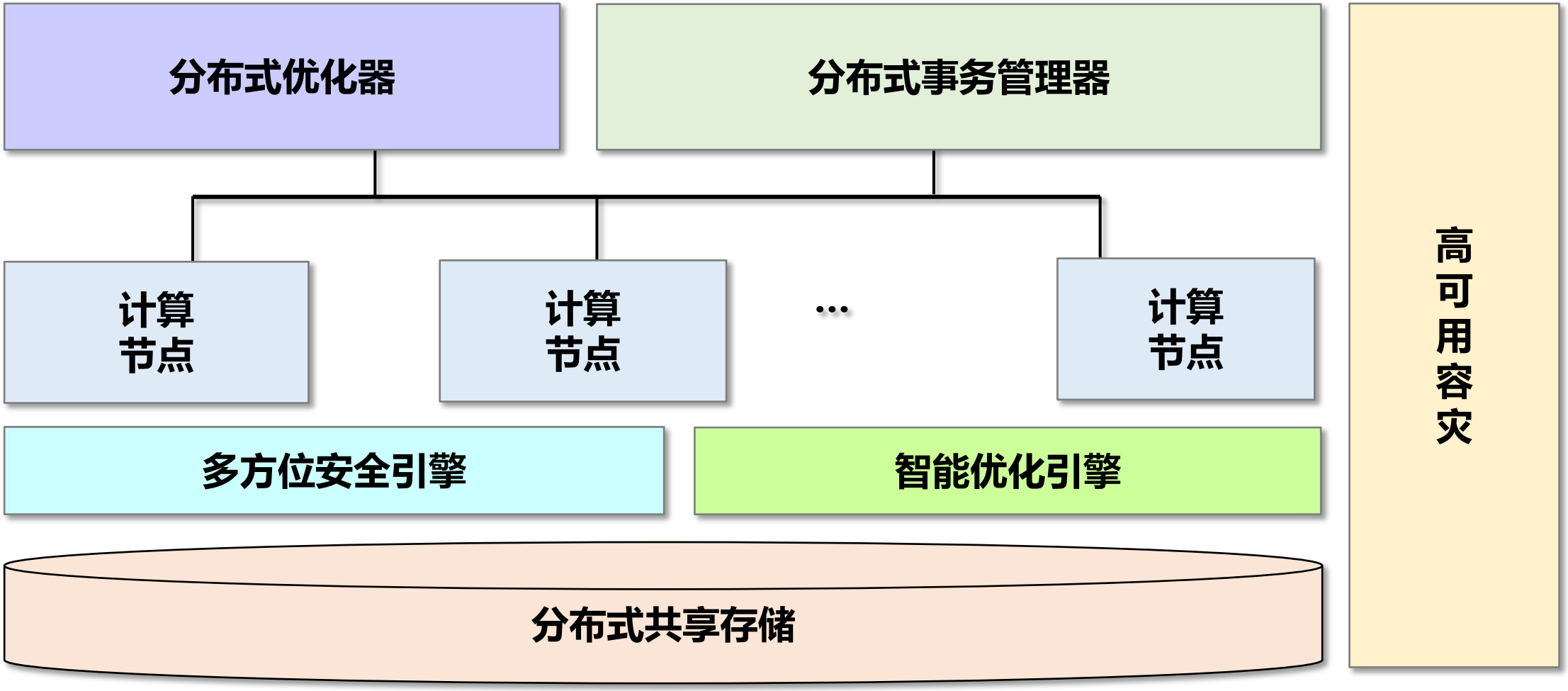
第15章 GaussDB 简介

清华大学计算机系 李国良

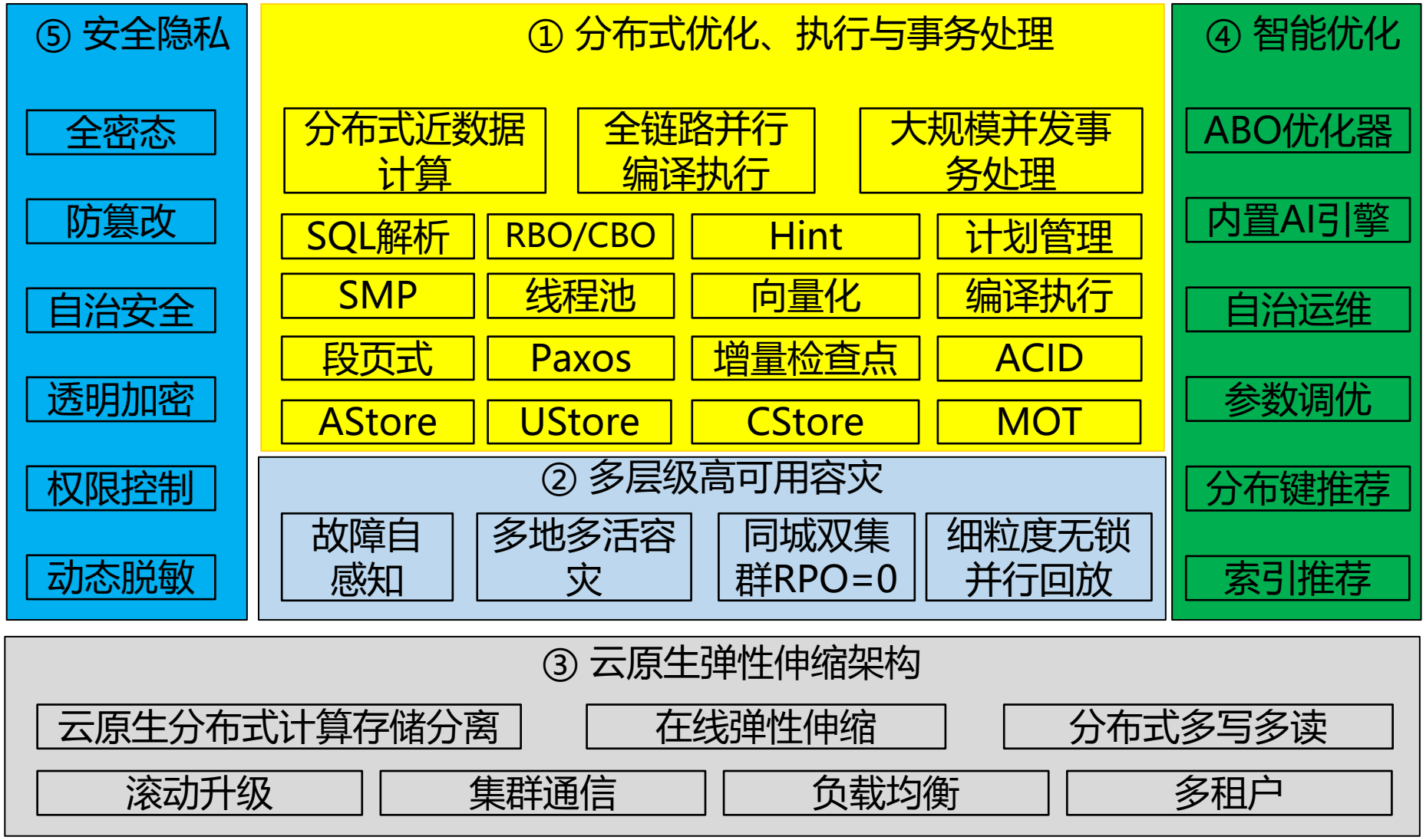
liguoliang@tsinghua.edu.cn
http://dbgroup.cs.tsinghua.edu.cn/ligl

目录

GaussDB技术架构



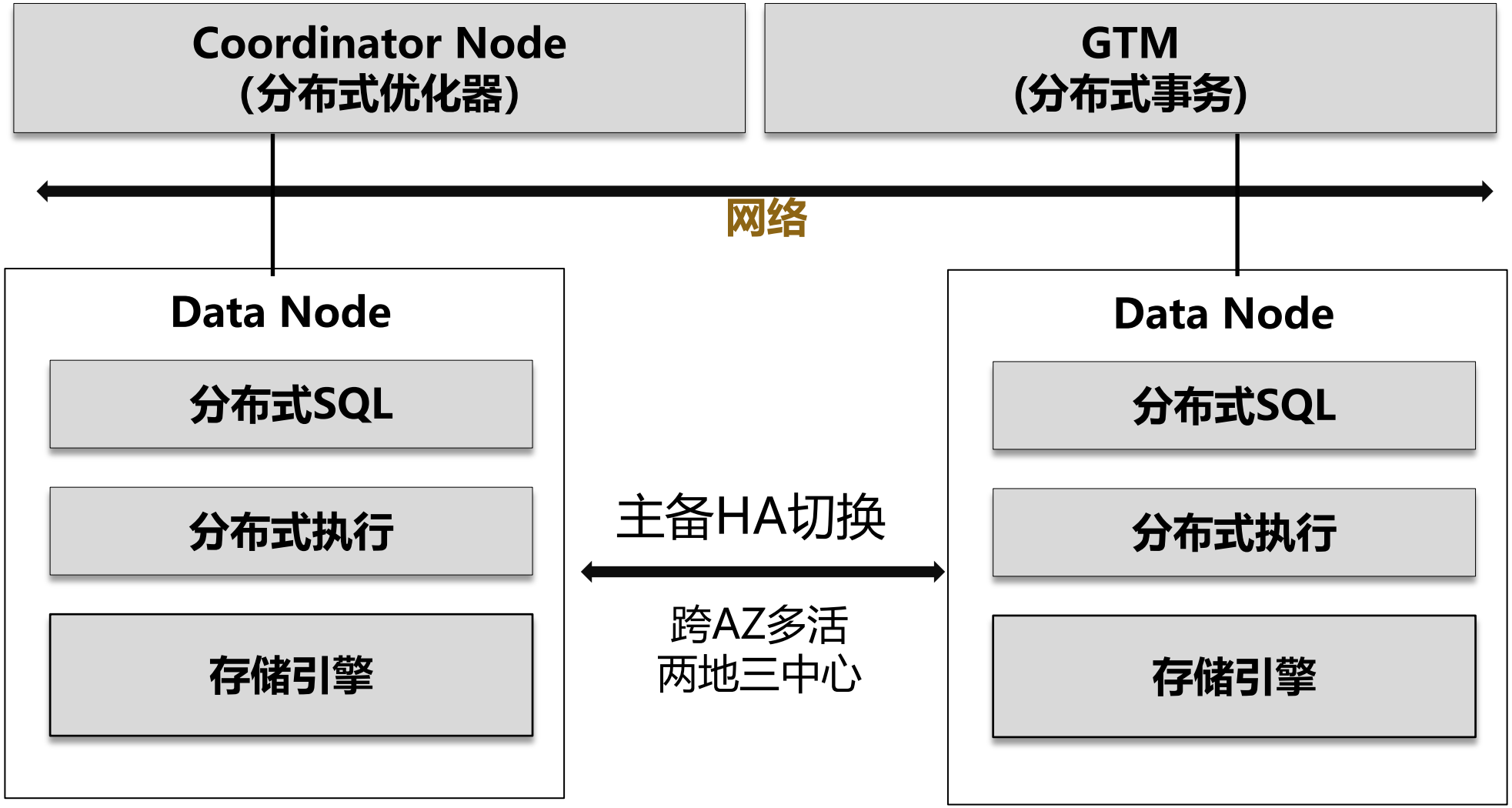
GaussDB技术架构



GaussDB技术架构特点

- 原生分布式计算架构(Shared Nothing)
  - 支持分布式SQL、分布式执行、分布式存储、分布式事务
  - 可插拔存储引擎（内存表、行存表、列存表）
- 云原生架构：计算-内存-存储分离、日志即数据
- 多层级高可用：故障自感知、多副本恢复、两地三中心、同城双集群、异地多活
- 基于机器学习的智能优化器：ABO、智能优化、库内AI
- 安全管理：安全认证、访问控制、统一审计、计算-存储-传输的全密态、数据脱敏、防篡改的存储策略
- HTAP：行列混存、智能行转列、新鲜度为0的透明行列路由、行列混存优化器

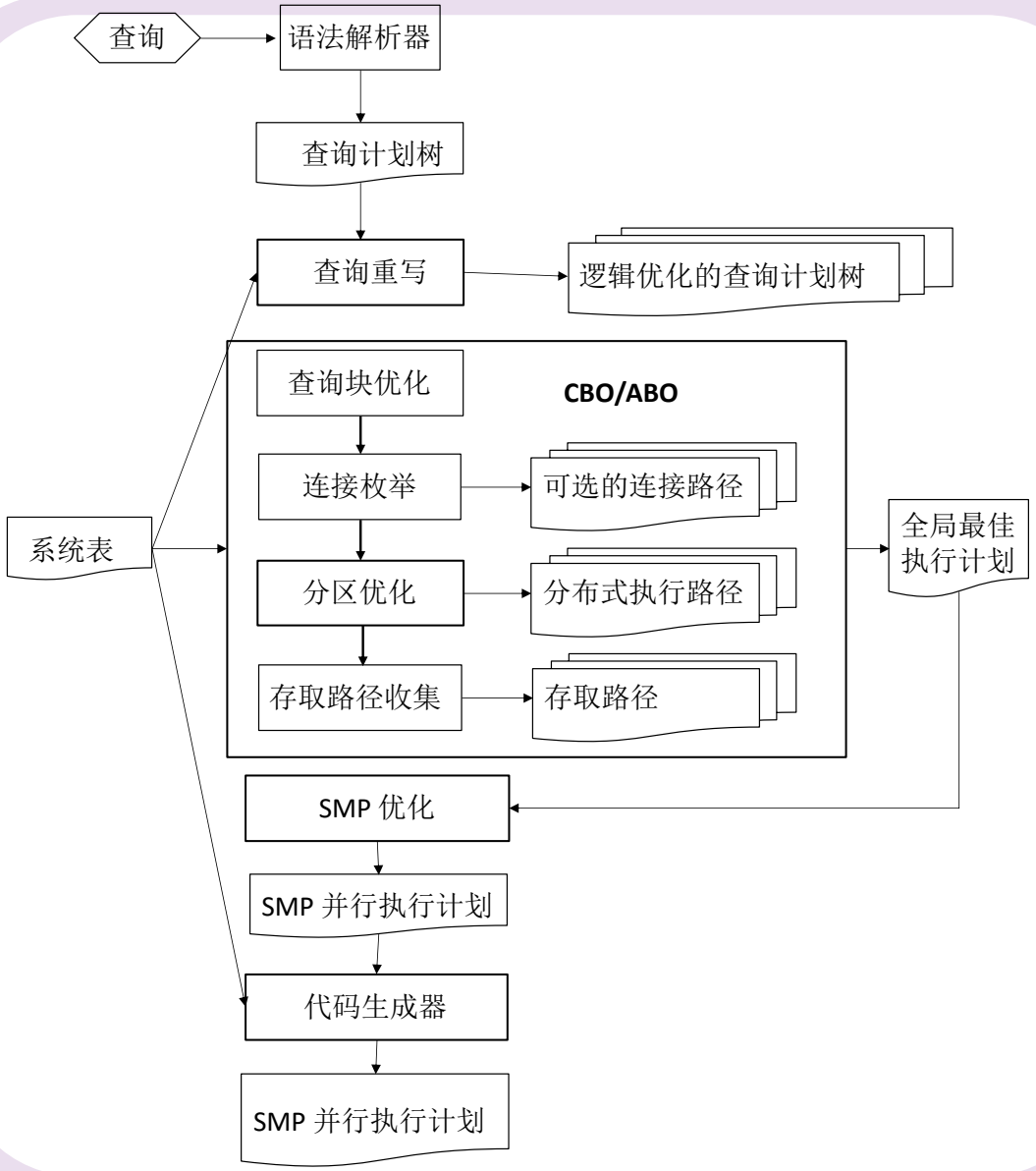
GaussDB分布式架构



GaussDB分布式架构

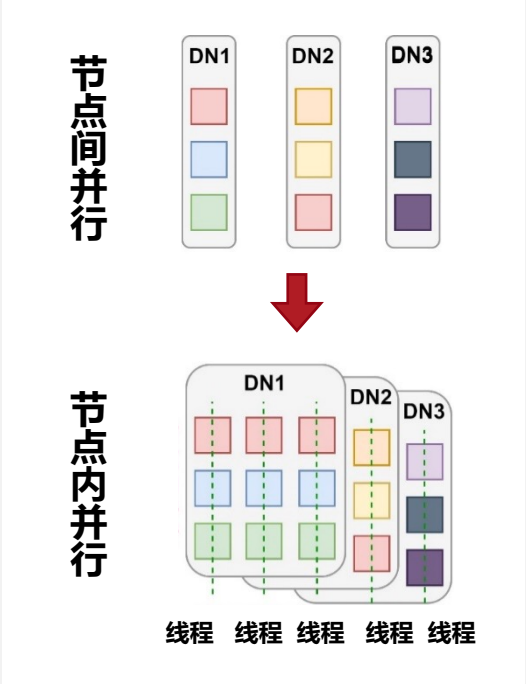
- 分布式执行框架
  - CN（协调节点）与DN（数据节点）相互配合
- 分布式事务处理
  - 保证事务全局强一致性，提供高性能事务处理能力
- 分布式查询
  - RBO、CBO、ABO
- 分布式并行执行
  - 支持节点间并行、线程间并行和单指令多数据并行，提高复杂查询的性能
- 分布式高可用
  - 支持节点级、可用区级、城市级等多层级故障恢复

GaussDB分布式优化器



GaussDB分布式并行执行

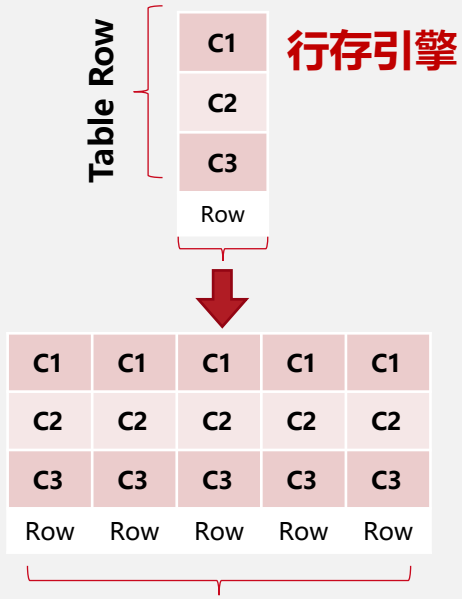
并行执行(Parallel)



□ 多机多线程并行执行

- 充分利用当前多核特点，通过多线程并发执行，提高系统吞吐量

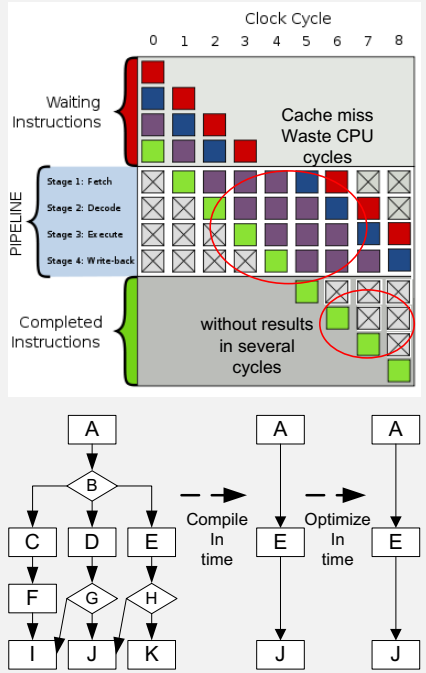
向量执行(SIMD)



□ 利用CPU的局部性及算力

- 利用CPU的局部性，通过向量执行提高执行效率

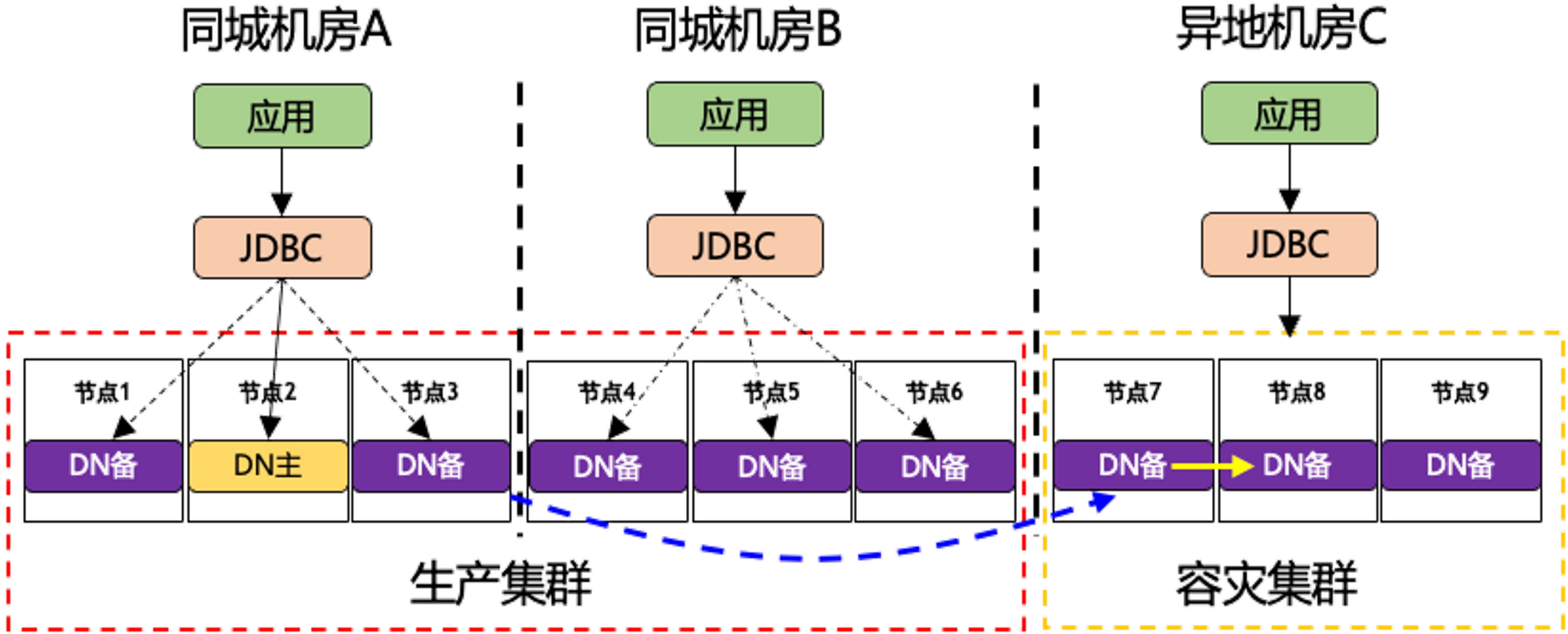
编译执行(LLVM)



□ 提高CPU指令的利用率

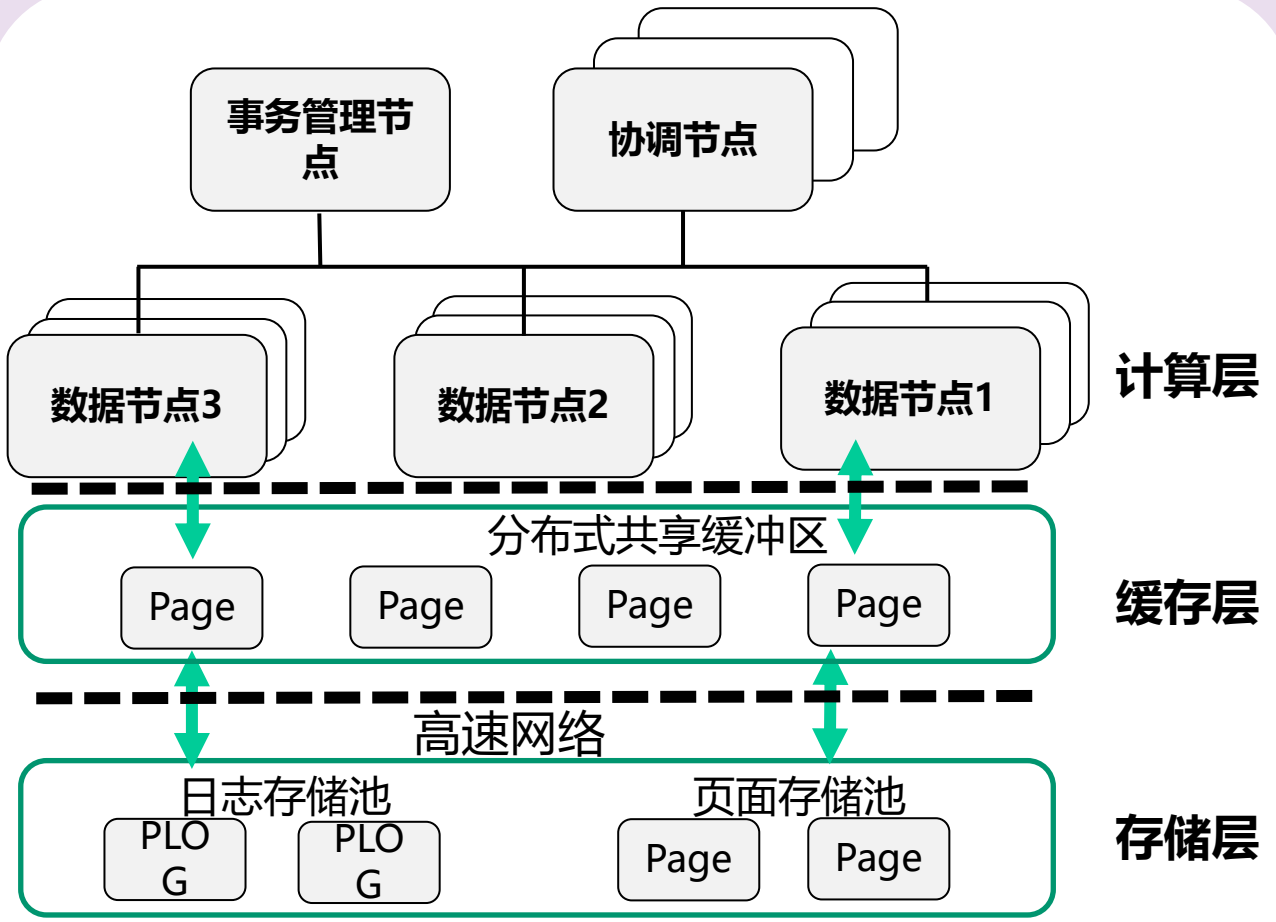
- 从解释执行向编译执行转变，大幅降低执行算子指令数量，提高运算效率

GaussDB两地三中心高可用架构



GaussDB云原生架构

- 计算存储分离架构
- 协调节点CN
  - 调度用户查询请求，负载均衡后交由DN处理
- 数据节点DN
  - 处理查询的实际执行过程
  - 缓存一定量的记录数据和日志数据
- 分布式共享缓冲区
- 分布式共享存储
  - 页面和日志逻辑上分离管理，物理上存储于相同的分布式存储，保证有效管理与高效传输

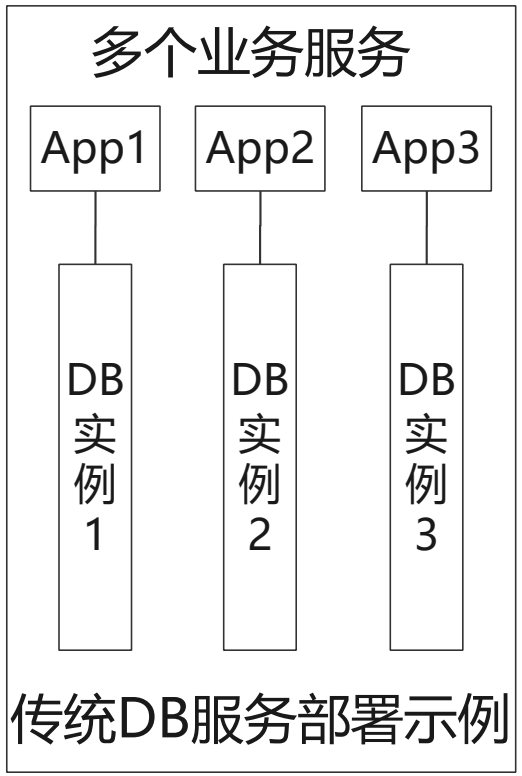


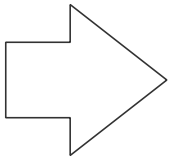
GaussDB多租户

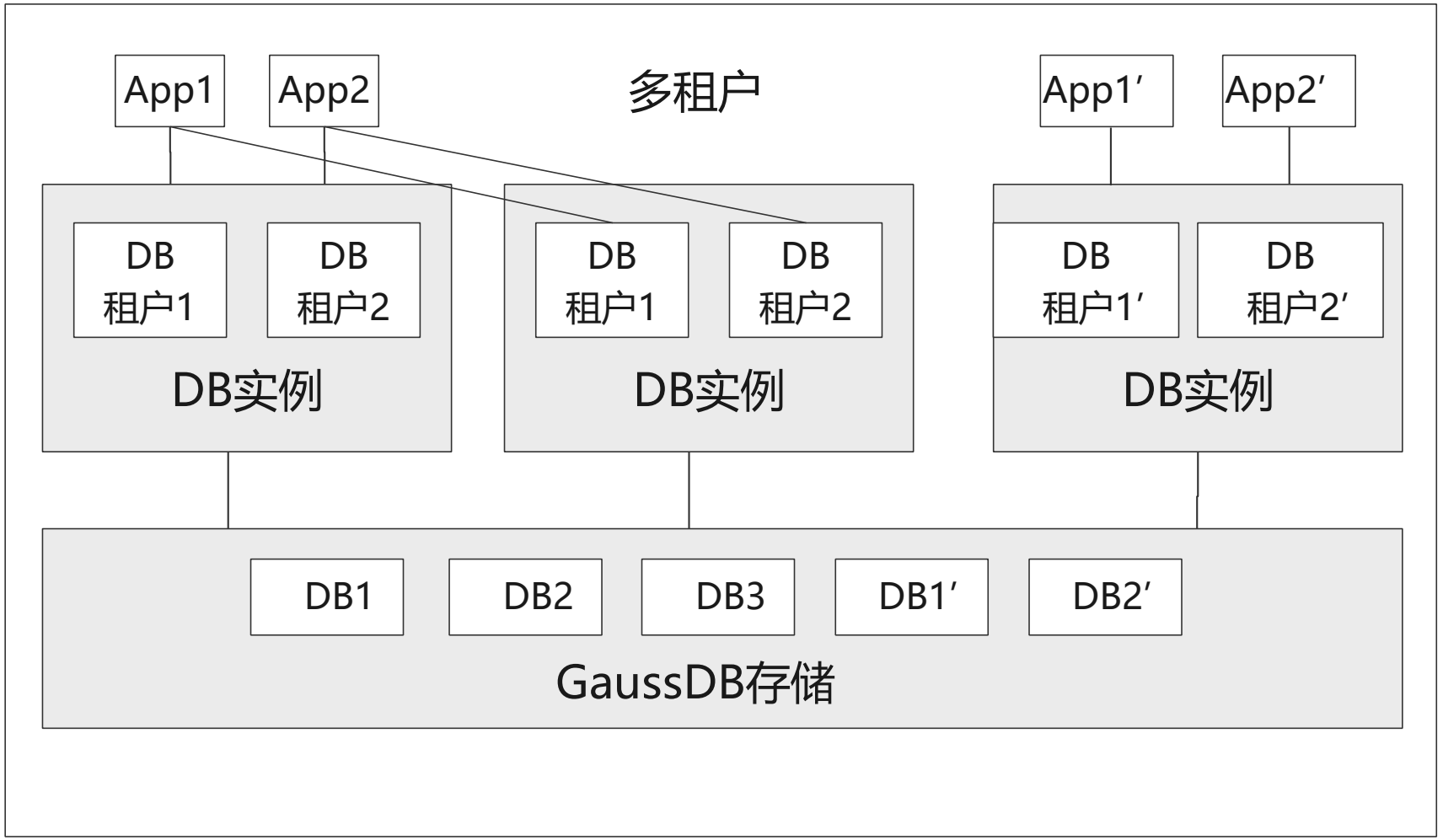
- 数据库
  - 资源管控与隔离
    - CPU、内存、磁盘
  - 数据隔离
    - 数据多租
    - 租户索引
  - 日志隔离
  - 弹性伸缩
- 租户
  - 资源弹性调度
  - 租户资源管控

GaussDB多租户

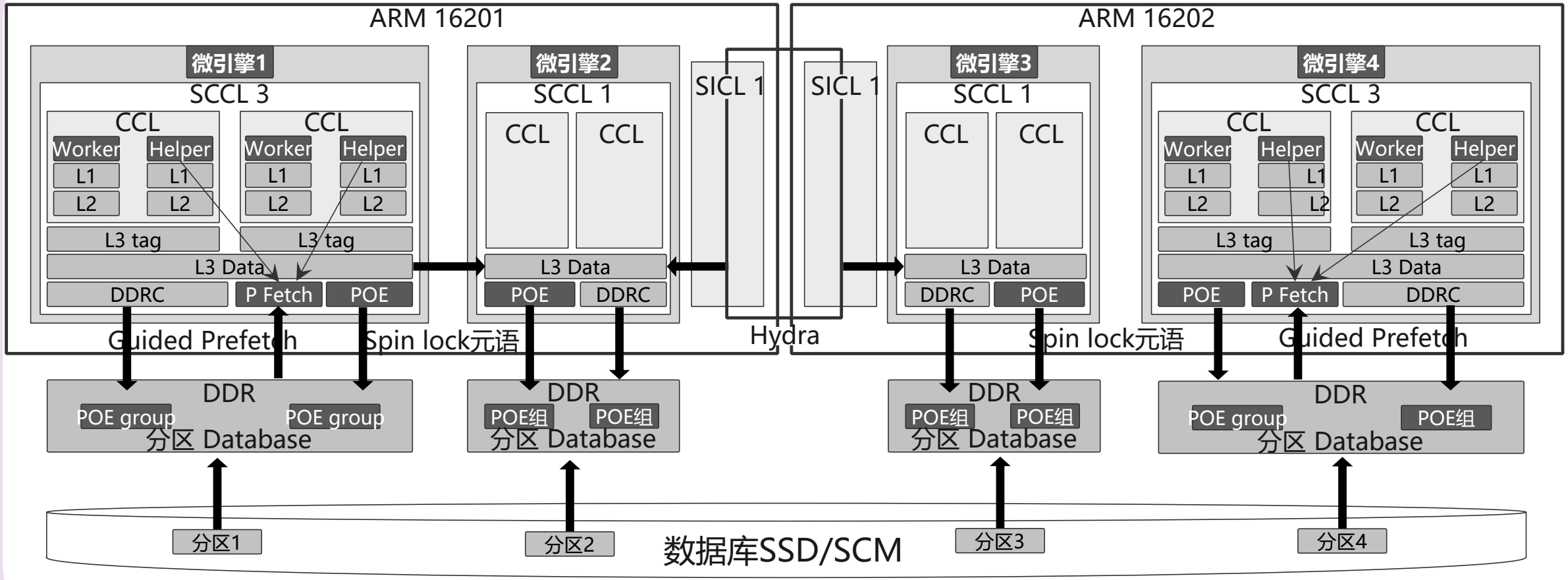
相互隔离的业务应用





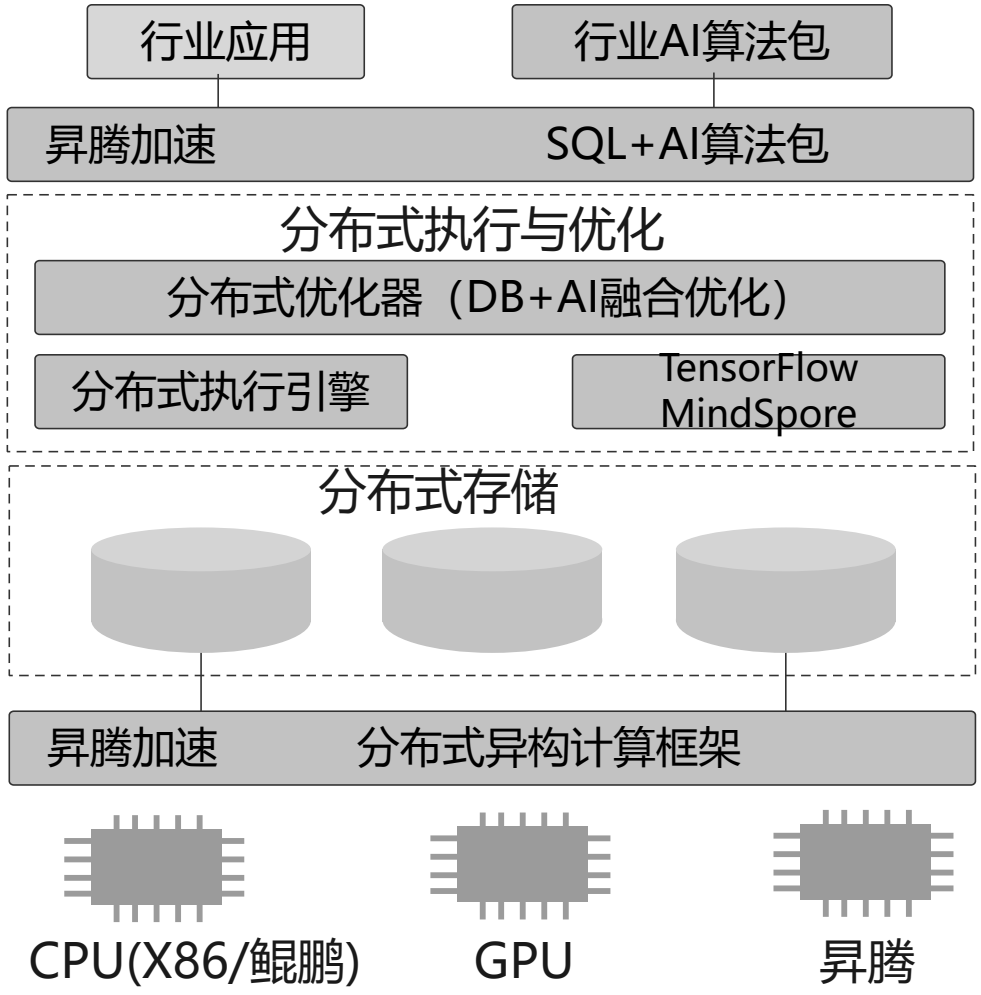


GausDB面向鲲鹏NUMA的架构

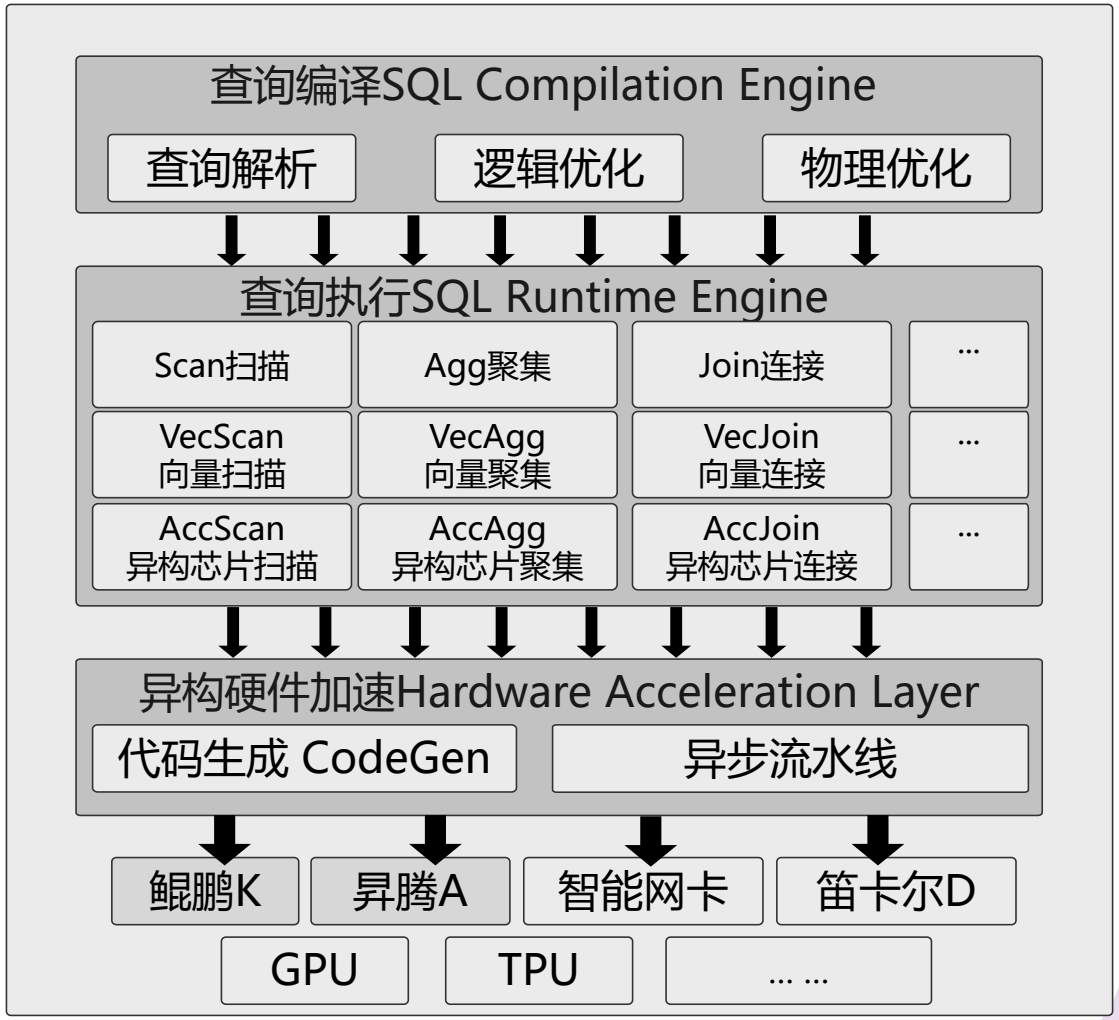


与昇腾结合的AI加速与计算加速

AI算子加速



DB算子加速



GausDB面向昇腾GPU的架构

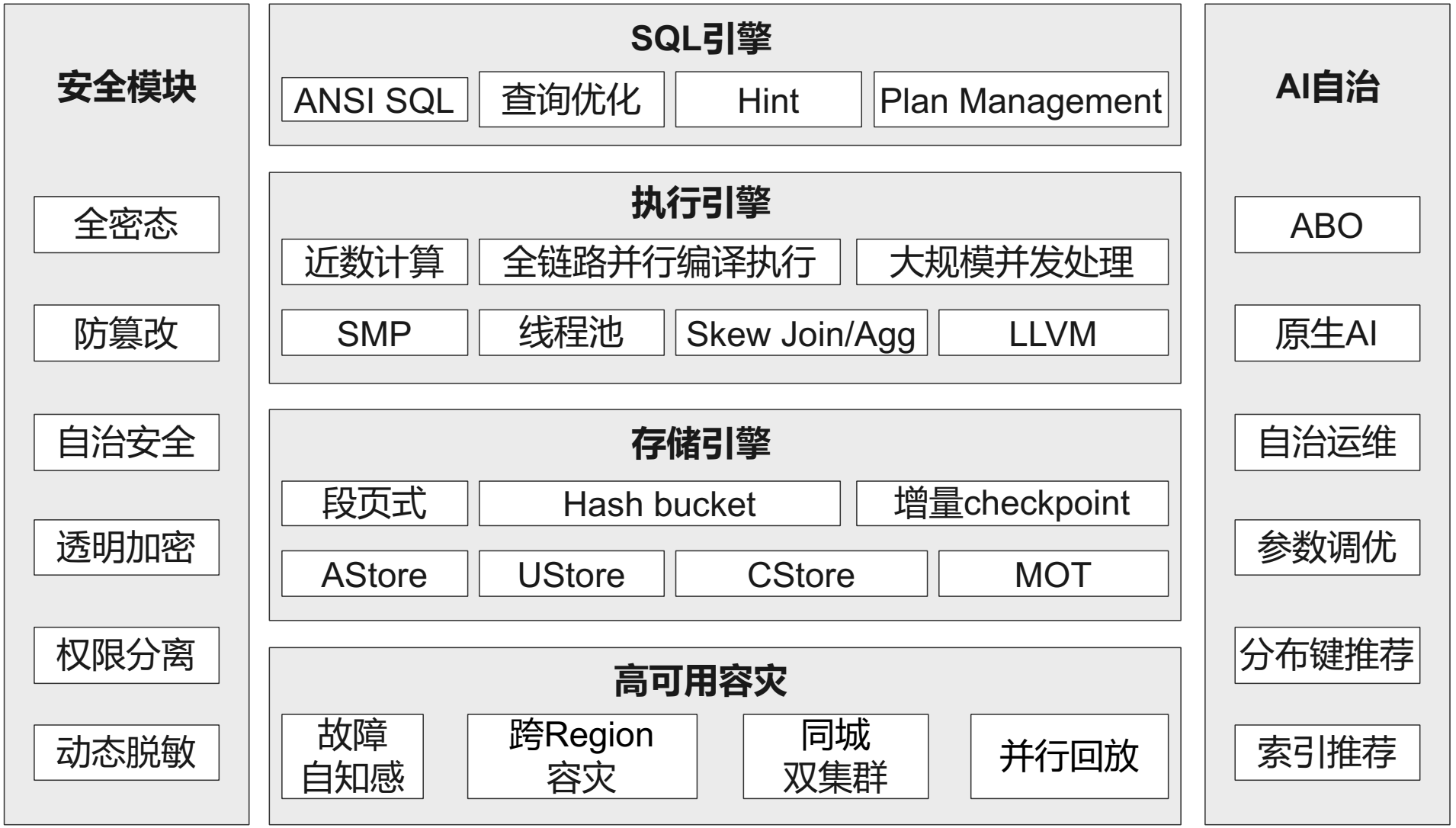
- 支持和优化AI算法中不同类型的算子操作，并通过SQL语句来支持GaussDB内AI算子并发原语机制
- 库内集成Tensorflow/MindSpore深度学习框架
  - 数据库内部实现CNN、DNN等神经网络算法，并基于昇腾芯片进行加速优化
- 数据库的内置AI算法包
  - 利用数据库优化器、索引、剪枝等技术实现机器学习模型训练与推理过程的加速
- 对vector、cube等计算模型的加速能力
  - 实现传统数据库内聚集操作（Aggregation）、连接操作（Join）的加速

GausDB面向新硬件的异构计算

- 支持NUMA架构，提供大规模数据计算能力
- 并发原语机制
  - 通过核间通讯机制，无需通过内存变量的原子性修改来达成协同一致
- 异步流水线机制
  - 执行事务时，可根据任务特点，将其切分成多个子任务，由特定线程执行
- 共享数据结构的全局访问
  - 利用单独的核来进行处理，减少数据访问冲突
- 新型的NUMA事务处理及存储引擎架构
  - 基于数据进行事务分发、基于NUMA的数据老化回收等，减少核间冲突、提升系统效率

目录

GaussDB单机架构—openGauss



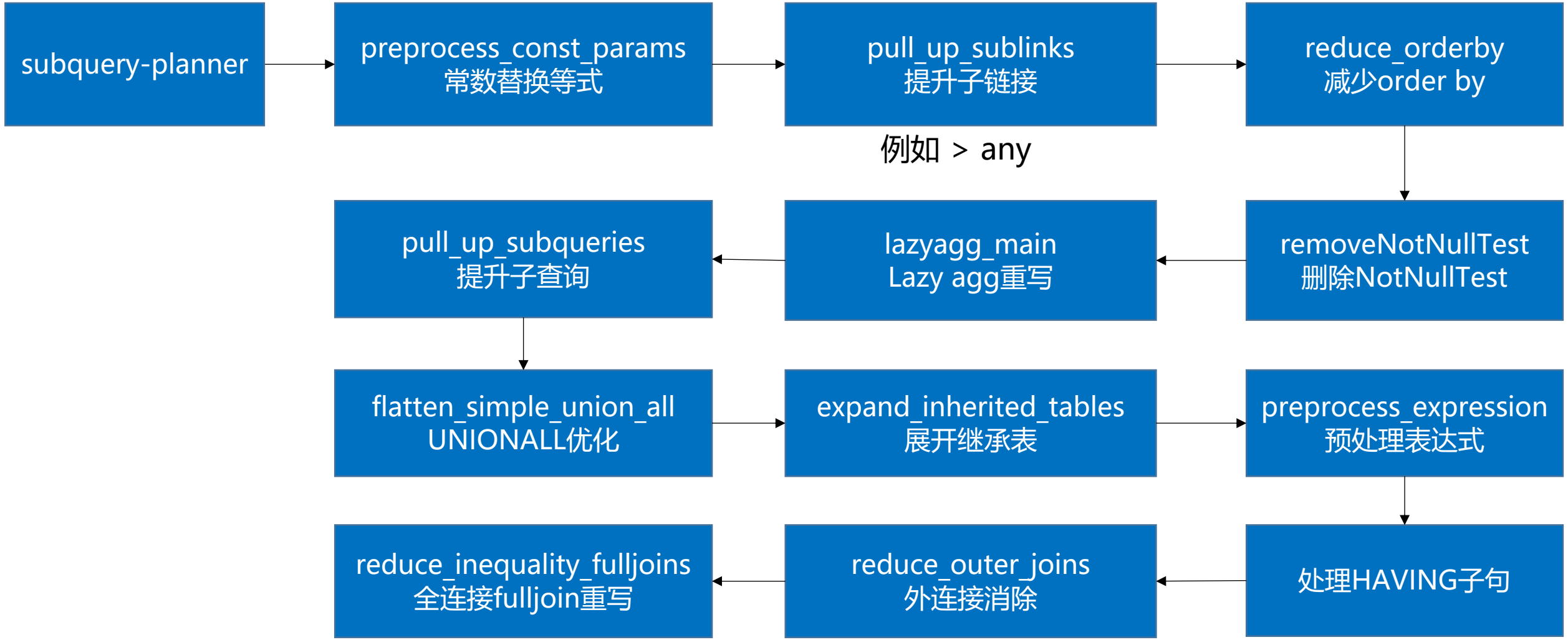
GaussDB单机架构—openGauss

- SQL引擎
  - 支持查询重写、统计信息、基数估计、代价估计、物理计划选择等功能
- 执行引擎
  - 支持近数计算、LLVM、并发处理、并行执行、向量化执行等功能
- 存储引擎
  - 段页式管理方式，支持追加更新行存、就地更新行存、列存、内存引擎等
- 高可用容灾
  - 支持故障自感知、故障自恢复、多副本存储、主备高可用部署等
- 高智能
  - 基于AI的自监控、自诊断、自优化：智能基数估计和计划生成；库内AI；智能调优和运维
- 安全
  - 支持认证、权限控制、异常检测、审计等传统数据库安全能力
  - 支持全密态、防篡改、隐私保护等新型安全能力

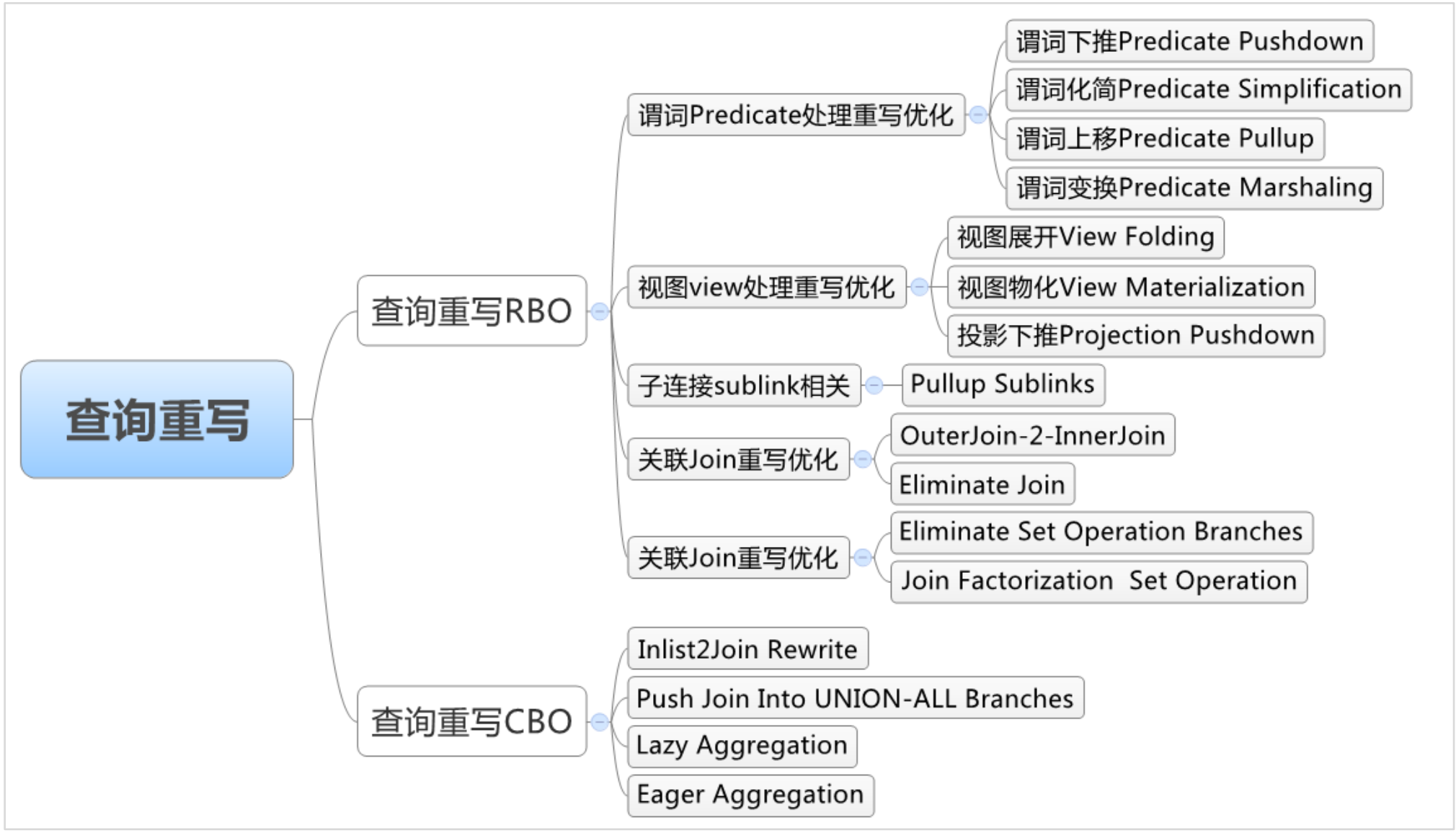
GaussDB查询优化

- 基于规则的查询优化（RBO）
  - 根据预定义的启发式规则对SQL语句进行优化
- 基于代价的查询优化（CBO）
  - 利用统计信息，通过基础和代价估计对SQL语句对应的候选物理计划进行代价估算，并从候选计划中选择代价最低的物理计划作为最终的执行计划
- 基于人工智能的查询优化（ABO）
  - 收集执行计划的特征统计信息，借助人工智能模型估计基数和代价，并选择高效的执行计划
  - 智能基数估计和代价估计
  - 智能计划选择和生成

GaussDB查询重写



GaussDB查询重写



GaussDB代价估计

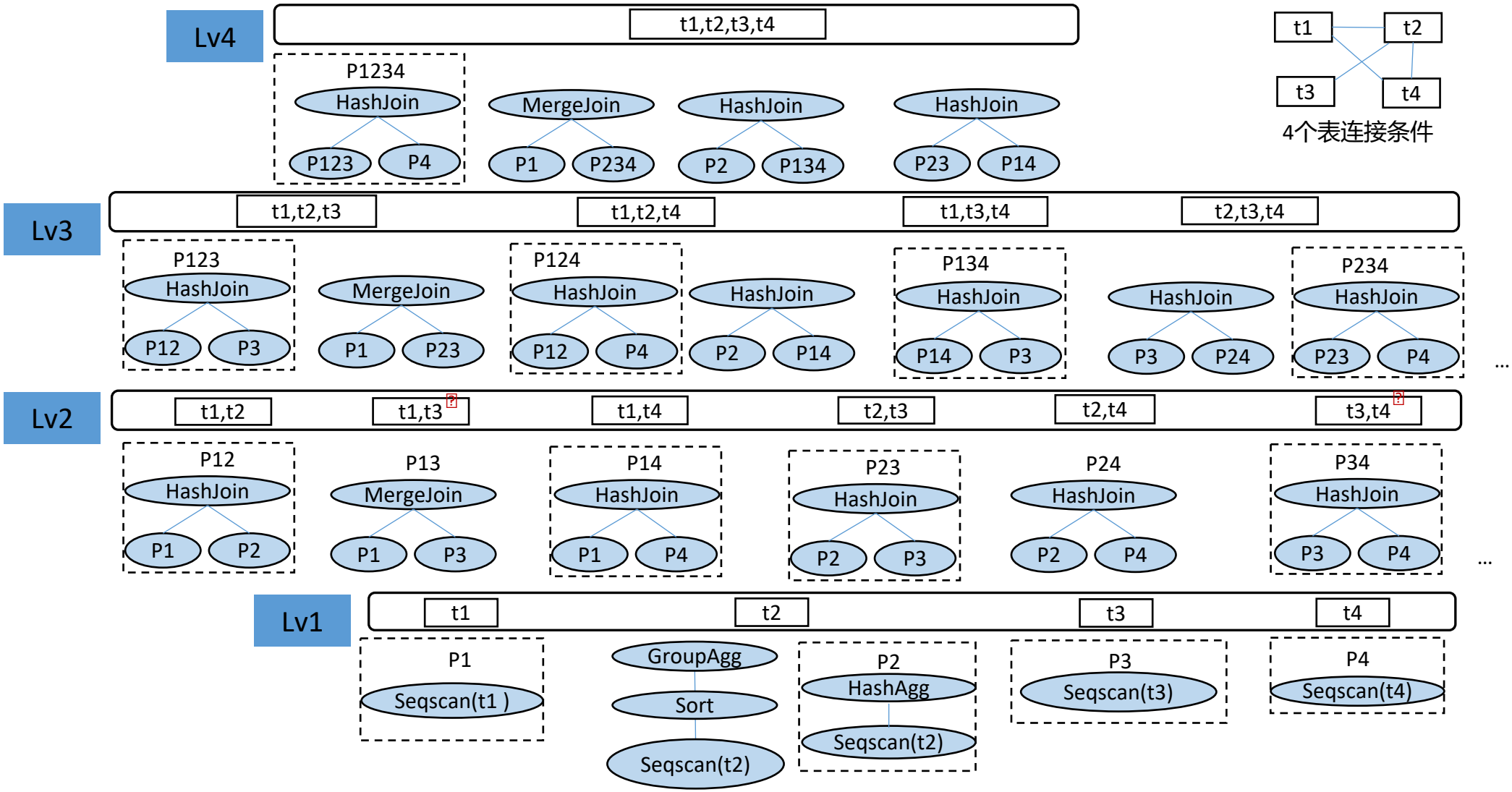
- 总代价=IO代价+CPU代价+通信代价
  - 处理页面产生IO代价
  - 处理元组（表达式计算）产生CPU代价
  - 分布式数据库在不同节点传输数据产生通信代价
- 统计信息
  - 统计信息描述了用户表中数据的分布特征为后续行数估算、代价估算提供数据基础。
  - Table-Level表级别统计信息：tuples总元组数、page总页面数
  - Column-Level列级别统计信息：Distinct值、NullRatio、MCV（Most Common Value）、直方图
  - 扩展统计信息：Multi-Column Statistics多列统计信息、View-Statistics视图统计信息

GaussDB计划生成

- 物理计划选择
- 自底向上模式
  - 拆解逻辑计划，确定每个表的物理扫描算子，和物理连接算子（以及其他算子），从而最终确定完整的物理执行
- 自顶向下模式
  - 通过采用自顶向下的方法遍历逻辑计划，结合动态规划、代价估算和分支界定技术，逐步确定每个逻辑算子对应的物理算子，获得最优的物理计划
- 随机搜索模式
  - 表较多的情况下通过随机化的方法对计划进行搜索，获得次优的执行计划

GaussDB主要采用自底向上与随机搜索相结合的方式

GaussDB路径搜索



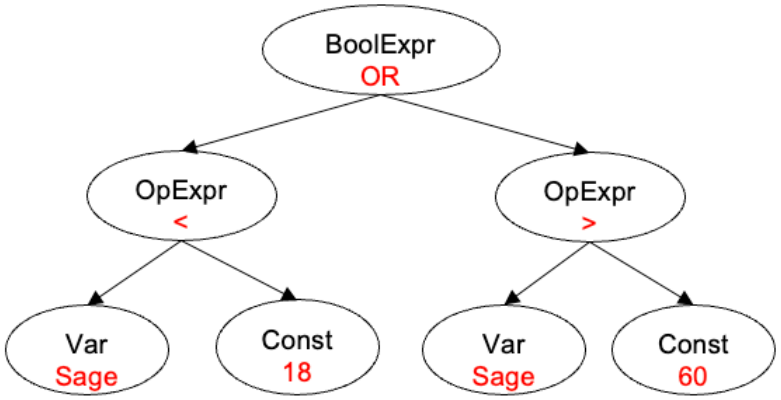
GaussDB查询执行

- 火山模型
- 并行执行
  - 并行计划生成
    - 生成每一层路径时，都增加一个并行路径，最终根据代价选择最优路径。
  - 并行计划框架
    - 处理不同节点和不同线程间数据分配
    - 对于扫描算子，增加一条并行的路径，计算并行的启动代价、数据分配代价以及算子执行代价，与串行的路径一起进行后续的路径选择
- 向量化执行
- 编译执行

GaussDB编译执行

- 为了进一步提升执行速度，GaussDB执行引擎采用LLVM、向量化引擎等多种技术，充分结合硬件技术高效执行
- 编译执行技术
  - LLVM Lib库，提供代码生成所需的功能接口
  - LLVM API层，封装部分LLVM Lib库函数实现优化功能，供LLVM实现层调用
  - LLVM实现层，优化LLVM IR函数，实现与原有执行器中的相同更高效的功能

SELECT Sno, Sname FROM Student WHERE Sage < 18 OR Sage > 60;



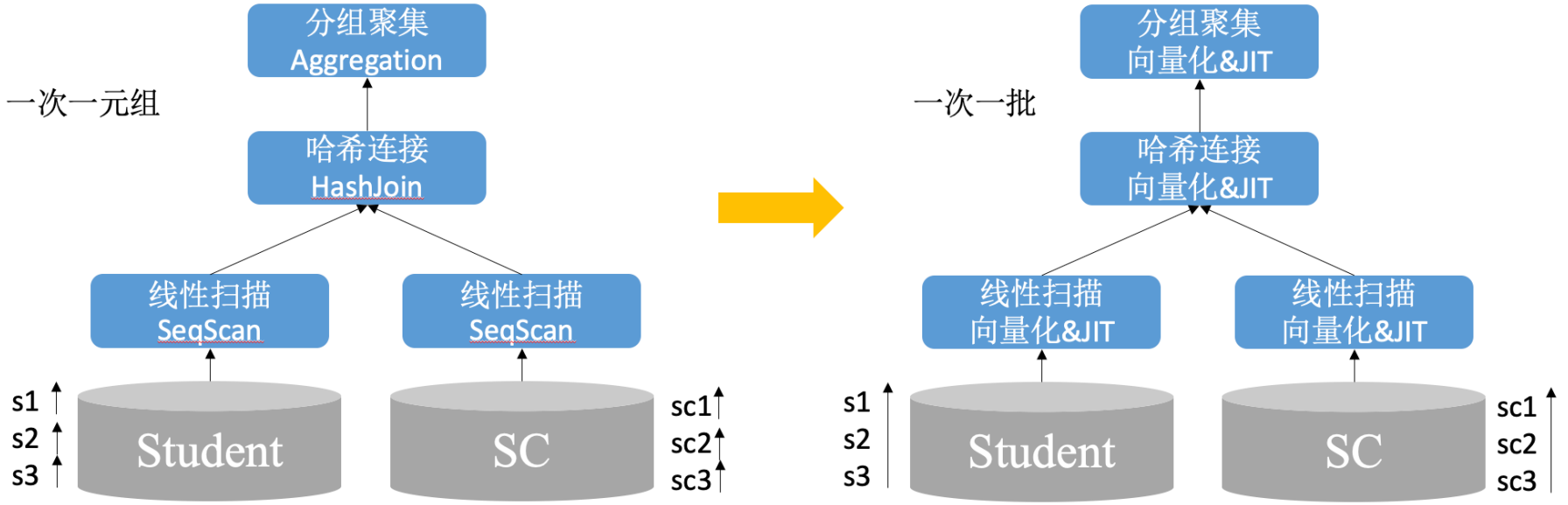
通用函数执行流程

Var	Const	OpExpr	Var	Const	OpExpr	BoolExpr
Sage	18	<	Sage	60	>	OR

编译执行代码生成函数的执行流程

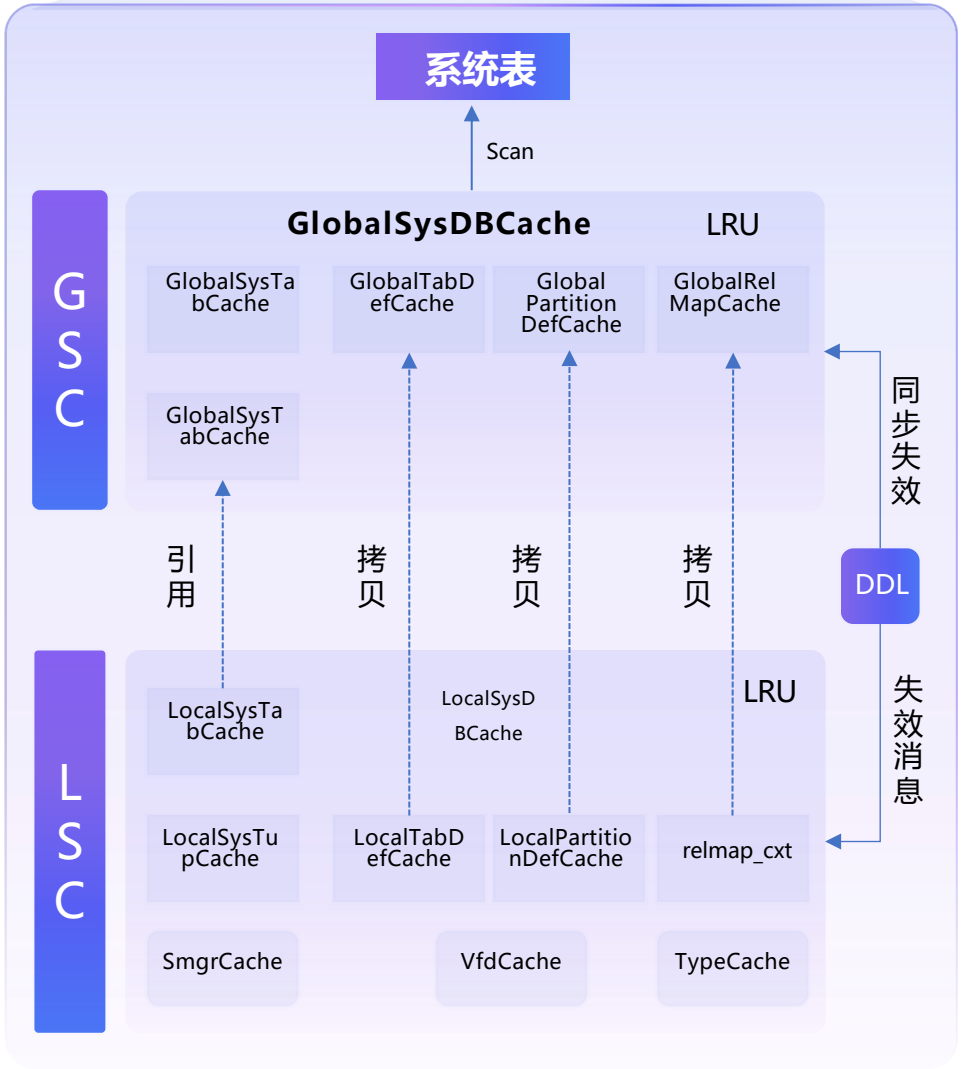
GaussDB向量化执行

- 向量化执行技术
  - 一次处理一批元组，大大减小了遍历执行节点的开销
  - 一次一批元组的数据处理方式为某些表达式计算的SIMD（单指令多数据）化提供了机会，进而带来性能的提升
  - 一次一批元组的处理方式天然对接列存，很方便在底层扫描节点批量装填向量化列数据

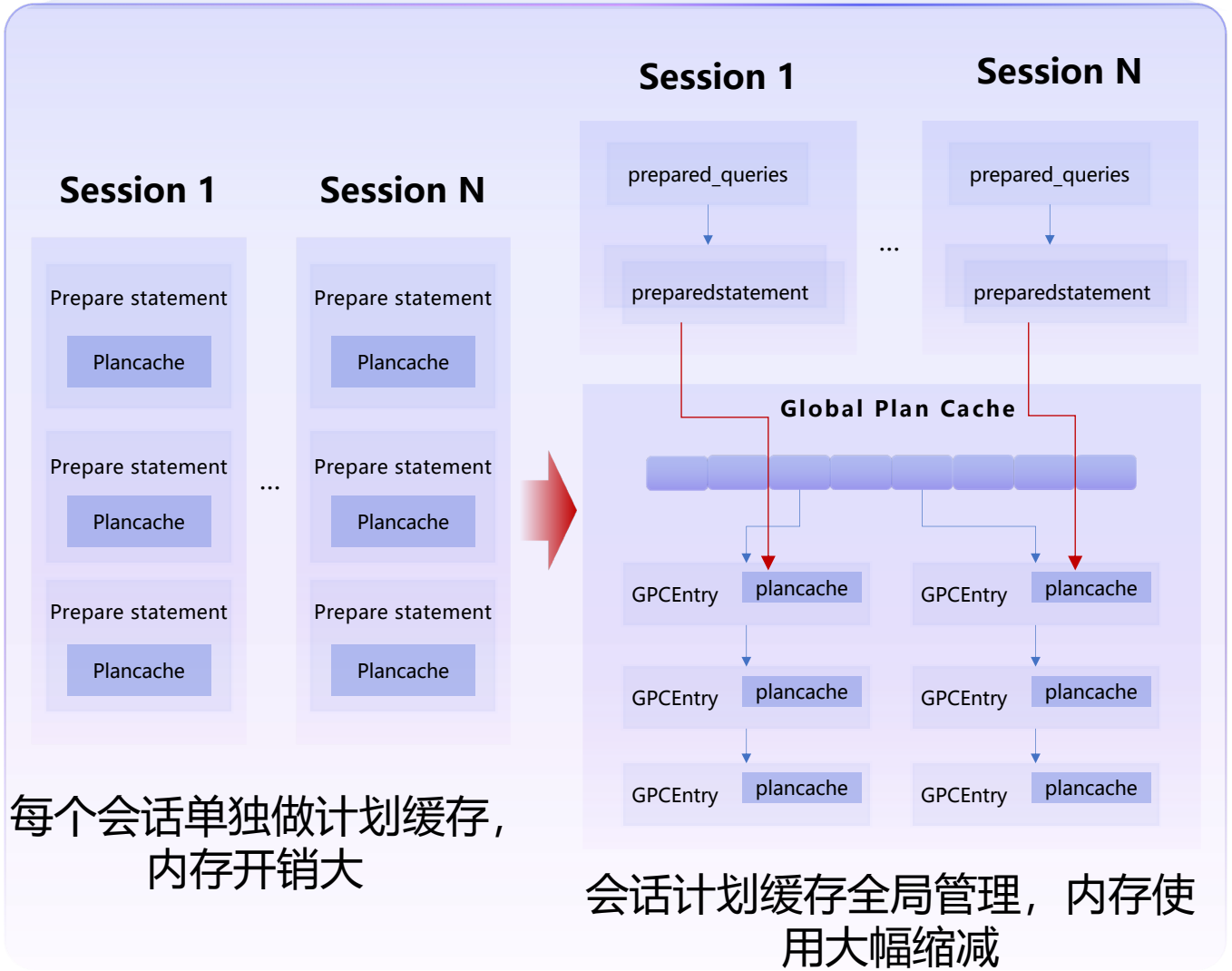


GaussDB大并发：全局共享数据结构

全局系统缓存（GSC）



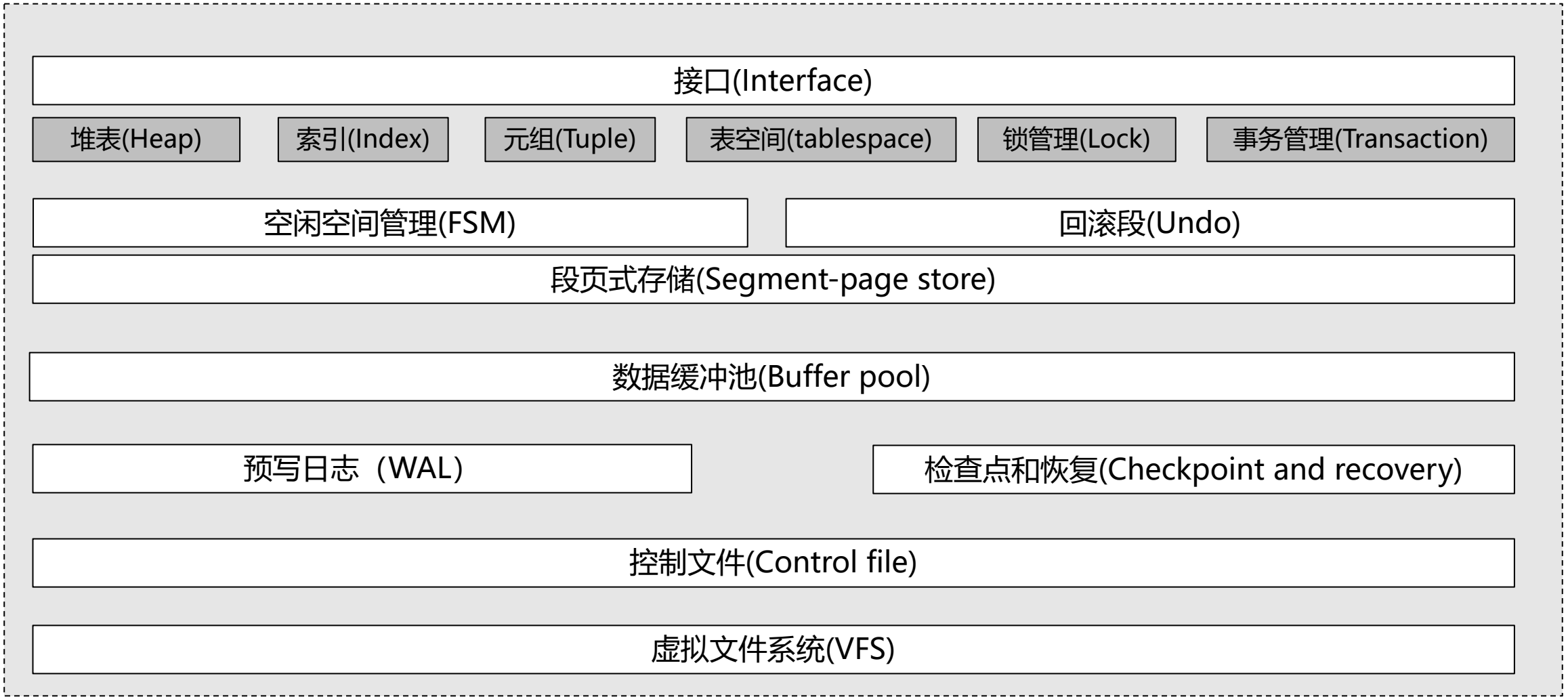
全局计划缓存（GPC）



目录

GaussDB存储概览

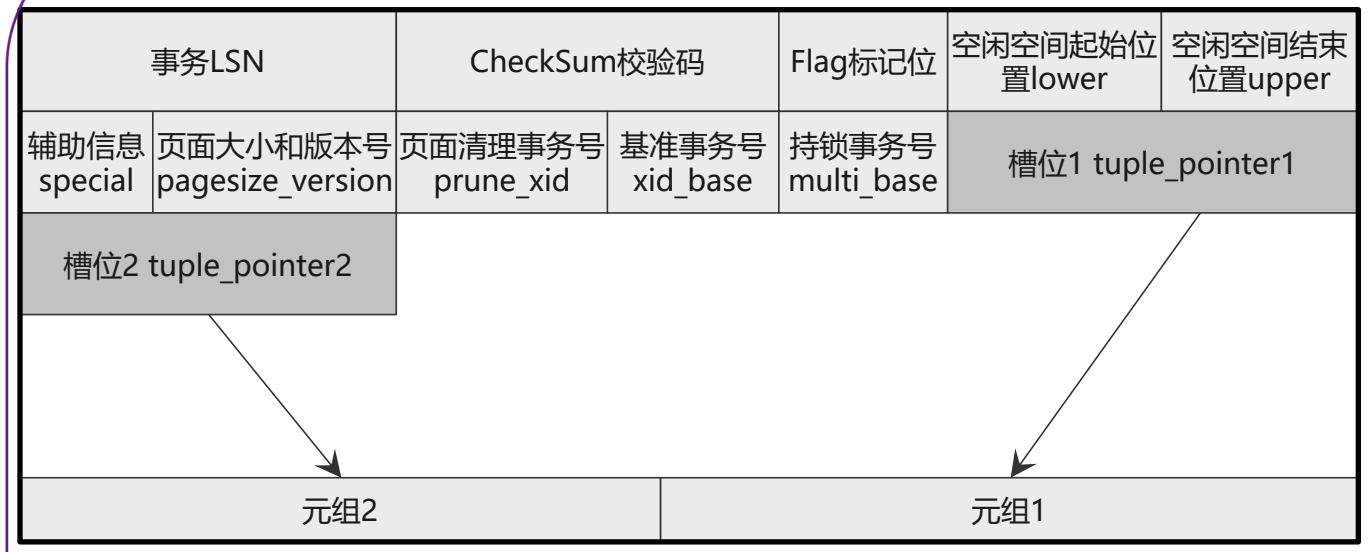
GaussDB存储概览



GaussDB行存储引擎—页面结构

- 行存储以页面为单位
  - 页头（HeapPageHeader）记录页面的公用信息以及关键标识；
  - 页头后放置指向元组的行指针（tuple_pointer），并向页面尾部扩展。
  - 行指针指向的元组（tuple），从页面尾部向页面头部延展
- 元组结构
  - 每个元组在系统中的唯一标识被称为CTID

堆表页面组织



GaussDB MVCC多版本

元组版本号分发管理

- 采用全局递增的事务号（作为一个元组的版本号，每个写事务都会获得一个新的事务号。
- 一个元组的头部会记录两个事务号xmin和xmax，分别对应元组的插入事务和删除（更新）事务。
- xmin和xmax决定了元组的生命期，亦即该版本的可见性窗口。

快照维护

- 快照是指在某一个特定的时刻点，数据库中所有事务的运行状态。
- 每个事务都会选择一个可见性判断的“时刻点”，在这个时刻点对应的快照中，正在执行的和尚未开始的事务，对其不可见；在这个时刻点，已经结束的事务，对其可见。
- 使用时间戳法来实现快照。

xmin状态 \ xmax状态	xmax对于查询可见	xmax对于查询不可见
xmin对于查询可见	记录不可见（先插入，后删除）	记录可见（先插入，未删除）
xmin对于查询不可见	不可能发生	记录不可见（未插入，未删除）

GaussDB行存多版本：追加更新

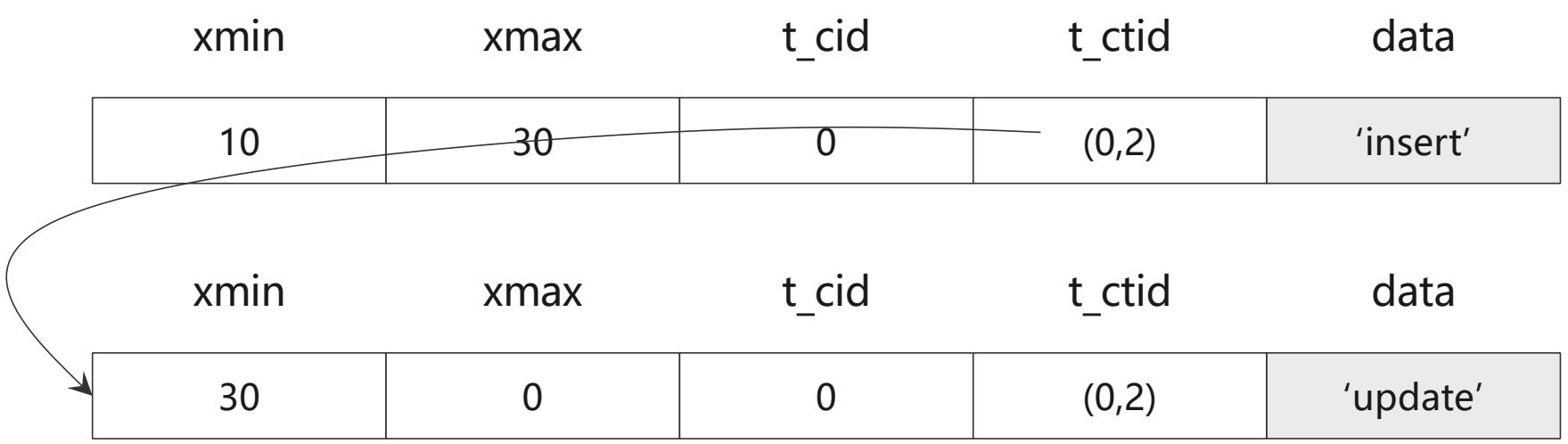
- GaussDB行存储的多版本支持追加更新，即在更新时并不是就地更新，而是在原有页面中保留上一个版本，然后在这个页面（如果空间不够会在新页面中）创建一个新的版本，来进行历史版本的累积更新。例如：

xmin	xmax	t_cid	t_ctid	data
10	0	0	(0,1)	‘insert’

- 假设在一个事务号xid为10的事务中，插入数据’ insert’ ，该行数据落入编号为0的数据页面上
  - 该行结构如上图所示。
  - xmax为0，说明该记录为有效记录。

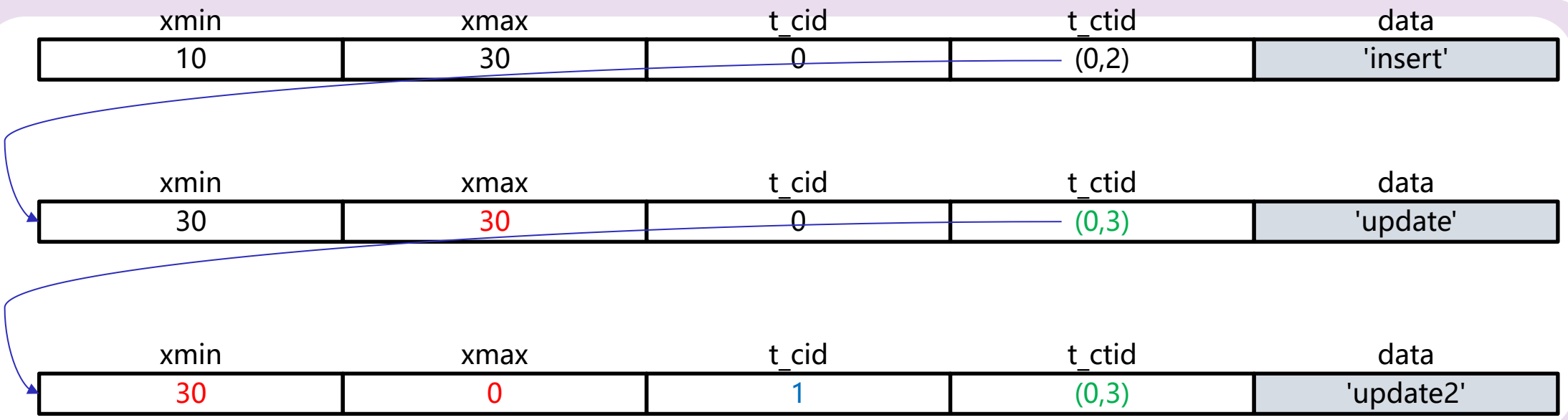
GaussDB行存多版本：追加更新

- xid=30的事务中对该行做两次更新操作，第一次更新会发生如下变化：
  - 原有行xmax更新为30，使得原有行失效。
  - t_ctid更新为新的ctid，指向新版本的行。



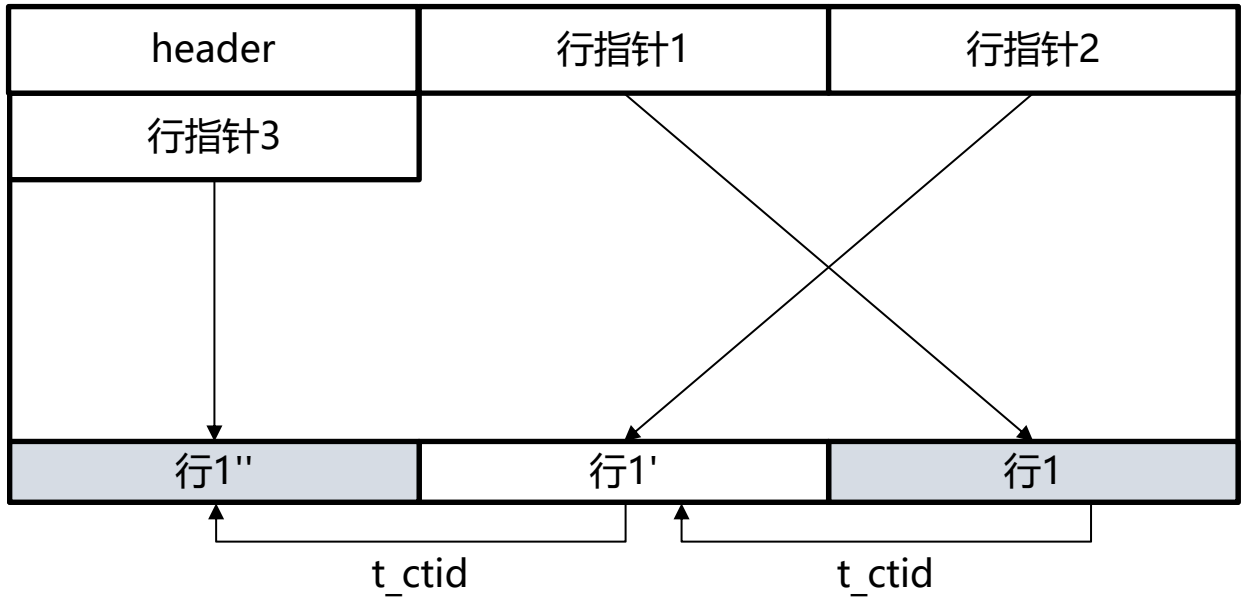
GaussDB行存多版本：追加更新

- 第二次更新会发生如下变化：
  - 第二个版本也变为历史版本
  - 通过t_ctid指向最新版本的记录
  - 第二个版本的记录在一个事务中被删除， xmin、xmax均为30
  - 第三个（最新）版本的xmin仍为30，但事务内的第二次更新，操作序号cid从0增加为1



GaussDB行存多版本：追加更新

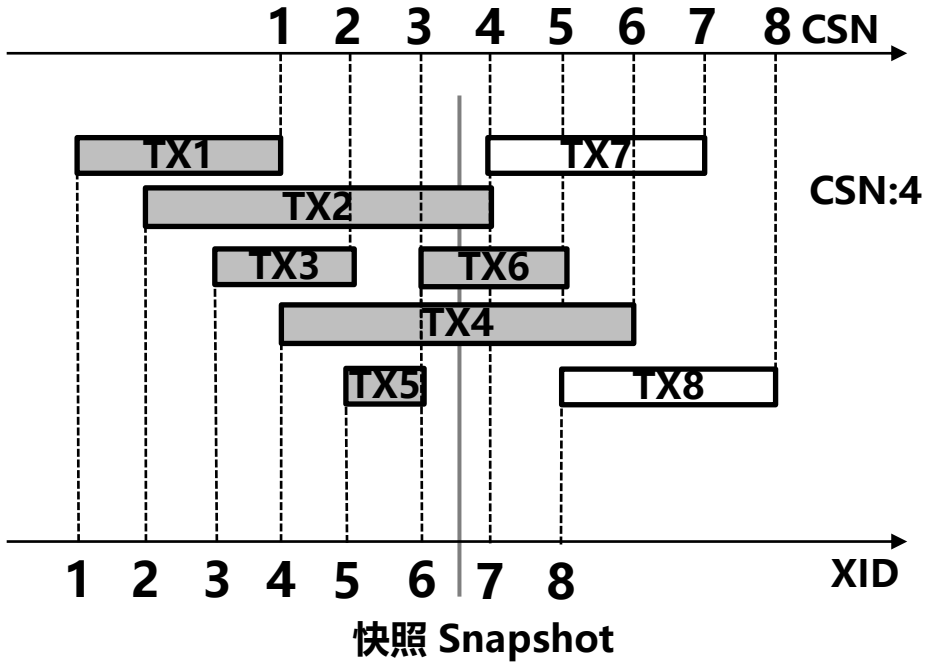
- 经过两次更新后的页面如下图所示。
  - 不同于原地更新，三次操作产生的同一行的3个版本均存储在该页面中。



- 同一页面中会存有同一条数据的不同版本
  - 在读写这些不同版本时互不冲突，具备良好的并发性能。

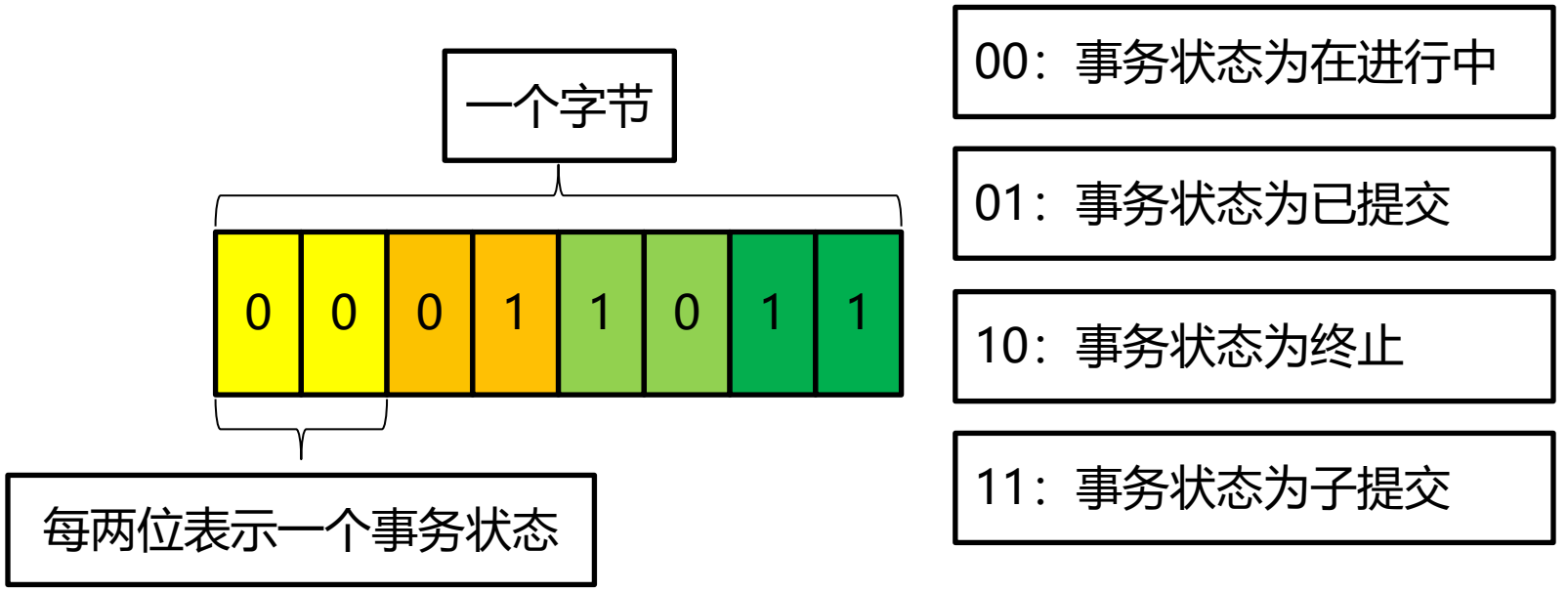
GaussDB行存多版本：追加更新

- 为了实现多版本的并发控制，还需要为每个事务分配事务号，并设计能够确定快照与事务之间可见关系的机制
  - 每个事务有一个单独的事务状态存储区域，记录了该事务的状态信息和提交顺序号（CSN）
  - 图中每个非只读事务的事务号为XID，在事务提交时会增加CSN取值

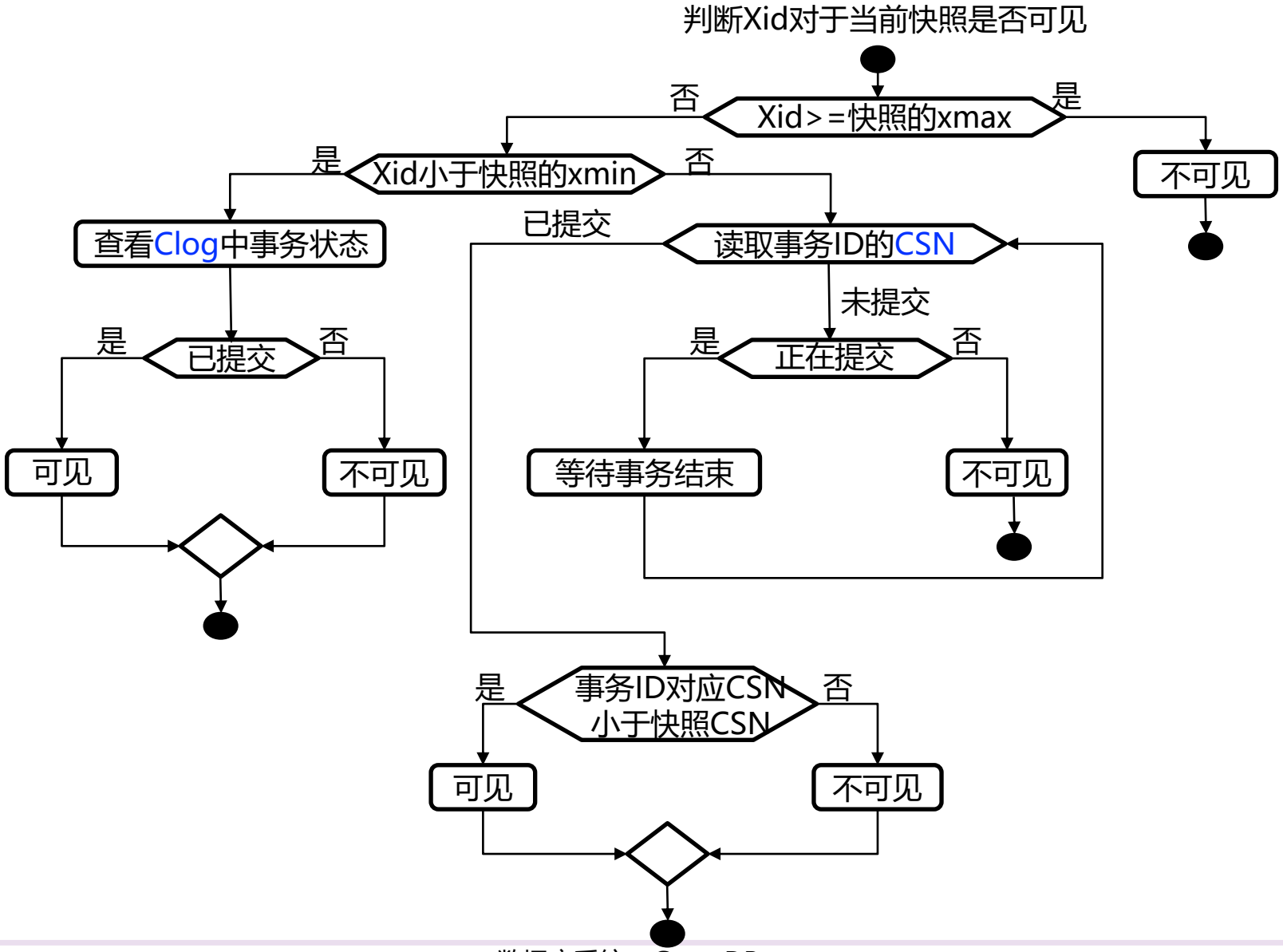


GaussDB行存多版本：追加更新

- CSNLog：记录XID与CSN的映射关系，为每个事务生成一个唯一递增的CSN，用于将事务与其可见性进行关联
- Clog：记录事务ID的运行状态：运行中/提交/回滚
- 当事务结束后，使用CLOG记录是否提交，使用CSNLOG记录该事务提交的序列

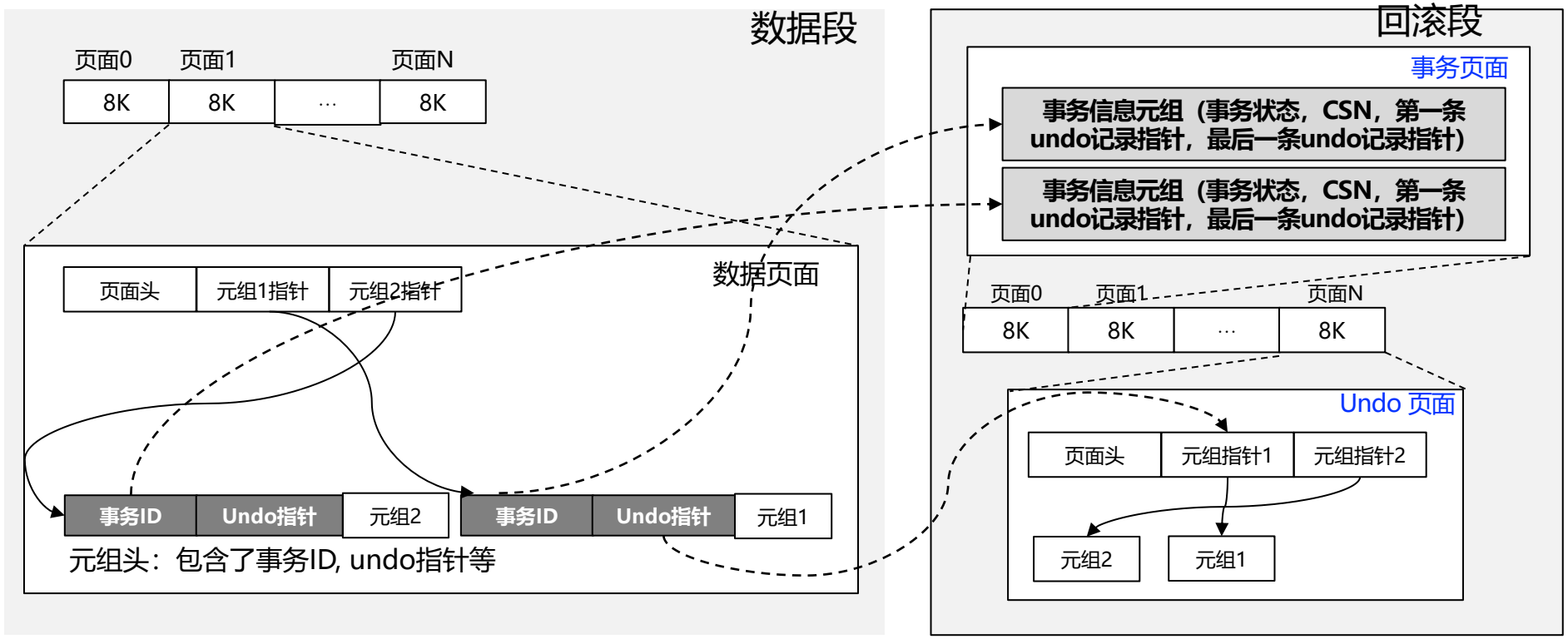


GaussDB行存多版本：追加更新



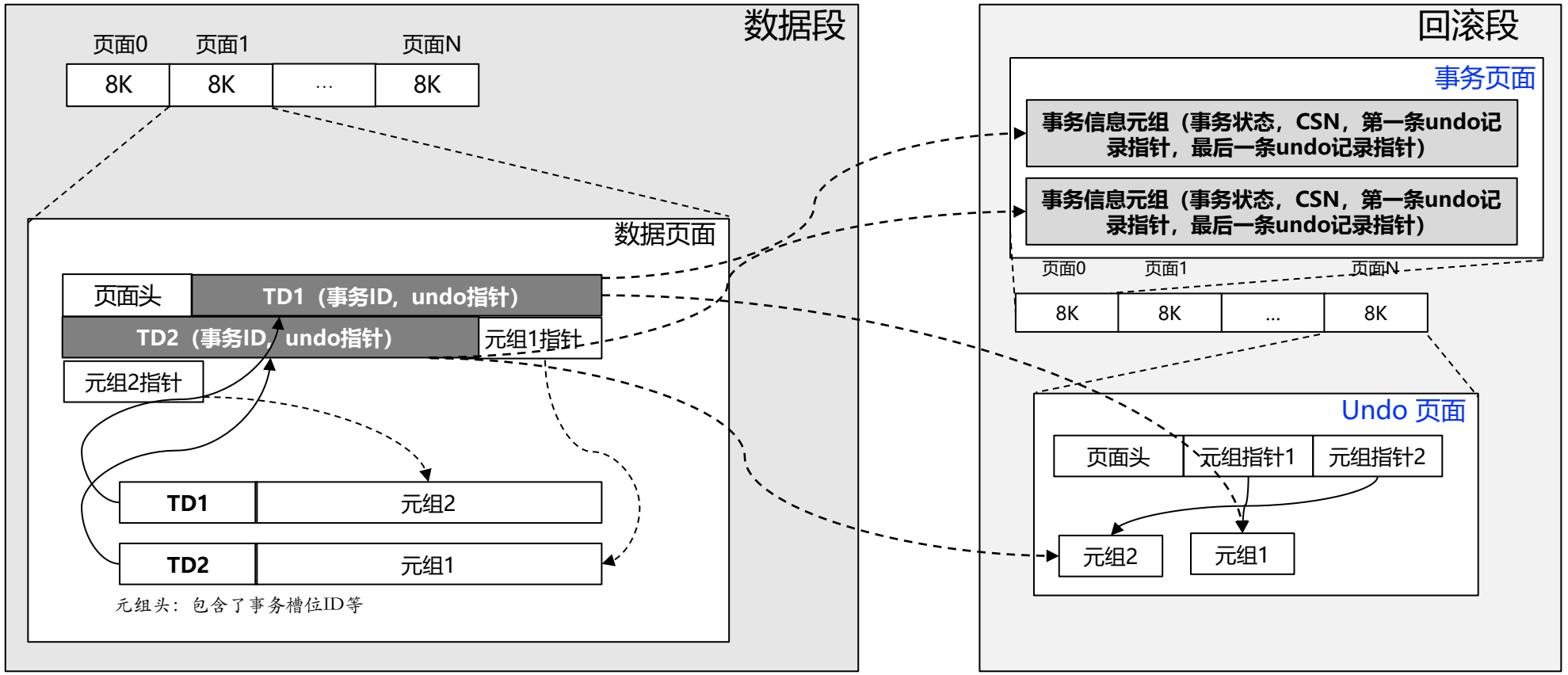
GaussDB行存多版本：原位更新

- 追加更新行存储将老版本元组和新版本元组存储在统一空间
  - 老版本回收困难，而且可能需要扫描所有数据来清除老版本信息
- 原位更新行存通过单独的回滚段来存储老版本数据
  - ① 最新版存在数据段，并通过指针指向存放于回滚段的老版本



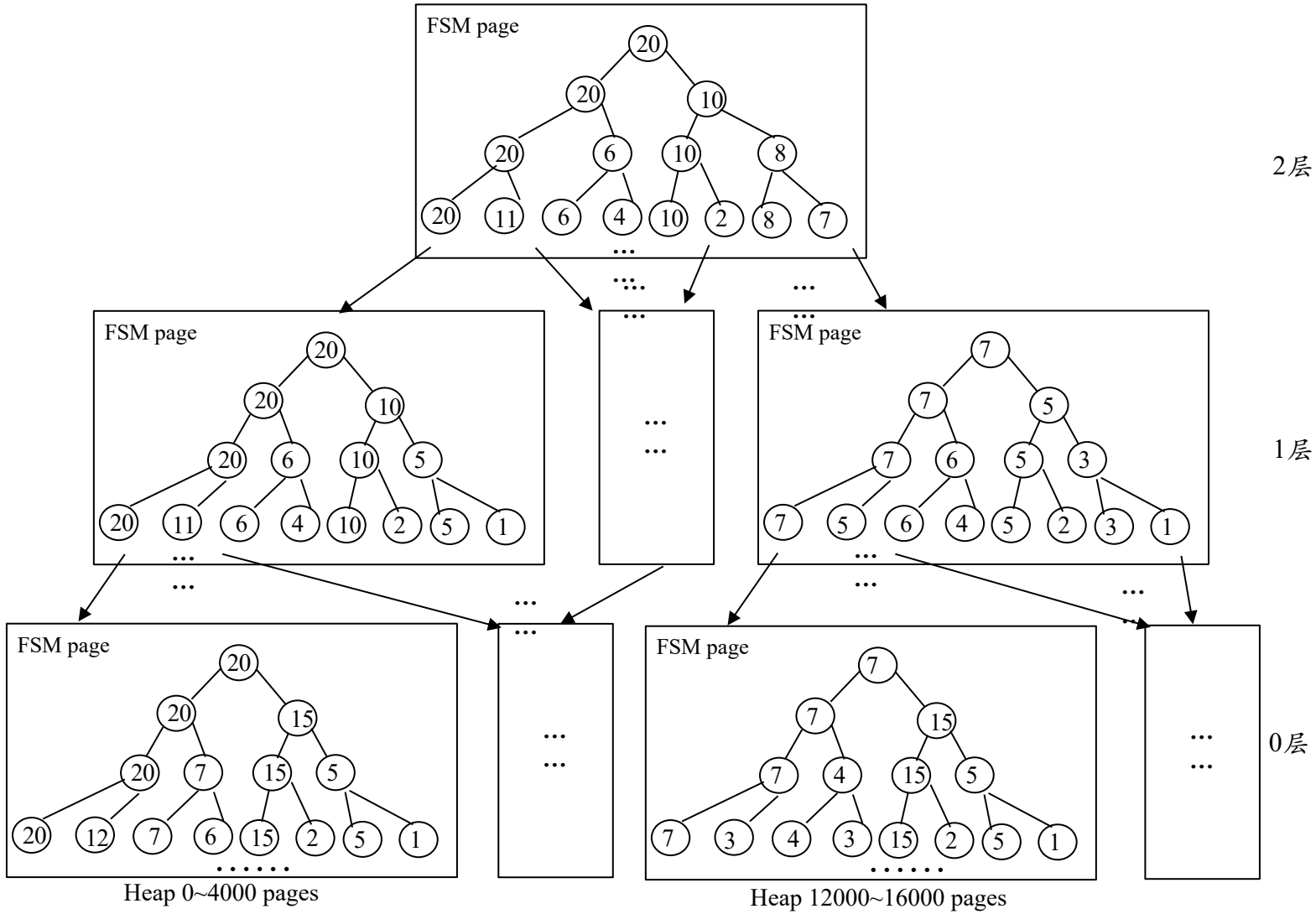
GaussDB行存多版本：原位更新

② 元组不直接存储老版本的指针，而是通过事务槽号来维护时间戳信息，从而节省存储空间



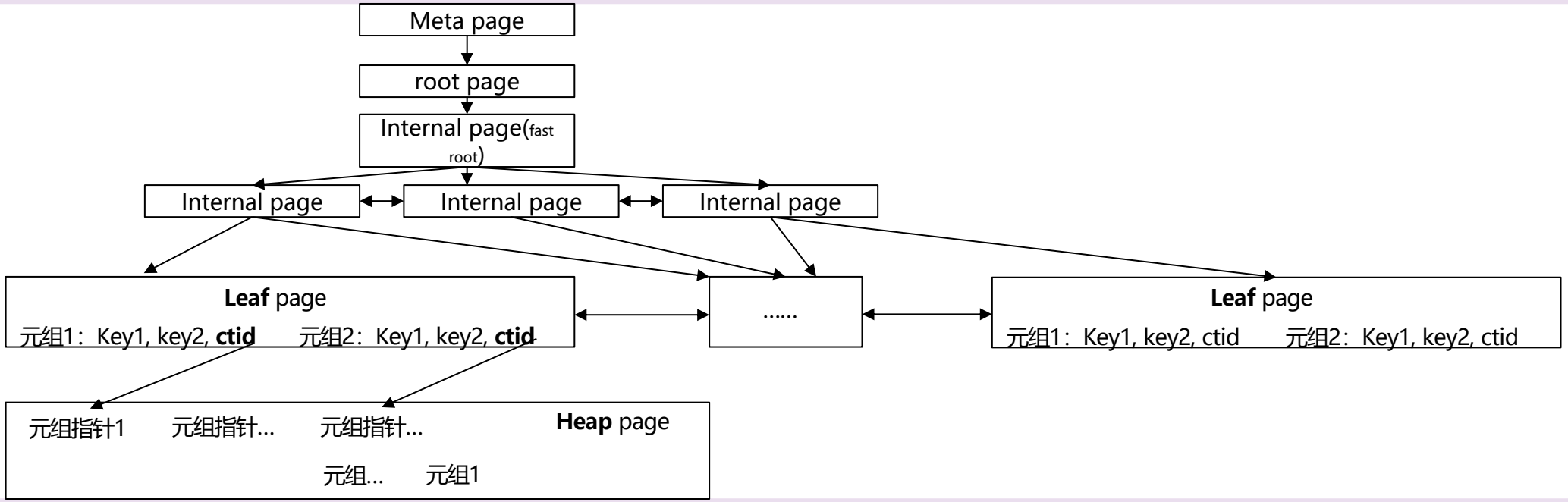
GaussDB行存储引擎—空闲空间管理

- 在数据插入的时候需要高效找到空闲空间足够的页面
- 插入数据后还要能够高效更新页面的空闲空间大小
- 采用三层大根堆的数据结构来管理空闲空间FSM(free space map)



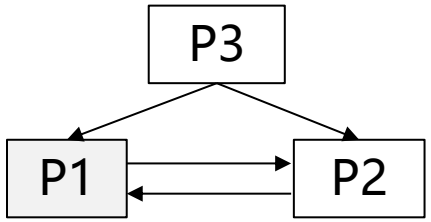
GaussDB行存储引擎—B+树索引

- 行存储索引采用Blink-tree的并发优化算法
- 索引叶子页面存储事务信息，非叶子页面不存储事务信息
- 索引空闲空间管理包括空页面（被删空）的回收重用和基于访问页面时按需整理页面空间。因为插入索引数据都是有序插入到叶子页面的，按需整理页面即可。



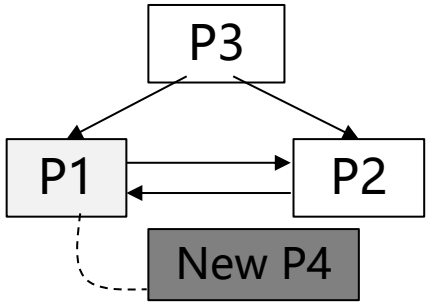
GaussDB行存储引擎—B+树索引

索引插入



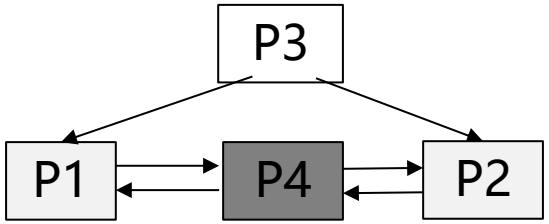
1

查找到待插入的页面P1，
持有页面P1写锁



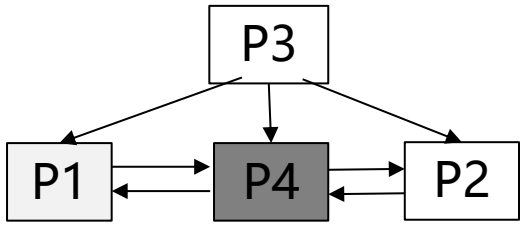
2

插入P1页面，空间不够，
分裂页面P1；
创建New P4页面并持有
页面写锁



3

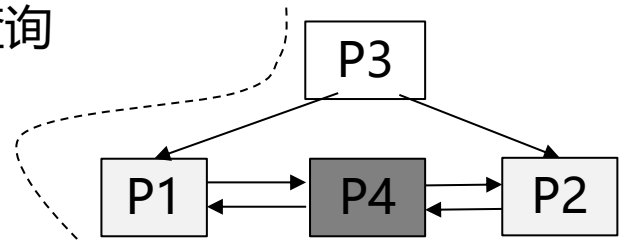
持有页面P2的写锁(修改左
兄弟指针为P4)
释放P2的写锁；
修改P1, P4左右兄弟指针



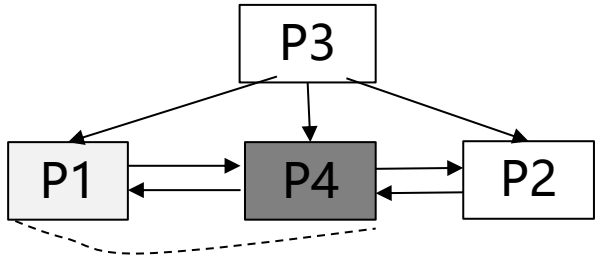
4

持有页面P1, P4的写锁
持有父页面P3的写锁，插入指
向页面P4的索引元组
释放P4, P1, P3的页面写锁。

索引查询



从P3获取到满足条件的叶子页面P1; 释放P3页面读锁
读取P1获取页面读锁，P1正在分裂，加页面读锁等待

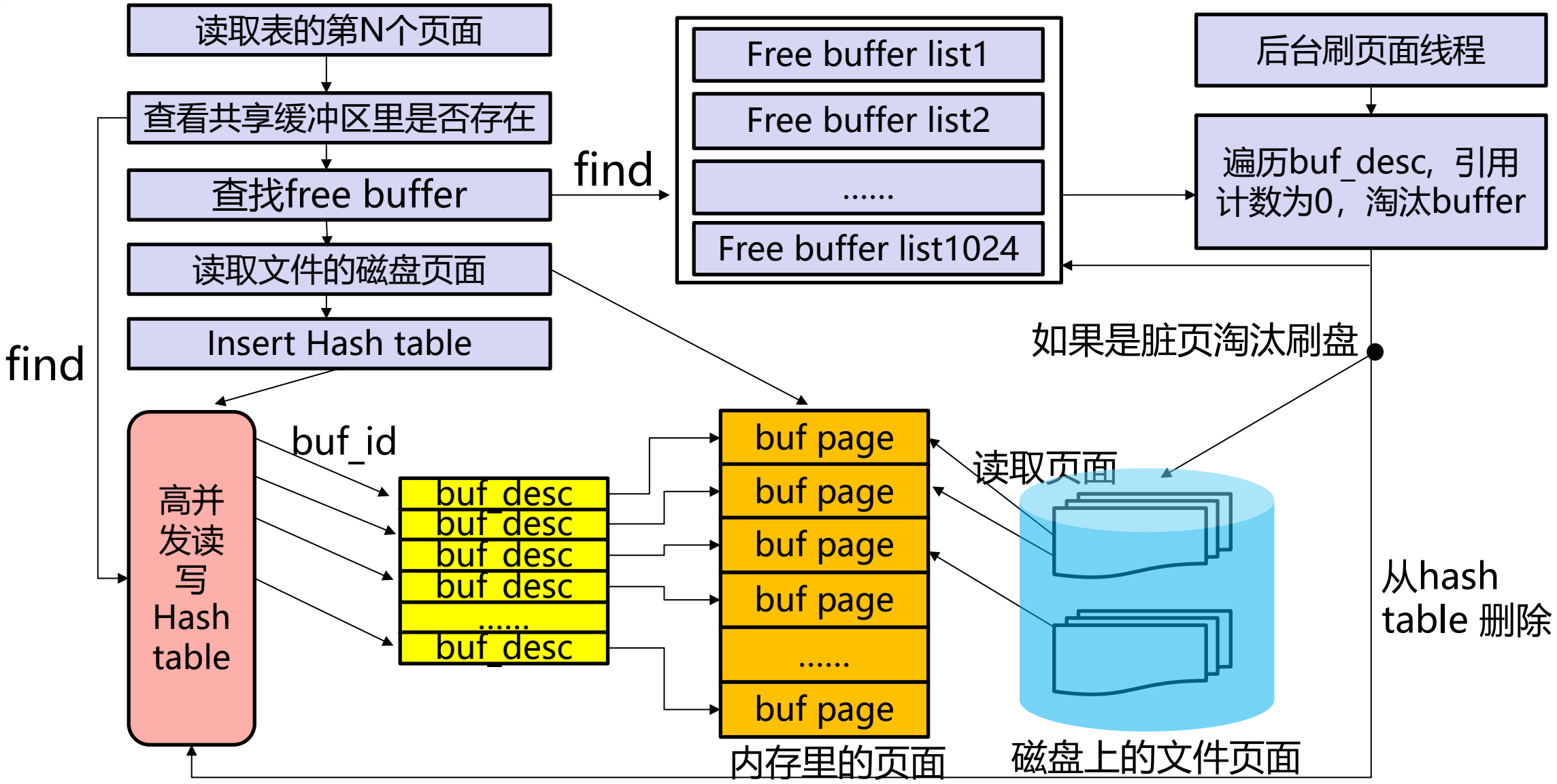


持有P1页面的读锁后，P1发生过分裂，可能满足条件的
索引元组迁移到了P4，释放P1的读锁，继续遍历P4。

GaussDB行存储引擎—缓冲区

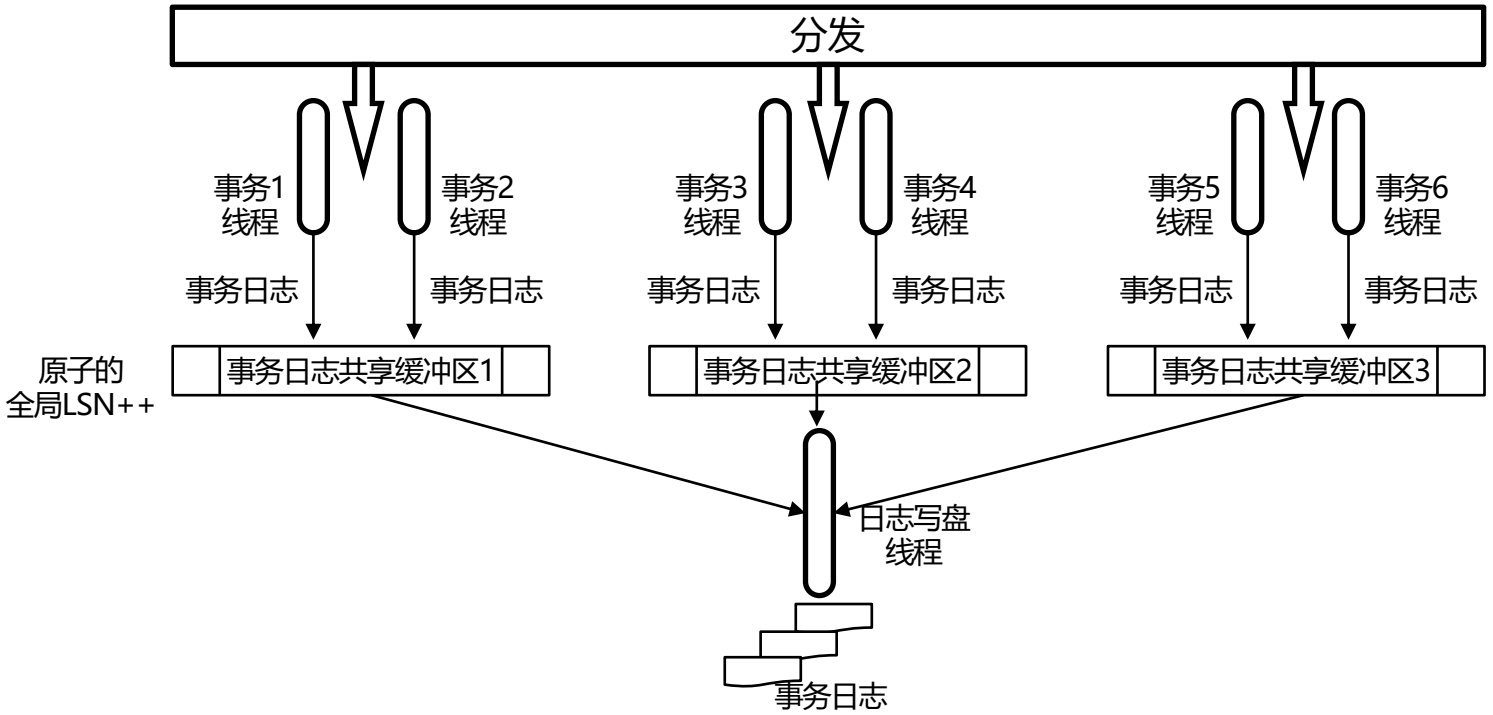
- GaussDB对事务的读写请求，都会被先传递至共享缓冲区
- 请求页面利用Ring Buffer操作批量读、批量写、以及页面清理，会先在缓冲区搜索该页面，如未命中，则获取一个空的槽位（可能需要淘汰掉缓冲区中不常用的页面），再与文件系统进行交互将所需页面读到槽位中，加锁并使用。
- 高并发Hash table：采用分区锁避免读写hash table冲突
- Buffer页面的冷热管理
  1）buffer ring策略：避免顺序扫描读取页面导致大量淘汰温热buffer
  2）never evict策略：基于规则避免页面被淘汰
  3）LRU策略：识别冷热页面，提高buffer命中率
- 空闲Buffer的高效查找算法
- Buffer pool在线弹性扩缩

GaussDB行存储引擎—缓冲区



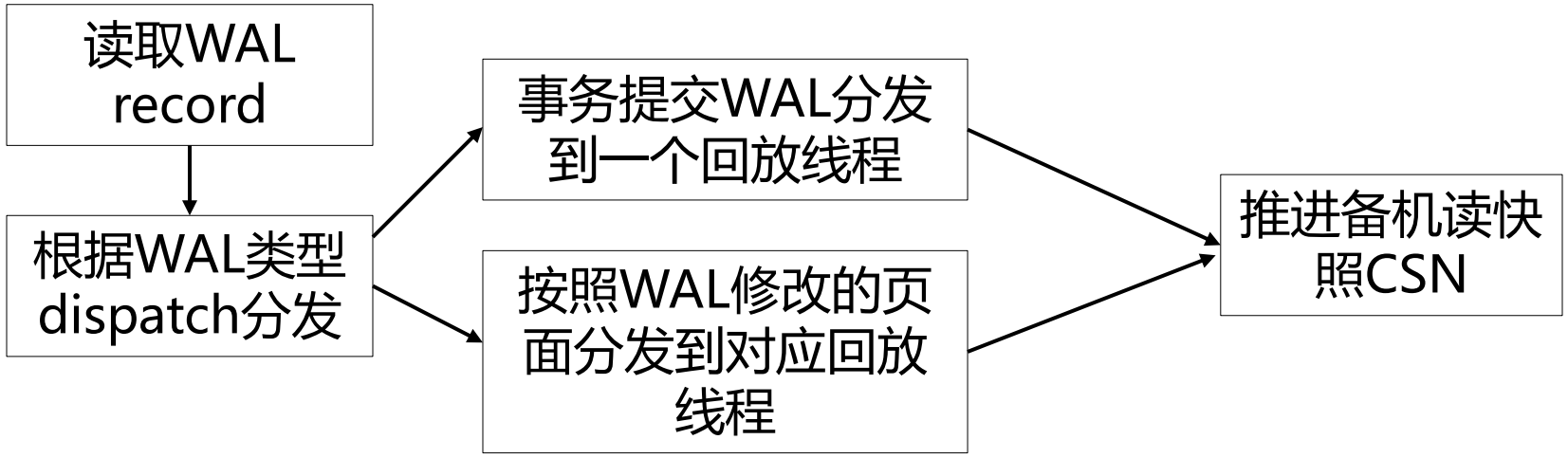
GaussDB行存储引擎—并行日志系统

- 事务提交仅保证事务相关的预写日志WAL持久化完成
  - WAL模块在内存中维护了WAL缓冲区用于各个事务的WAL日志并发写入缓存区，提高性能
  - 后台刷WAL日志线程需要顺序从WAL 缓冲区读取WAL日志完成持久化
  - 在每个事务提交之前，需要判断已经刷新的WAL LSN是否已经包含了本事务的WAL



GaussDB行存储引擎—备机支持读

- 主备通过Raft复制WAL, 备机读取WAL记录并按照页面并行恢复
- 备机按照WAL CSN顺序回放，同时推进备机读的快照CSN，保证事务的因果序。
- 针对清理页面垃圾版本空间的WAL记录回放：
  ① 主机计算可回收CSN时间点（考虑备机的读快照CSN），会影响主机垃圾版本回收效率、影响性能；
  ② 备机回放WAL时，维护多版本，根据备机的最老快照读决定回收。



GaussDB列存储引擎

- 列存引擎的基本单位是压缩单元（CU），即表中一列的一部分数据组成的压缩数据块。整个表被按列划分为若干个CU
- 支持差分编码、游程编码、字典编码、LZ4、zlib等压缩算法，并能为每个列自适应地选择压缩方式

CU_ID	Min	Max	Row_Count	CU_Mode	Size	CUPtr
C0	100	102	1000	RLE	8000	...
C1	aaa	aab	1001	LZ4	8008	...
C2	9	10	1000	Delta | RLE	8000	...
C3	10	11	1002	Delta | RLE	8016	...
...	min	max	1000	...	8000	...
...	NULL	NULL	1002	...	8016	

GaussDB列存储引擎—页面结构

- 为了管理列存表的CU，列存储引擎使用压缩单元描述符（CUDesc）表来记录CU的元信息
  - CUDesc的一行对应一个CU，记录了CU的事务时间戳信息、大小、存储位置等信息

- CU文件结构

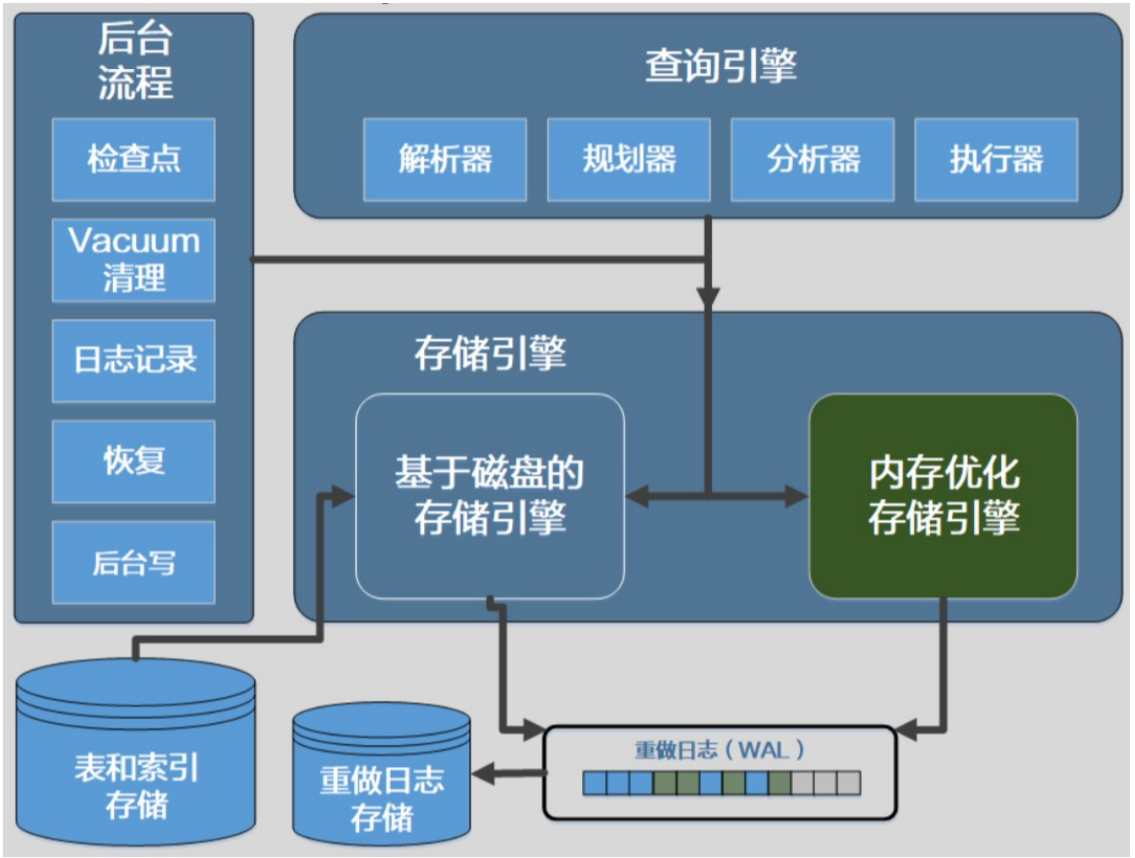
CU CRC 校验码	CU magic	CU info 属性	压缩后NULL值位图位数	压缩前数据长度	压缩后数据长度	压缩后NULL值位图内容	压缩后数据内容

GaussDB列存储引擎—索引设计

- 列存储的B+树索引
  - 与行存类似，列存B+树的索引页面上存储键到ctid（页面号，元组槽位）的映射，ctid记录的是CU的编号，还需要通过CUDesc进一步查找
- 列存储的稀疏索引
  - CUDesc存储每个CU中数据的最小值和最大值。读CU前可以根据查询条件进行判断，如不在不在最小值和最大值之间，则可以不读取该CU，大量减少I/O的开销
- 列存储的聚集索引
  - 不同CU间最小值和最大值的区间有大量交集时，稀疏索引可能无法提升效率
  - 聚簇索引对部分区间内的数据排序（一般区间会包含多个CU），由此减小CU之间数据的交集，使CU内部的数据有序，提升CU文件本身的压缩效率。

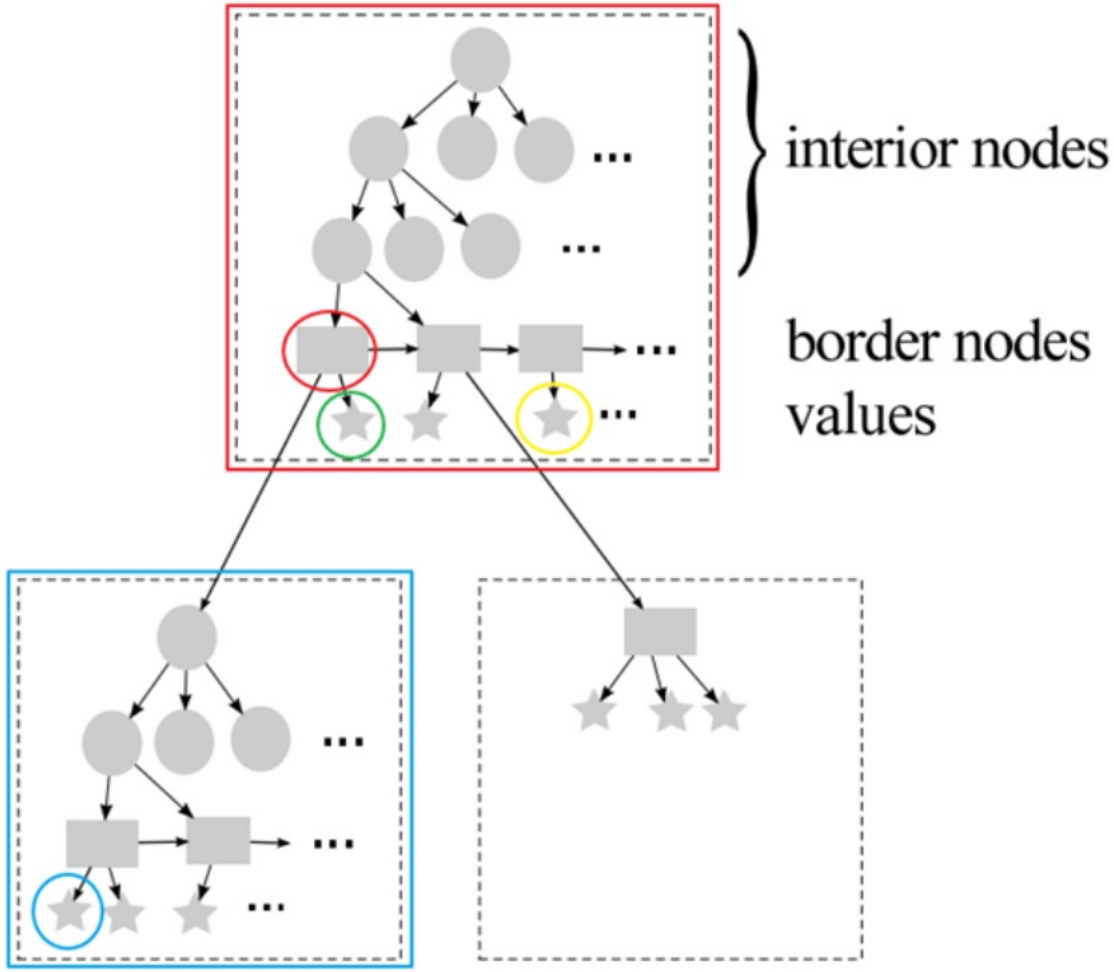
GaussDB内存引擎

- 全内存态的数据存储与事务处理
- 低延迟（Low Latency）：提供快速的查询和事务响应时间。
- 高吞吐量（High Throughput）：支持峰值和持续高用户并发。
- 高资源利用率（High Resource Utilization）：充分利用硬件。
- 具备并行持久化、检查点、高可靠等能力



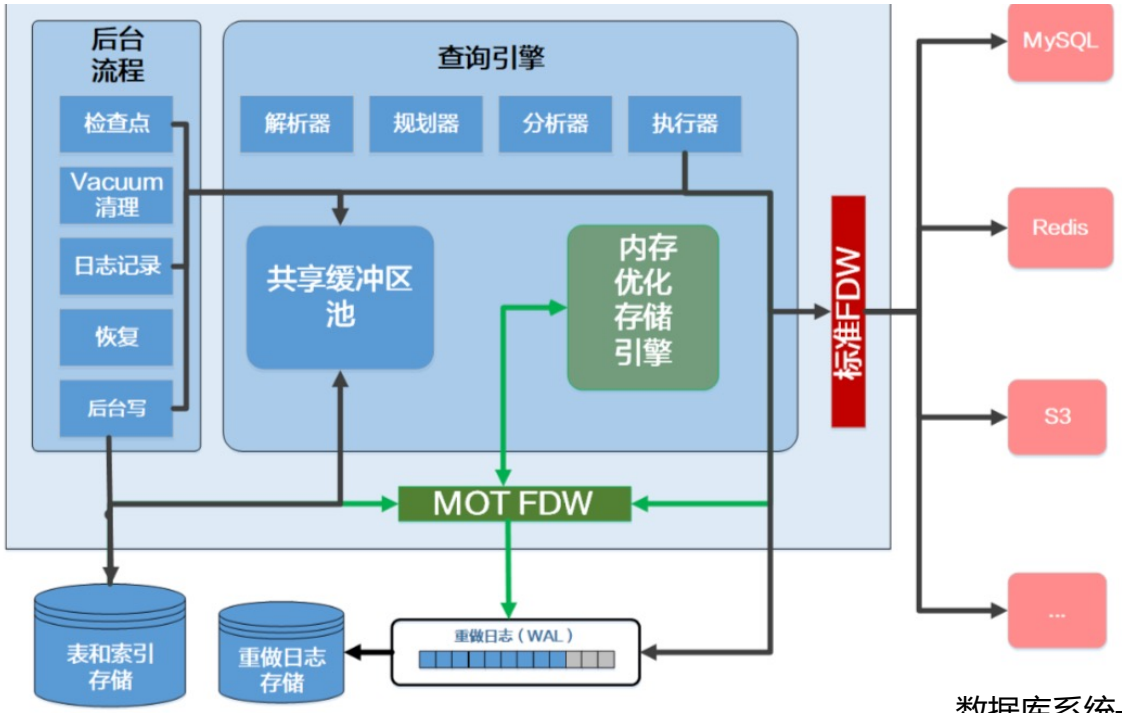
GaussDB内存引擎

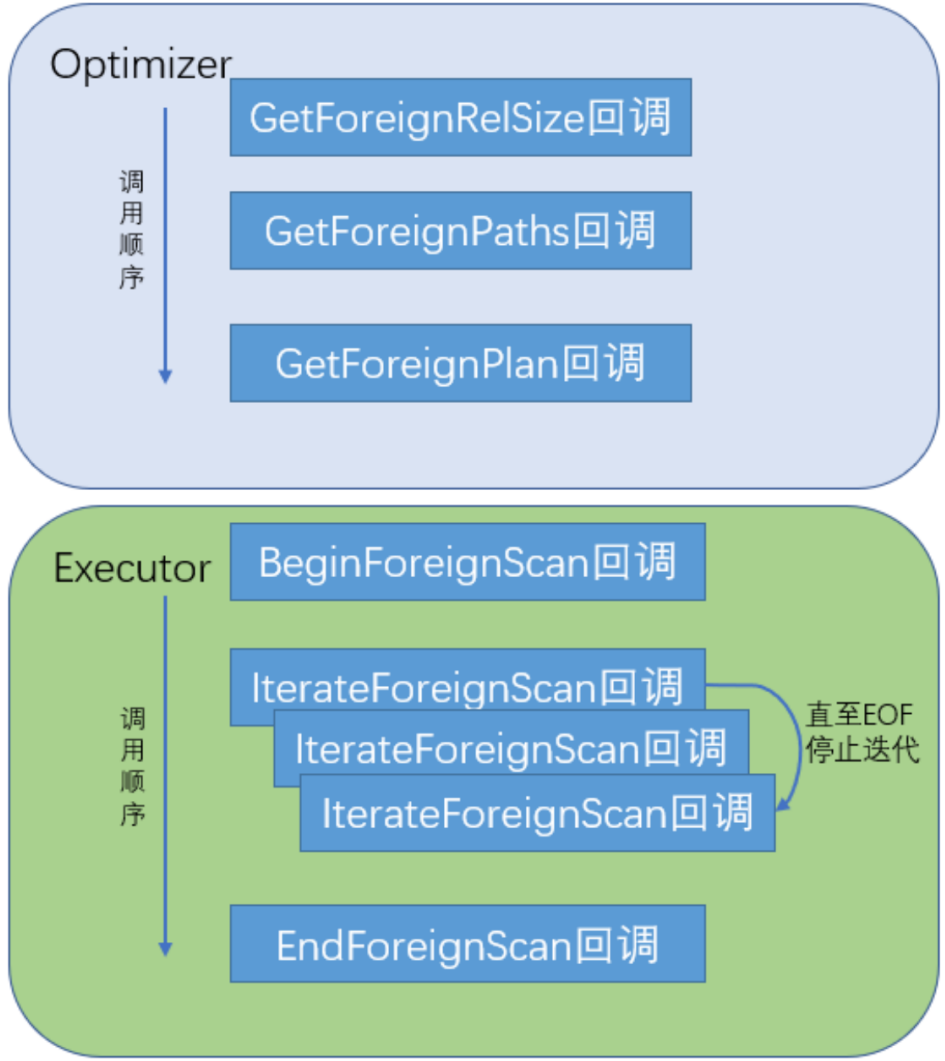
- 内存优化数据结构
- 免锁事务管理
- 免锁索引
- NUMA-aware的内存管理
- 高效持久性
- 高SQL覆盖率和功能集
- 编译执行
- MOT和普通表的无缝集成



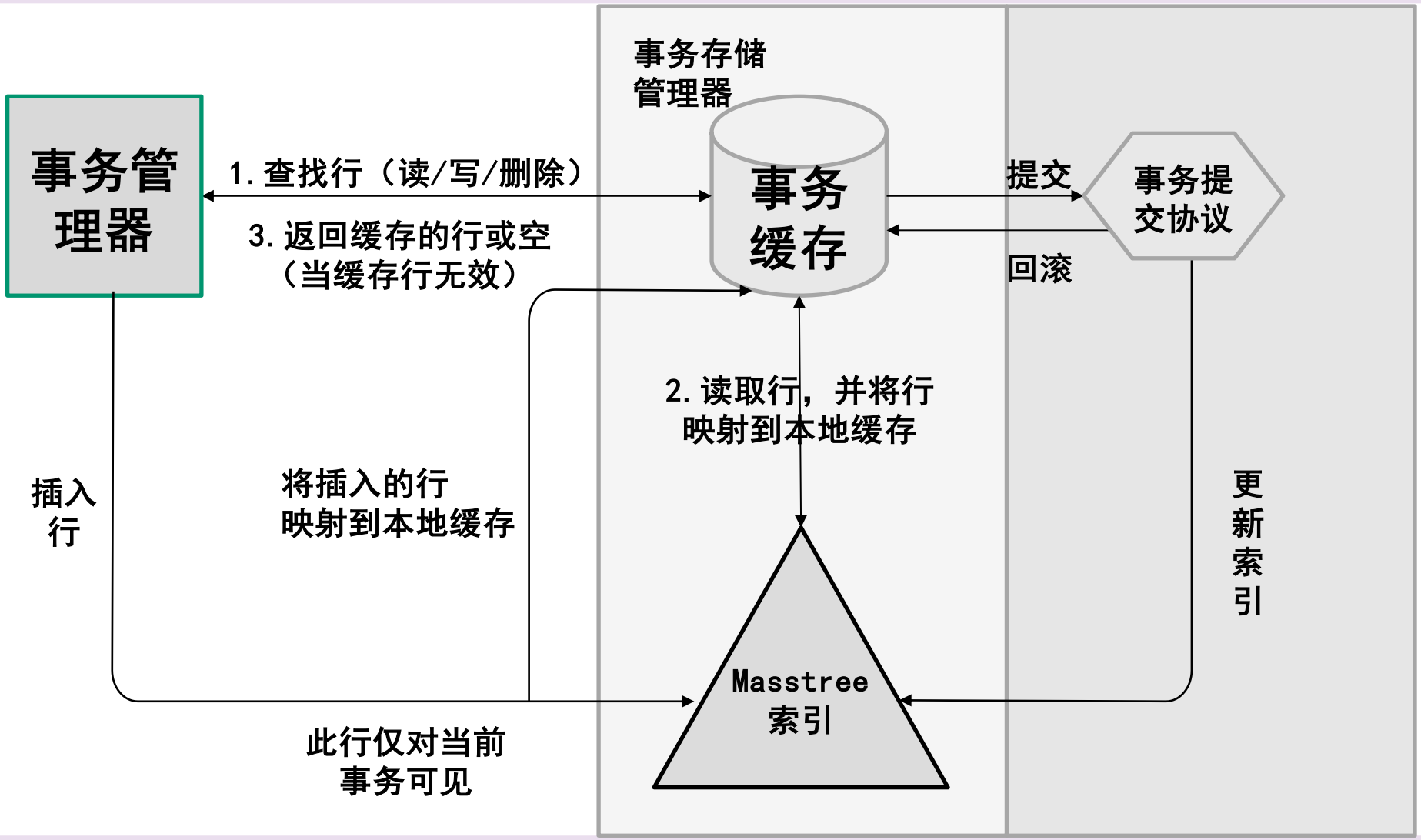
GaussDB内存引擎：FDW

- 采用外部数据封装器FDW（Foreign Data Wrapper）方式将内存引擎插入数据库
  - 优化器可以获取内存引擎的元信息
  - 内存引擎可以按照执行器预期的方式将查询结果返回



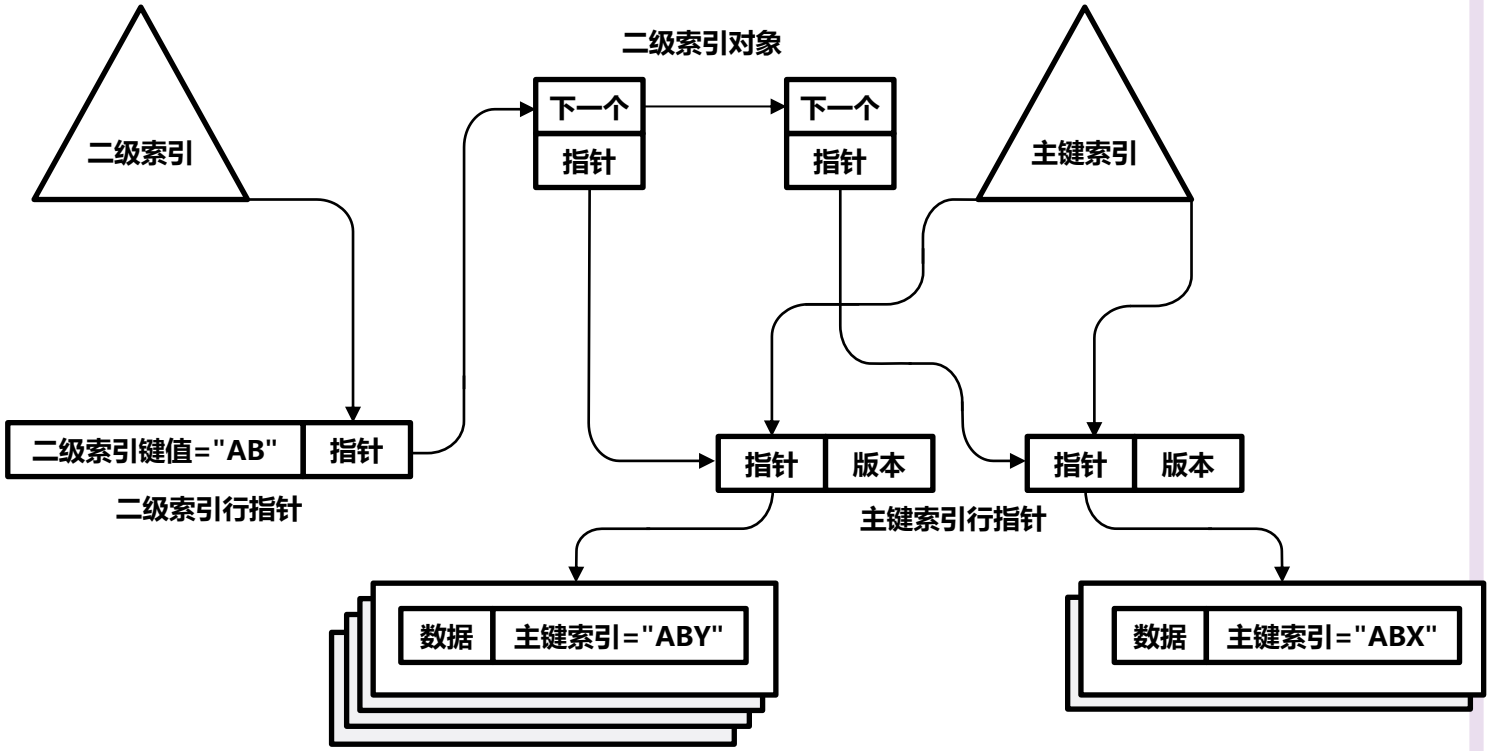


GaussDB内存引擎—并发控制



GaussDB内存引擎—索引设计

- 要求必须存在主键
- 主键索引存储各个行记录的行指针，由行指针来对行记录数据进行内存地址的记录
- 非主键索引被称为二级索引，不再以主键为键值，但仍然存储键值对应元组的指针



GaussDB HTAP

HTAP负载透明路由

- 透明路由
- 智能列选择
- 代价模型
- 计划生成

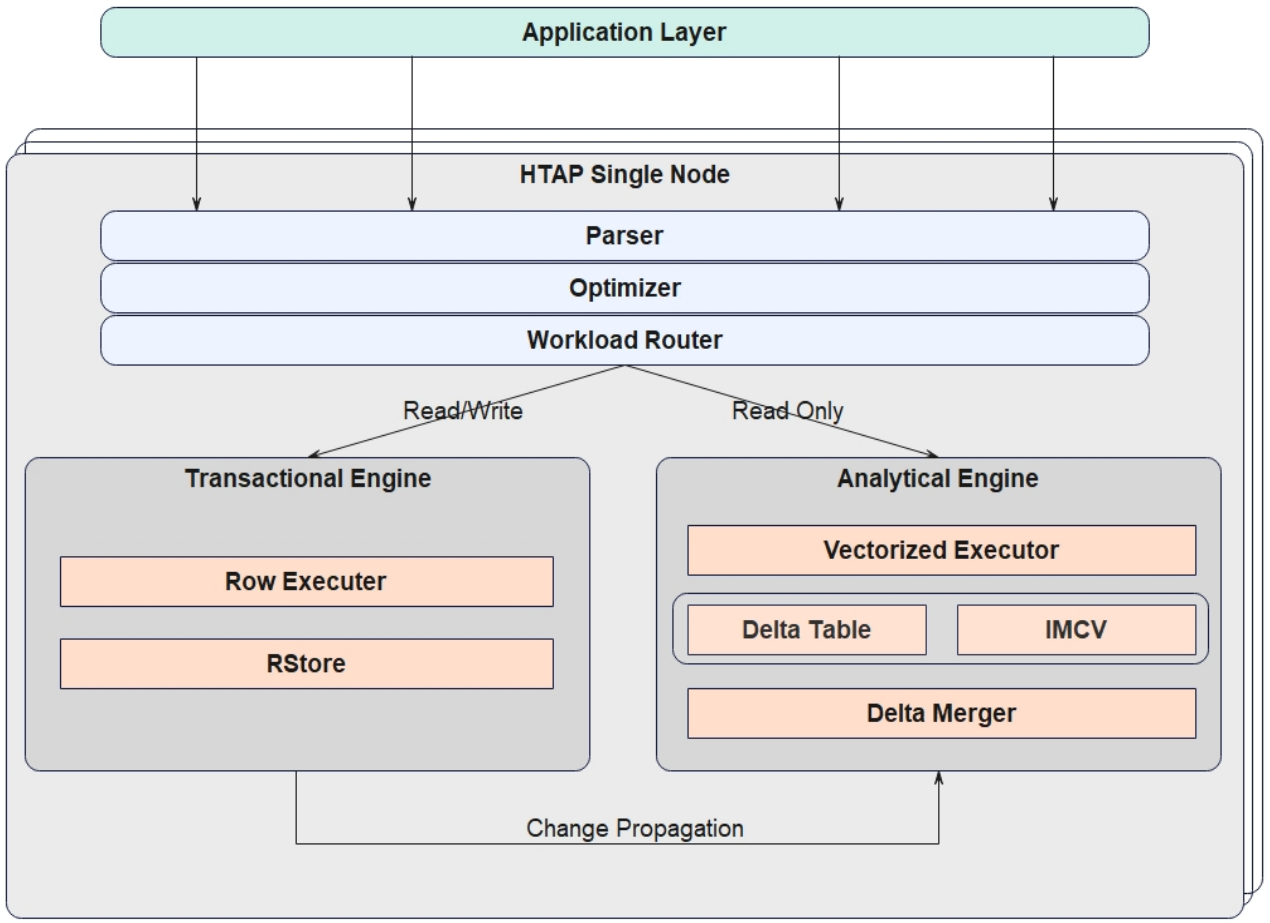
HTAP行列混合执行架构

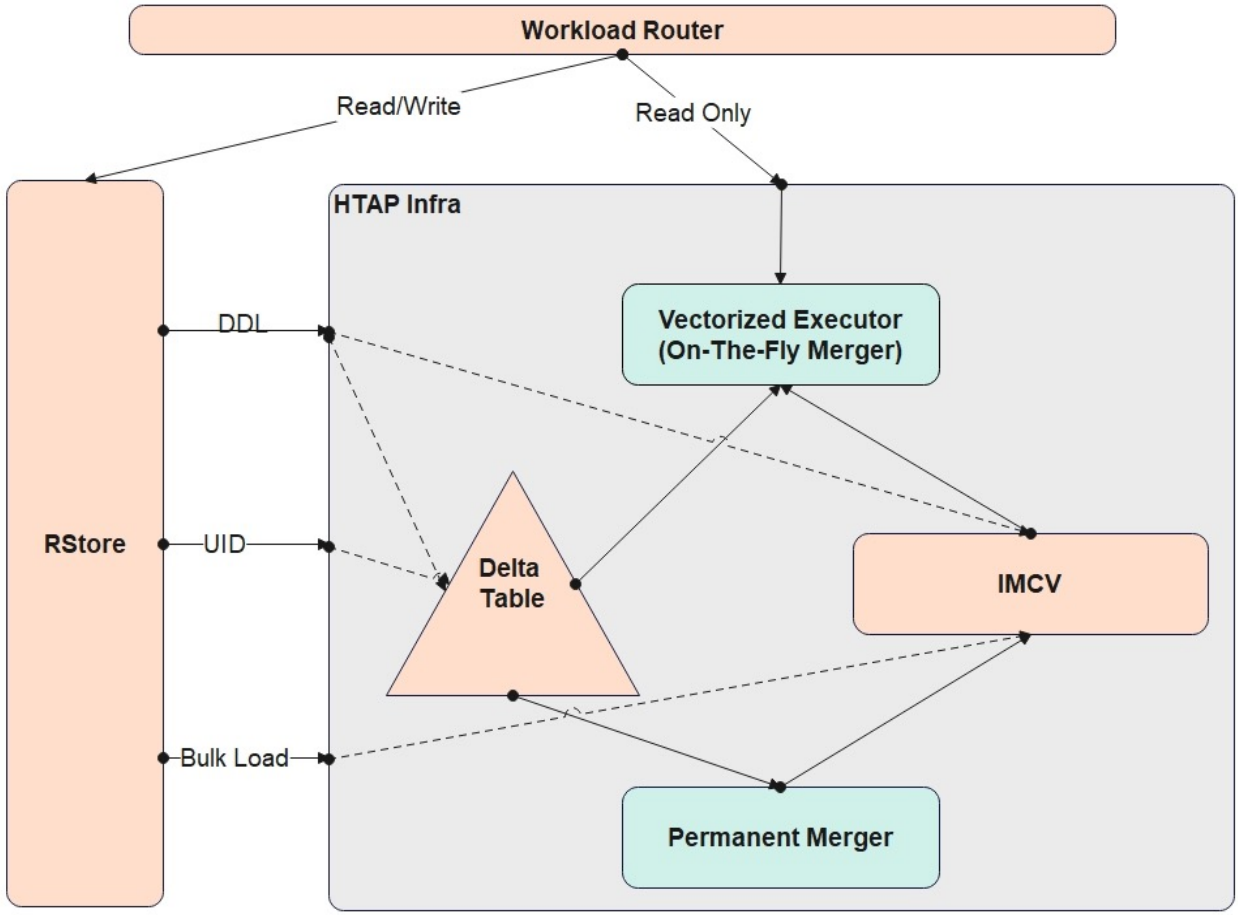
- 列存算子
- 行转列算子
- 列转行算子
- 向量化执行
- 行列混合执行框架

HTAP In-Memory Column Store (IMCS)

- 列存储
- 行delta存储
- 数据实时转换
- 列数据合并
- 数据压缩

GaussDB HTAP

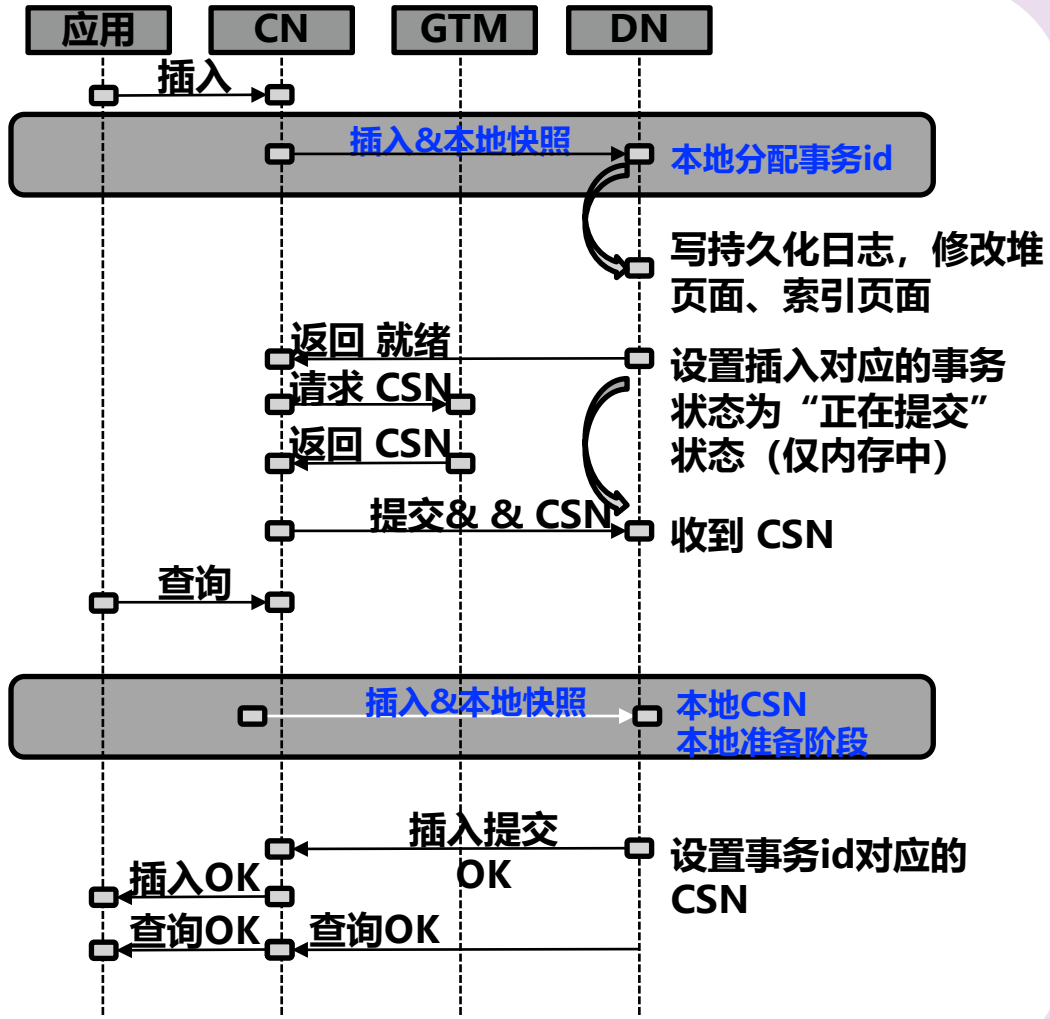




目录

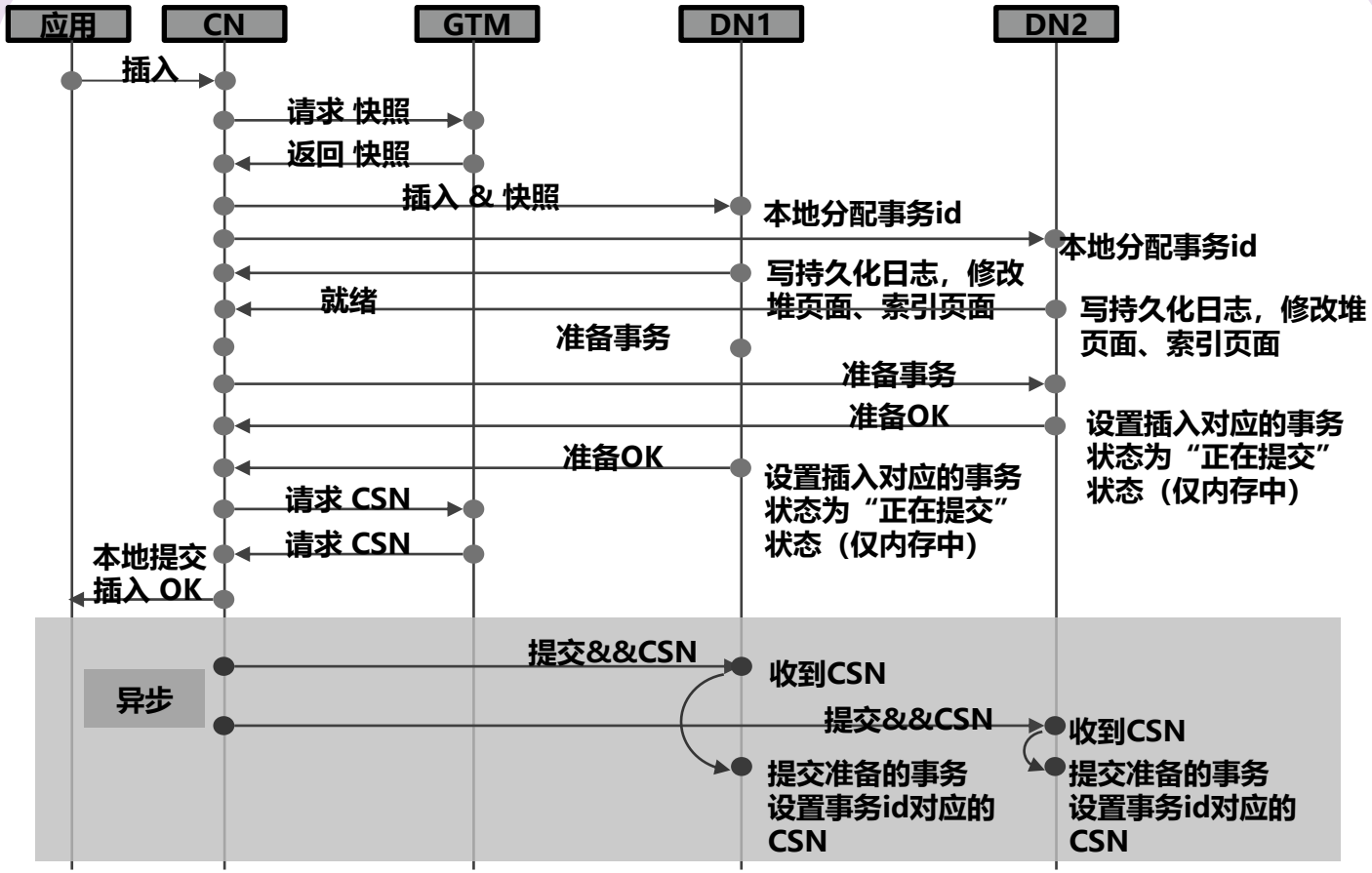
GaussDB分布式事务

- 全局事务管理器GTM处理全局时间戳请求，即CSN（Commit Sequence Number，待提交事务的序列号）
  - 获取本地最新的CSN和准备阶段事务号
  - 如果CSN状态为“提交中”则进行等待
  - 如果row.CSN < localsnapshot.csn || xid in prepared_xid list可见，否则不可见。



GaussDB分布式事务

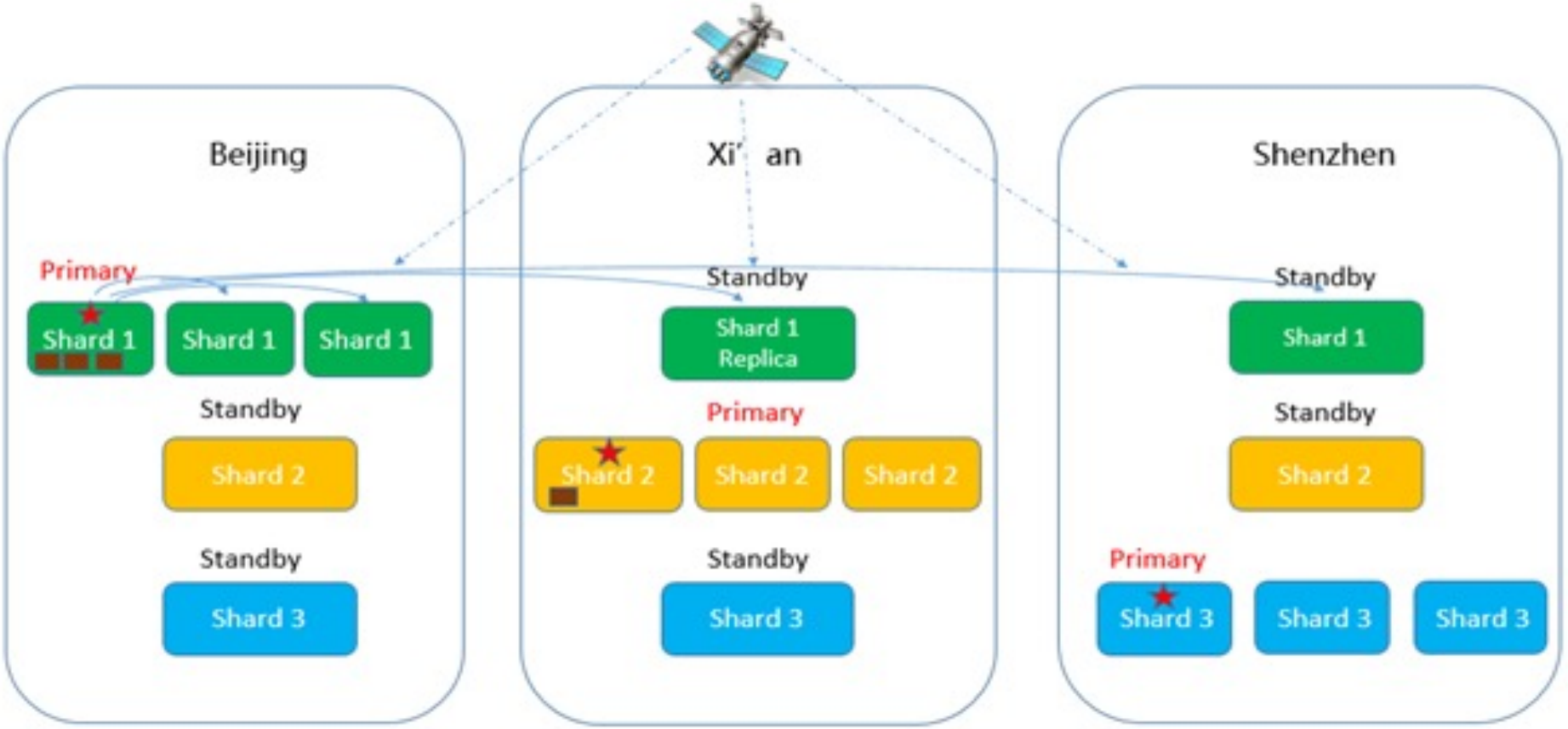
- 在分布式事务（含有多个DN）中，第二阶段事务提交改为异步方式，只同步做两阶段提交的准备阶段
  - DN处于准备状态的事务依赖对应CN上的事务是否提交，如已提交，且CSN比快照CSN小，即为可见；
  - DN上处于准备状态的事务，
  - CN上的事务不处于提交状态，
  - 则必须判断是否是残留状态，
  - 如果是则进行回滚。



GaussDB分布式高精度时钟

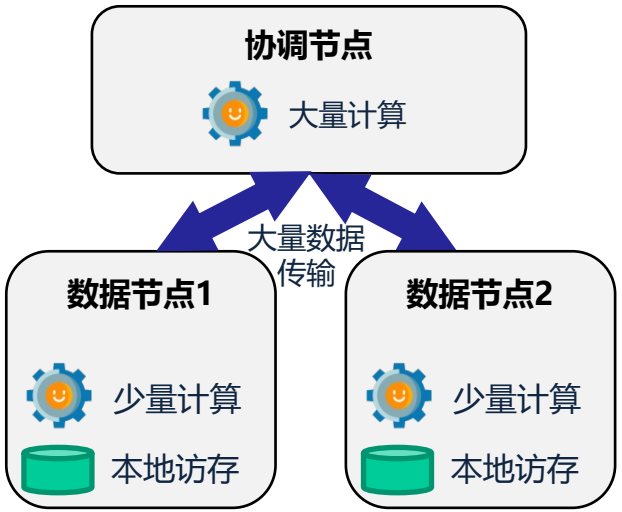
基于高精度时钟的全球数据库技术

- 全球分布式事务模型
- 一致性就近备机查询
- 跨地域高精度时钟同步技术
- 全球多活异地容灾管理

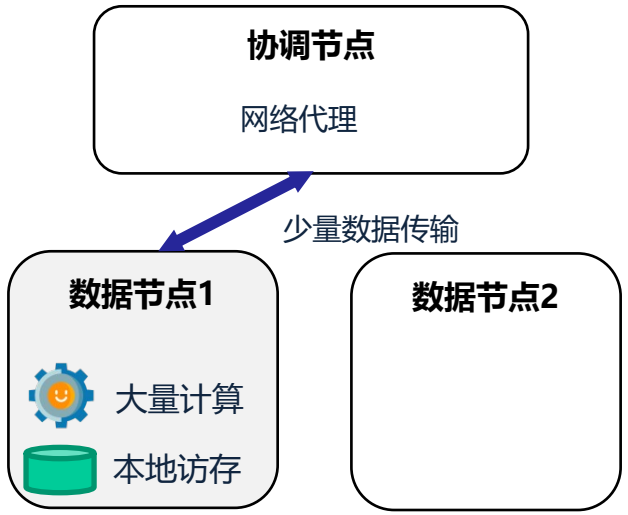


GaussDB分布式查询优化

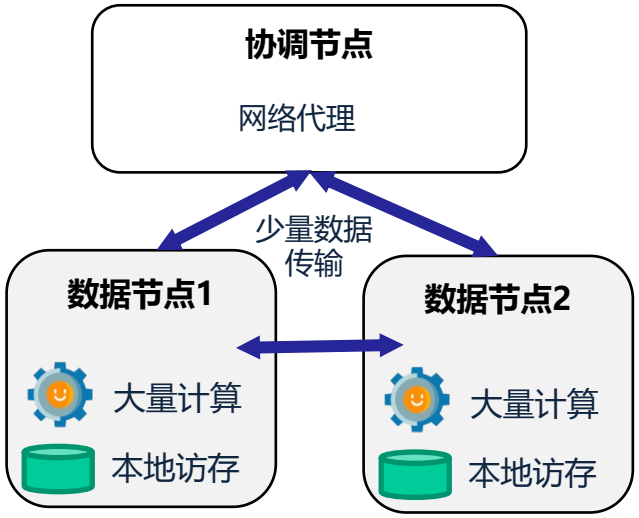
传统分发-汇聚的分布式计划



协调节点旁路技术

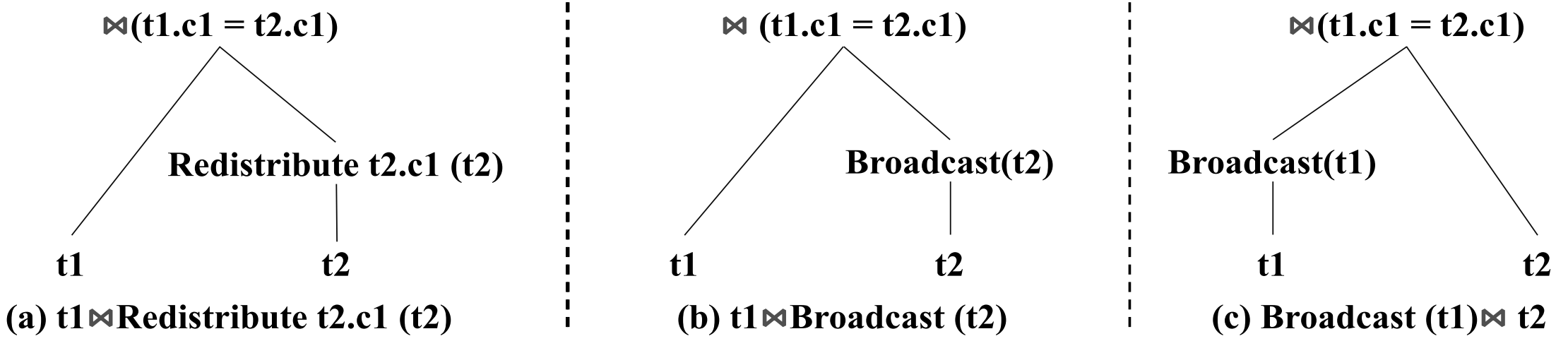


数据节点自协同的流式计划



GaussDB分布式查询优化

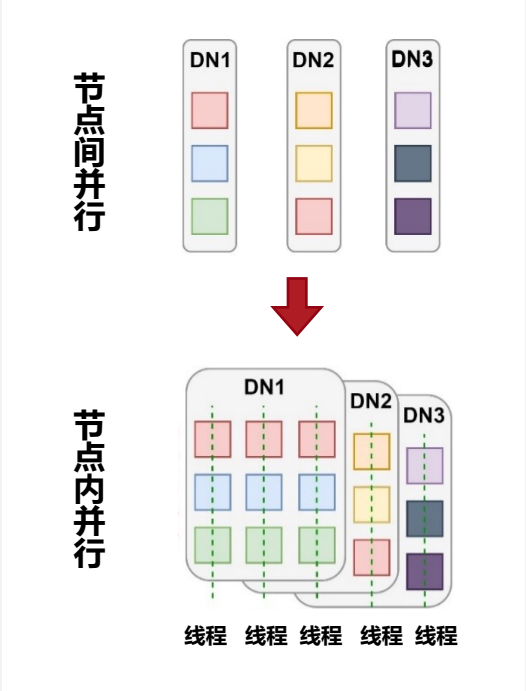
- 生成分布式物理计划时，会考虑数据是否处于同一个数据节点，如果不是，那么会添加相应的数据分发算子。例如：



- 根据分发算子的数据量以及网络通信开销， GaussDB优化器计算计划的代价，并根据代价从中选出最优的物理计划。

GaussDB分布式并行执行

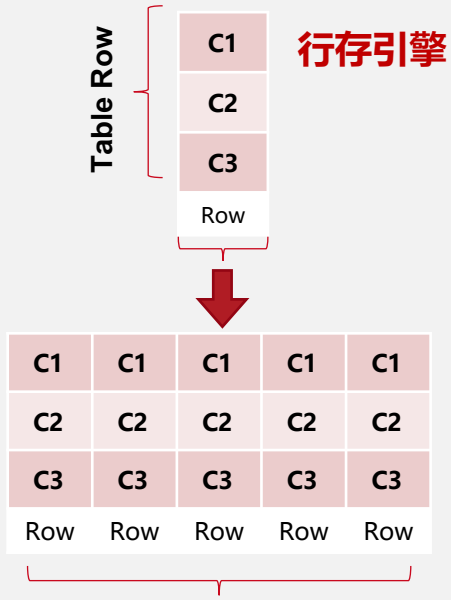
并行执行(Parallel)



□ 多线程并行执行

- 充分利用多核特点，通过多线程并发执行，提高系统吞吐量

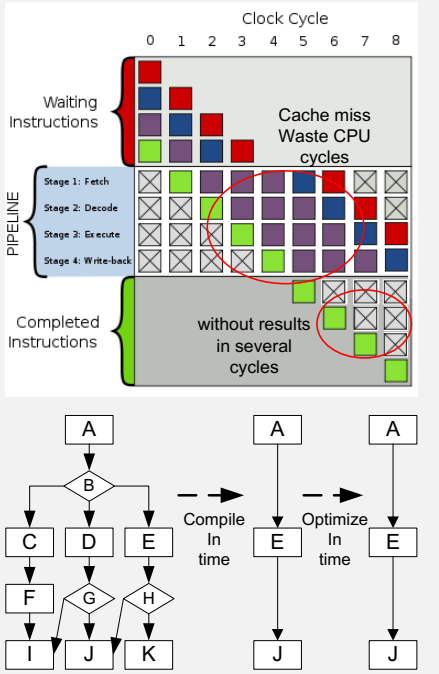
向量执行(SIMD)



□ 利用CPU的局部性及算力

- 利用CPU的局部性，通过向量执行提高执行效率

编译执行(LLVM)



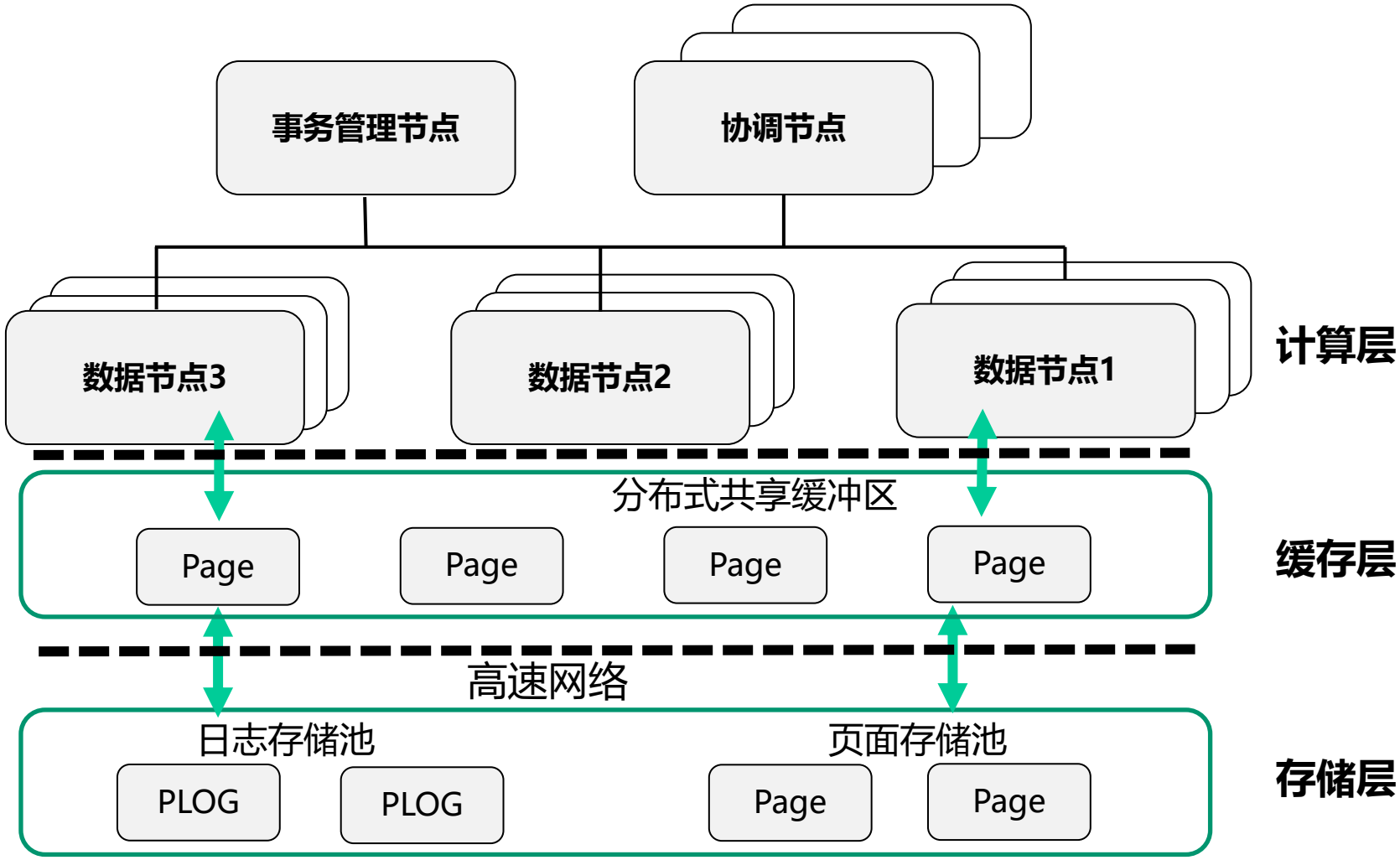
□ 提高CPU指令的利用率

- 从解释执行向编译执行转变，大幅降低执行算子的指令数量，提高运算效率

目录

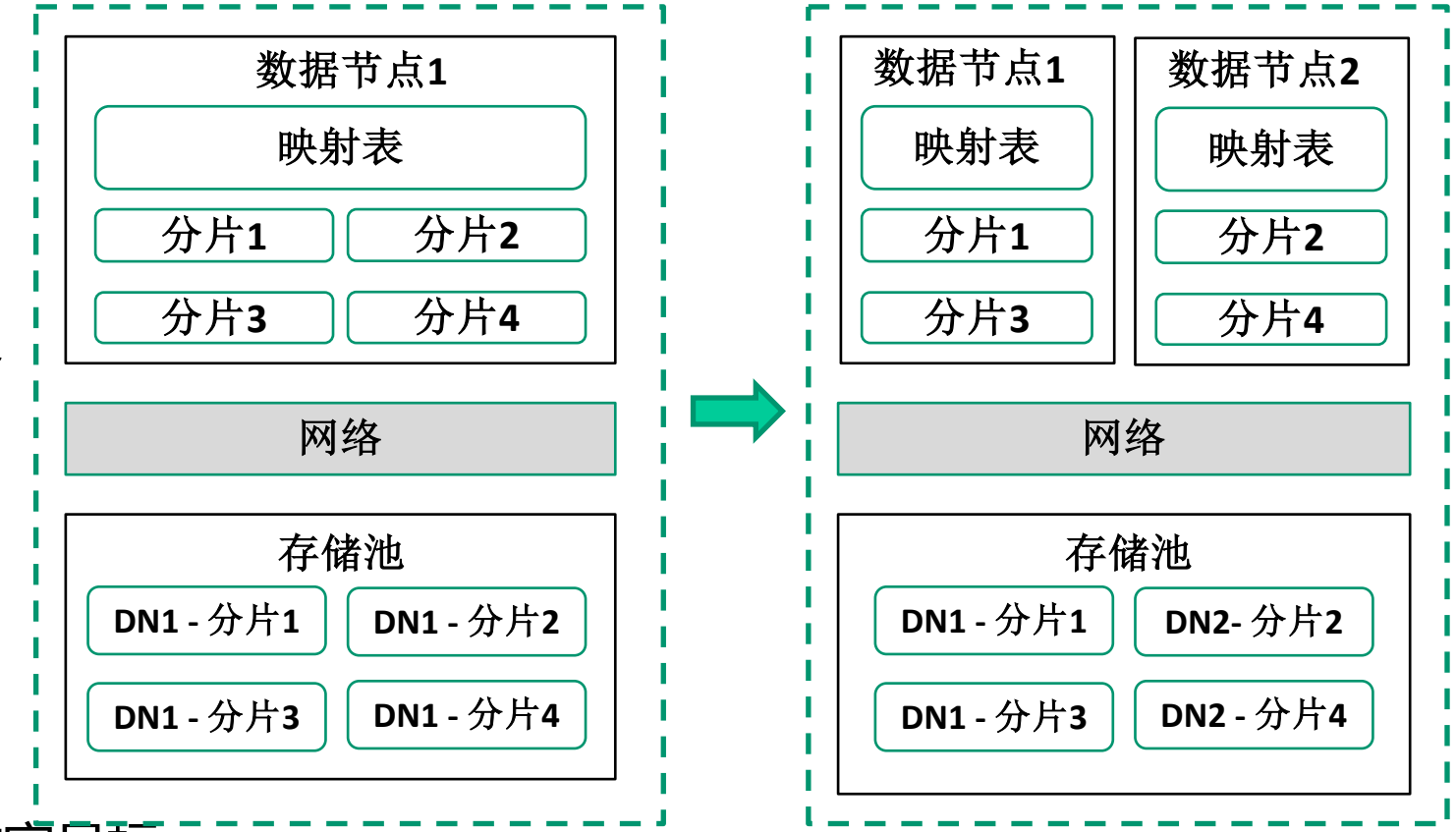
GaussDB云原生

- 分布式计算存储分离架构
- 存储节点日志按需回放
- 数据预分片
- 哈希桶聚簇存储
- 映射表并行迁移
- 哈希桶聚簇存储
- 自调节查询下发
- 读写负载均衡



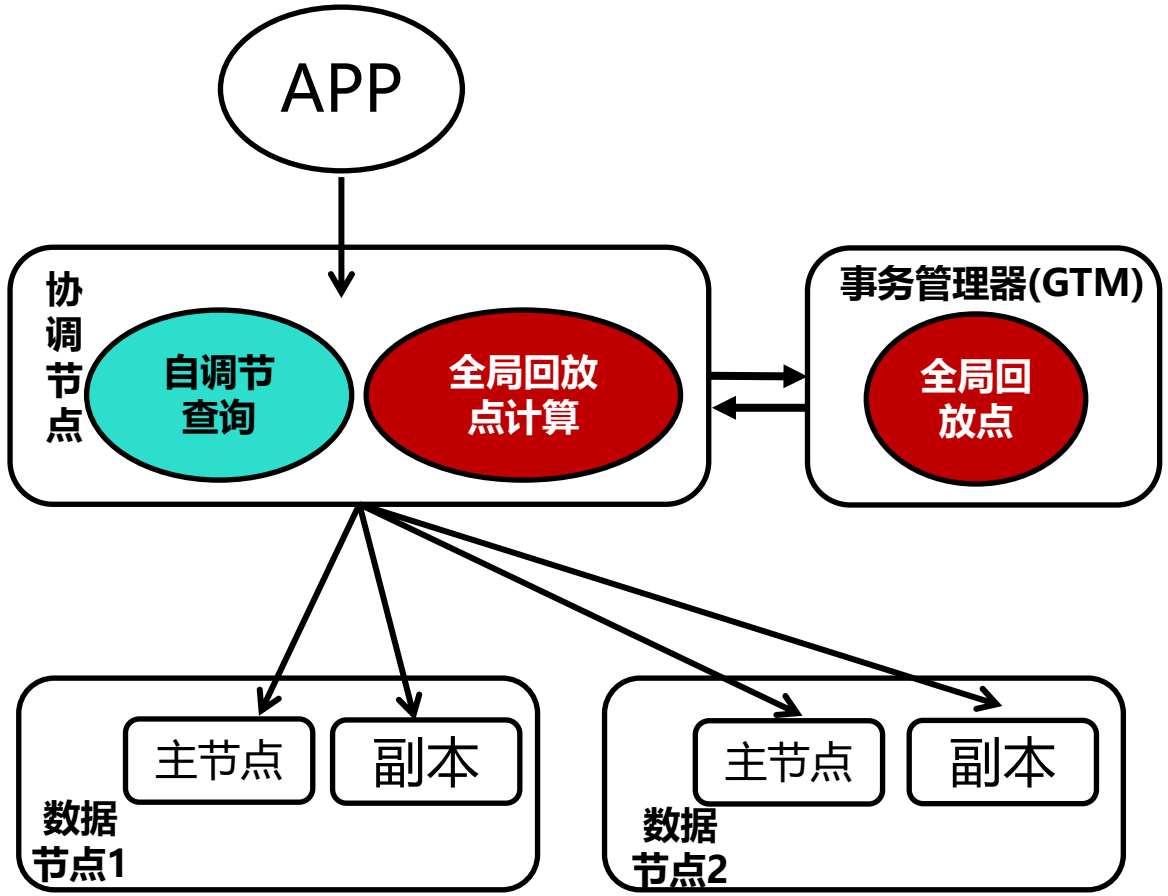
GaussDB云原生

- 分布式计算存储分离：计算与存储解耦，实现计算资源池和存储资源池独立扩缩容，分层弹性
- 存储节点按需回放: 存储节点实现日志回放，大幅减少计算节点和存储节点带宽压力。
- 数据预分片: 结合底层分布式存储，实现数据预分片能力，缩短扩容时间，实现业务平滑扩容
- 哈希桶聚簇存储：把表文件拆分为多个分片，采用一致性哈希算法，扩容时，只需**重分布少量数据**可实现扩容
- 映射表并行迁移：用户数据不迁移，仅迁移少量元数据映射表，实现数据在数据节点的重分布，达成扩容目标



GaussDB云原生

- 全局回放点计算：收集所有备机回放进度，计算全局一致性点
- 自调节查询下发：通过全局一致点信息，选择满足要求的节点下发查询，保证分布式强一致读取
- 读写负载均衡：将读请求下发给同一分片，不同的副本节点，实现读请求的负载均衡



目录

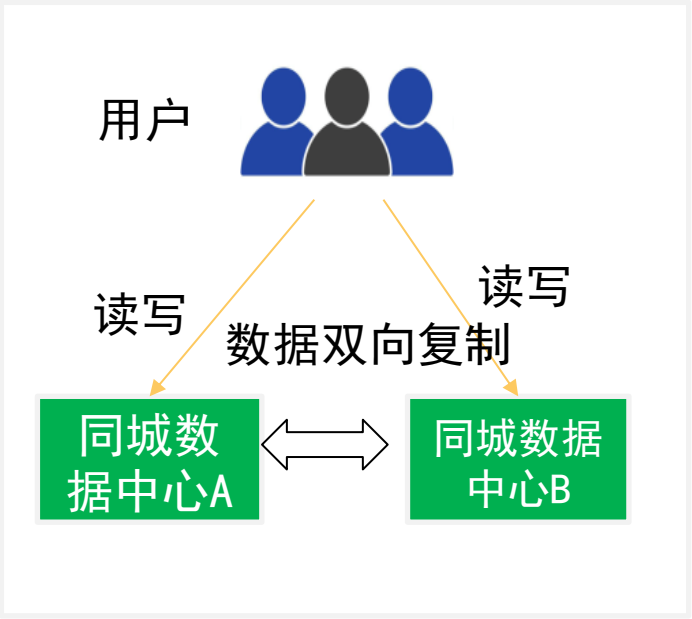
GaussDB 高可用容灾关键技术

- GaussDB设计了多副本高可用、两地三中心、双集群高可用、跨地域多活等技术，全面提升分布式数据库的高可用能力

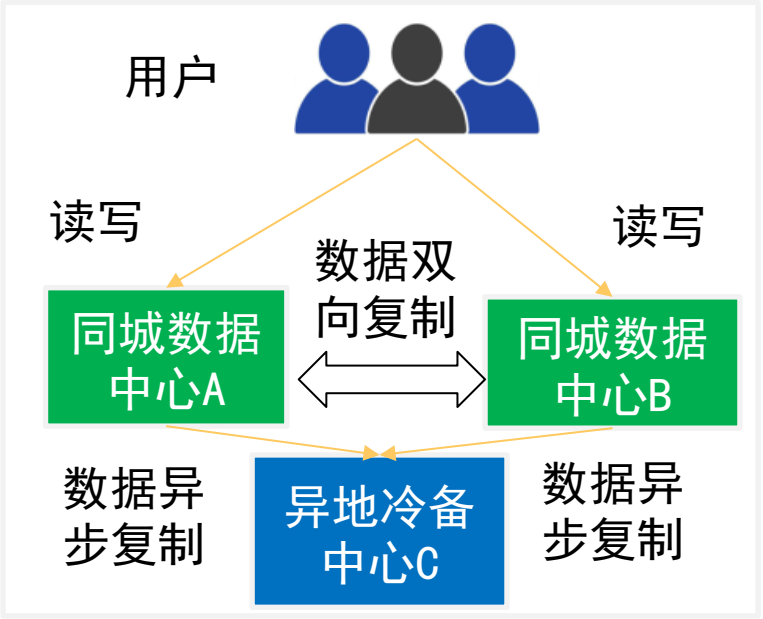
容灾级别	容灾方案	故障场景	关键技术
同城	三可用区	节点、机房级故障	多副本一致性算法
	同城双集群	节点、机房级故障	并行逻辑日志解析与回放
异地	两地三中心	节点、机房级、城市级故障	物理日志流式复制技术
	异地双活	节点、机房级、城市级故障	并行逻辑日志复制技术

GaussDB高可用

同城跨AZ

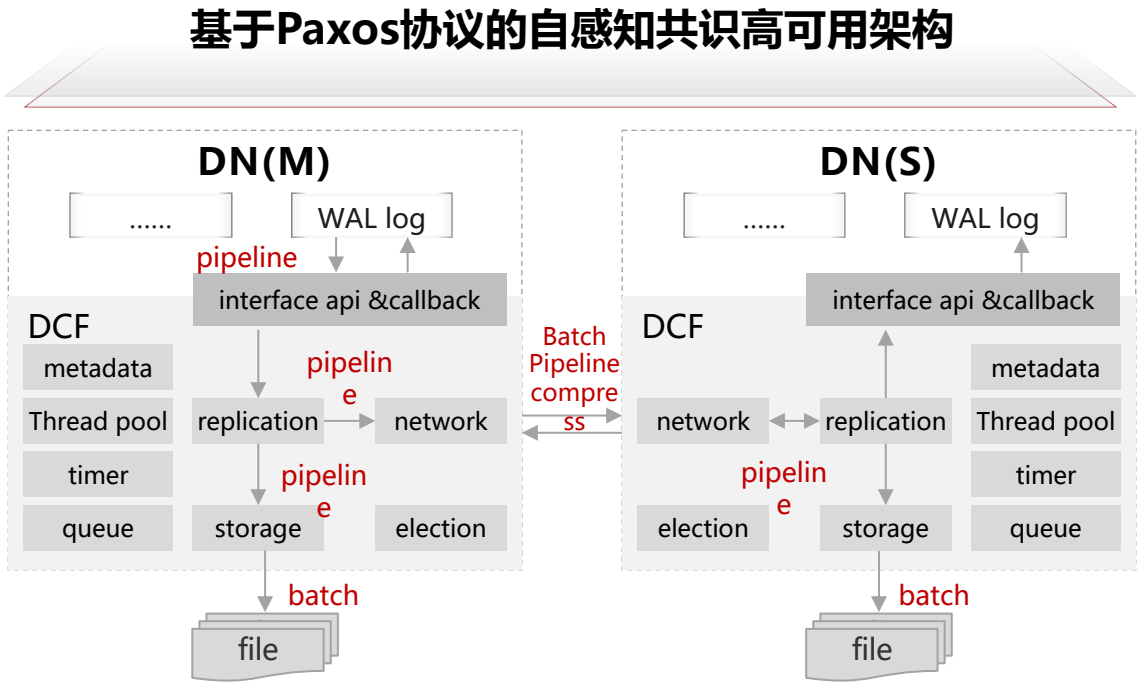


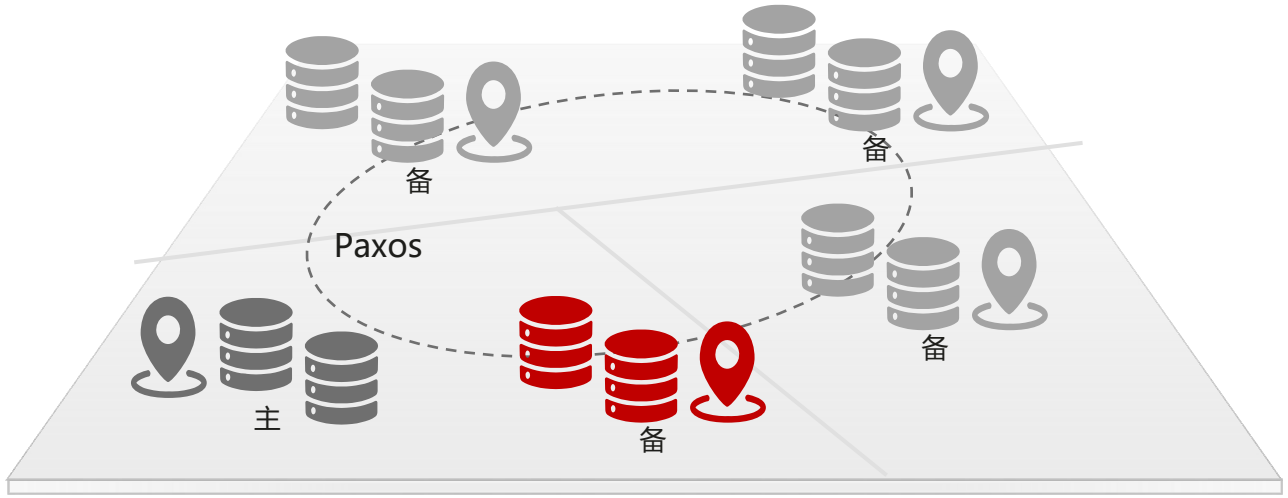
两地三中心



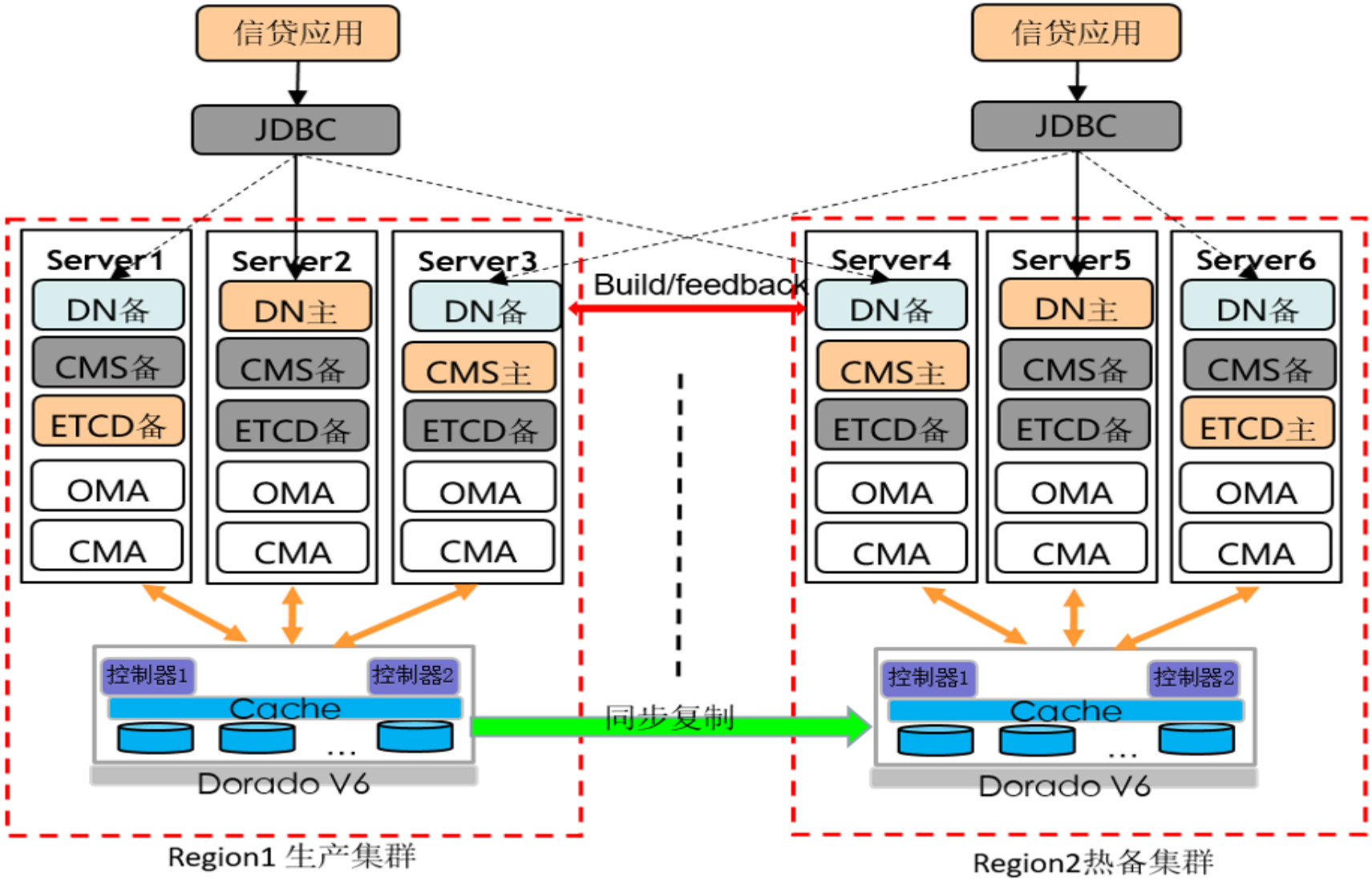
异地多活

GaussDB多副本高可用

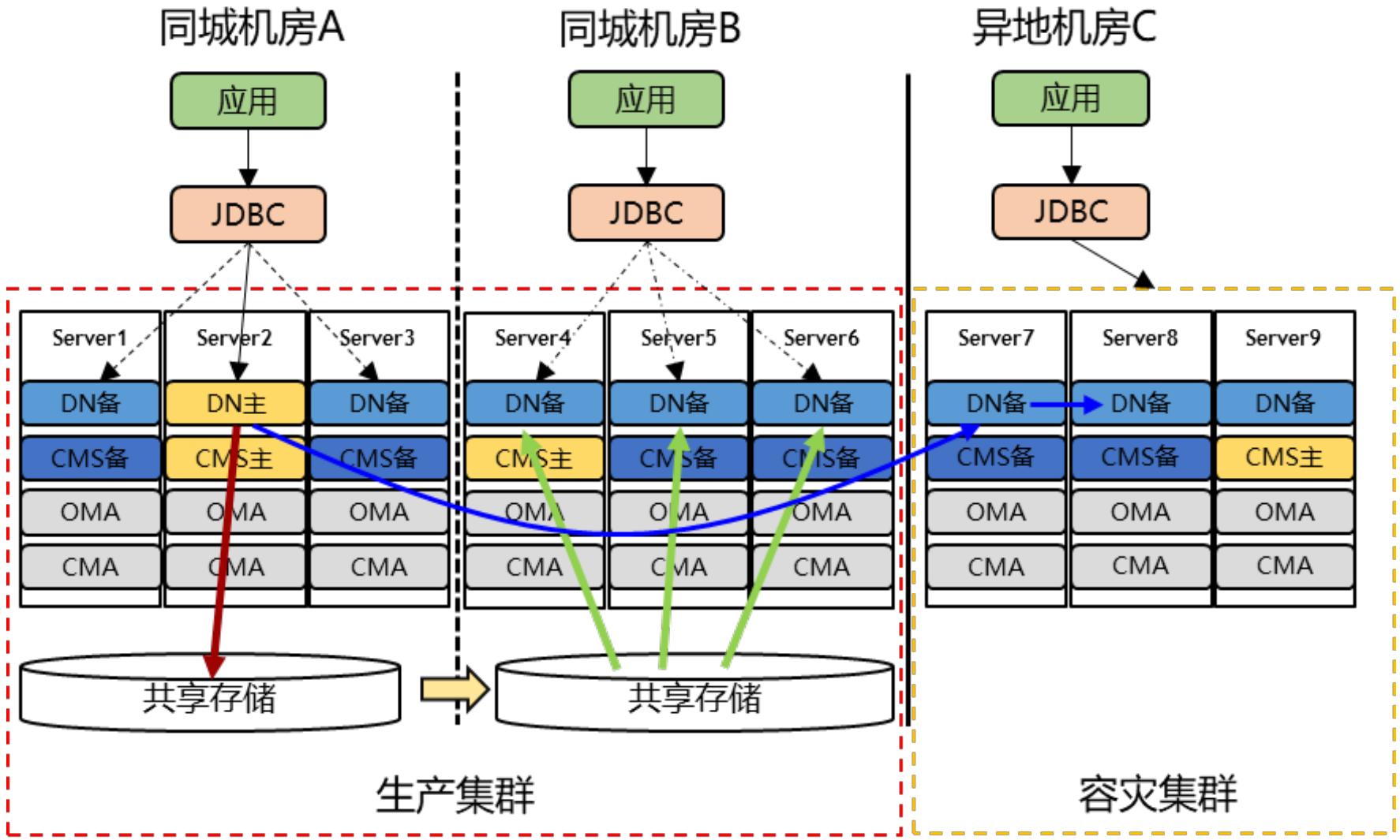




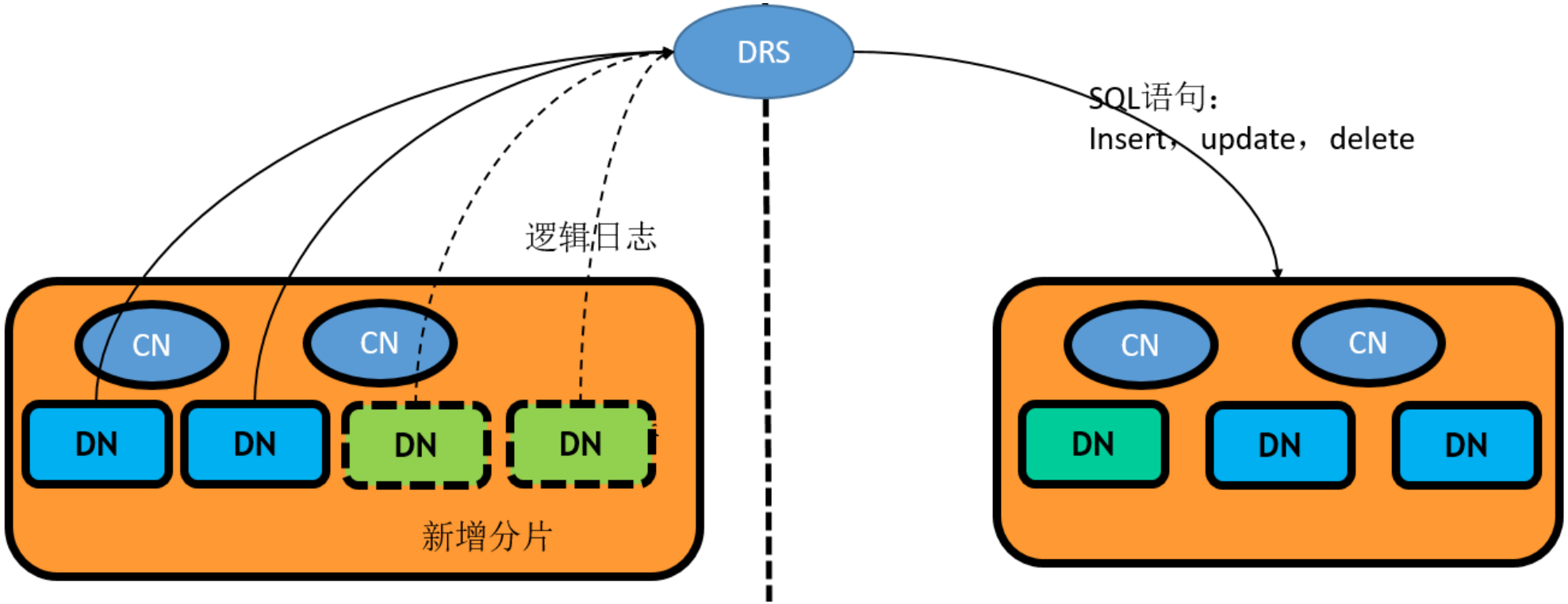
GaussDB 同城双集群RPO=0



GaussDB两地三中心

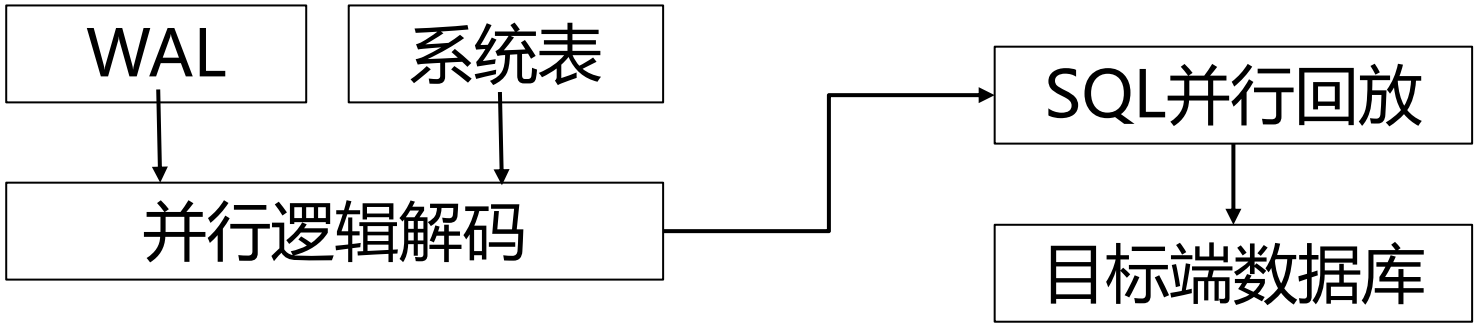


GaussDB多地多活



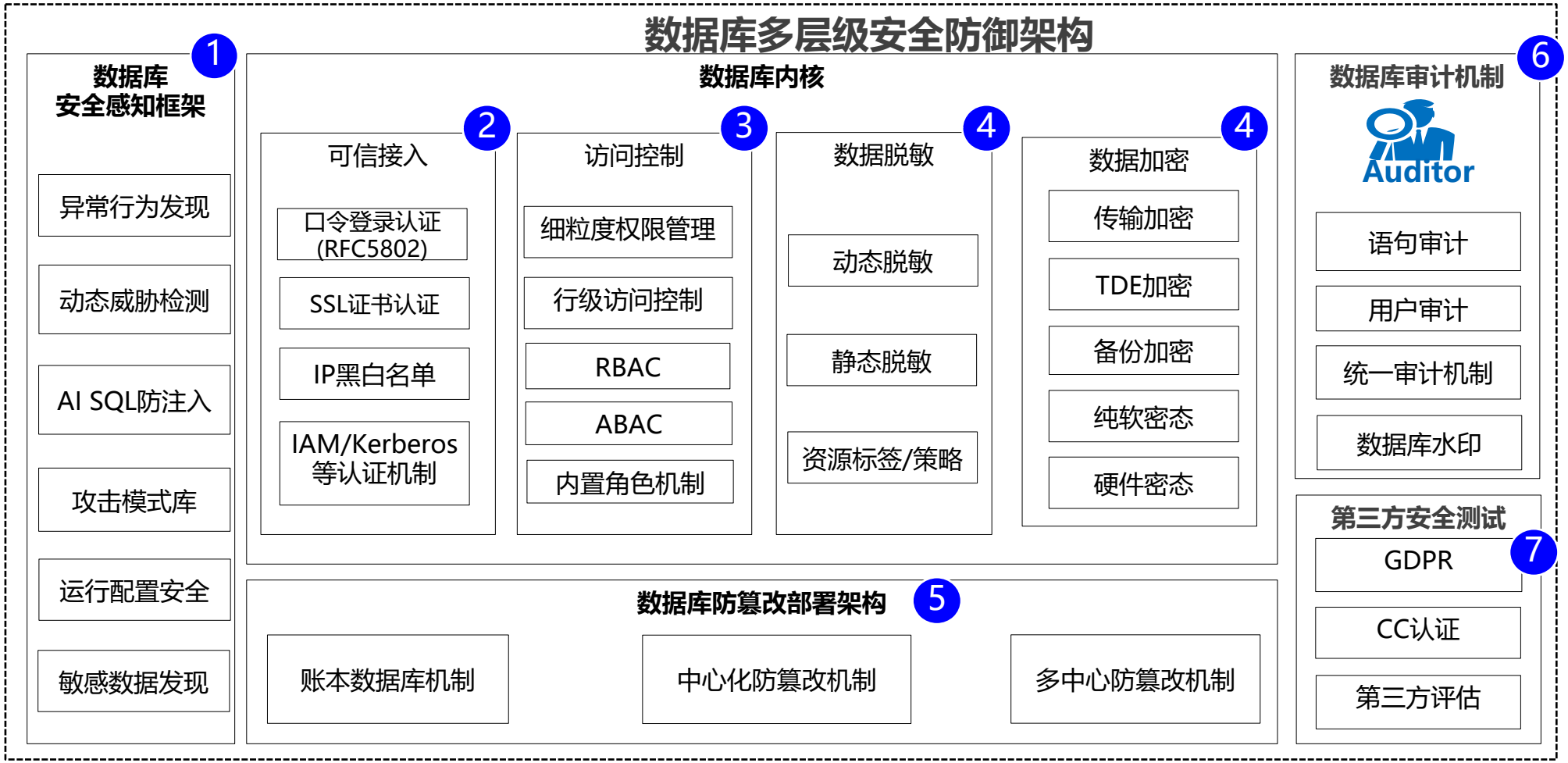
GaussDB逻辑复制

- 从WAL记录并行反解析事务的SQL，反解析依赖于系统表元信息。
- 反解析的事务SQL在目标端数据库按照在源端的csn顺序提交。有行冲突的事务必须是串行的，无法并行执行。对无行冲突的事务，可以并行，但是原则上也要按照csn提交顺序提交事务才可以保证事务的因果序。
- 逻辑复制支持发布订阅功能，支持创建逻辑复制槽。



目录

GaussDB安全全景



1 攻不破

2 进不来

3 拿不走

4 看不懂

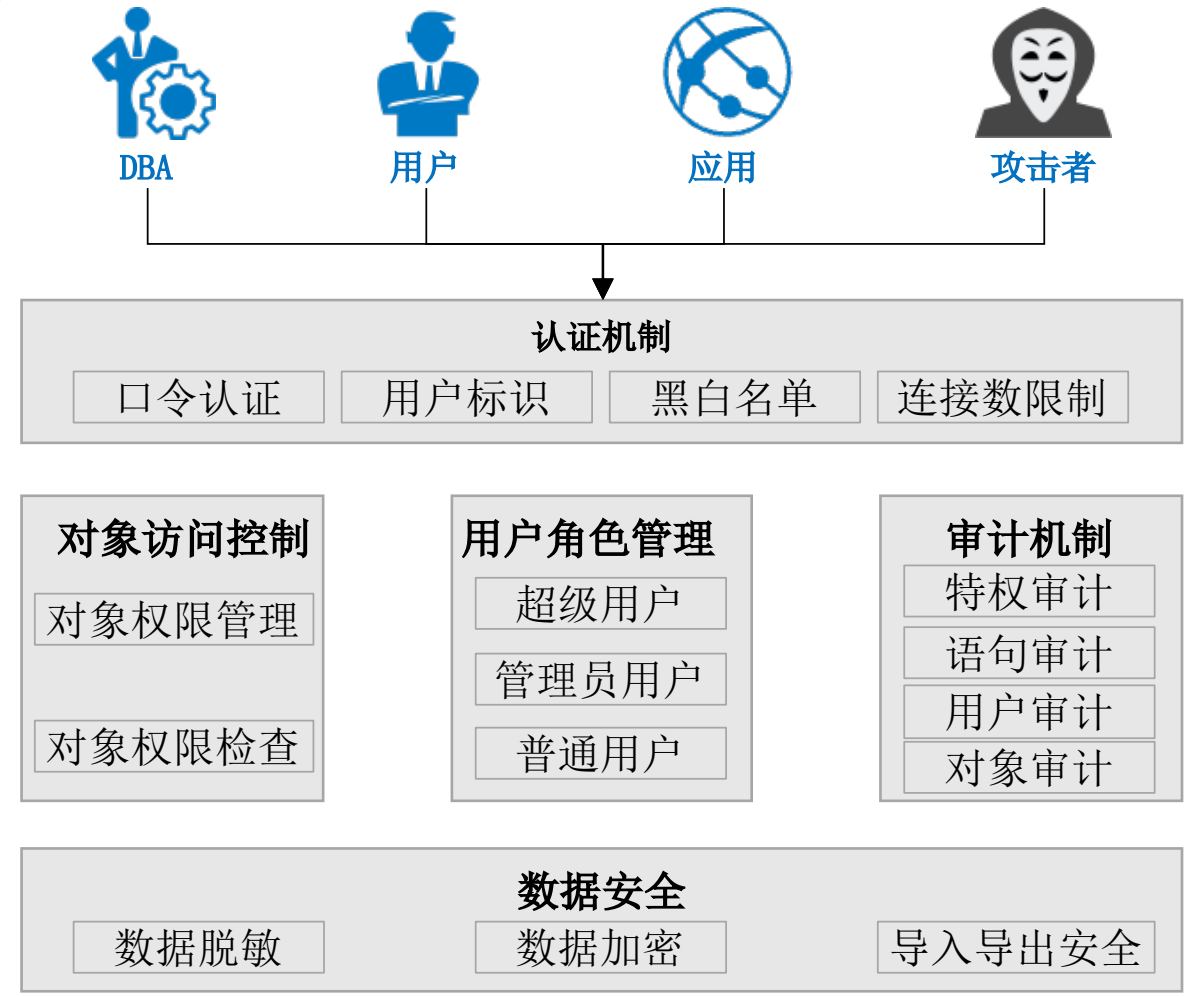
5 改不了

6 赖不掉

7 信得过

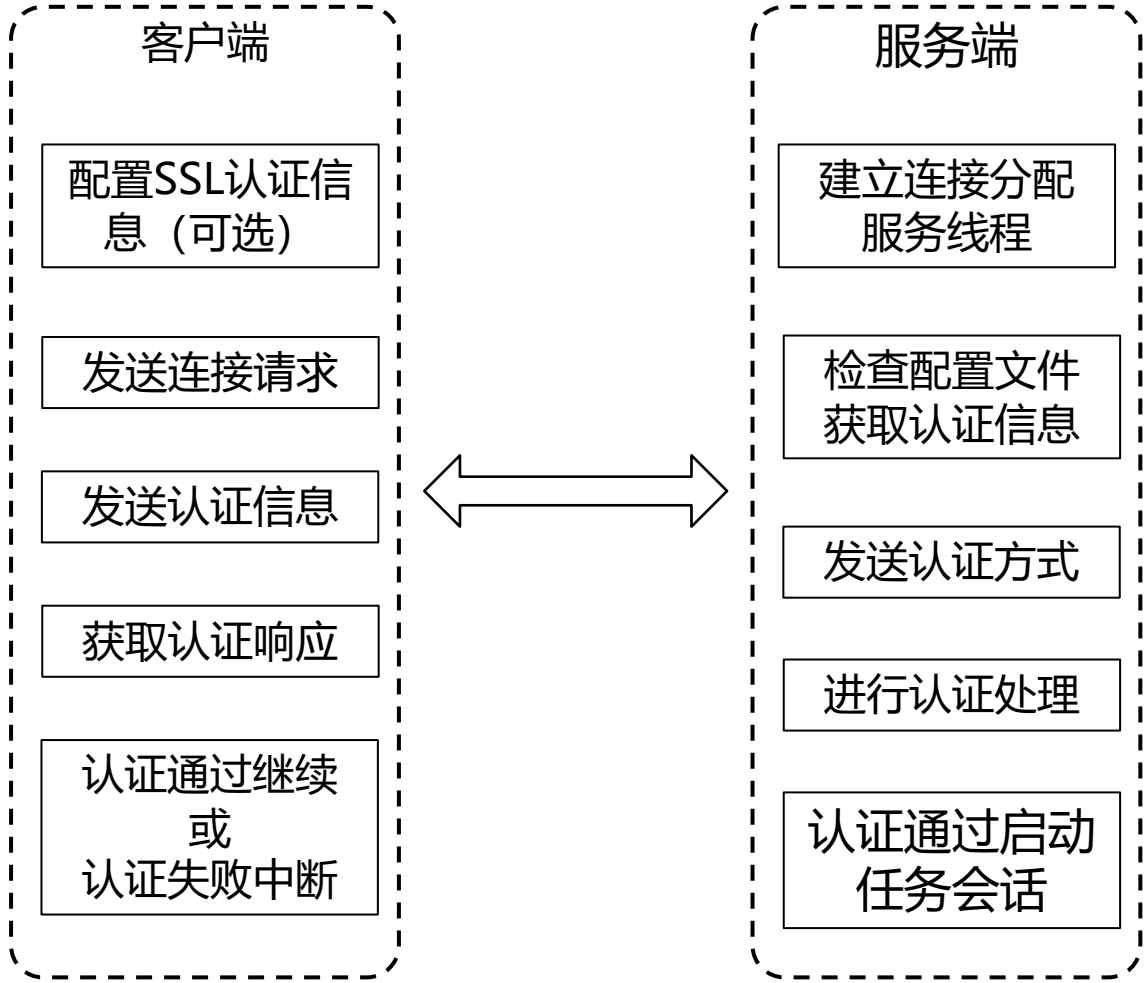
GaussDB安全机制体系

- 安全认证
  - 限制用户对数据库的访问
- 基于角色的访问控制
  - 控制用户对数据库资源的访问
- 审计
  - 日志记录与行为追溯
  - 统一审计
- 数据安全
  - 保障数据的隐私和安全
  - 全密态
  - 防篡改
- 异常行为分析和SQL防注入

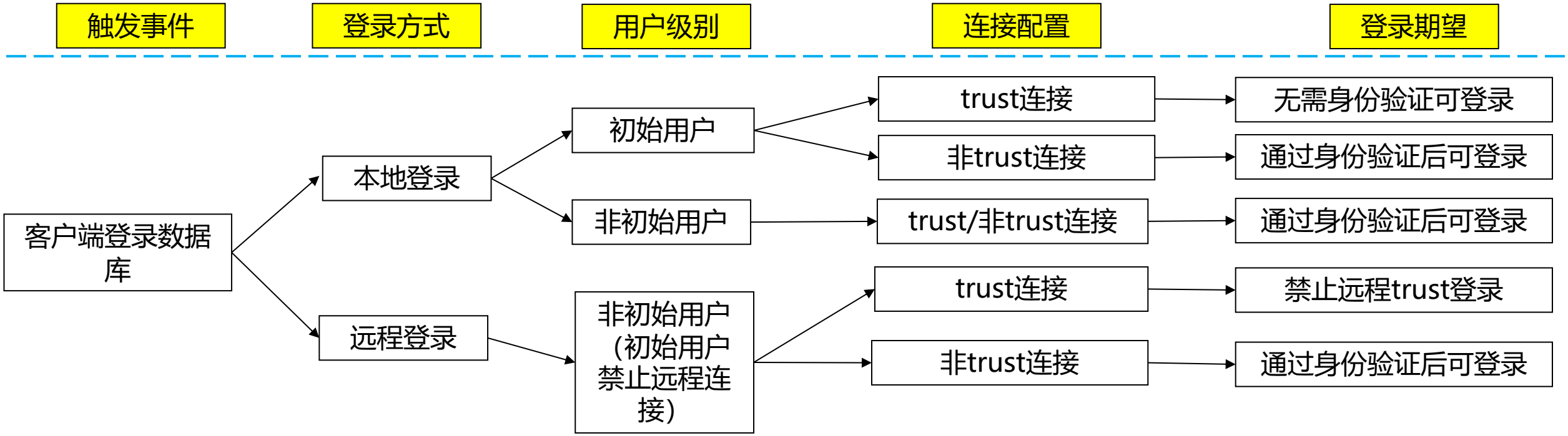


GaussDB安全认证

- 工作模式：客户端/服务端模式
- 解决访问源可信问题
  - 将访问方式、访问源IP地址、认证方法等存放在服务端的配置文件中
- 认证方法
  - Trust认证：无需提供口令
  - 口令认证：SHA256加密口令认证
  - 证书认证：使用SSL客户端认证
- 安全认证机制
  - RFC5802认证机制



GaussDB安全认证

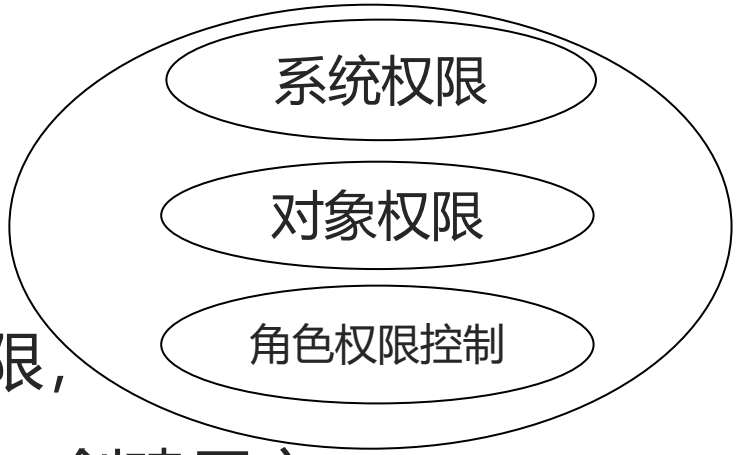


登录所使用的用户名　　服务器端IP地址

```
gsql -d db -U username -W 'passwd#123' -h 192.168.0.10 -p 8000 -r
```

请求登录的数据库　　对应用户名的密码　　服务端端口号

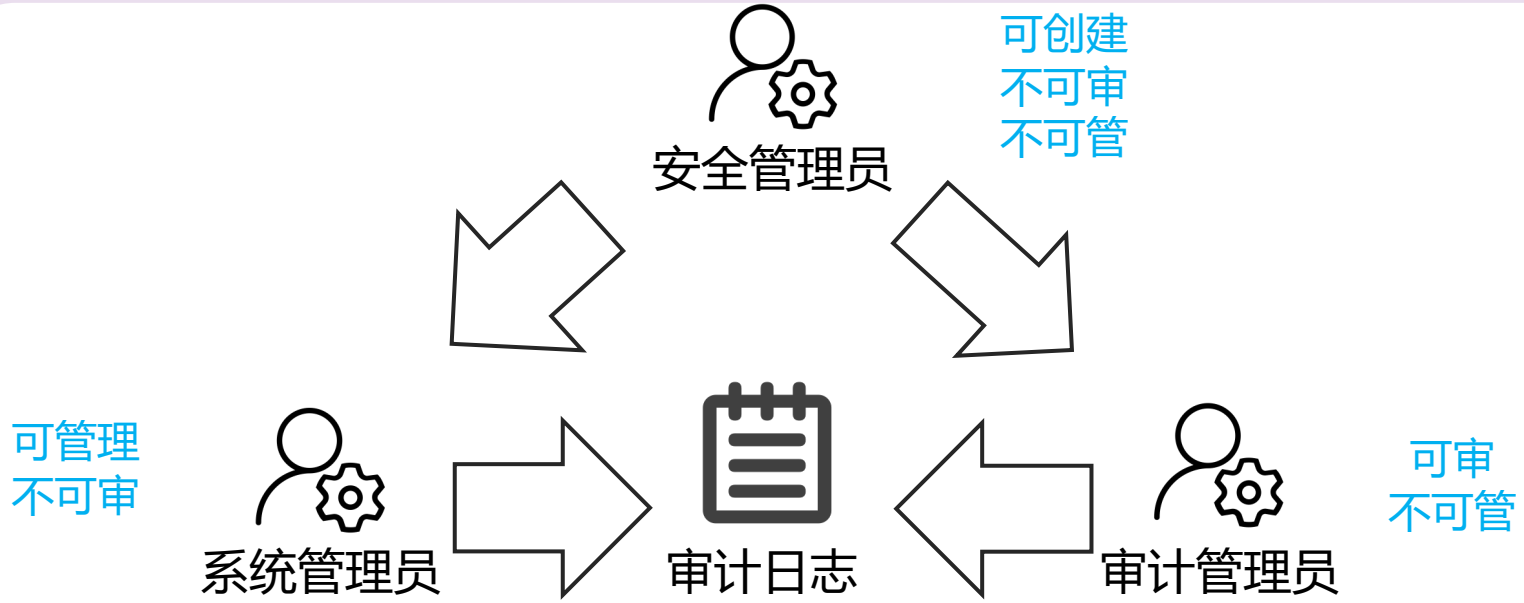
GaussDB权限管理:基于角色的访问控制



- 用户（USER）创建、访问数据库对象和执行SQL语句。
- 角色（ROLE）通常用来组织和管理权限。
- 系统权限：又称用户属性，系统规定用户使用数据库的权限，比如连接数据库（LOGIN），创建数据库（CREATEDB），创建用户（CREATEROLE）等。
- 使用角色控制哪些用户拥有哪些对象的哪些权限，可以大大简化权限的管理。
- 通过将多种权限打包成角色授予用户，使得用户获得该角色的所有权限；若改变角色的权限，则继承该角色的用户权限会被自动修改。一个用户也可以继承多个不同角色的权限。

GaussDB权限管理:基于角色的访问控制

- 角色
  - 包含角色组中所有成员权限
  - 管理员将权限赋予角色，用户从角色继承相应权限
  - 每个数据库对象都有对应的访问控制列表ACL
- 三权分立
  - 管理员细分为安全管理员、系统管理员和审计管理员



GaussDB基于标签的访问控制

主体：用户 —访问→ 客体：表或表列

获取主体安全标签　获取客体安全标签 ------- 安全标签：等级L1-L1024：范围G1-G1024

获取访问动作

强制访问控制规则校验

插入（INSERT）
主体等级小于等于客体等级且
主体范围是客体范围的子集

修改（UPDATE）
主体等级等于客体等级且
主体范围等于客体范围

查询（SELECT）
主体等级大于等于客体等级且
主体范围是客体范围的超集

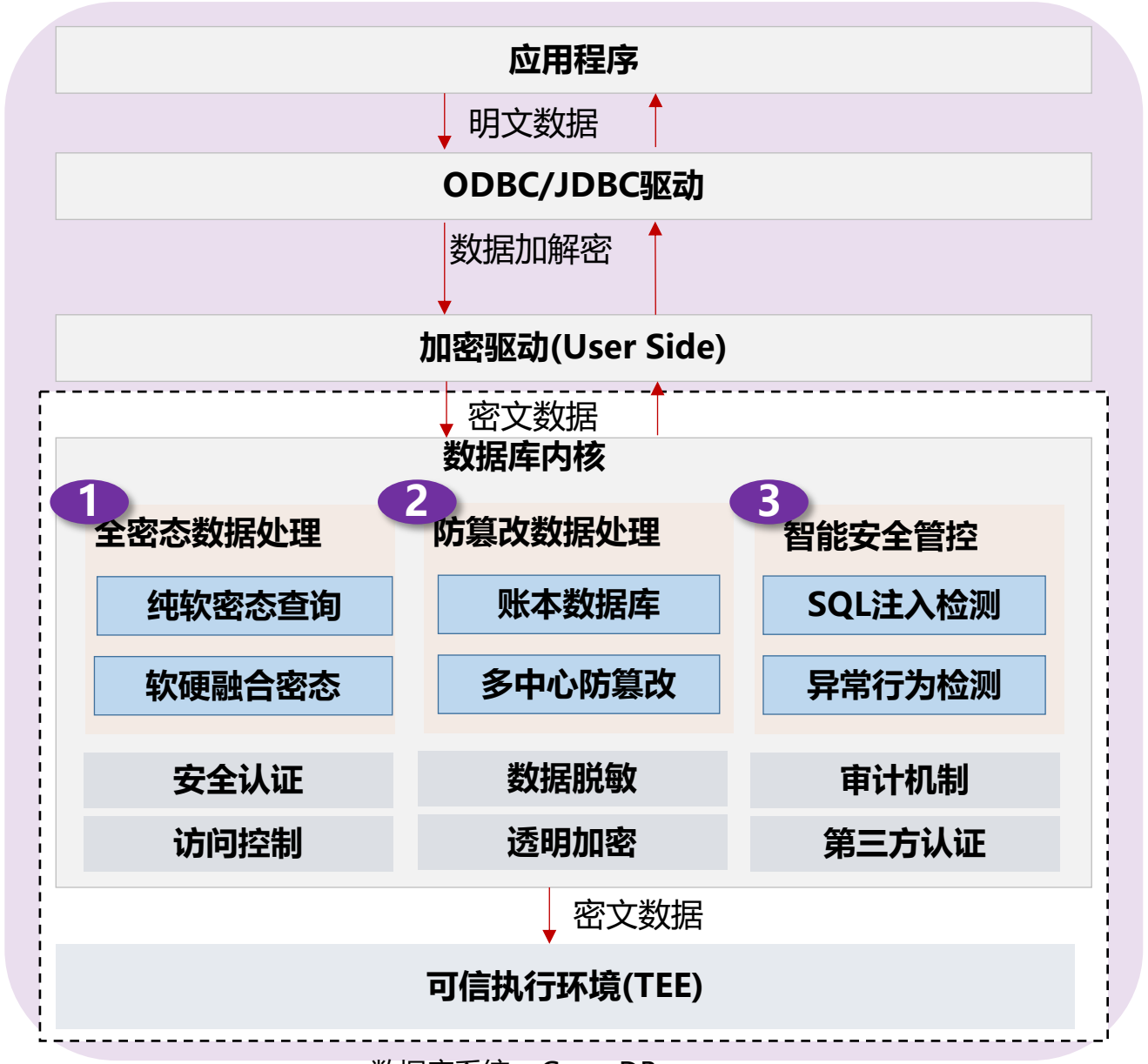
删除（DELETE）
主体等级等于客体等级且
主体范围等于客体范围

→ 符合规则，允许访问
不符合规则，拒绝访问

GaussDB审计

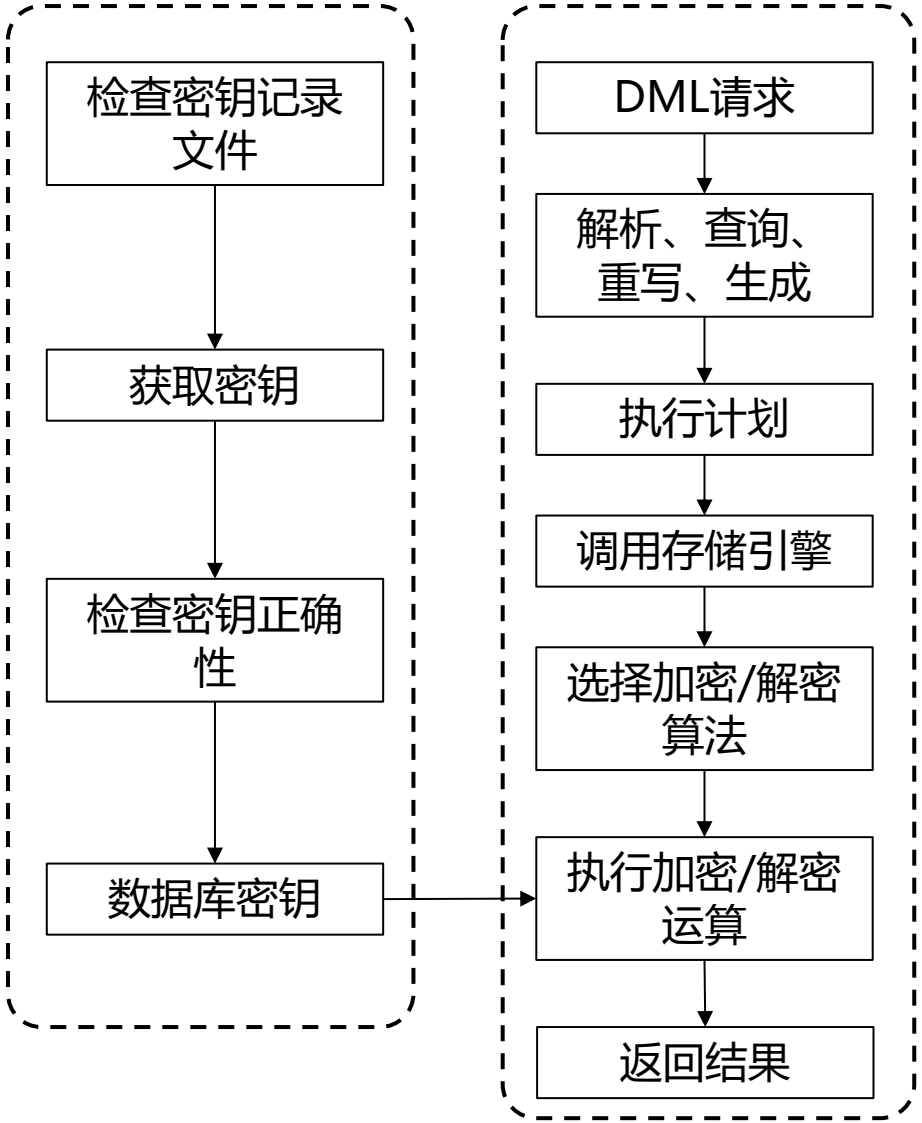
- 目的：快速发现和追溯异常行为
- 审计日志
  - 内容：事件发起者、发生时间、事件内容
  - 审计线程自动记录审计日志
  - 保存方法：记录到数据库表或操作系统文件
- 审计追踪
  - 审计文件按先后顺序进行标记
  - 以特定格式存储
- 统一审计
  - 通过规则在数据库内部有选择性执行有效审计

GaussDB数据安全管理



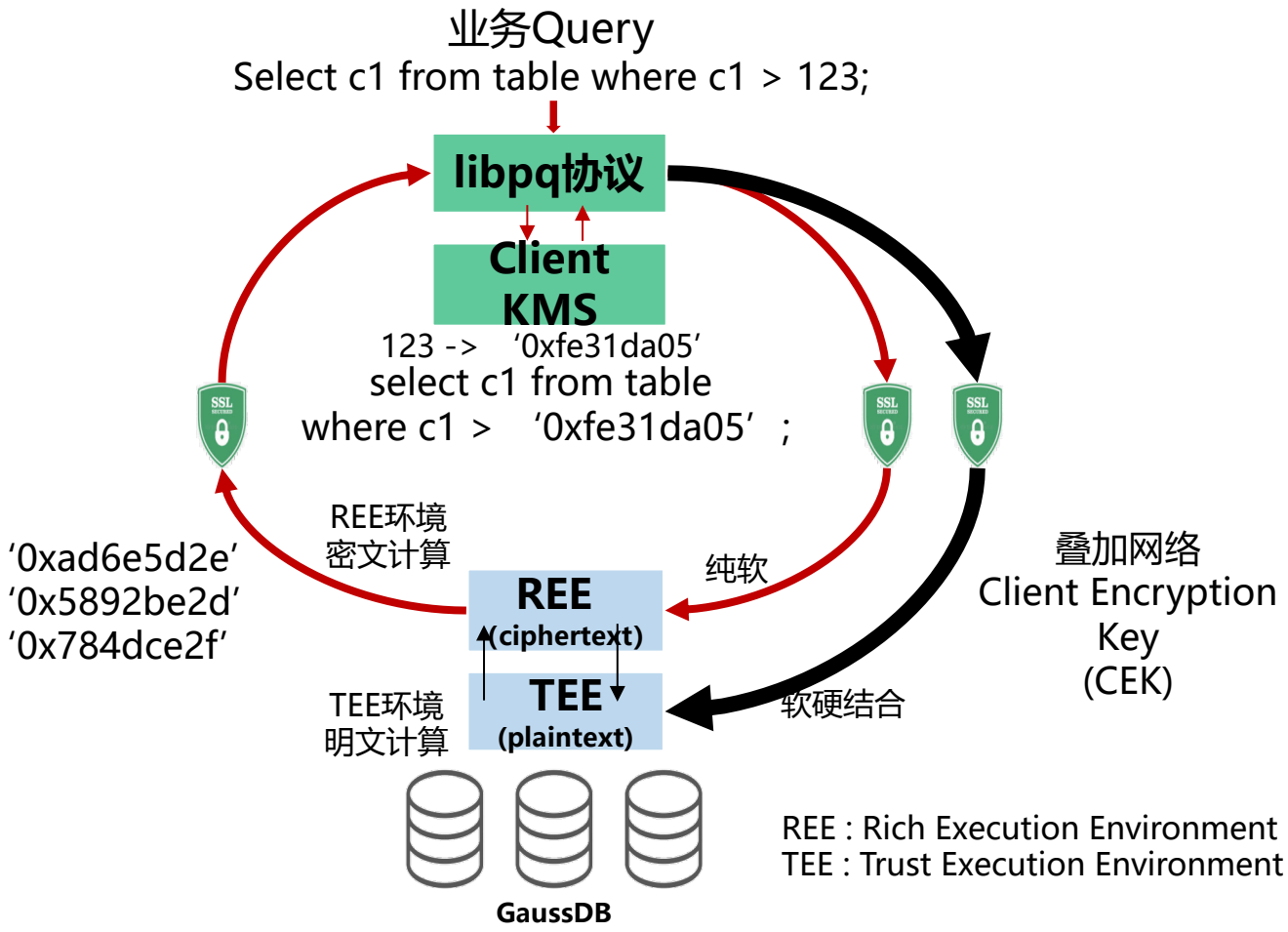
GaussDB数据加密

- 数据加密
  - 数据经过加密后以密文形式存储
- 数据脱敏
  - 对敏感信息进行特殊处理，掩盖敏感数据
  - 主要使用动态脱敏技术
- 透明加密
  - 全程加密更好地保护静态存储的数据
  - 防止第三方人员绕过认证机制
  - 数据库密钥由系统密钥管理系统生成
  - 数据库密钥密文以文件存储于数据库系统中
  - 数据进入到数据库系统后一直出于加密状态

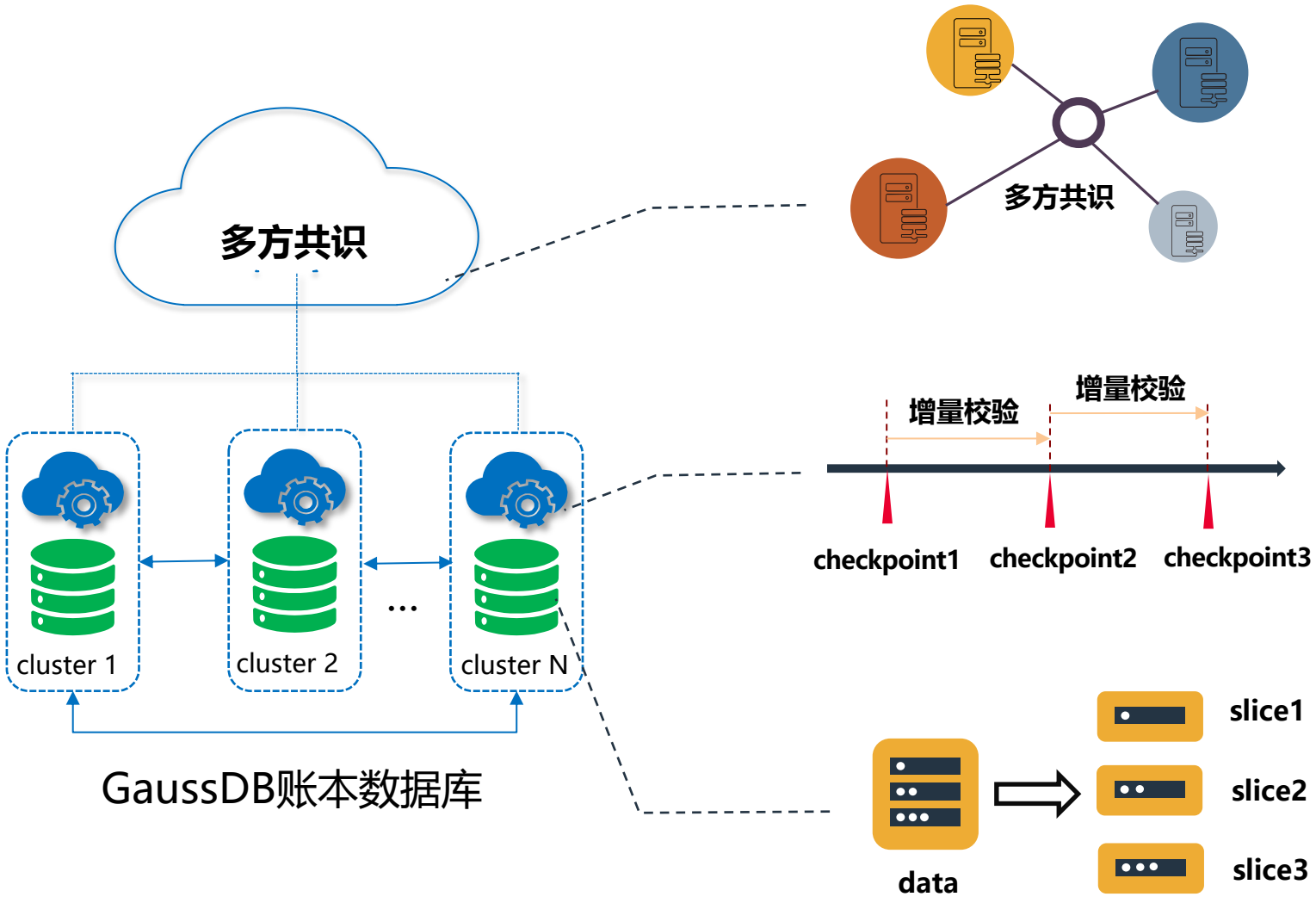


GaussDB全密态

id	c1 (plaintext)	c1 (ciphertext)
C10001	256	0xad6e5d2e
C10002	157	0x5892be2d
C10003	97	0x124f4ed2
C10004	685	0x784dce2f
C10005	58	0x324f4ed2



GaussDB防篡改



多方共识防篡改：

数据的修改需要取得参与节点的一致。

挑战：高性能共识协议。

高效篡改校验算法：

保证用户在不接触敏感信息的条件下校验数据库中数据的一致性和完整性。

挑战：高性能、高并发的篡改校验算法

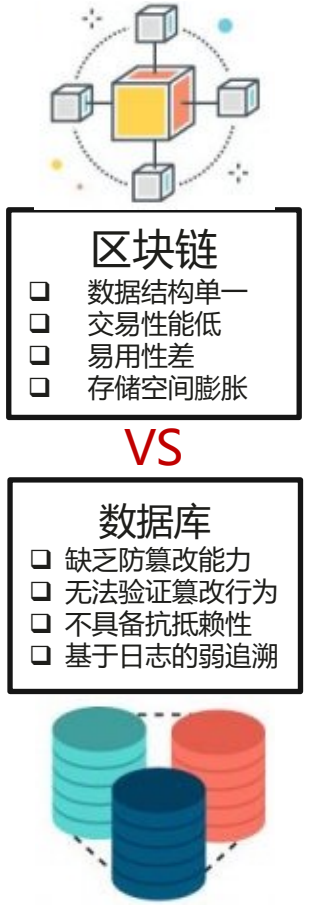
账本数据存储：

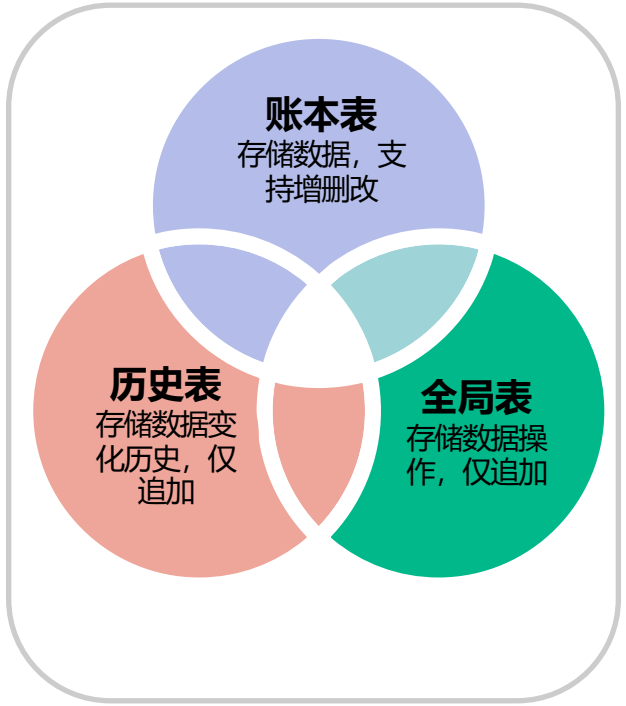
忠实记录用户数据操作的完整信息，保证操作链可追溯。

挑战：保证追溯信息充分的情况下降低额外存储开销

GaussDB防篡改

- 防篡改账本数据库：使用Multi-Set方式生成数据及SQL操作的校验数据，提供高性能、强追溯的防篡改
- 对数据增删改操作，系统会记录行级数据变化追溯信息及操作的追溯信息，并通过算法高效生成校验信息
- 通过对校验信息的保护和校验，起到对重要数据的完整性保护，识别、阻止未授权更改





防篡改校验数据

目录

GaussDB智能概览

原生智能内核

1 智能优化器

- 智能重写
- 统计信息推荐
- 智能代价
- 智能路径搜索
- 自适应计划
- 智能基数估计

2 库内原生智能引擎

- 参数自优化
- 分布式模型训练和推理
- 分布式模型管理
- 分类算法
- 回归算法
- 降维算法
- 聚类算法

智能运维平台

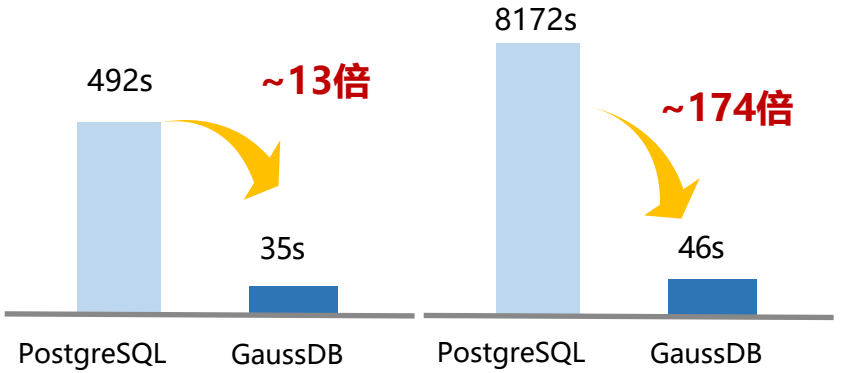
智能计算平台

- 人工智能算法库
- 监控数据管理

3 智能服务平台

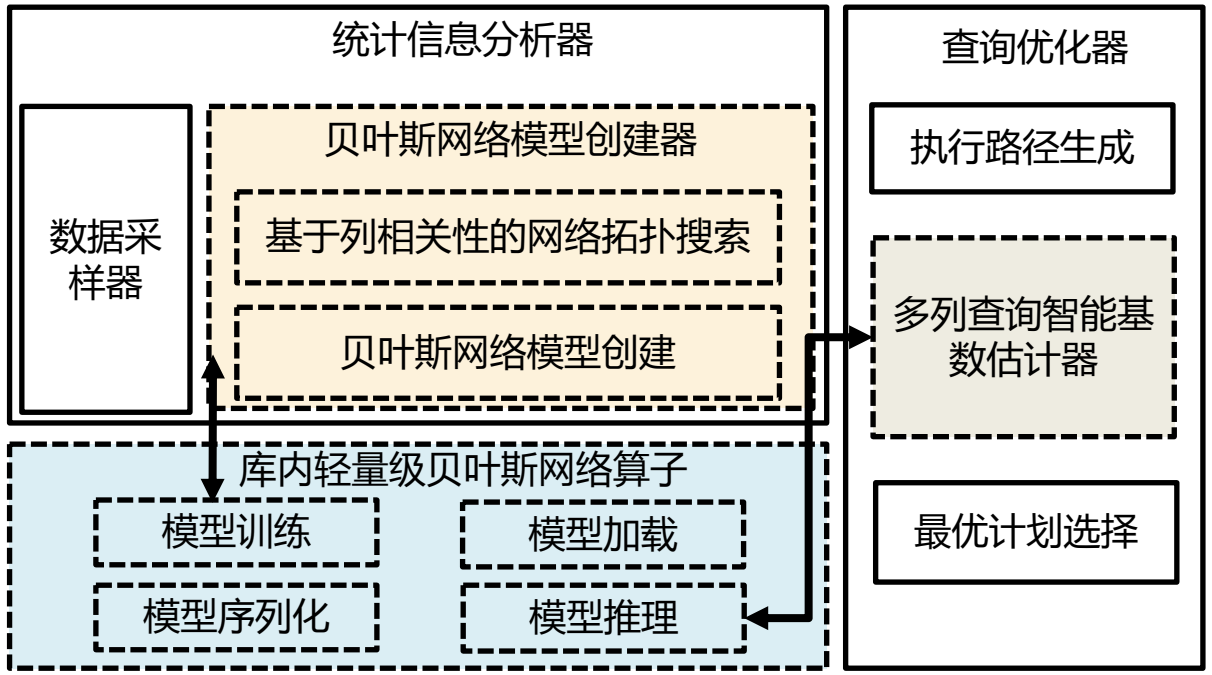
- 自监控
- 自诊断
- 自优化

智能优化

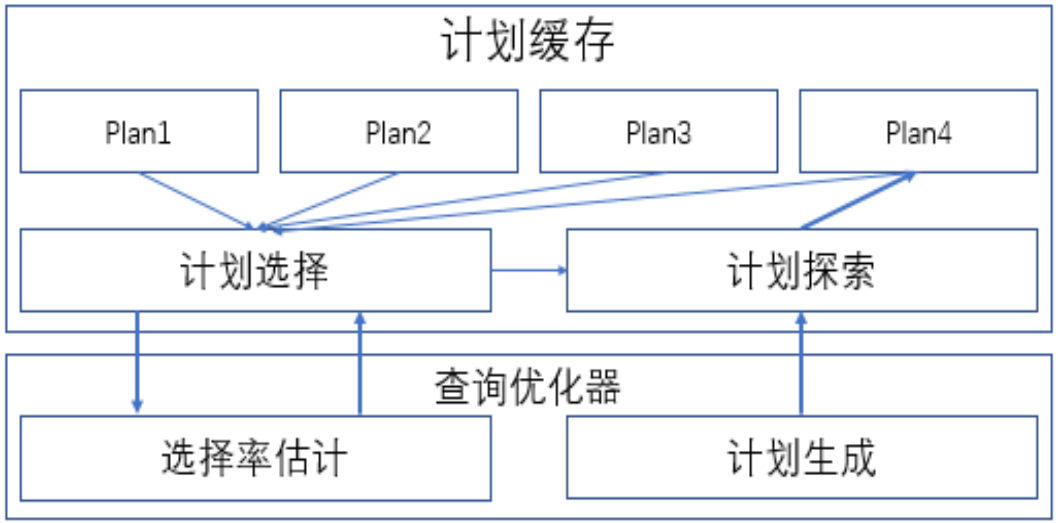


关键技术	应用场景	实际效果
智能索引推荐	**银行	性能 12% 冗余索引少83%
	**银行	性能 36%
智能分布列推荐	**银行	性能 15%
	**保险	性能 26%
	**银行	性能 20%
智能参数优化	**云	性能 32%
系统根因分析	**银行	效率 5倍
	**云	诊断率 90%

GaussDB智能优化器

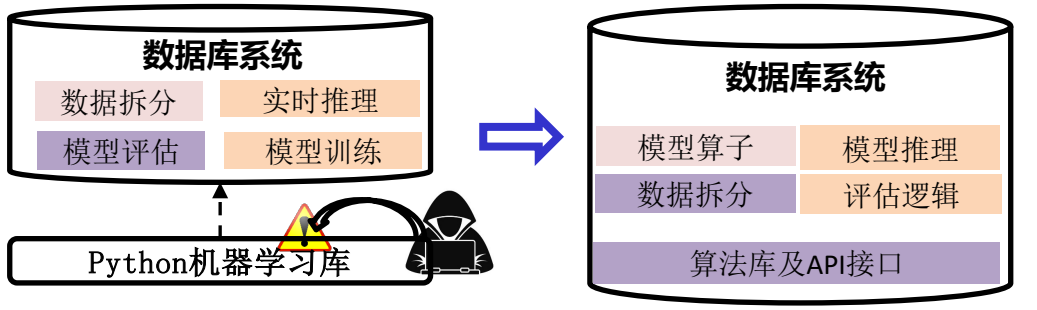


智能基数估计：利用贝叶斯网络对数据样本分布进行建模



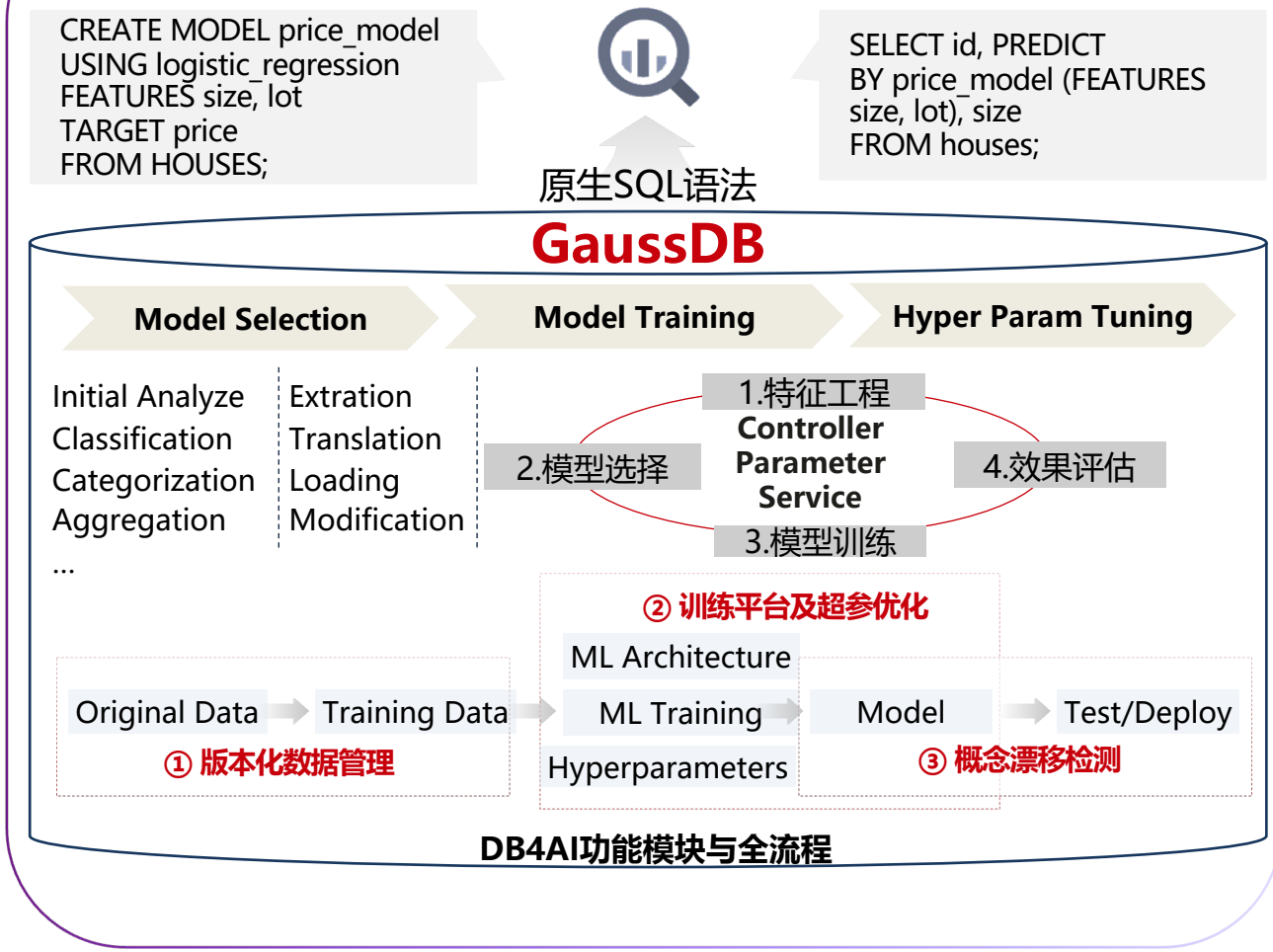
自适应计划选择：基于多版本探测的缓存计划自适应选择

GaussDB库内原生AI引擎

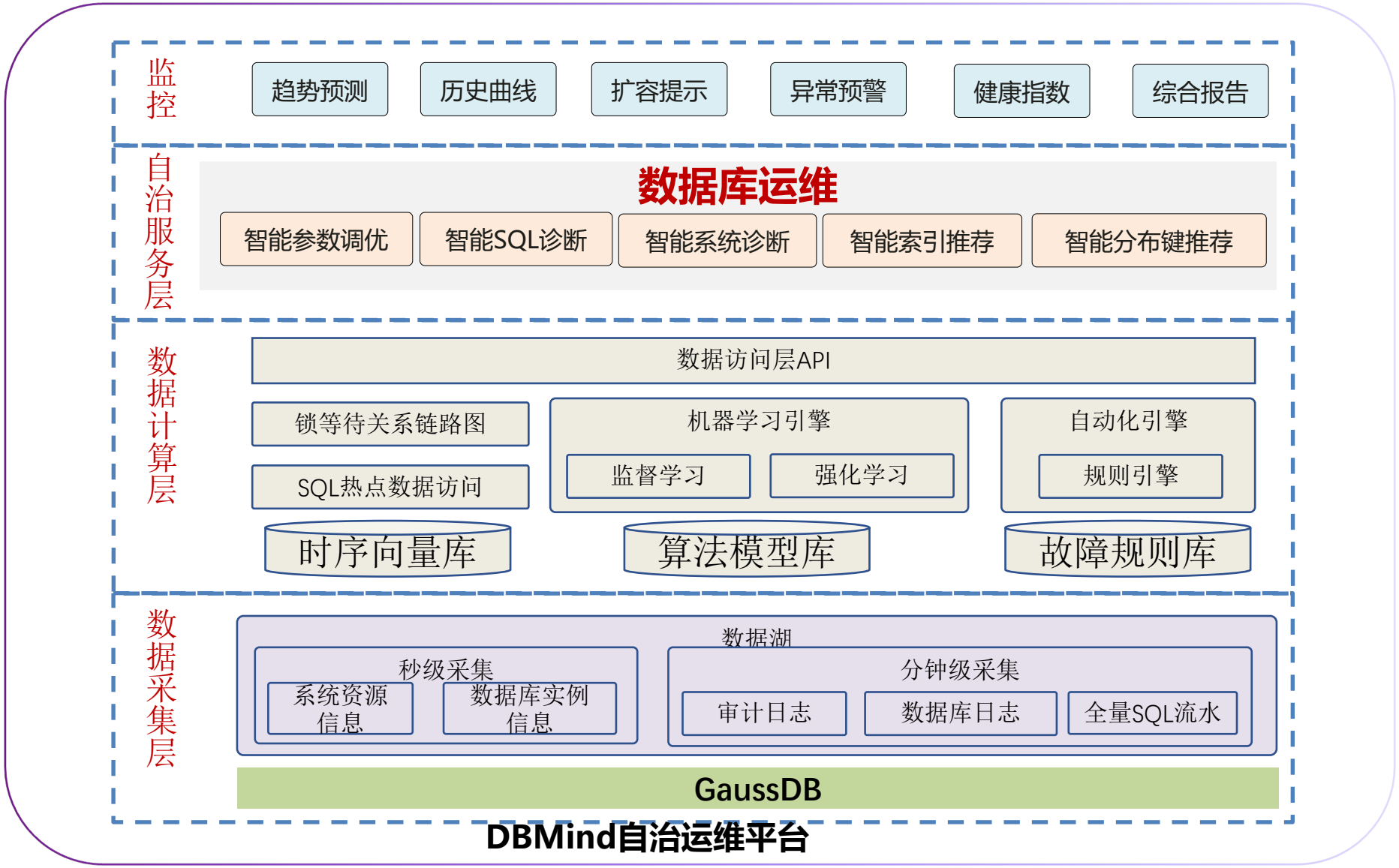


数据库集成AI算法库（PG14）　　数据库和AI融合(GaussDB)

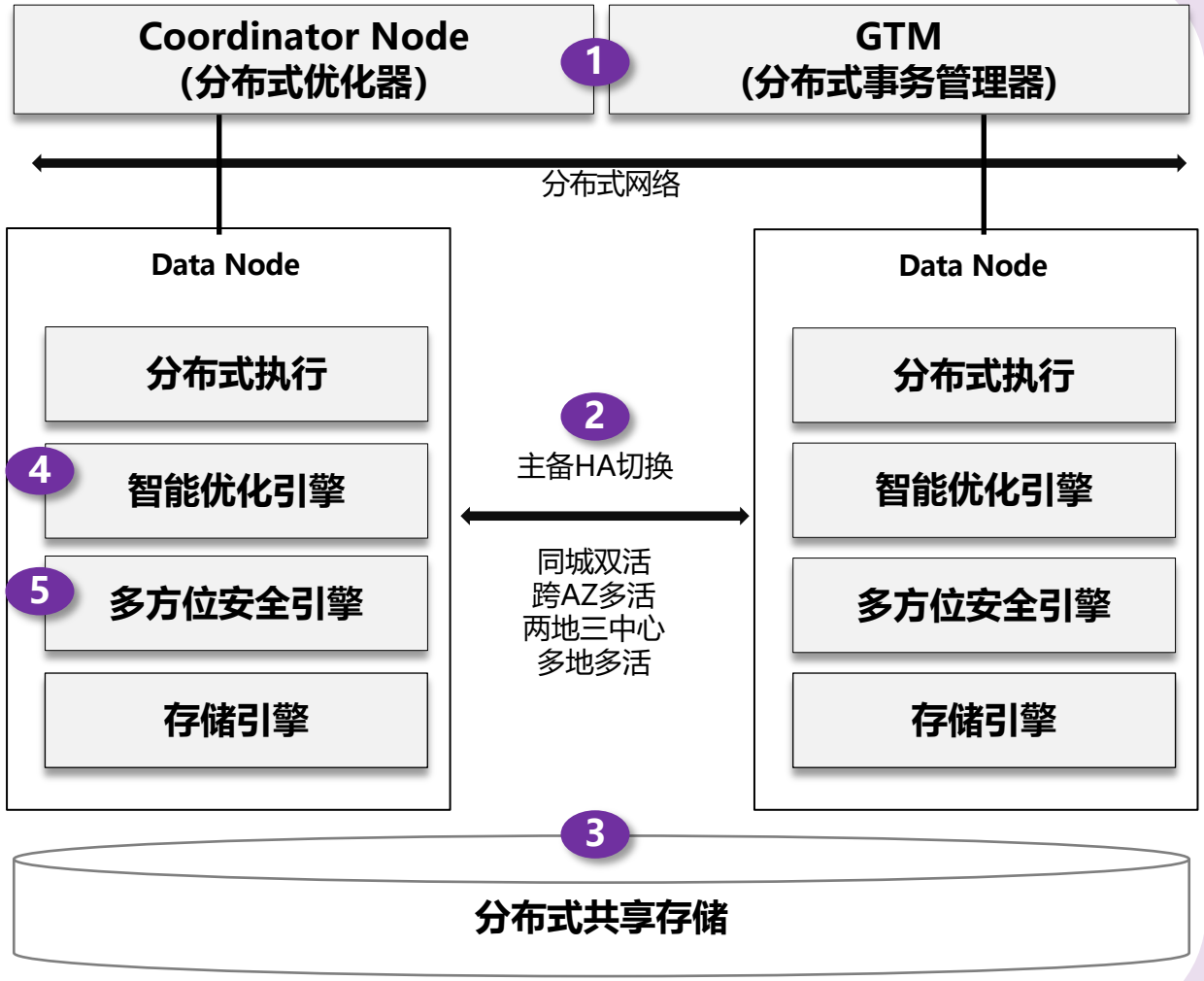
- 原生SQL-like语法，无需学习Python和AI知识，方便数据科学家使用；
- 构建原生AI执行算子，并行训练，提升执行效率；
- 数据库内全流程管理能力，支持数据清理、模型管理、超参优化和模型漂移，简化使用代价。



GaussDB自治运维平台



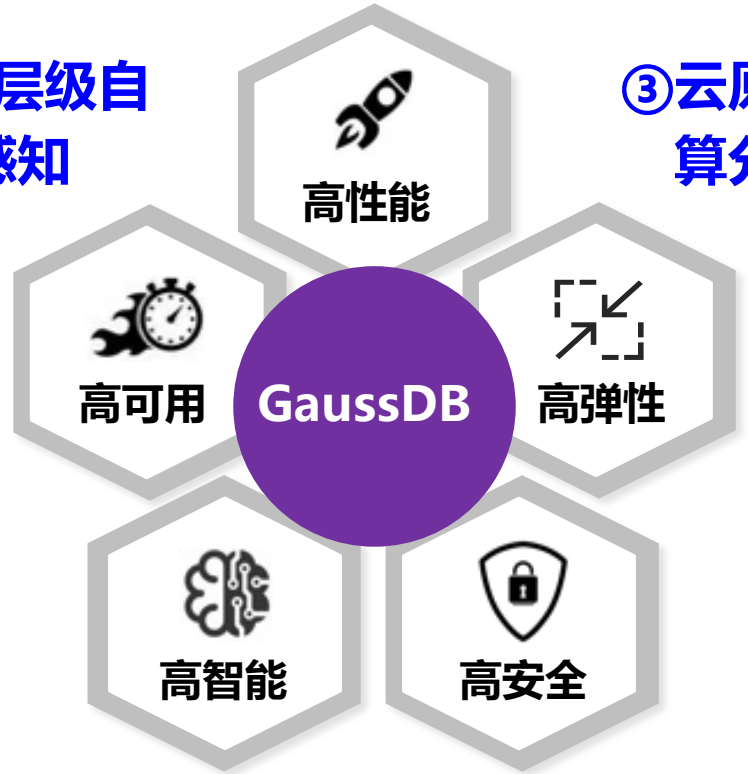
GaussDB小结



① 分布式并行

②多层级自感知

③云原生存算分离



④智能优化引擎

⑤全密态、防篡改