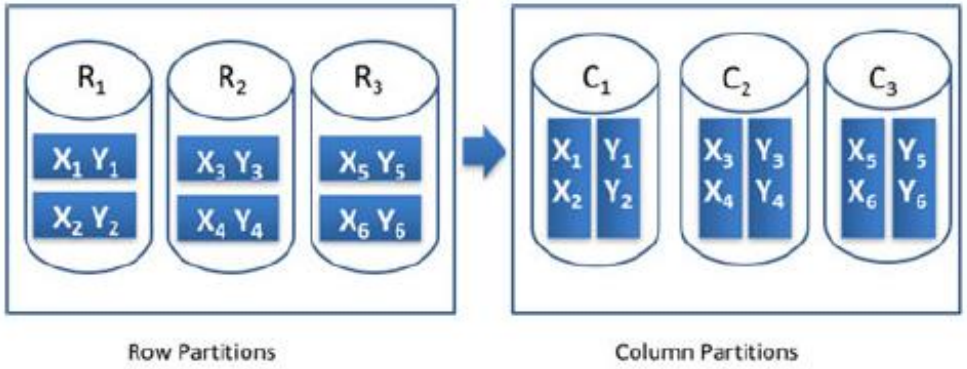
分布式存储与计算实验室

Janus

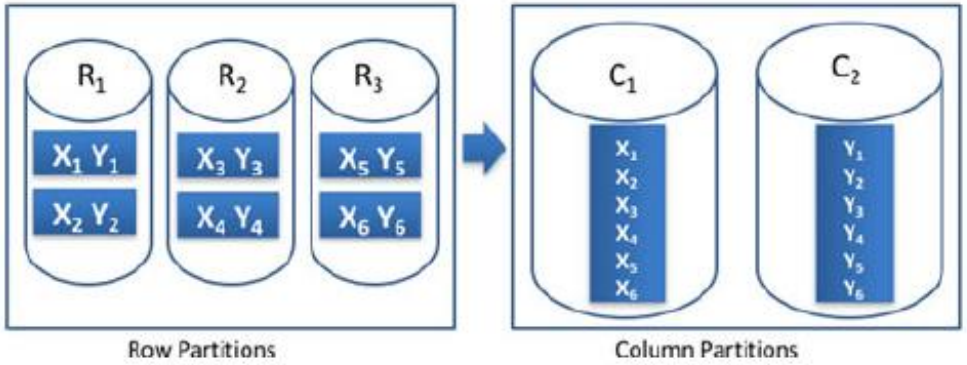
1. 问题：分布式场景会涉及多个分区，每个分区都有对应batch，一个事务可以涉及多个batch

2. TP与AP分区是多对一关系

3. 分布式多batch如何保证一致性
   1. Batch中保存元数据，包含batch的id，具备涉及数据的版本号
   2. 事务提交时将batch的版本号信息保存到每一个行分区中。



(a) Partitioning - Case 1



(b) Partitioning - Case 2

02 隔离架构设计

Janus

1. 核心思想：数据按照正确顺序处理，一个事务数据原子处理

2. 基于图依赖的算法
   1. 每个batch都有唯一的ID，由行分区和版本号组合
   2. 每个batch会被处理为图中一个节点
   3. 入边是该batch的依赖项

3. 两种依赖
   1. 分布式事务依赖，互相依赖
   2. 顺序依赖

4. 依赖对象apply后才能处理，即没有入边数据

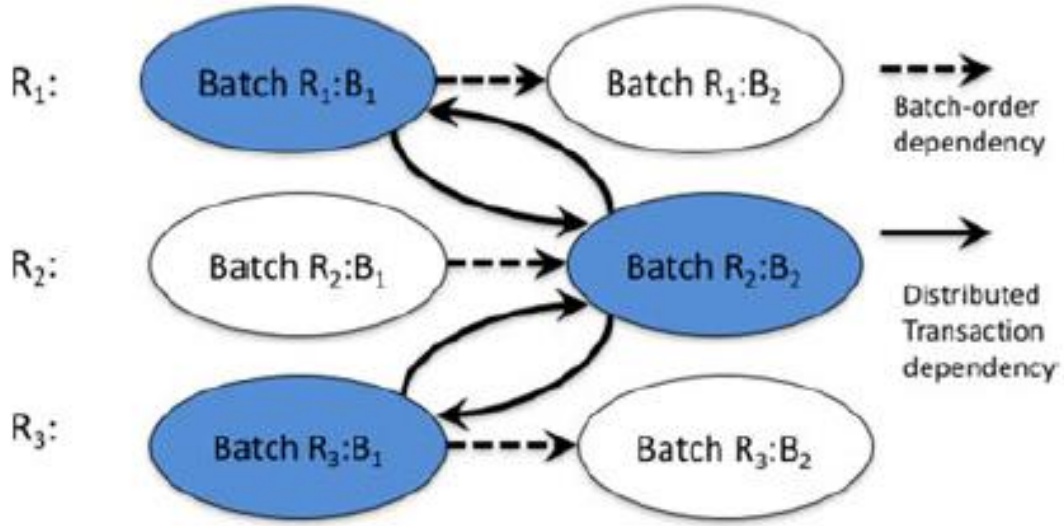


Fig. 4. Batch dependency graph.

02 隔离架构设计

关键问题解决方法：

1．隔离性：通过解耦硬件资源
2．数据新鲜度：数据更新日志批量传输同步
3．事务一致性：AP侧对事务排序还原

优点：
1．隔离性好
2．最佳优化对应的工作负载，充分发挥对应的机器性能

缺点：
1．数据新鲜度低
2．事务一致管理复杂

2.3 面向HTAP的内存数据库 -混合设计架构

02 混合架构设计

分布式存储与计算实验室

Oracle Dual-format

1. 内存中维护数据两种存储格式
2. 面向OLAP分析表以列格式存储到内存中，对DML数据操作轻量级维护
3. 降低OLTP索引维护
4. 无缝内置到Oracle数据库引擎中，不影响原有功能（数据库恢复、灾难恢复、备份、复制、和集群高可用）

Figure 1: Dual-Format Database Architecture

[Oracle Database 12c Release 1 (12.1.0.2) New Features]

02 混合架构设计

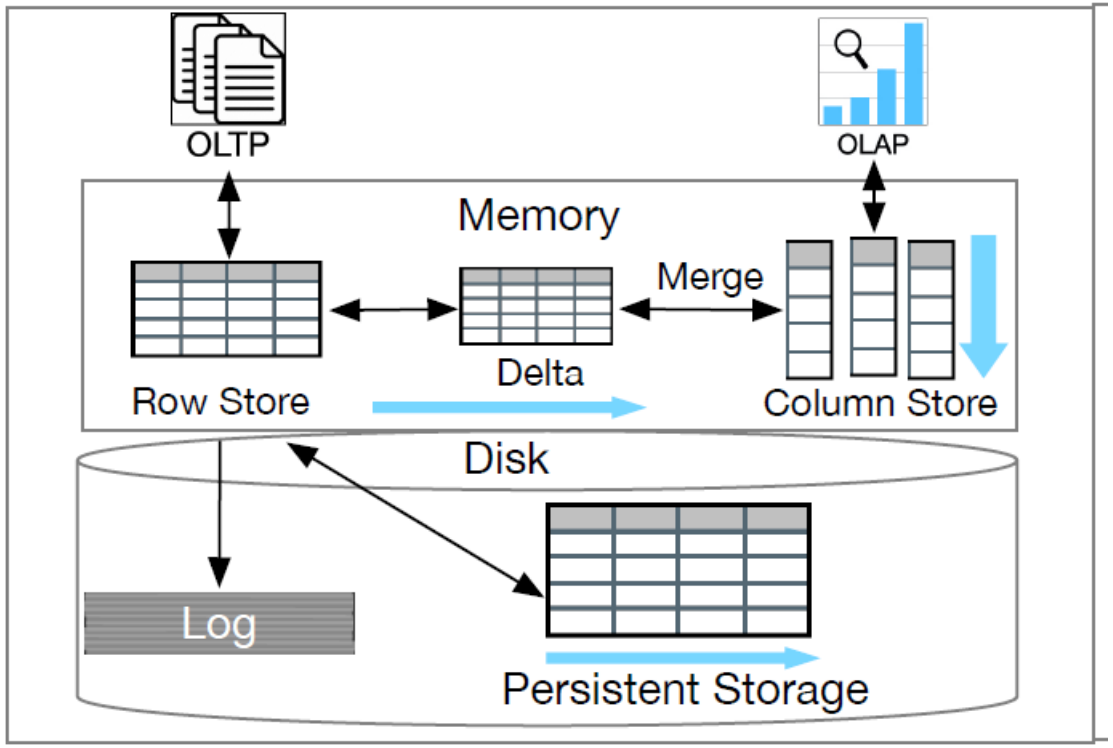
分布式存储与计算实验室

1．列式格式数据来源与持久化后的行存数据。

2．IMCU：内存管理单元，为表中数据连续分配内存进行列式数据管理

3．IMCU支持多种面向预期请求的数据压缩方式：
   1．DML：最小压缩，支持快速扫描
   2．Query：允许就地访问的压缩
   3．CAPACTITY：高压缩比，访问数据需要解压

4．支持同表多种压缩方式



(a) Primary Row Store+In-Memory Column Store

02 混合架构设计

确保一致性读

1. 数据库事务具有单调递增的版本标记SCN

2. 为IMCU添加元数据跟踪单元SMU，负责跟踪该IMCU管理的数据是否有效

3. 行存事务发生数据更改，会产生对应的日志保留到SMU中

4. 列式扫描步骤：
    1. 先扫描SCN获取变更日志
    2. 提取变更日志中修改的列值
    3. 后扫描列式数据
    4. 将修改的列值与扫描列值合并返回上层

5. 周期性重建IMCU

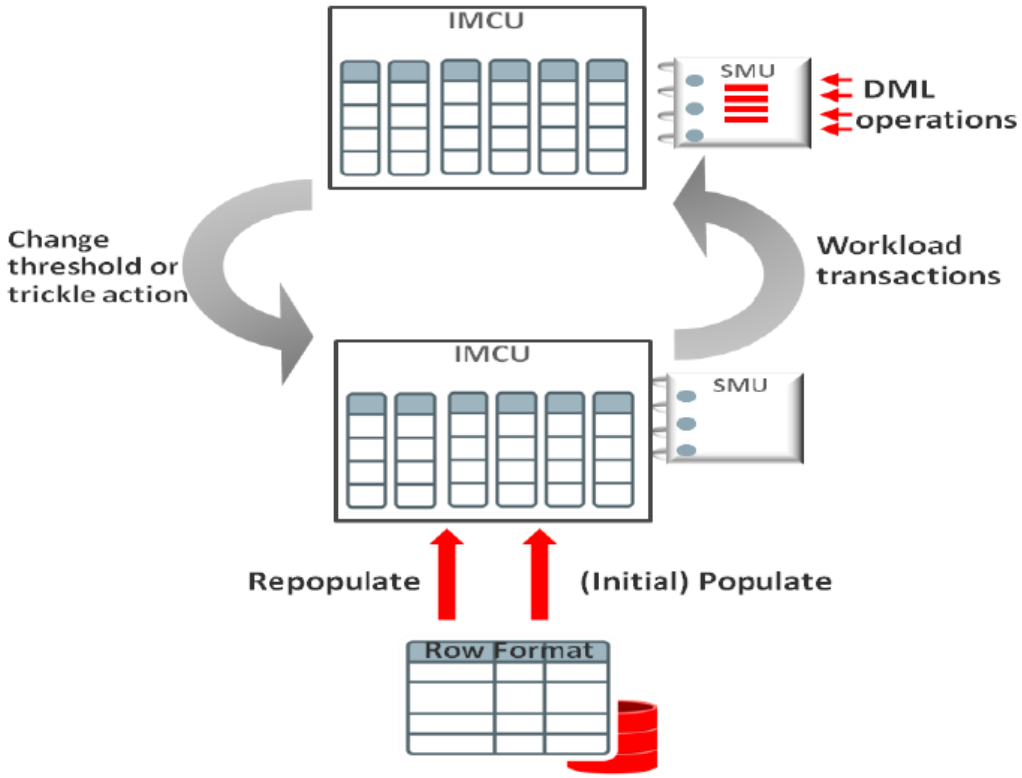


Figure 6: Change tracking and repopulate

02 混合架构设计

关键问题解决方法：

1．隔离性：通过内存管理隔离
2．数据新鲜度：AP侧主动查询事务更新的日志数据
3．事务一致性：数据版本号记录

优点：
1．介于两者设计之中
2．无需日志传输大开销，来获取较高数据新鲜度
3．无需与TP资源竞争明显

缺点：
硬件资源共享，对TP和AP仍有干扰

2.4 面向HTAP的内存数据库 -调度式设计架构

02 调度设计架构

分布式存储与计算实验室

问题：当前没有那种机制保证不会对TP或AP引擎的损耗
1．隔离系统吞吐量高，共享系统实时性高

2．查询频率高时隔离系统实时性提高

Adaptive HTAP：
1．当前业务场景主要考虑数据的新鲜度

2．将HTAP作为调度问题处理

3．通过弹性资源应对不同的工作负载请求+调度算法

4．内存系统设计

5．支持可变数据新鲜度要求的查询工作负载

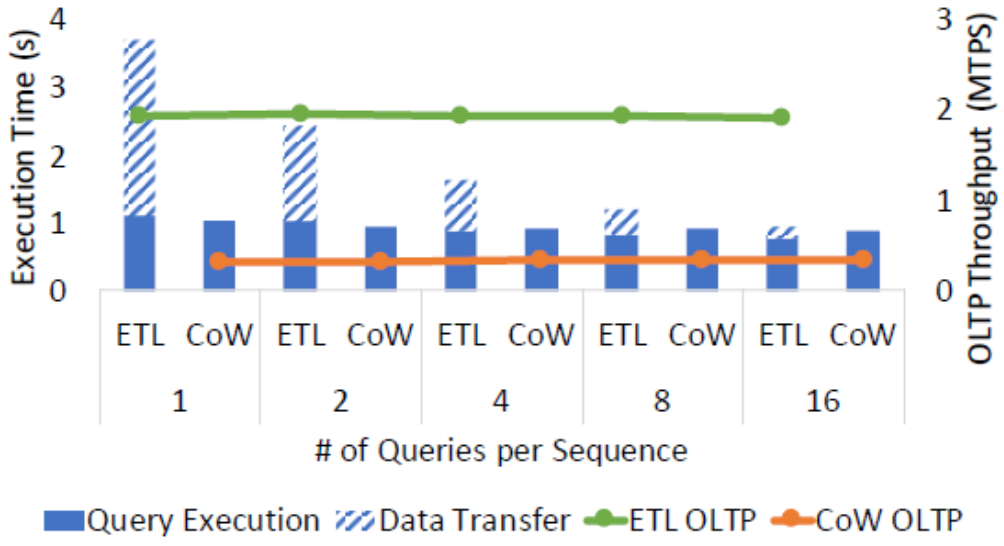


Figure 1: Trade-off between ETL- & CoW-based HTAP.

02 调度设计架构

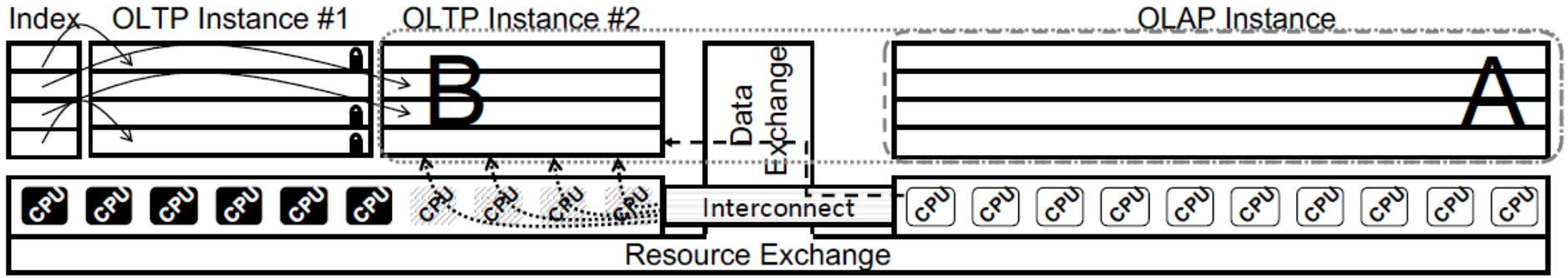


Figure 2: System architecture

OLTP引擎设计

1．具有存储管理、事务管理、线程管理

2．内存中列式格式存储

3．在双实例中维护和管理数据，当请求发生时切换实例（同时同步数据）

4．索引执行最新数据所在实例位置

OLAP引擎设计

1．包含存储管理、查询执行器

2．存储管理具有数据访问路径插件，决定数据如何获取，具有不同的数据访问方式
   1．访问AP存储和访问TP存储

02 调度设计架构

分布式存储与计算实验室

HTAP设计：

1．每个引擎有自己的存储空间，并通过访问路径设置支持彼此访问存储空间

2．HTAP具有单向依赖：从OLAP引擎到OLTP读取数据

3．核心思想：可选的共享OLTP引擎存储或复制到OLAP引擎

4．引入RDE引擎
   1．作为所有内存和CPU资源的所有者，
   2．并为OLTP和OLAP引擎分配资源和数据，
   3．决定了状态迁移
   4．记录TP数据的统计信息

5．三种状态：共享OLTP和OLAP、隔离的OLAP和OLAP、混合OLTP和OLAP

02 调度设计架构

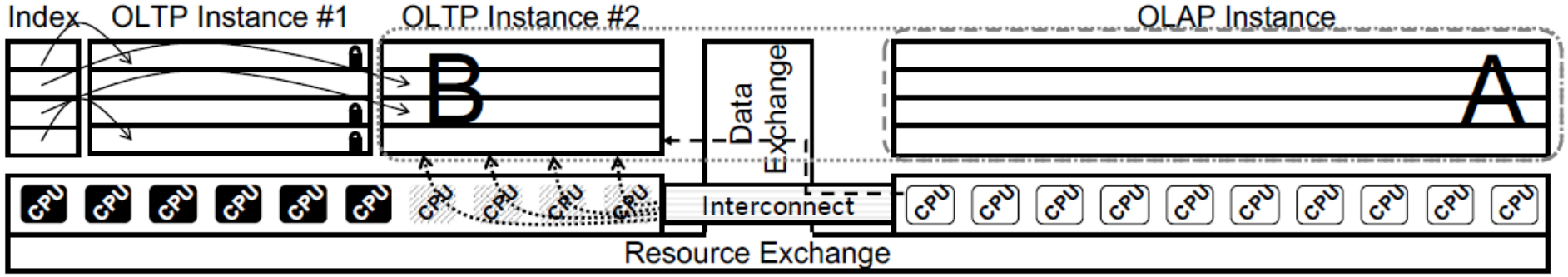


Figure 2: System architecture

状态一：共享架构

1．共享内存和CPU

2．AP引擎请求到达时切换TP活动实例，AP直接在TP非活动实例上查询

状态二：隔离架构

1．各个引擎服务对应的请求

2．AP引擎接受到请求时切换TP活动实例，将增量数据同步到OLAP存储

3．网络带宽比内存带宽小，延迟较高，隔离较好

状态三：混合架构

1．共享内存和CPU

2．AP引擎请求到达时切换TP活动实例，AP引擎通过套接字向TP侧访问最新的数据

02 调度设计架构

关键问题解决方法：

1．隔离性：通过调度引擎隔离硬件资源
2．数据新鲜度：根据需求获取不同数据新鲜度数据
3．事务一致性：不同状态适用该状态方案

优点：
实现HTAP动态调度适应多种工作负载

缺点
1．实现难度大
2．实时调度难
3．对硬件资源要求高

03 总结

03 总结

DSC 分布式存储与计算实验室

	思路	优点	缺点
共享设计	1. 基于硬件的COW	1. 新鲜度最高	1. 不同工作负载性能影响较大，并不能为特定工作负载做优化 2. 交叉工作负载对硬件资源干扰较强 3. 依赖OS的内存数据管理 4. 不易扩展
隔离设计	1. 资源物理隔离 2. 增量日志批量传输	1. 隔离性好 2. 最佳优化对应的工作负载，充分发挥对应的机器性能	1. 数据新鲜度低 2. 事务一致管理复杂
混合设计	1. 内存双格式处理	1. 介于两者设计之中 2. 无需日志传输大开销，来获取较高数据新鲜度 3. 无需与TP资源竞争明显	硬件资源共享，对TP和AP仍有干扰
调度式设计	1. 通过资源弹性管理 2. 结合调度算法进行状态切换	实现HTAP动态调度适应多种工作负载	1. 实现难度大 2. 实时调度难 3. 对硬件资源要求高

03 文献

分布式存储与计算实验室

[01] - Milkai, Elena, et al. "How Good is My HTAP System?." *Proceedings of the 2022 International Conference on Management of Data*. 2022.

[02] - Makreshanski, Darko, et al. "BatchDB: Efficient isolated execution of hybrid OLTP+ OLAP workloads for interactive applications." *Proceedings of the 2017 ACM International Conference on Management of Data*. 2017.

[03] - Lahiri, Tirthankar, et al. "Oracle database in-memory: A dual format in-memory database." *2015 IEEE 31st International Conference on Data Engineering*. IEEE, 2015.

[04] - Raza, Aunn, et al. "Adaptive HTAP through elastic resource scheduling." *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2020.