OPENGAUSS 体系架构

openGauss社区布道师/Maintainer 朱金伟

openGauss

https://opengauss.org

openGauss 目录

- **openGauss 开源社区介绍**
- openGauss 架构
- openGauss 核心技术及实践
- openGauss 未来技术发展方向

openGauss

https://opengauss.org

华为计算商业策略：硬件开放、软件开源、使能合作伙伴

伙伴 华为

应用
中间件
AI框架
数据库
操作系统
整机
部件
主板
华为云
处理器

使能合作伙伴

用三年时间让90%的软件跑在鲲鹏上

- 支持合作伙伴应用和软件的迁移

软件开源

将软件利润重分配给新ISV，重新分配软件价值链

- 开源AI 框架MindSpore
- openLooKeng开源数据虚拟化引擎
- openGauss开源数据库（OLTP单机版），构筑鲲鹏数据库生态
- openEuler开源操作系统，使能伙伴操作系统商业发行

硬件开放

降低整机门槛，打破现有格局，重新分配整机价值链

- 提供鲲鹏主板，SSD/网卡/RAID卡等部件，使能伙伴发展自有品牌部件、服务器和PC等

华为聚焦

- 鲲鹏处理器研发，全场景芯片，鲲鹏云服务

openGauss

https://opengauss.org

华为GaussDB演进历程

内部自用孵化阶段→联创产品化阶段→共建生态

2001-2011

- 企业级内存数据库

2011-2019

- G行核心数据仓库、DWS华为云商用
- Z行核心业务系统替换商业数据库
- 支撑公司内部40+主力产品，在全球70+运营商规模商用3万+套，服务全球20+亿人口。

2019-2020

- 2019.5.15 GaussDB全球发布
- 构筑合作伙伴生态
- 兼容行业主流生态，完成金融等行业对接
- 2020.6.30 openGauss开源。

2021~

- 分享企业级数据管理能力
- 引领生态建设
- 促进数据库教育事业发展

openGauss

https://opengauss.org

openGauss产品：商用+自用+开源相结合，内核将长期演进

华为公司内部配套、公有云的GaussDB、开源openGauss 共代码基线

https://opengauss.org

openGauss社区：共建生态，共议发展方向

2019.9.19	2020.06.30	2020.08	2020.09	2021.03	2021~
openGauss宣布开源	openGauss源代码正式开放	openGauss社区首届Meetup	HC大会伙伴案例发布	社区首个LTS版本发布	5个城市用户组社区理事会筹备组成立

openGauss社区组织架构

委员会

SIG(18个)

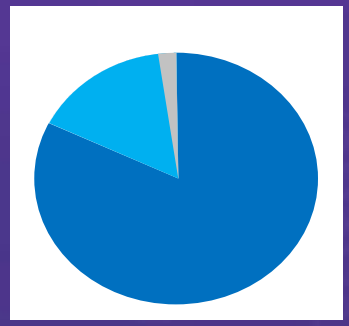
- SQLEngine
- StorageEngine
- Connectors
- Docs
- Infra
- Security

openGauss社区角色

- TC 决策社区技术发展方向
- Maintainers 看护项目代码
- Committers 看护代码质量和合入
- Contributors 需求、Bug反馈和开发实现

开源协议类型：Mulan PSL v2

openGauss社区运营进展



- 官方网站访问量：110万+；官方网站访客人数：12万+；
- 官方网站安装包下载量：30万+
- 微信公众号关注人数：1630；微信社群人数：718
- 社区用户D0（安装包下载IP数）：1968
- 社区参与者D1（Star、Fork、Watch）：722
- 社区贡献者D2（提交Issue和PR）：412

openGauss

https://opengauss.org

openGauss 定位

把企业级数据库能力带给用户和伙伴

价值

openGauss提供面向多核的极致性能、全链路的业务和数据安全、基于AI的调优和高效运维的能力，全面友好开放，携手伙伴共同打造全球领先的企业级开源关系型数据库；

关键特性

高性能

- 两路鲲鹏性能150万tpmC
- ① 面向多核架构的并发控制技术；
- NUMA-Aware数据结构；
- SQL-Bypass智能选路执行技术；
- ④ 面向实时高性能场景的内存引擎；

高可用 & 高安全

- ② 业务无忧，故障切换时间RTO<10s；
- 精细安全管理：细粒度访问控制、多维度审计；
- ⑤ 全方位数据保护：存储&传输&导出加密、动态脱敏、全密态计算；

易运维

- ③ 基于AI的智能参数调优，提供AI参数自动推荐；
- 慢SQL诊断，多维性能自监控视图，实时掌控系统性能表现；
- 提供在线自学习的SQL时间预测、快速定位，急速调优；

全开放

- 采用木兰宽松许可证协议，允许对代码自由修改，使用、引用；
- 数据库内核能力完全开放；
- 开放运维监控、开发和迁移工具；
- 开放伙伴认证、培训体系及高校课程

openGauss

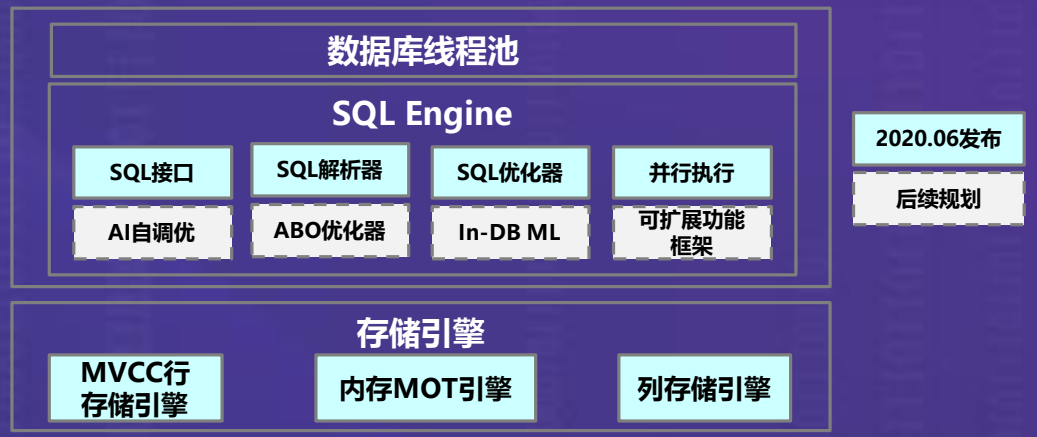
https://opengauss.org

openGauss 目录

- openGauss 开源社区介绍
- **openGauss 架构**
- openGauss 核心技术及实践
- openGauss 未来技术发展方向

openGauss 体系架构

openGauss架构

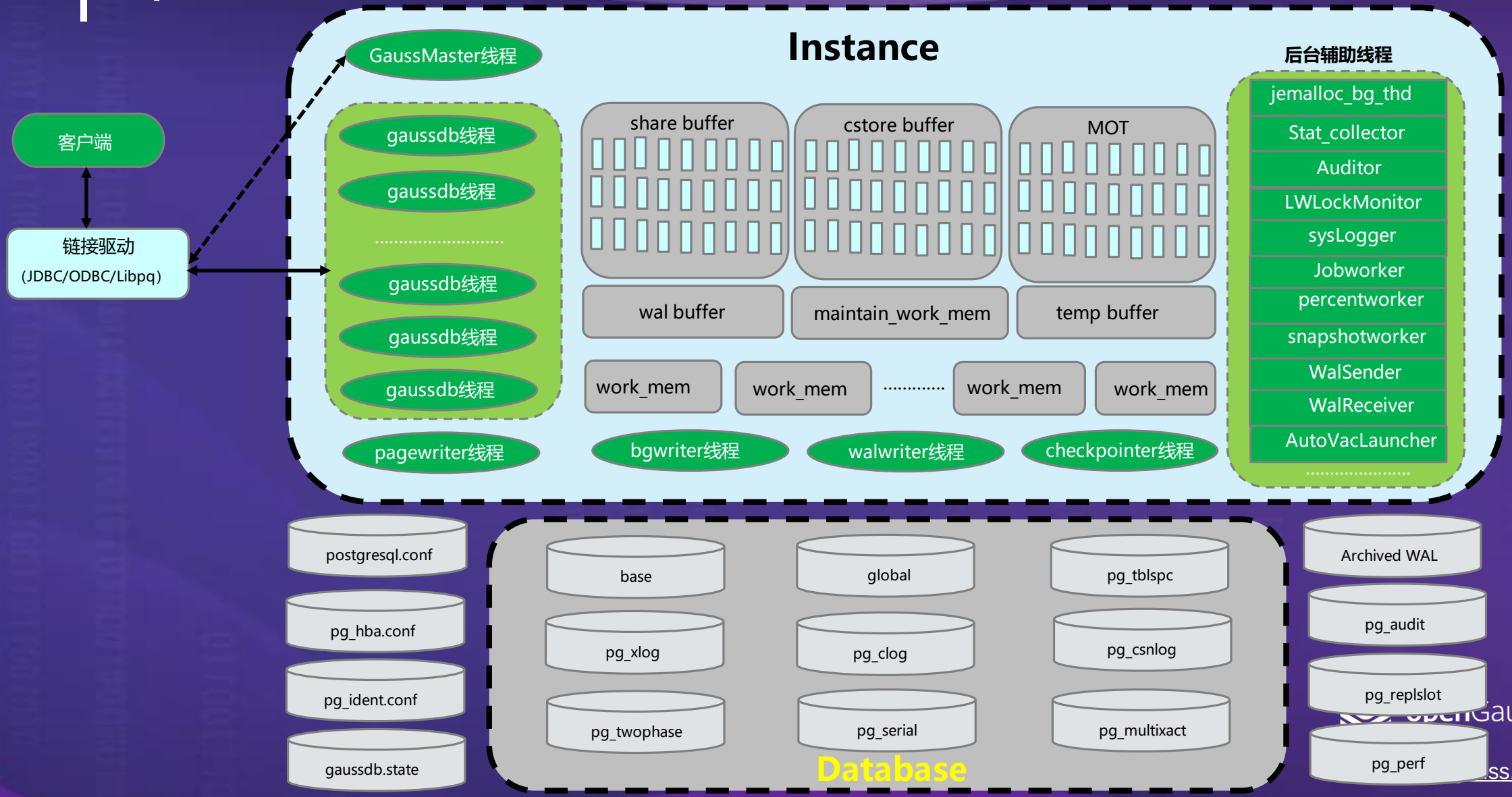


关键差异化因素		openGauss
运行时模型	执行模型	线程池模型，高并发连接切换代价小、内存损耗小，执行效率高，一万并发连接比最优性能损耗<5%
事务处理机制	并发控制	64位事务ID，使用CSN解决动态快照膨胀问题; NUMA-Aware引擎优化改造解决“五把大锁”
	日志和检查点	增量Checkpoint机制，实现性能波动<5%
	鲲鹏NUMA	NUMA改造、cache-line padding、原生spin-lock
数据存储与组织	多引擎	行存、列存、内存引擎，在研DFV存储和原位更新
查询优化器	优化器	支持SQL Bypass, CBO吸收工行等企业场景优化能力
	SQL解析	ANSI/ISO标准SQL92、SQL99和SQL2003 和企业扩展包

openGauss

https://opengauss.org

openGauss 体系结构



openGauss

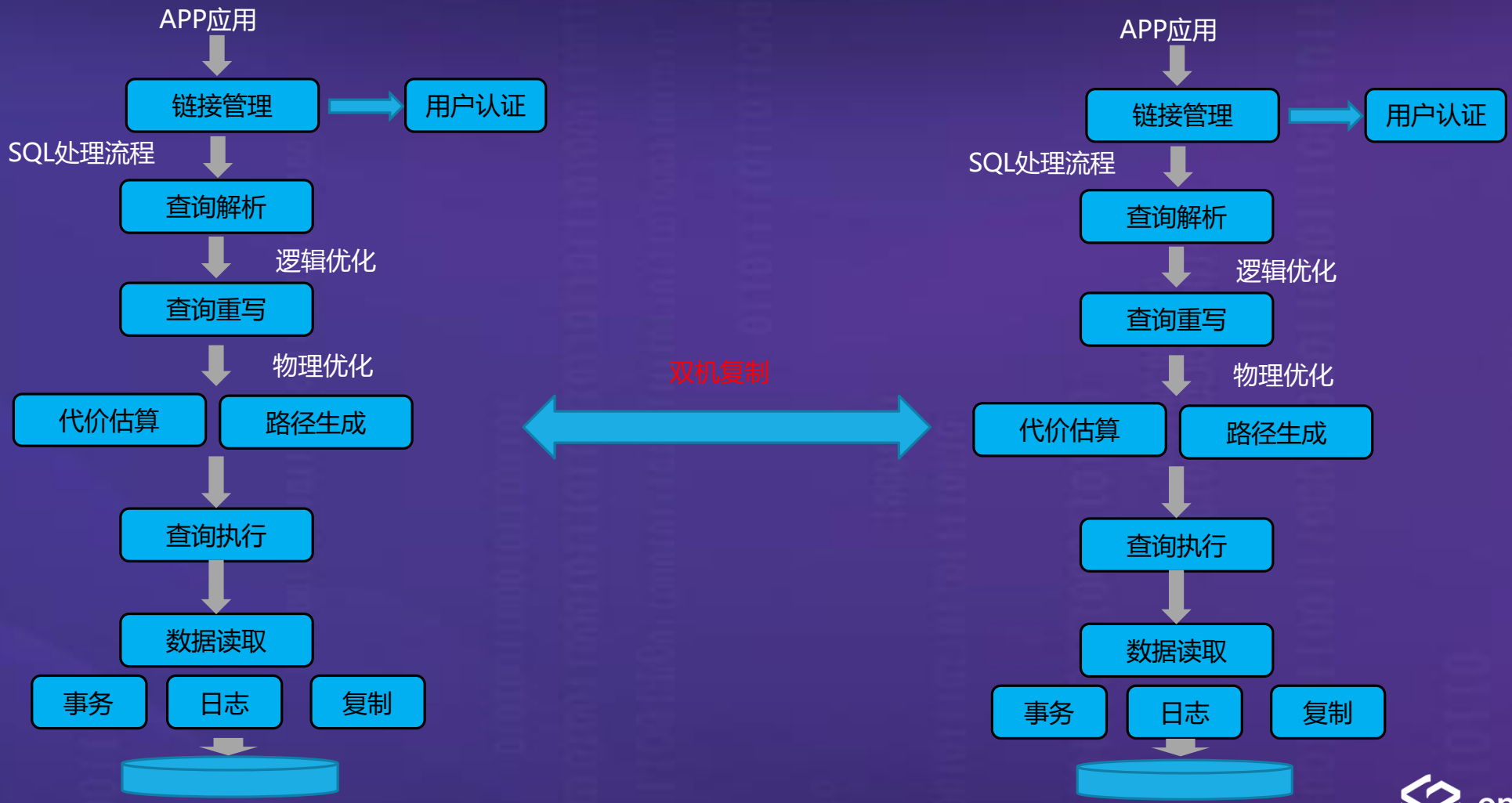
ss.org

openGauss 逻辑模块

openGauss

https://opengauss.org

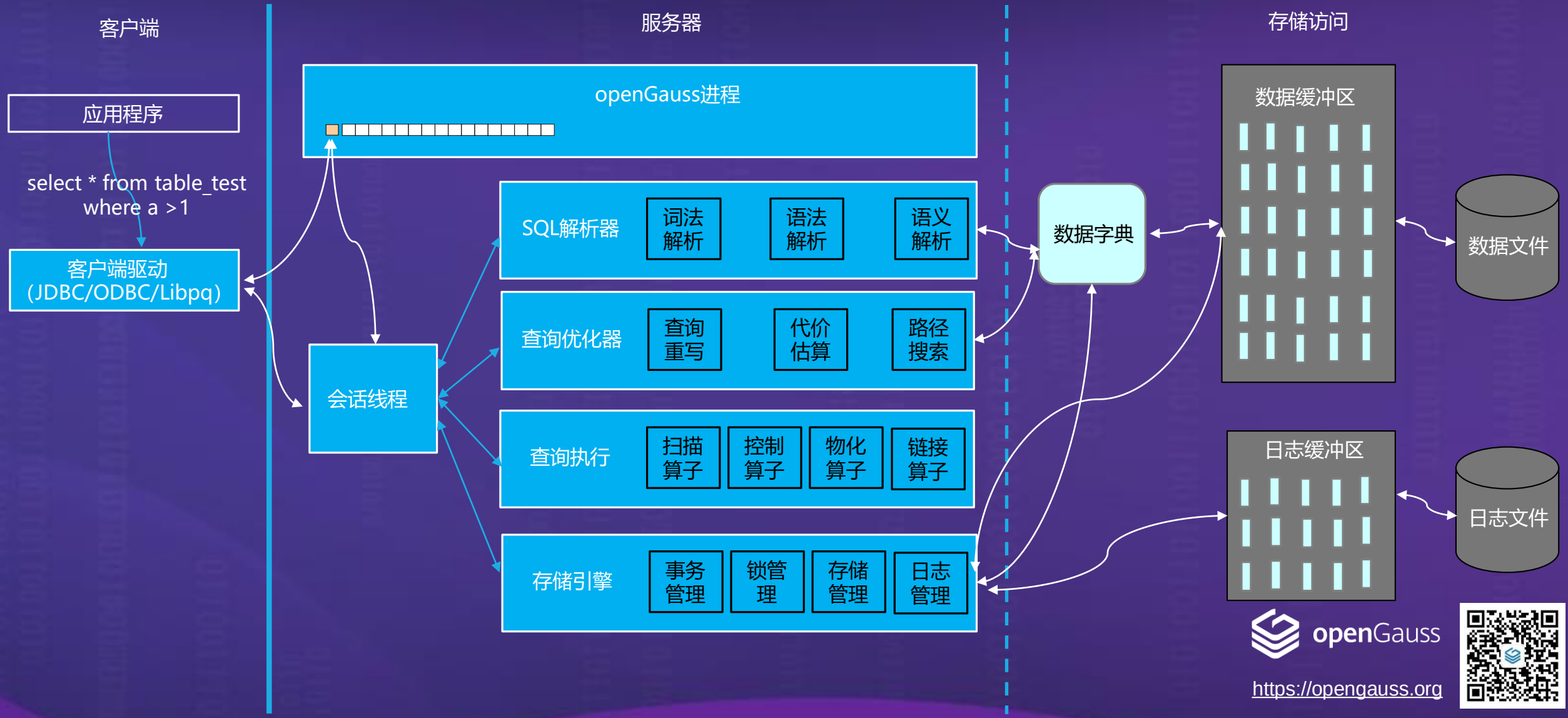
openGauss SQL命令处理流程



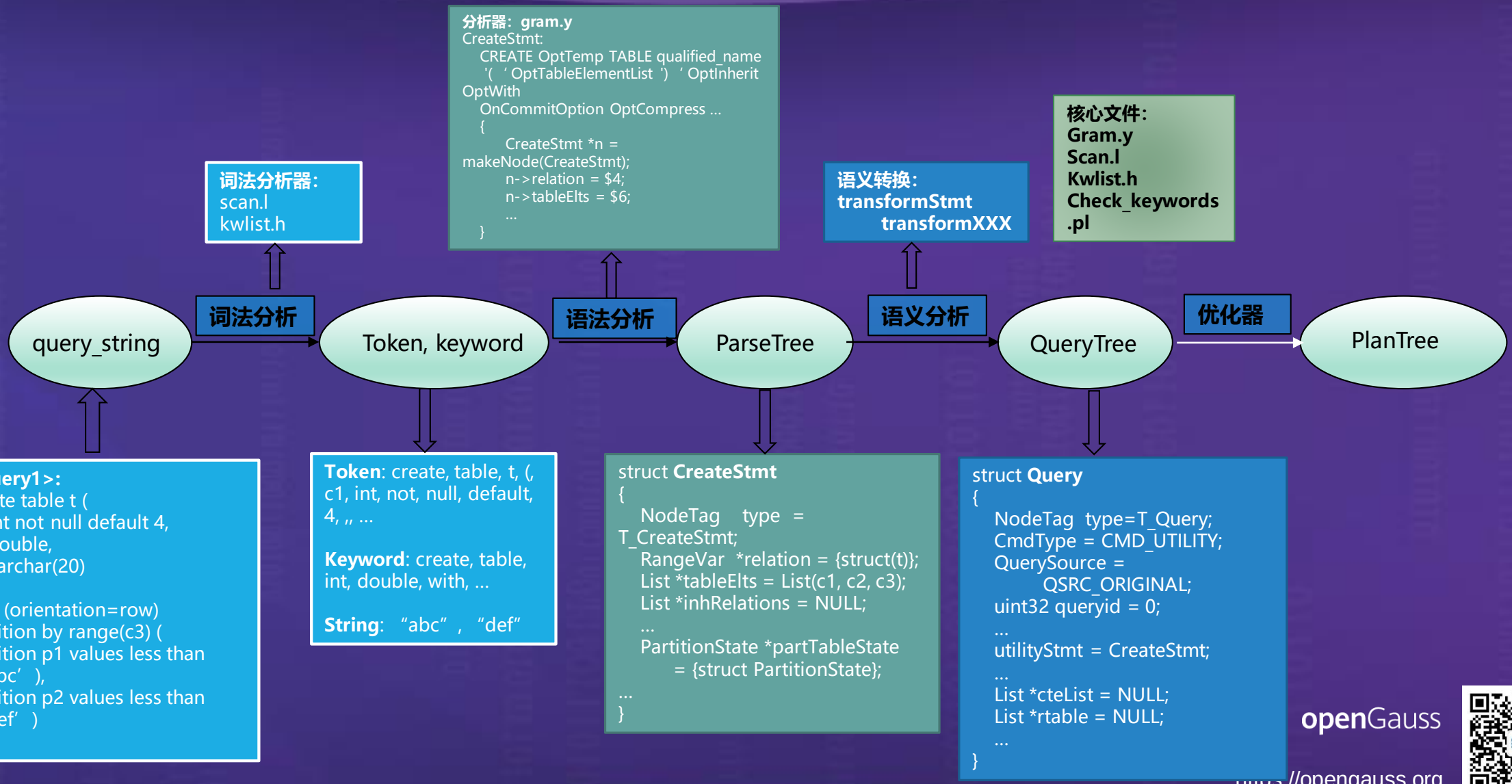
openGauss

https://opengauss.org

openGauss SQL命令处理流程示意图



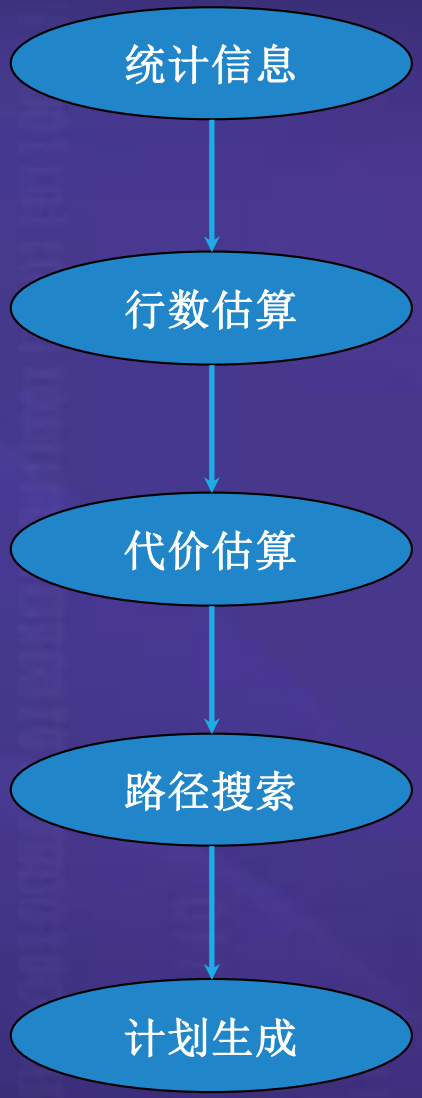
openGauss SQL引擎



openGauss

https://opengauss.org

openGauss 查询优化



物理优化的依据，来源于表信息的统计

代价估算的基础，来源于基表统计信息的推算

根据关系的行数，推算出当前算子的执行代价

依据若干算子的执行代价，通过求解最优路径进行路径搜索

将查询的执行路径转换成PlanTree能够输出给执行器做查询执行

物理优化的技术点：

1、统计信息模型 Table/Column-Level statistics

描述基表数据的特征包括唯一值、MCV值等，用于行数估算

2、行数估算 Row Estimation

估算基表baserel、Join中间结果集joinrel、Aggregation中结果集大小，为代价估算做准备

3. 代价估算 Cost Estimation

根据数据量估算不同算子执行代价，各算子代价之和即为计划总代价

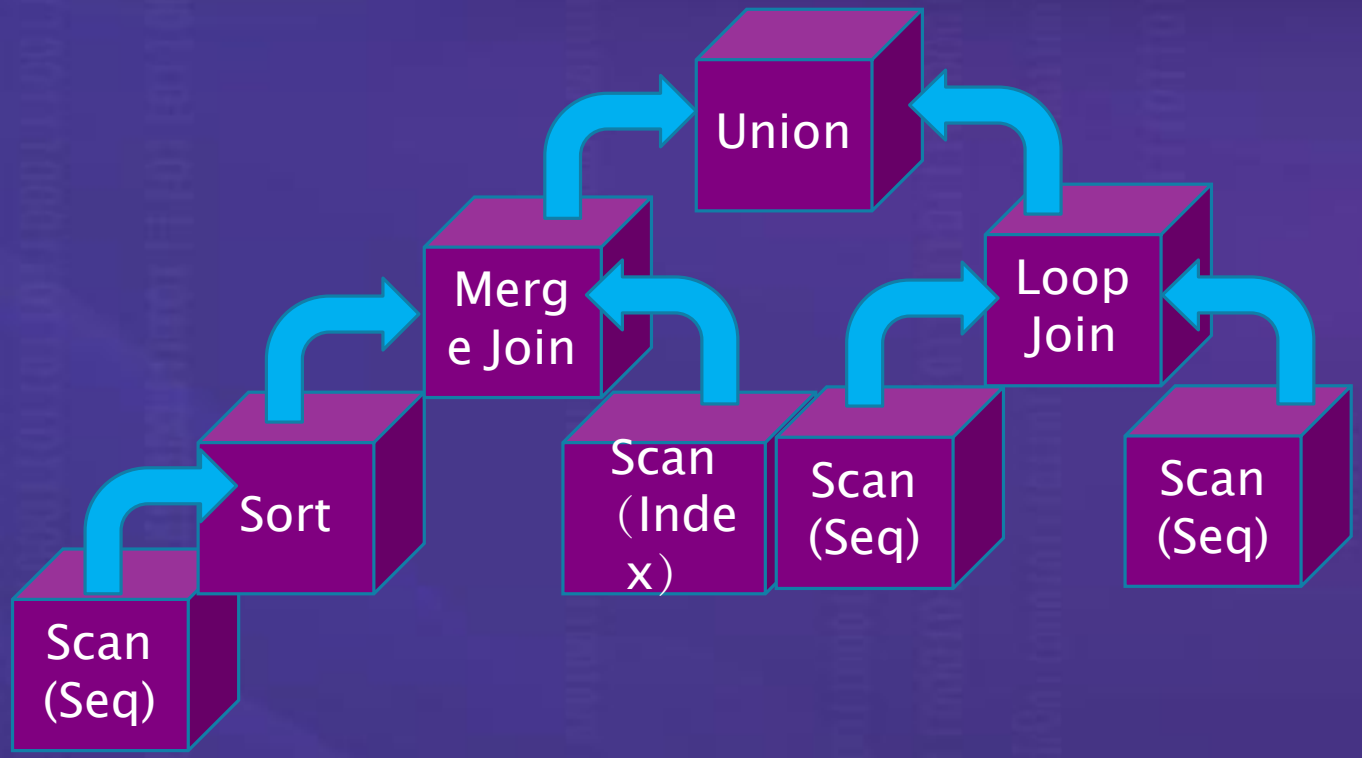
4. 路径搜索 Access Path Generation

通过求解路径最优算法（e.g. 动态规划、遗传算法）处理连接路径搜索过程，以最小搜索空间找到最优连接路径

openGauss

https://opengauss.org

openGauss 执行引擎



关系数据库本身是对关系集合Relation的运算操作，执行引擎作为运算的控制逻辑主要是围绕着关系运算来实现的，算子可以分成以下几类：

1. **扫描算子（Scan Plan Node）**

   扫描节点负责从底层数据来源抽取数据，数据来源可能是来自文件系统，也可能来自网络。一般而言扫描节点都位于执行树的叶子节点，作为执行的数据输入来源，典型代表SeqScan、IndexScan、SubQueryScan

   关键特征：输入数据、叶子节点、表达式过滤

2. **控制算子（Control Plan Node）**

   控制算子一般不映射代数运算符，是为了执行器完成一些特殊的流程引入的算子，例如Limit、RecursiveUnion、Union

   关键特征：用于控制数据流程

3. **物化算子（Materialize Plan Node）**

   物化算子一般指算法要求，在做算子逻辑处理的时候，要求把下层的数据进行缓存处理，因为对于下层算子返回的数据量不可提前预知，因此需要在算法上考虑数据无法全部放置到内存的情况，例如Agg、Sort

   关键特征：需要扫描所有数据之后才返回

4. **连接算子（Join Plan Node）**

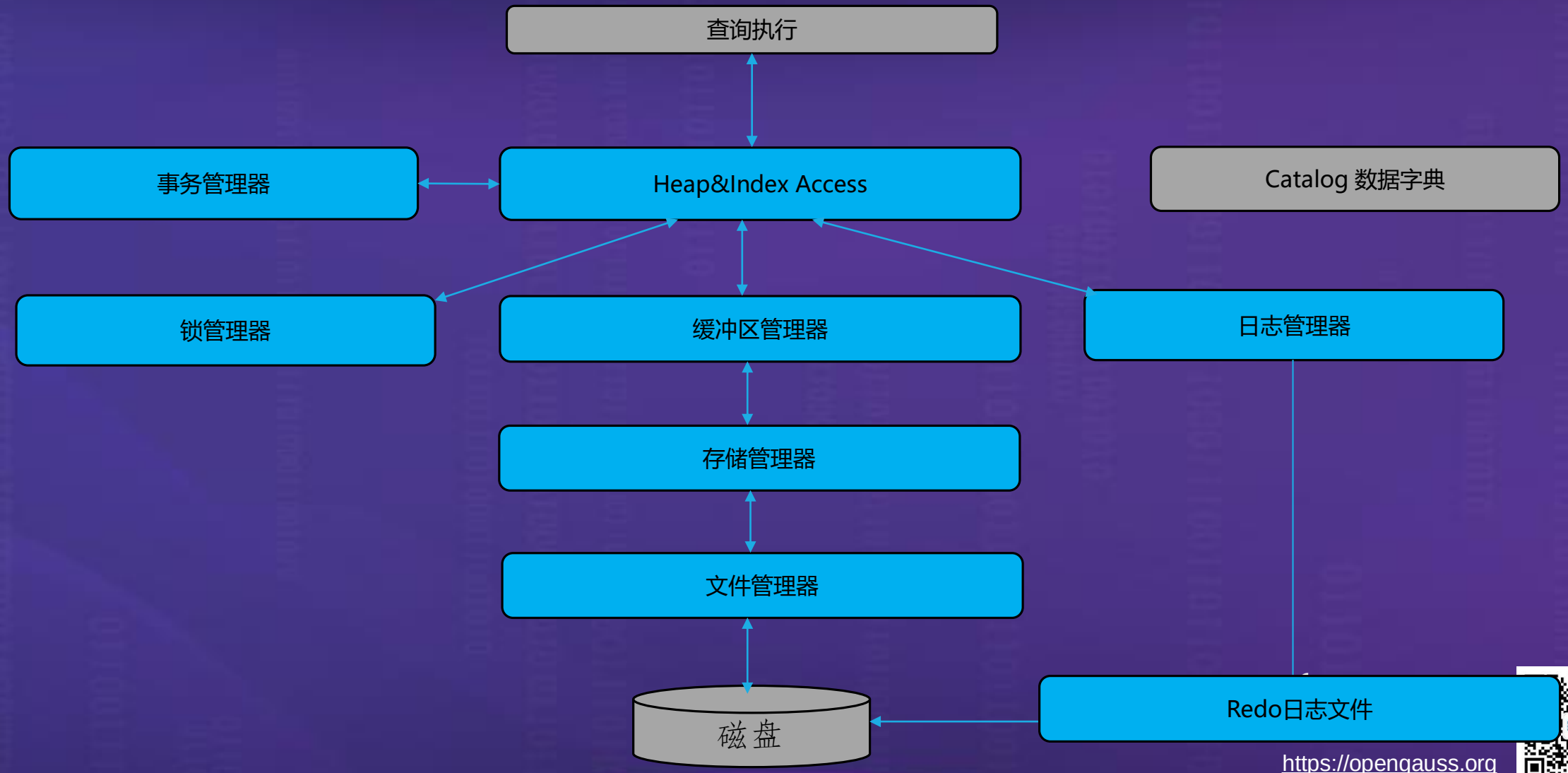
   这类算子是为了应对数据库中最常见的关联操作，根据处理算法和数据输入源的不同分成MergeJoin,SortJoin,HashJoin。

   关键特征：多个输入

openGauss

https://opengauss.org

openGauss 存储引擎



https://opengauss.org

openGauss 目录

- openGauss 开源社区介绍
- openGauss 架构
- **openGauss 核心技术及实践**
  - **高性能**
    - NUMA多核优化
    - 行列混合引擎
    - 线程池原理
    - MOT内存表原理
    - 增量检查点原理
    - 增量检查点原理
  - 可用性
    - 极致RTO优化
  - 安全性
    - 全密态等值查询原理
  - 易运维
    - AI4DB和DB4AI
- openGauss 未来技术发展方向

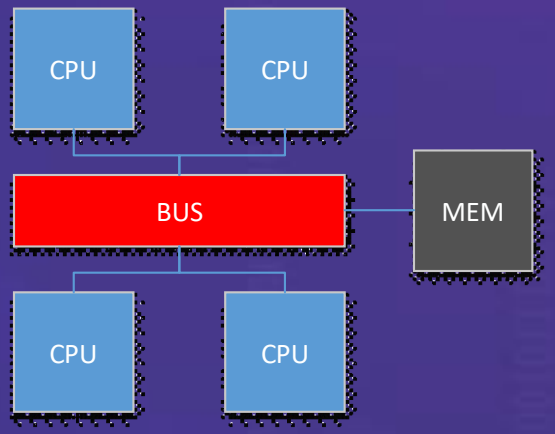
openGauss

https://opengauss.org

SMP概念

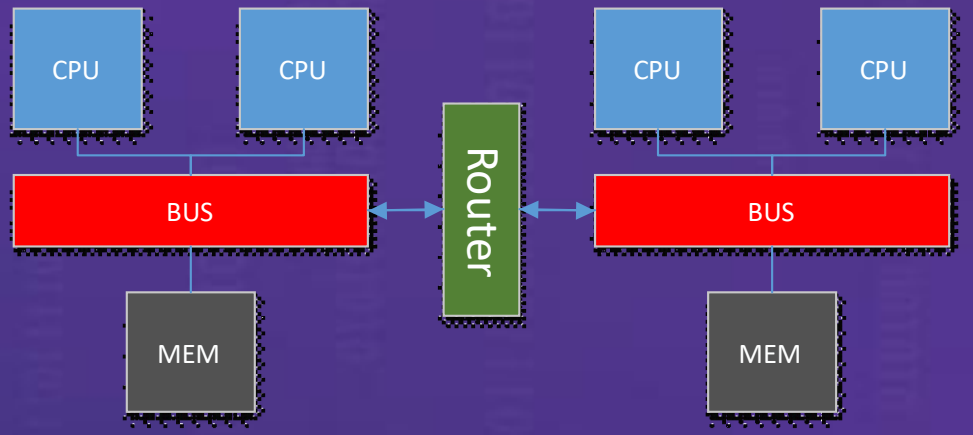
- SMP(Symmetric Multi Processing)：对称多处理器；
- NUMA(Non-Uniform Memory Access)：非一致存储访问结构；

SMP



- 所有的CPU共享全部资源，如总线，内存和I/O系统等。
- 操作系统管理着一个队列，每个处理器依次处理队列中的进程。
- 并发访问同一资源，通过硬件、软件的锁机制解决资源争抢。
- 因为共享内存访问主线，导致SMP架构CPU核数存在上限。

NUMA

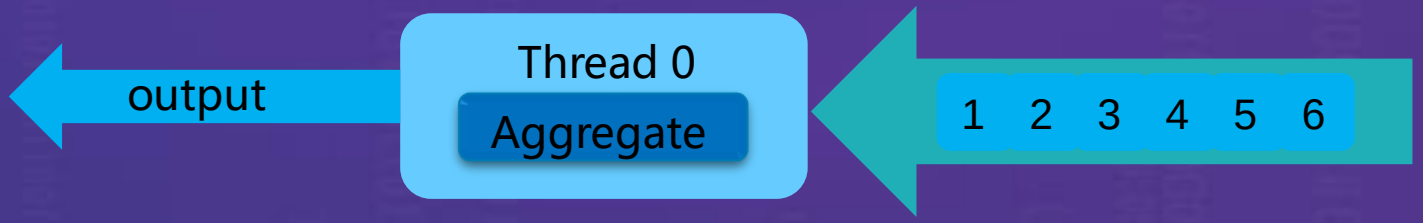


- 具有多个 CPU 模块，每个 CPU 模块由多个 CPU组成，并且具有独立的本地内存、 I/O 槽口等。
- 节点之间通过互联模块进行连接和信息交互，因此每个 CPU 可以访问整个系统的内存 。显然，访问本地内存的速度将远远高于访问远地内存（系统内其它节点的内存）的速度

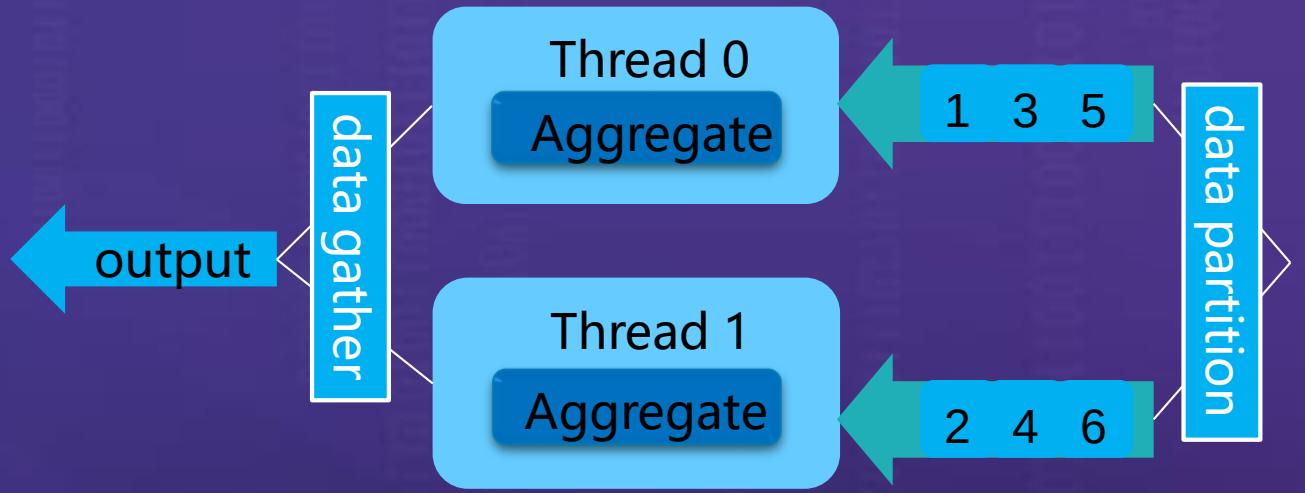
为什么要用SMP

- 提升scale up能力：多进程或多线程技术，充分利用现代服务器单机多核能力的软件架构
- 改变CPU利用率低： 当前服务器CPU核数40/48/60核，未来CPU核数进一步提升(众核架构)
- 定制化使用资源：针对不同的查询，不同的算子，选择不同的计算资源

串行执行：



并行执行：



openGauss

https://opengauss.org

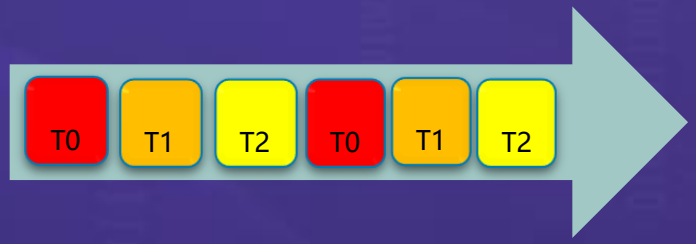
SMP并行架构

- 并行执行框架：为并行执行线程之间进行数据分配、交换和汇总(Scan类：磁盘；stream：网络)
- 并行计划生成：一阶段计划生成：在路径生成阶段，加入并行路径，最终根据代价，决定所选择的计划
  两阶段计划生成：第一步生成原有的串行计划，第二步在将串行计划改造成适应并行的计划

Scan类算子：

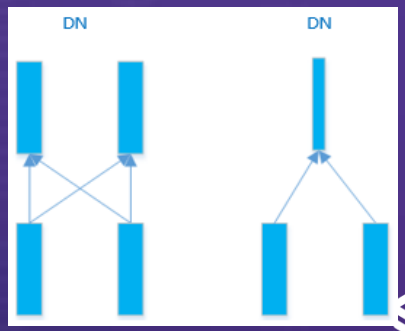
行存：page
列存：CU

数据采用round robin方式分配



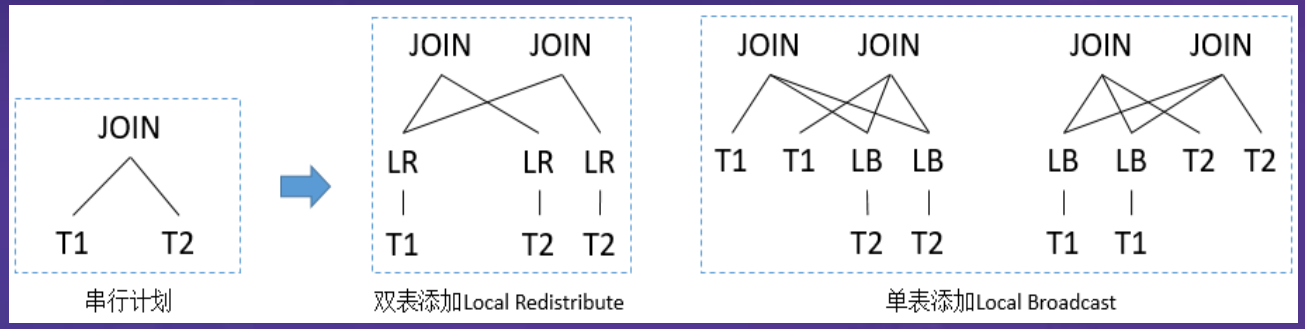
DN内数据分配：

- Local Redistribute：在节点内，按照hash值，在并行线程之间做数据重分布
- Local Broadcast：将数据广播到节点内的并行线程之中
- Local RoundRobin：将数据轮询的发送到节点内的并行线程之中
- Local Gather：把并行线程中的数据汇总到一个线程中

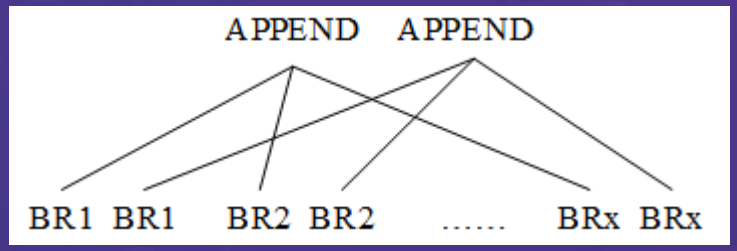


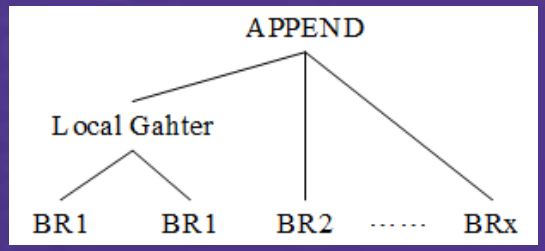
openGauss SMP示例

Join算子



Append算子

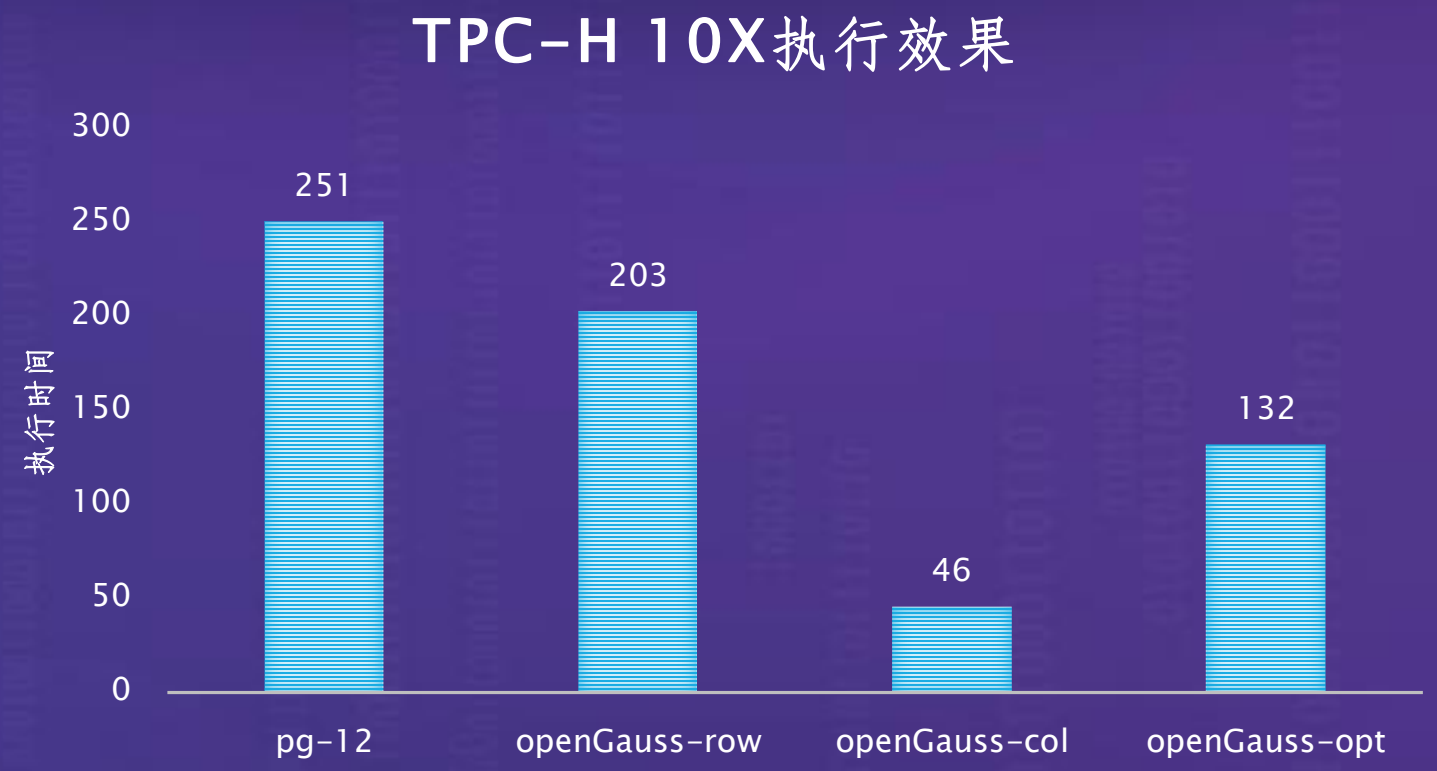




openGauss

https://opengauss.org

openGauss SMP应用效果

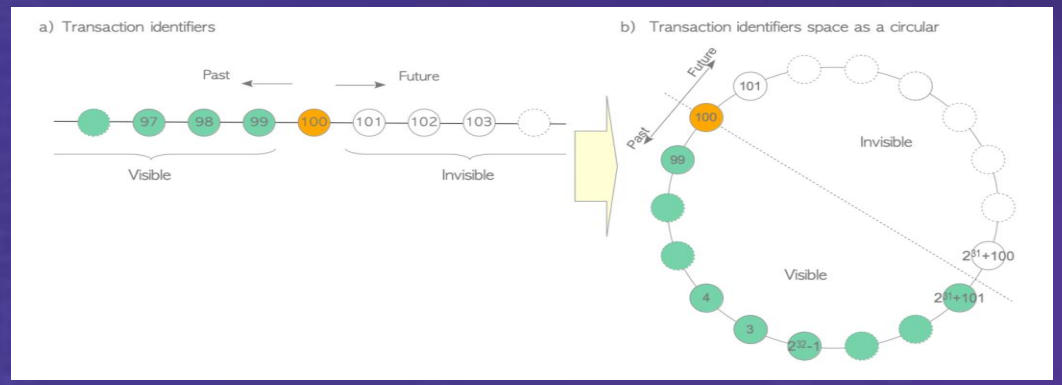


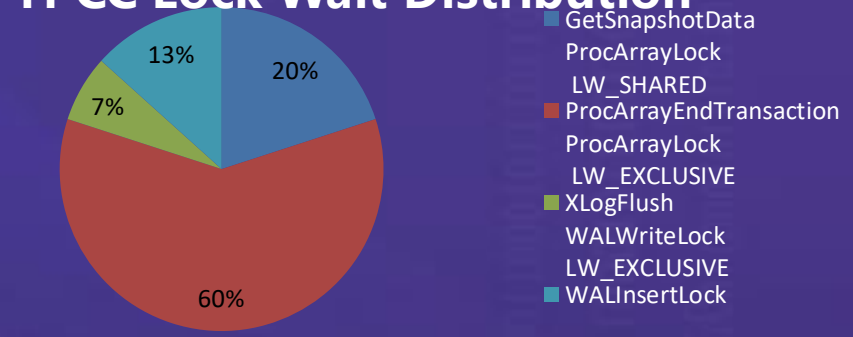
openGauss

https://opengauss.org

openGauss CSN快照

整改前：活跃事务ID快照



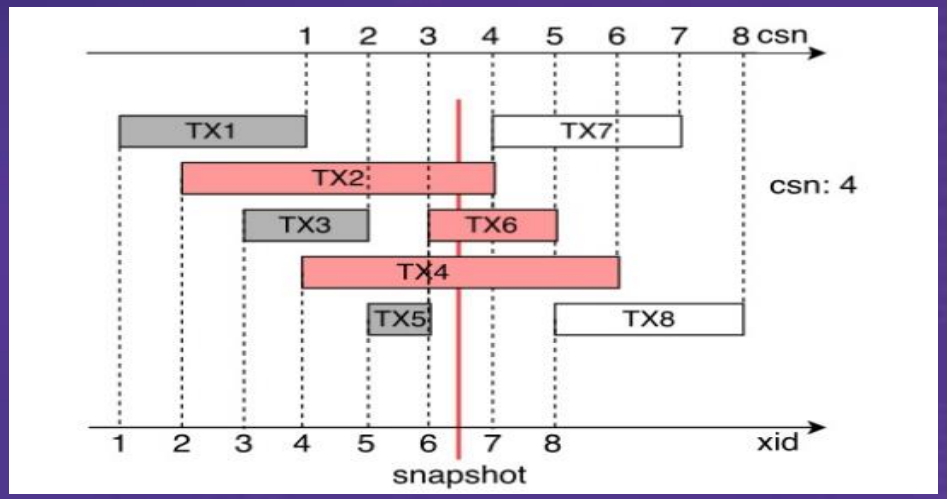


1、事务启动获取事务快照，需要获取ProcArrayLock。

2、事务结束，清理事务状态快照时，需要获取ProcArrayLock。

3、并发连接增大时导致全局事务管理器上获取的快照变大。

整改后：CSN提交快照



1、上图中每个非只读事务在运行过程中会取得一个XID，在事务提交时会推进CSN，同时会将当前CSN与事务的XID映射关系保存起来。

2、上图中，棕色竖线表示取snapshot时刻，如果不使用CSN方案，那么棕色竖线对应snapshot的集合应该是{2,4,6}。

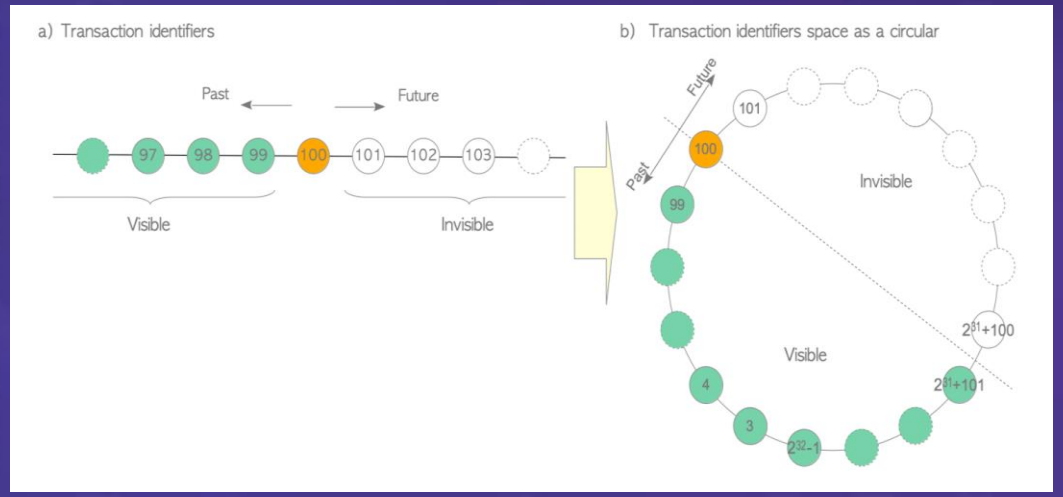
如果采用CSN方案，会获取当前的CSN值，也就是3，事务TX2、TX4、TX6、TX7、TX8的CSN分别为4、6、5、7、8，对于该snapshot而言，这几个事务的修改都不可

openGauss

https://opengauss.org

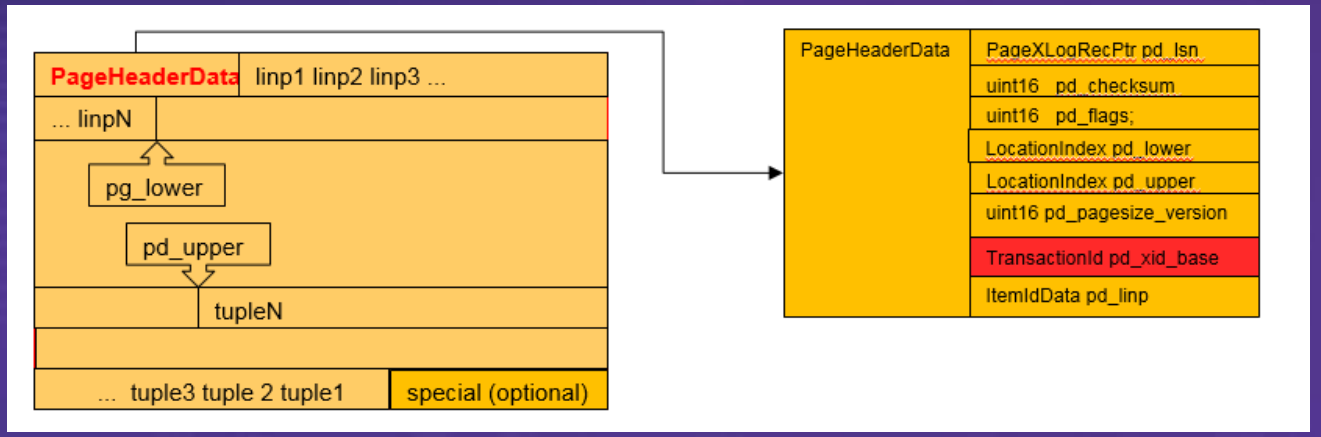
openGauss 64位事务ID

整改前：32位事务ID

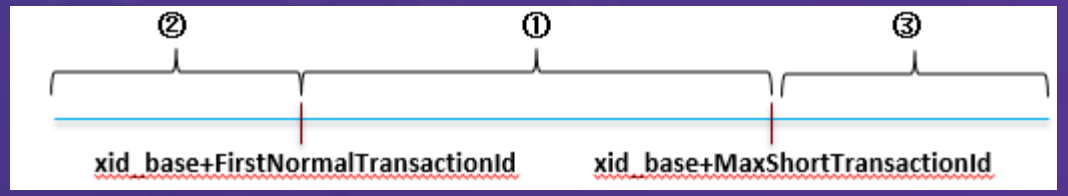


1、32位事务ID机制中可以将事务ID理解为一个循环可重用的序列串。对其中的任一普通XID来说，都有20亿个相对它来说过去的事务，和20亿个未来的事务。可以看出同一个数据库中存在的最旧和最新两个事务之间的年龄是最多是2^31。
为了避免这种问题，需要Vacuum freeze定期的对最旧事务进行回收，以推进该值；当事务ID的差值超过2^31会造成业务中断，强制全库回收事务ID。
2、TP场景事务ID增长迅速，导致32bit事务ID会频繁发生回卷
3、Vacuum机制来回收过老事务ID，效率低，IO开销大。
4、事务ID回收不及时，会导致停库对业务造成严重影响。

整改后：64位事务ID



页面头部增加xid base

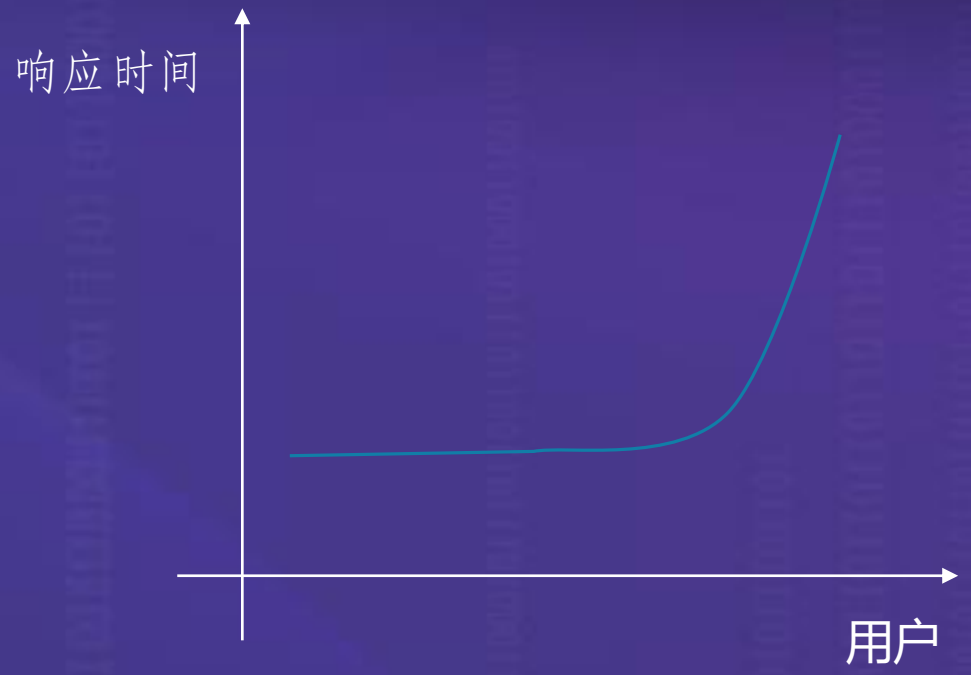


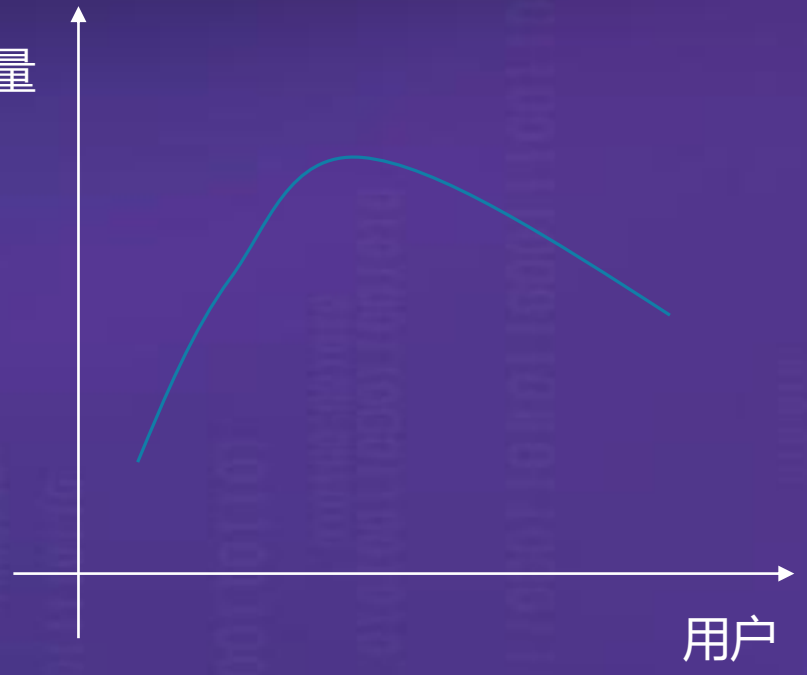
1、解决事务ID回卷问题，防止频繁vacuum对系统造成性能影响
2、64位事务ID特性平滑升级，对现有业务无影响，高效率，低风险。
3、老版本数据页面在页面头部增加xid_base(64bit), 避免对页面中的tuple进行修改。

openGauss

https://opengauss.org

openGauss 大并发业务场景问题





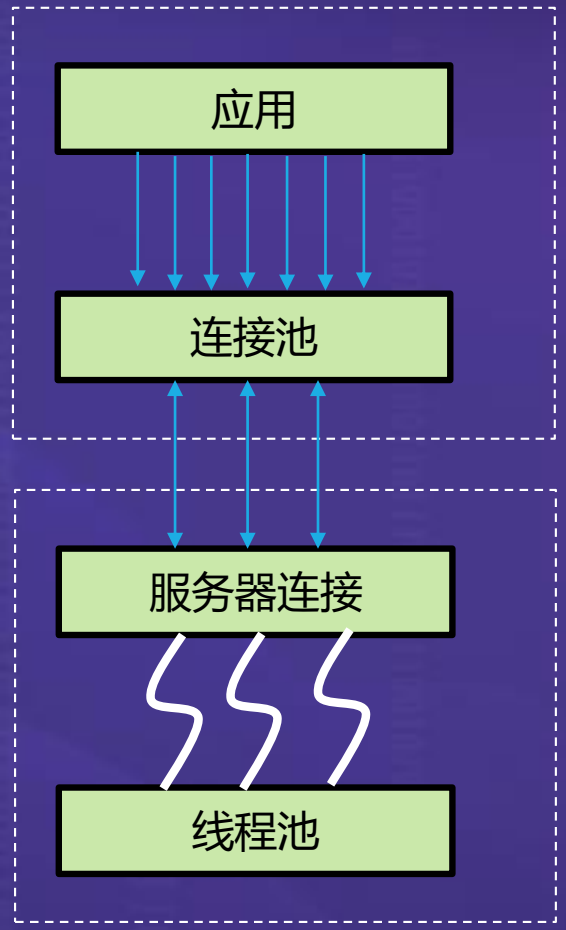
1、连接数越多，后台需要的线程数越多，资源开销越大。

2、在CPU核数固定的情况下，OS调度开销大，不同线程访问数据库资源冲突大。

openGauss

https://opengauss.org

openGauss 大并发问题解决方案

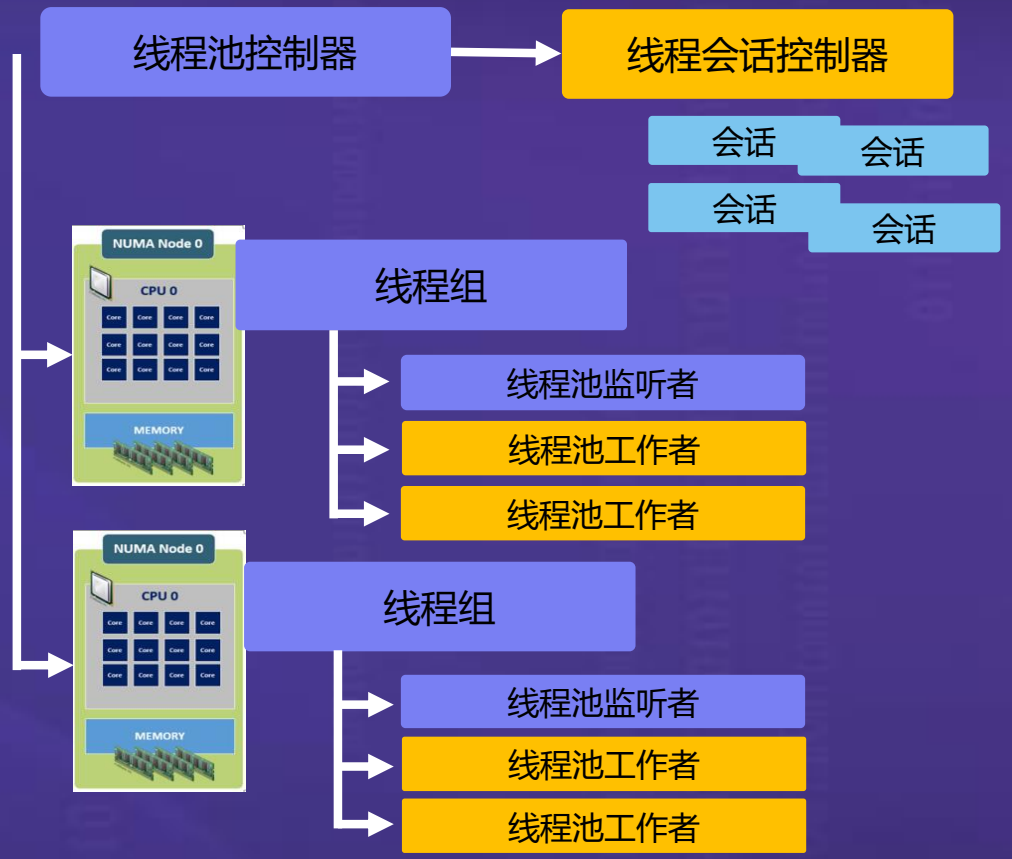


1、连接池一般在客户端设置，连接池避免了连接的频繁创建和销毁。连接复用。

2、线程池在数据库服务器上配置，控制数据库服务器活动线程数目。线程复用。对系统的业务起到流控作用，防止出现雪崩。

3、在高并发场景下，可以将连接池和线程池结合起来使用。

openGauss 线程池实现原理



1、主线程监听连接请求，分配会话，把会话分配给一个线程组。

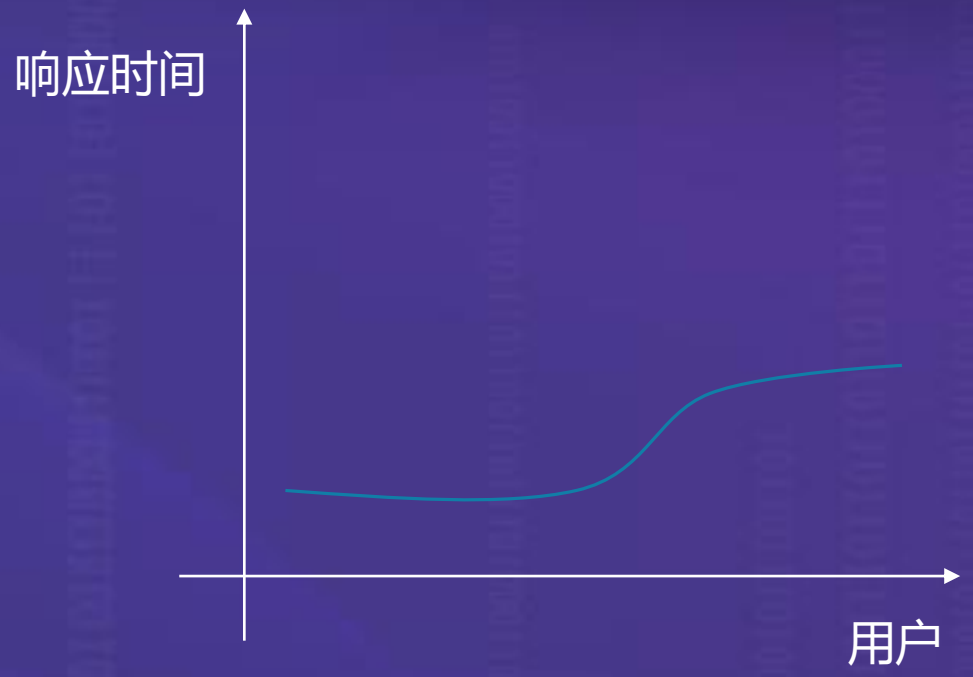
2、每个线程池组有一个监听线程负责监听epoll列表中所有的客户连接，避免“惊群”效应。

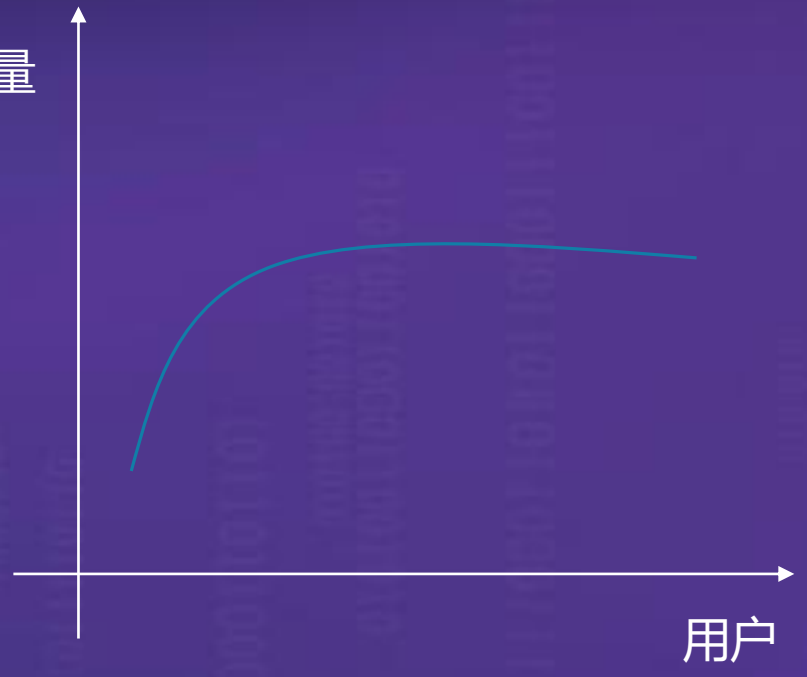
3、每个线程组可以和一个NUMA 节点绑定。

openGauss

https://opengauss.org

openGauss 线程池下大并发性能结果





1、高并发下保持性能的稳定性。

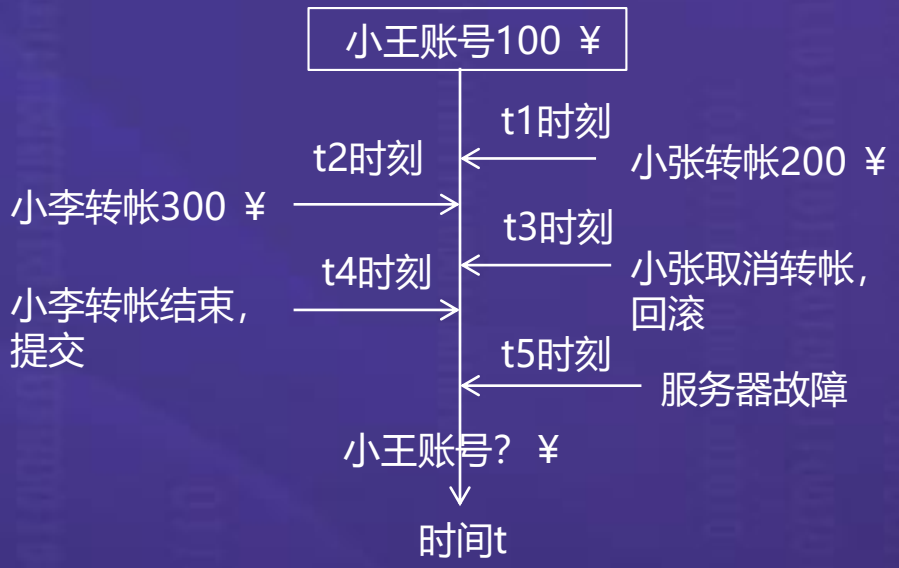
openGauss

https://opengauss.org

openGauss 数据库事务

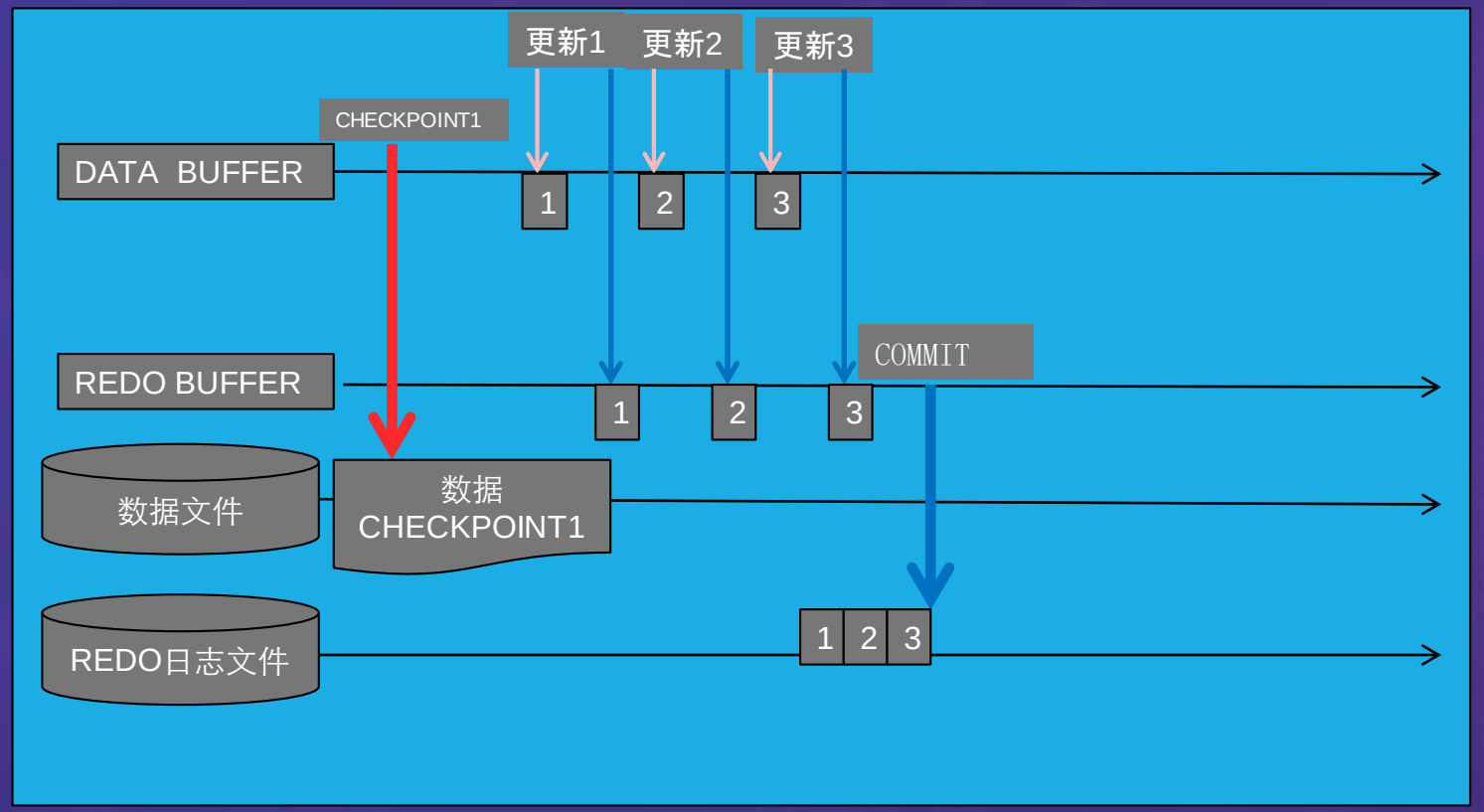
- **事务的ACID特性**
  - 原子性（Atomicity）：事务所包含的操作要么全部完成，要么什么也没做。
  - 一致性（Consistency）：在一致性数据库上执行事务后，数据库仍需保持为一致性的状态。
  - 隔离性（Isolation）：没有结束的事务在提交之前不允许将其结果暴露给其它事务。
  - 持久性（Durability）：当一个事务的结果提交后，系统保证该结果不会因以后的故障而丢失。

- 范例：银行转账（小张，小李都给小王转账）



- ACID保障服务器重启后，小王的账号为400 ￥。
- 原子性保障：小张取消转账后，小王账号仍然为100 ￥。
- 一致性保障：小李的账号少300 ￥，小王账号多300 ￥。保证他们两个账号的总和不变。
- 隔离性保障：小张转账的200 ￥对小李不可见，不然结果为700 ￥。
- 持久性保障：服务器故障重启后，小李的转帐仍然是成功的。
- 小王的账号始终为400 ￥。

openGauss 数据库事务机制



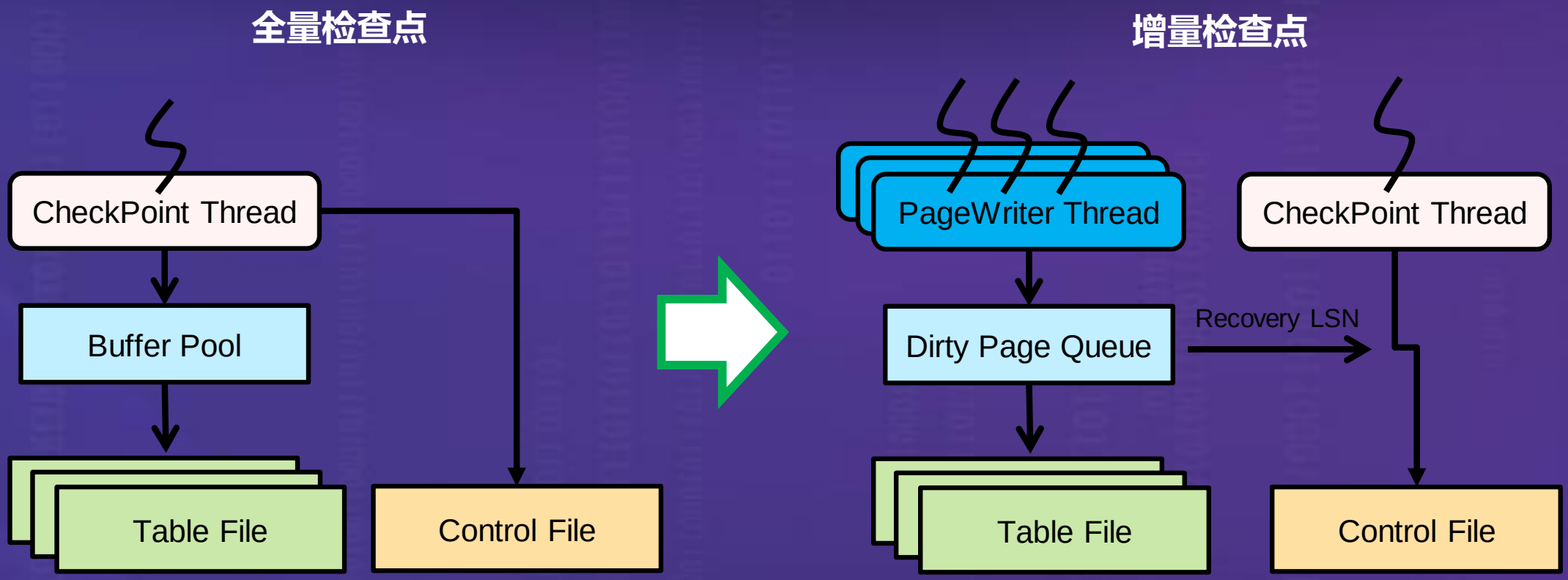
1、事务的持久性（Durability）主要通过预写式日志WAL算法实现。在事务提交时，采用预写式日志方式，把REDO日志写到磁盘。

2、检查点：将脏缓冲队列上的全部数据写出到数据文件。

openGauss

https://opengauss.org

openGauss 增量检查点技术原理

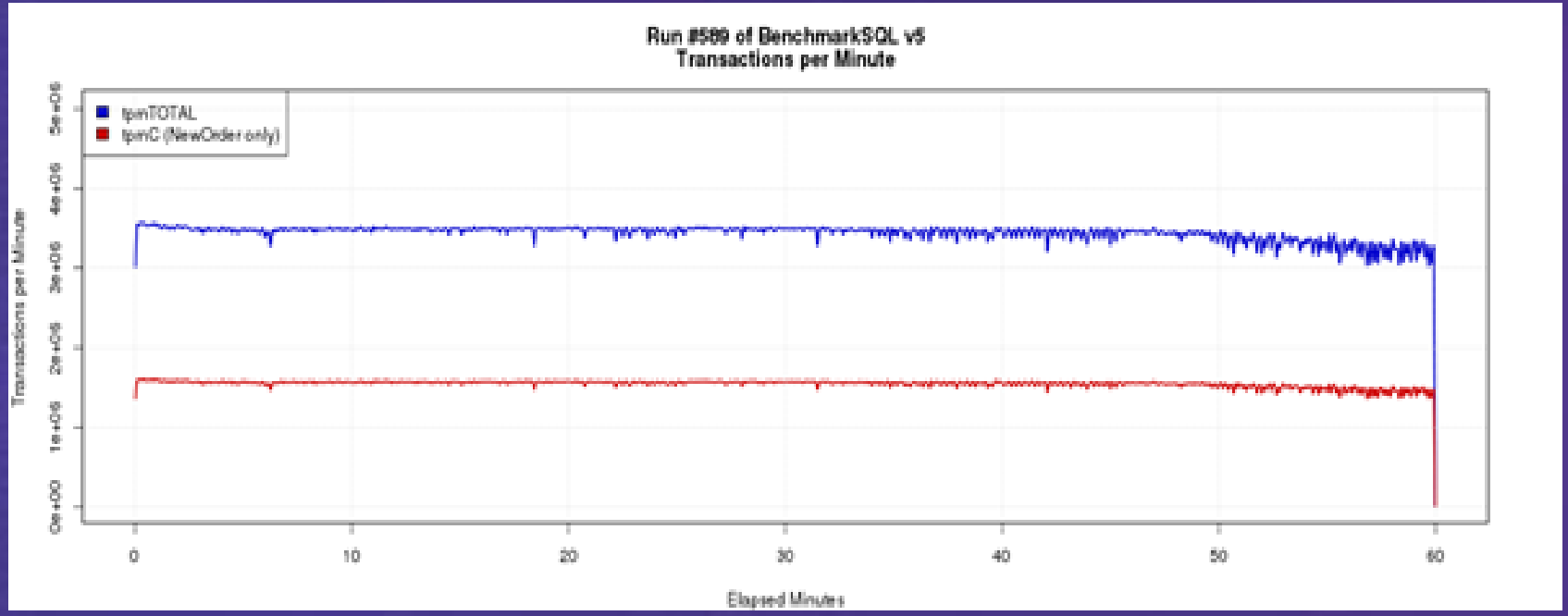


1. 每一个脏数据库块都会被记录到脏页队列，按照第一次对此数据块修改时日志的LSN顺序来排列，如果一个数据块进行多次修改，该数据块在脏页队列中的顺序并不会发生变化。
2. 在执行增量检查点时，把一定LSN之前的脏页刷盘，不需要把所有dirty buffer 全部写到磁盘。

openGauss

https://opengauss.org

openGauss 增量检查点IO平稳

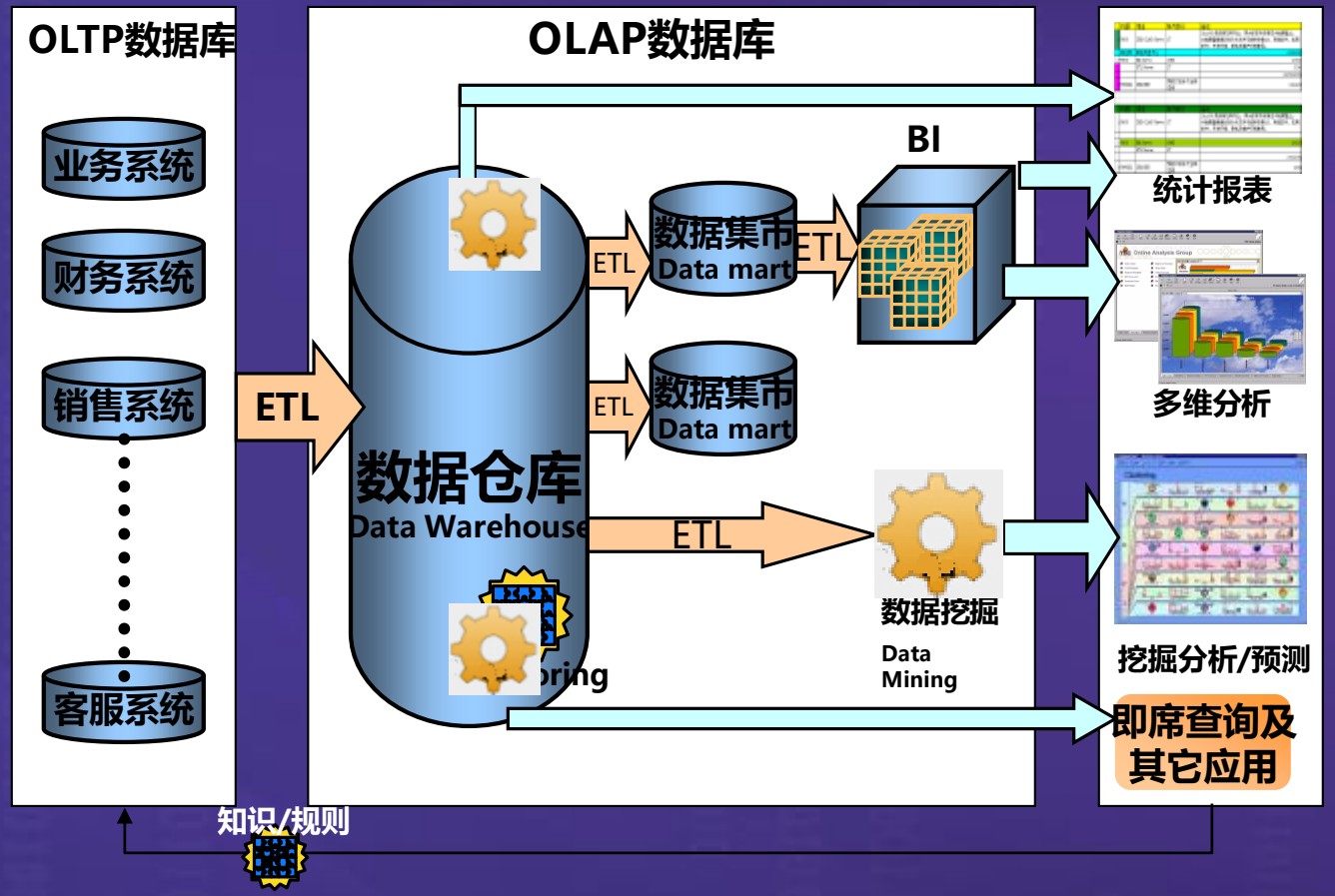


1. IO均分到各个阶段，性能更加平稳。
2. Checkpoint点效率和频率更高，减少宕机恢复时间，降低RTO。

openGauss

https://opengauss.org

openGauss 企业应用场景OLTP&OLAP



1、联机事务处理(OLTP)： 存储/检索**业务应用中活动的数据**以支撑日常的业务活动；

2、联机分析处理(OLAP)：存储**历史数据**以支撑复杂的分析操作，侧重决策支持；

openGauss

https://opengauss.org

openGauss 数据库 OLTP&OLAP业务特征

对比维度\ 数据库类型	OLTP	OLAP
用户	操作人员，低层管理人员	决策人员，高级管理人员
功能	日常操作处理	分析决策
事务支持	强事务语义	无/弱事务语义
查询	点查询为主	汇总，关联等复杂查询
插入/更新	大量插入/更新操作， 每次操作1条或很少记录	以批量入库（插入）为主， 很少或没有更新操作
数据内容	管理当前数据	管理历史数据
数据大小	通常几GB到几TB	通常几百GB到几PB

openGauss 行存&列存

Emp_no	Dept_id	Hire_date
1	1	2001-01-01
2	1	2002-02-01
3	1	2002-05-01

行存（Row Store ）	列存（Column Store）
容易修改记录	修改记录代价高，写一条记录时需要访问多次IO
读取数据代价高，可能读入不需要的列	可以只读取相关的数据
不同记录重复率低，压缩率低	同一列重复值高，压缩率高

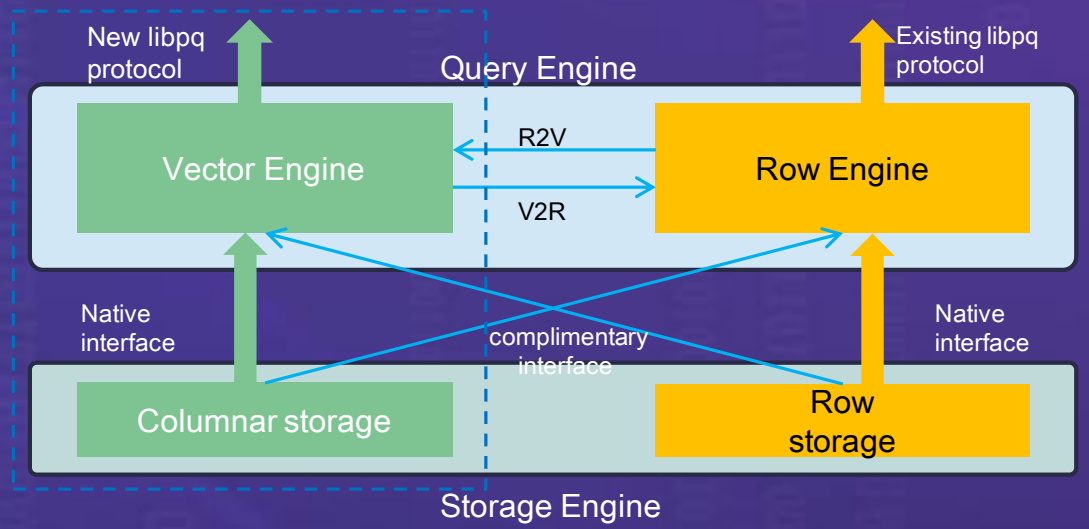
openGauss

https://opengauss.org

openGauss 行列混合引擎

行列混合引擎

表级别指定行存/列存，根据不同的场景选择不同的存储类型。



场景	行存	列存	优选
点查询	B+树索引，直接定位到行（页）	粗粒度索引，定位到CU	行存
数据更新	• 支持行级别锁 • 支持CU级别并发更新	支持CU级别锁，支持CU级别并发更新	行存
统计分析	Pipeline执行	天然和向量化引擎对接，降低CPU Cache Miss和指令Miss，效率成倍提升	列存
批量加载	并行批量加载	压缩率高，IO量更小	列存

openGauss

https://opengauss.org

openGauss 行列混合IoT场景

openGauss

https://opengauss.org

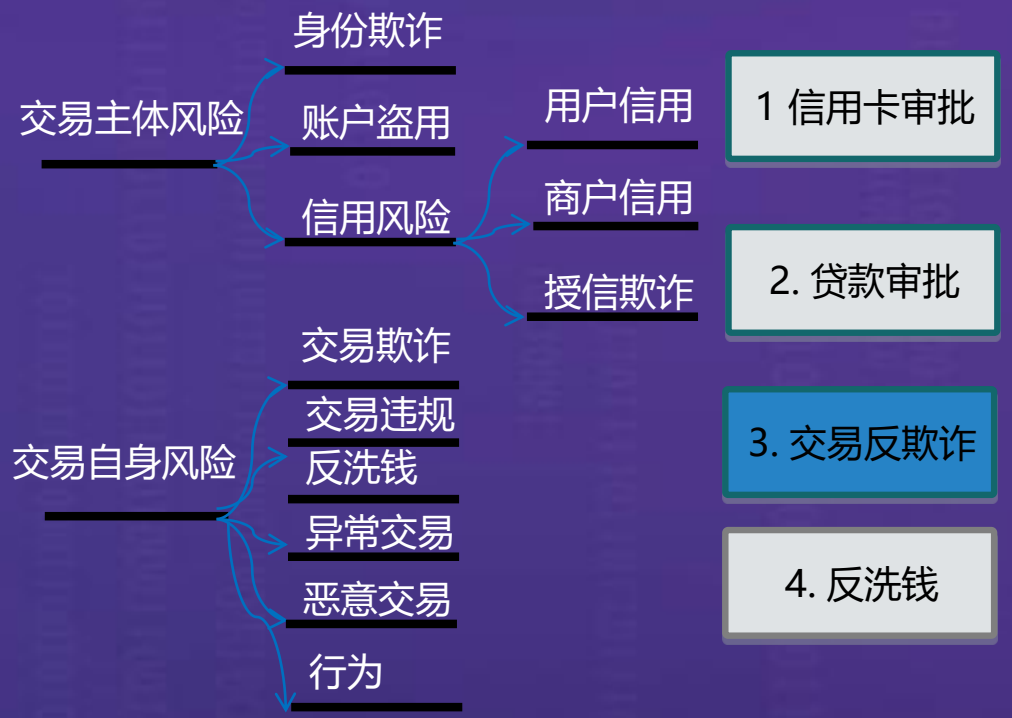
openGauss 金融风控场景

金融领域风控平台的要求是高并发，低时延，可业务编程。

移动支付占比70%，**网络伪冒**成上升趋势

2017年**个人数据大量泄漏1250**多起，加剧伪卡欺诈和典型诈骗

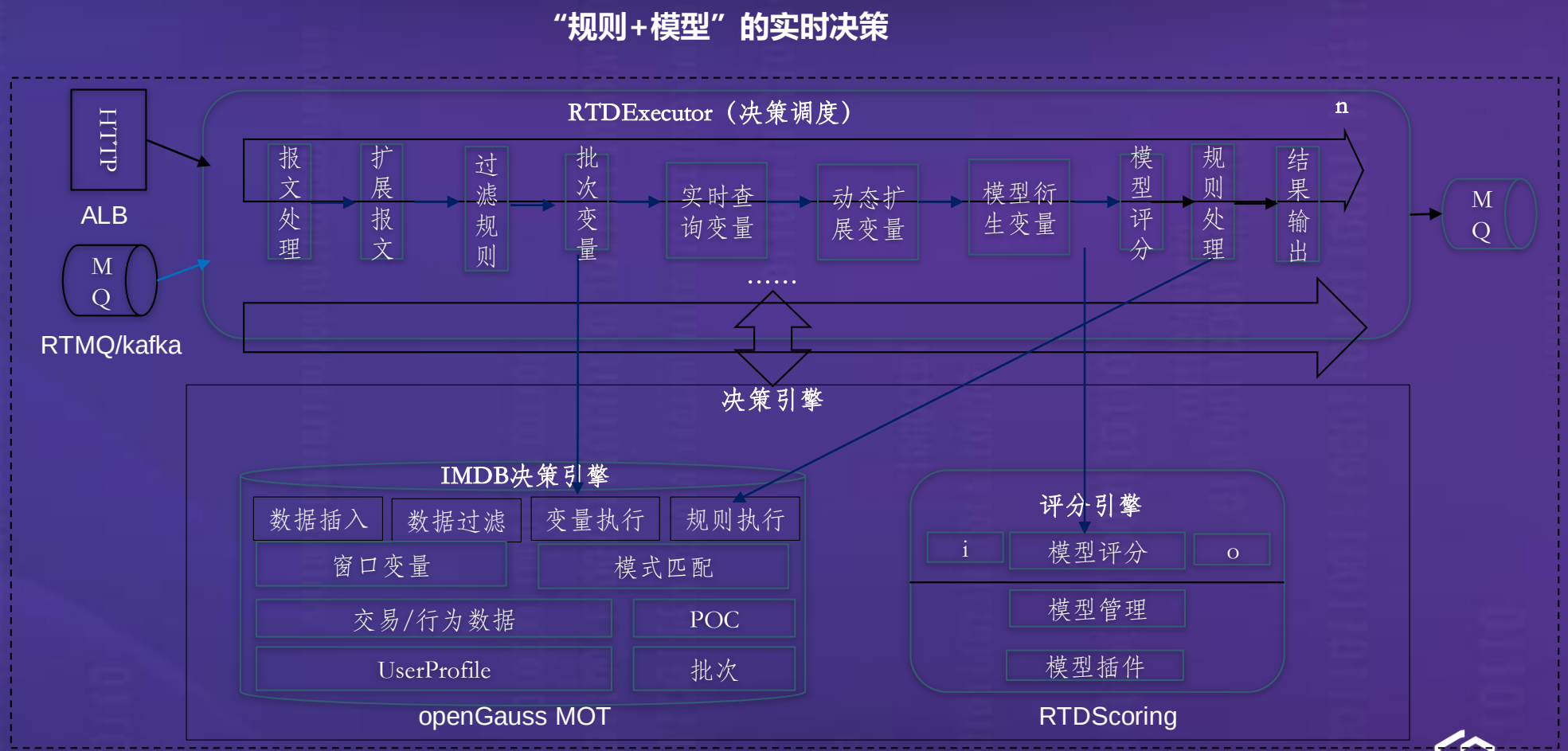
小额、高频交易量同比上升50%以上，传统风控系统**无法快速水平扩展应对压力**



openGauss

https://opengauss.org

openGauss MOT 风控实时处理



openGauss

https://opengauss.org

openGauss 磁盘引擎问题

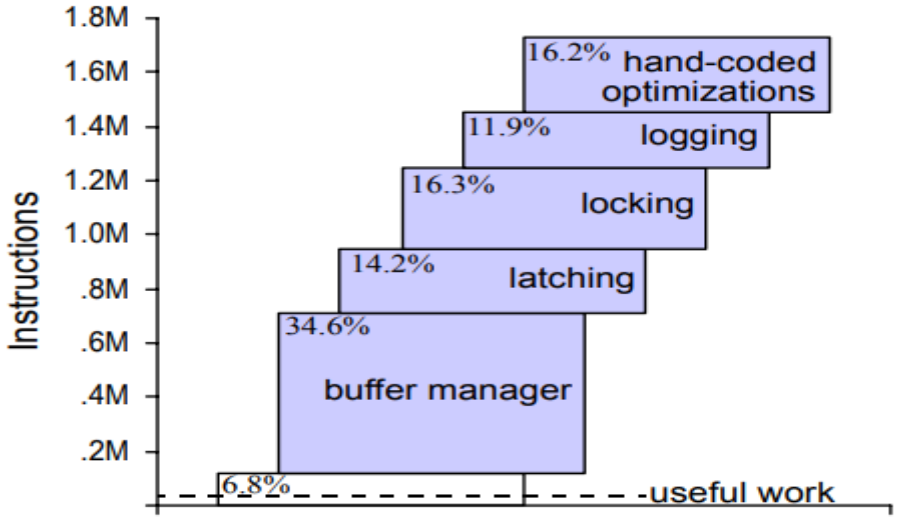


Figure 1. Breakdown of instruction count for various DBMS components for the New Order transaction from TPC-C. The top of the bar-graph is the original Shore performance with a main memory resident database and no thread contention. The bottom dashed line is the useful work, measured by executing the transaction on a no-overhead kernel.

OLTP Through the Looking Glass, and What We Found There

Stavros Harizopoulos
HP Labs
Palo Alto, CA
stavros@hp.com

Daniel J. Abadi
Yale University
New Haven, CT
dna@cs.yale.edu

Samuel Madden
Michael Stonebraker
Massachusetts Institute of Technology
Cambridge, MA
{madden, stonebraker}@csail.mit.edu

1、图灵奖获得者Micheal Stonebraker：磁盘架构70%左右指令在等待锁/闩，仅不到30%CPU做有用功。

2、内存数据库在保证ACID的前提下，以Lock-free无锁方式实现事务处理，能有效规避磁盘架构缺点：

CPU利用率高，一般可达80%以上，提升硬件投资回报；

时延低，没有锁/闩等待，事务处理延迟低，满足实时业务；

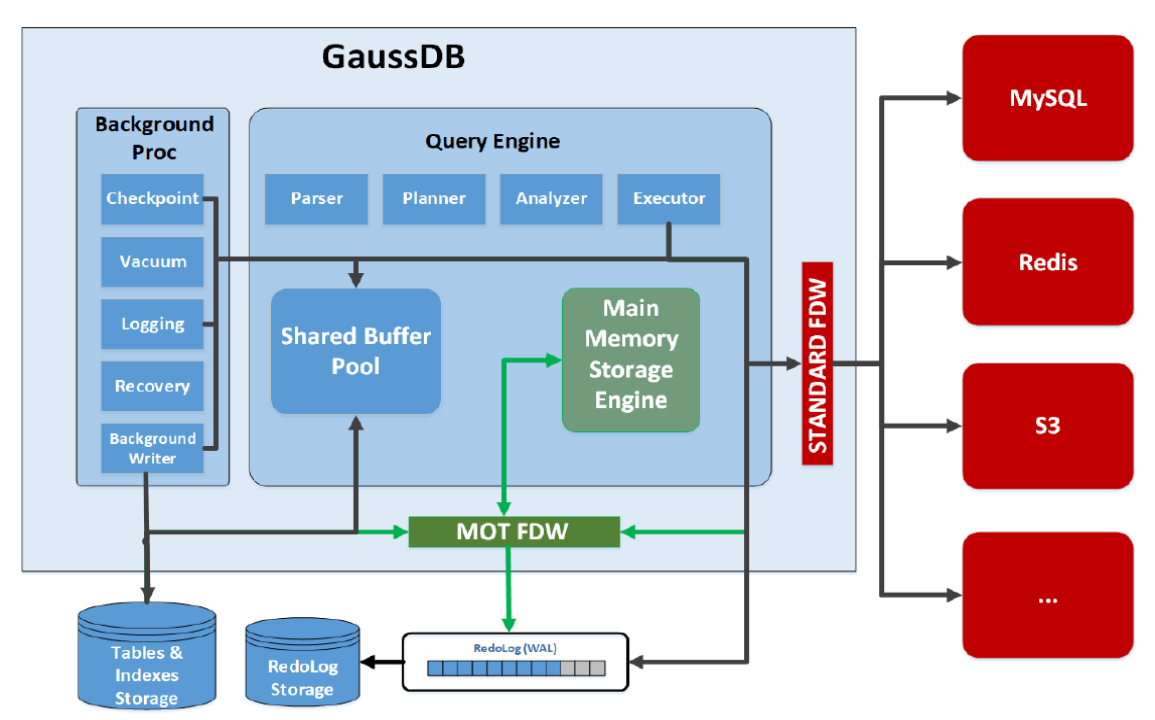
openGauss

https://opengauss.org

openGauss MOT原理

openGauss免锁，高吞吐，低时延

openGauss的融合内存引擎MoT架构



优势1：性能高、CPU利用率高、延迟低

① 高度优化的全内存免锁存储引擎
② 基于全内存优化实现的免锁索引
③ 高度优化的并发访问控制
④ 针对NUMA优化的内存管理，预缓存对象池
⑤ 针对NUMA高度优化的组提交

优势2：生态好、兼容好，功能完整

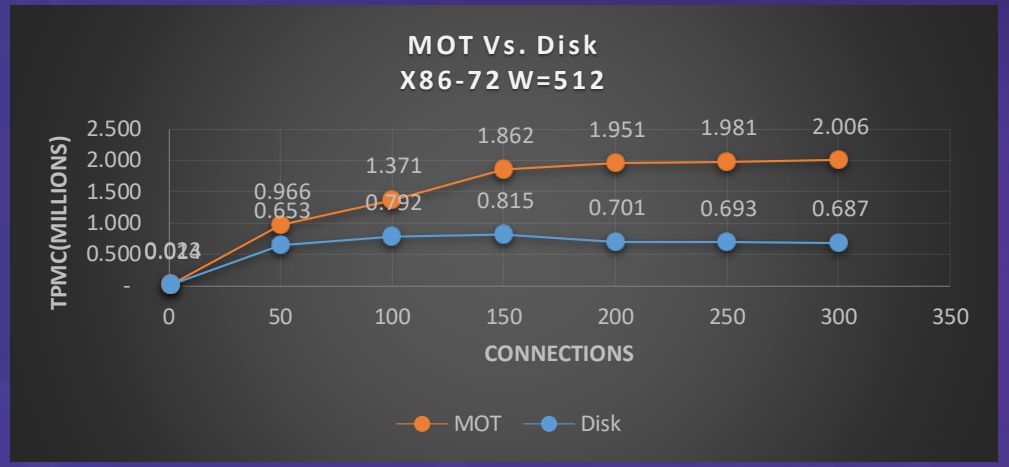
① 有效利用openGauss现有的查询引擎，兼容PG生态
② 兼容PG原生FDW和索引，SQL标准兼容度高，功能完整
③ 除PG原生FDW之外，还支持存储过程、用户定义函数等功能

openGauss

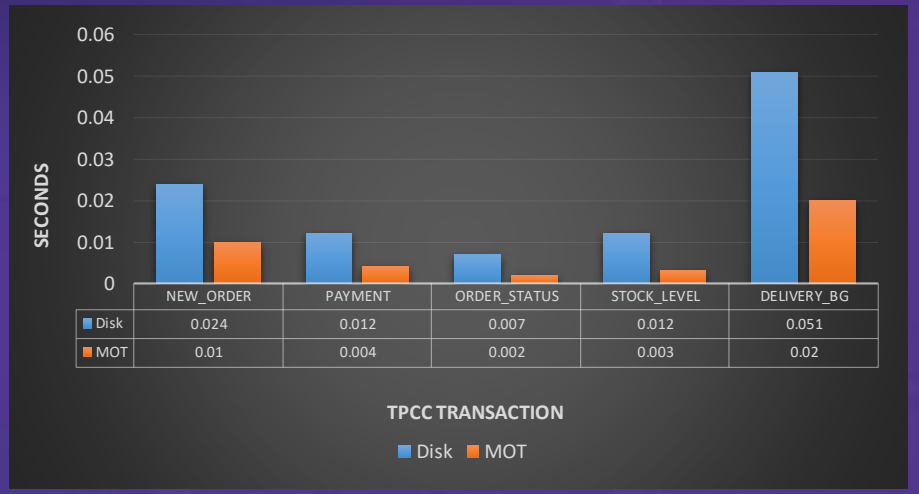
https://opengauss.org

openGauss MOT性能

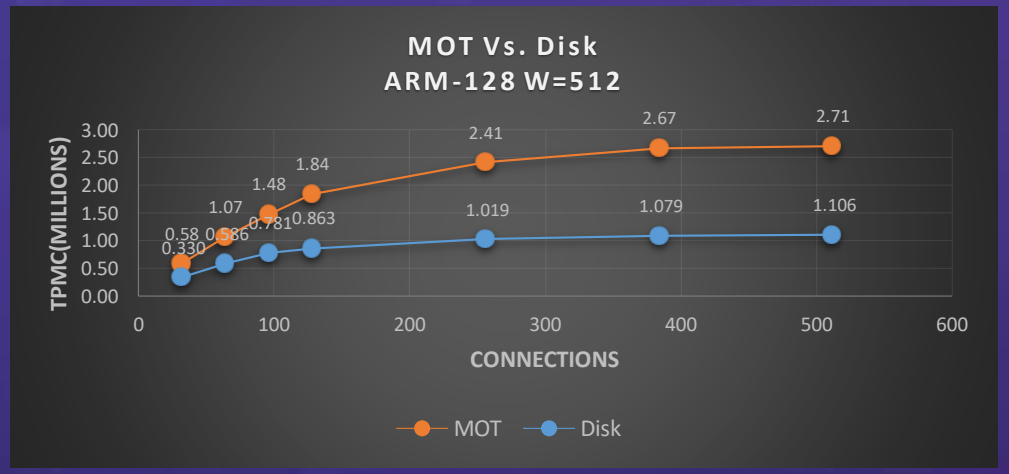
TPCC Throughput on X86-72 vCores



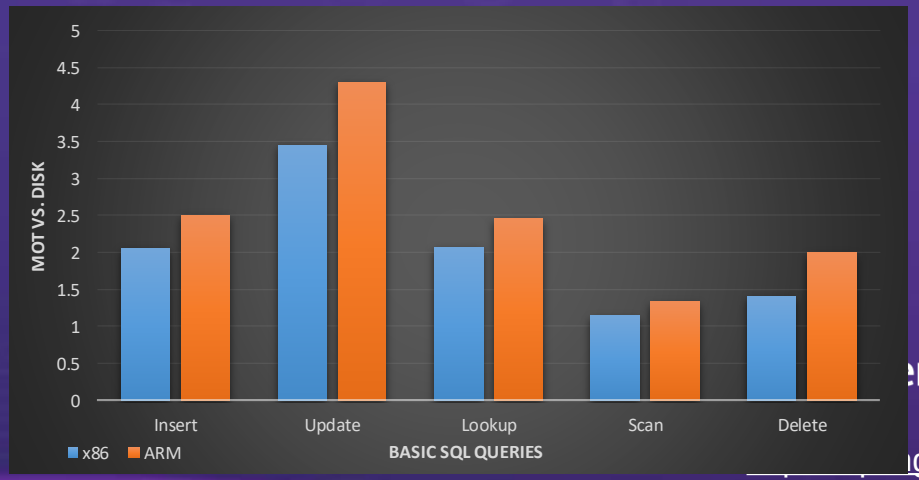
TPCC Latency



TPCC Throughput on ARM-128 Cores



Basic SQL Speed Up



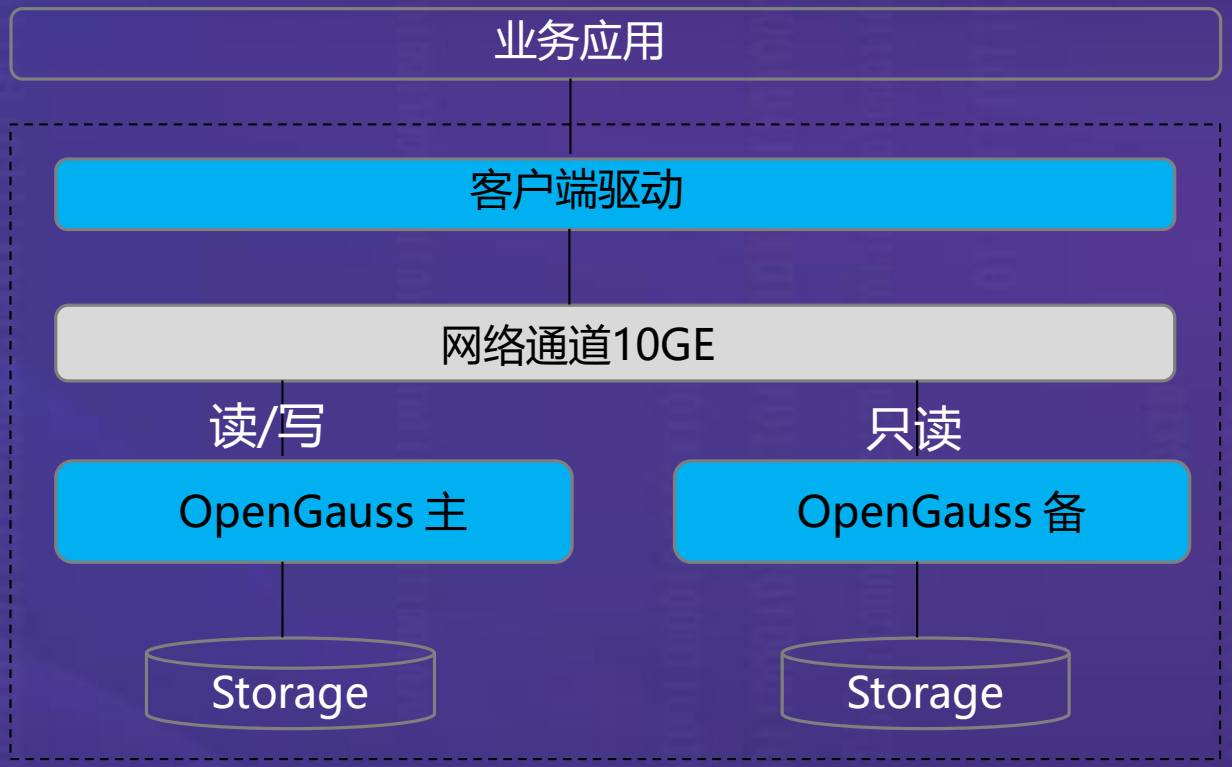
enGauss

gauss.org

openGauss 目录

- openGauss 开源社区介绍
- openGauss 架构
- **openGauss 核心技术及实践**
  - 高性能
    - NUMA多核优化
    - 行列混合引擎
    - 线程池原理
    - MOT内存表原理
    - 增量检查点原理
  - **可用性**
    - 极致RTO优化
  - 安全性
    - 全密态等值查询原理
  - 易运维
    - AI4DB和DB4AI
- openGauss 未来技术发展方向

openGauss 数据库双机企业部署场景



1、**最多可能丢失的数据的时长(RPO)：恢复业务系统后与中断时相比的数据损失量，反映恢复数据完整性的指标。**

2、**系统恢复正常所需要的最大时长(RTO)：业务从中断到恢复正常所需的时间，反映业务恢复及时性的指标。**

openGauss

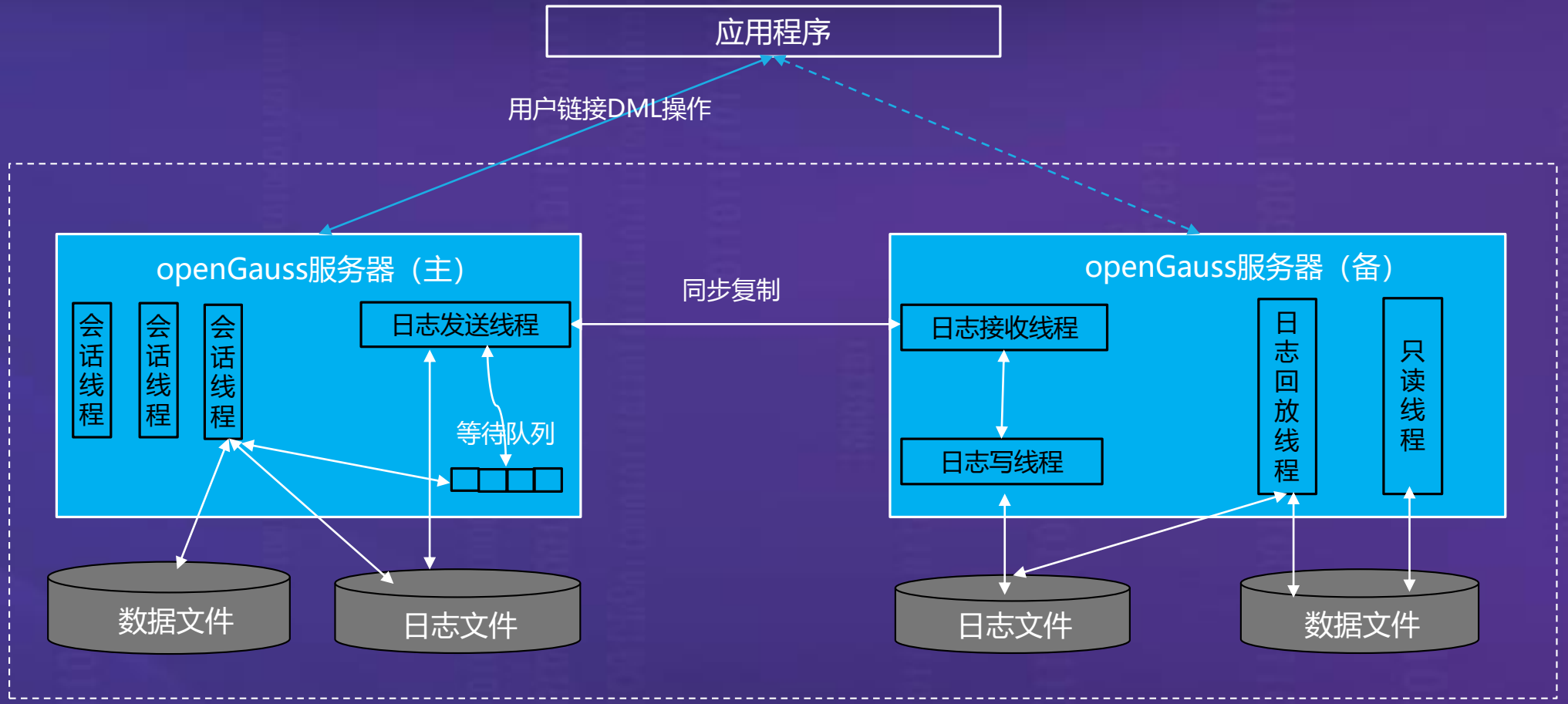
https://opengauss.org

openGauss 数据库双机原理

openGauss

https://opengauss.org

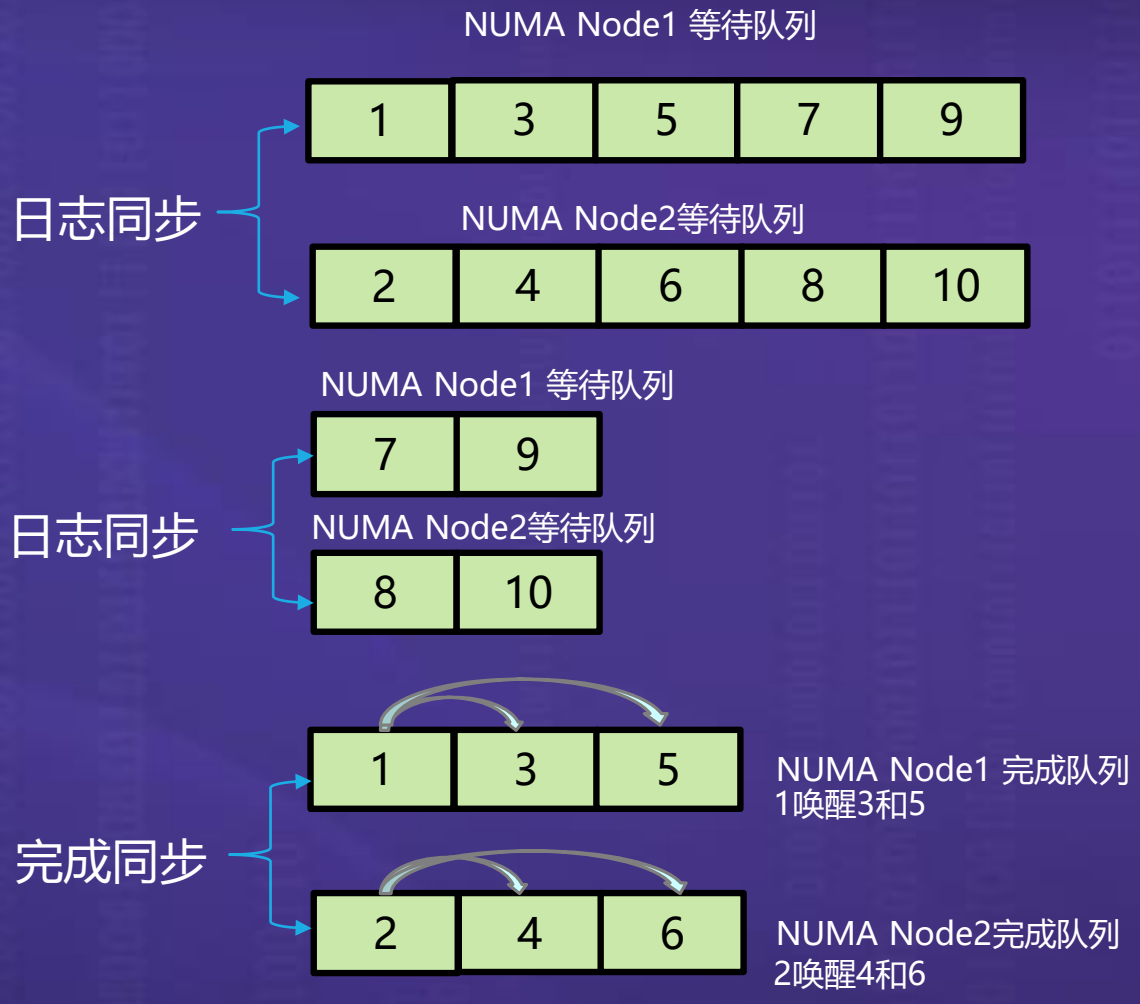
openGauss 双机处理流程



openGauss

https://opengauss.org

openGauss 日志传输



1、日志同步线程根据接收位置信息（图示为位置6）遍历等待队列，从等待队列中分离出完成队列，并且唤醒第一个会话线程，然后立即处理下一批日志同步任务，缩短唤醒会话线程的时间。

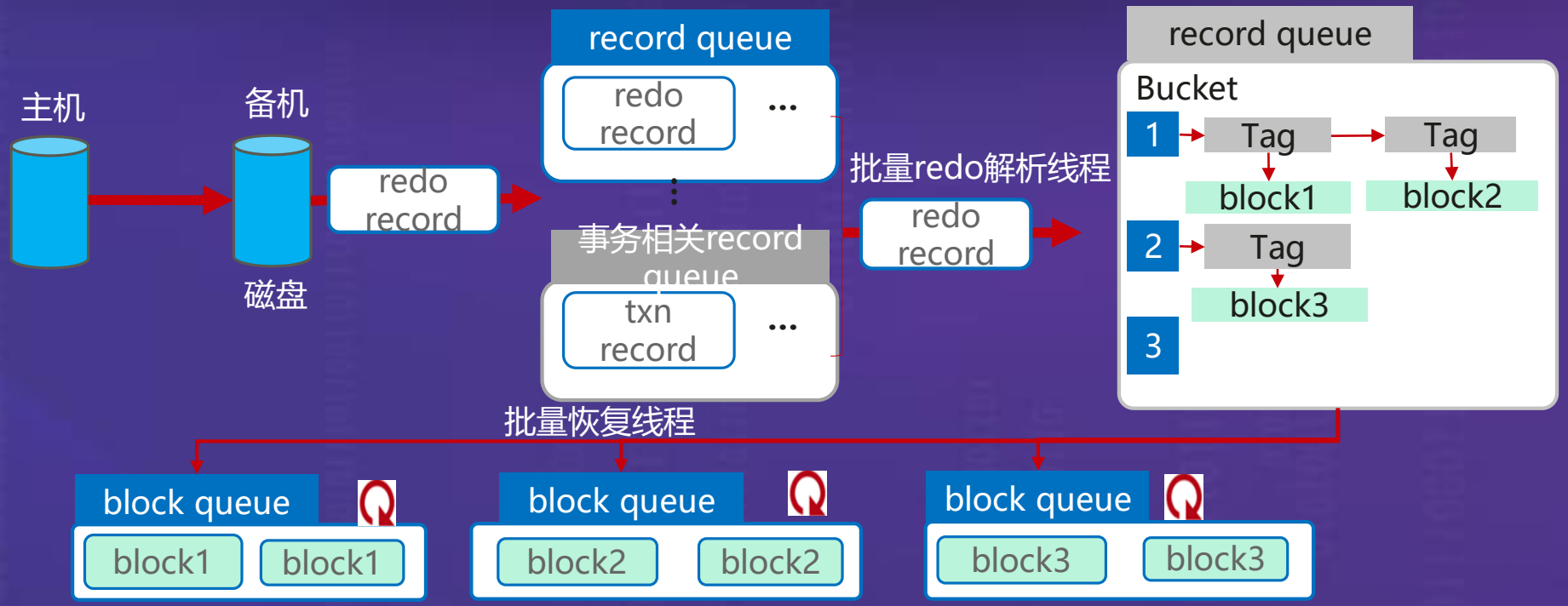
2、等待队列、完成队列根据会话线程组划分为多组，每组按照NUMA Node进行分布。

3、每个完成队列的首会话线程分别唤醒各自的完成队列。

openGauss

https://opengauss.org

openGauss 数据库极致RTO



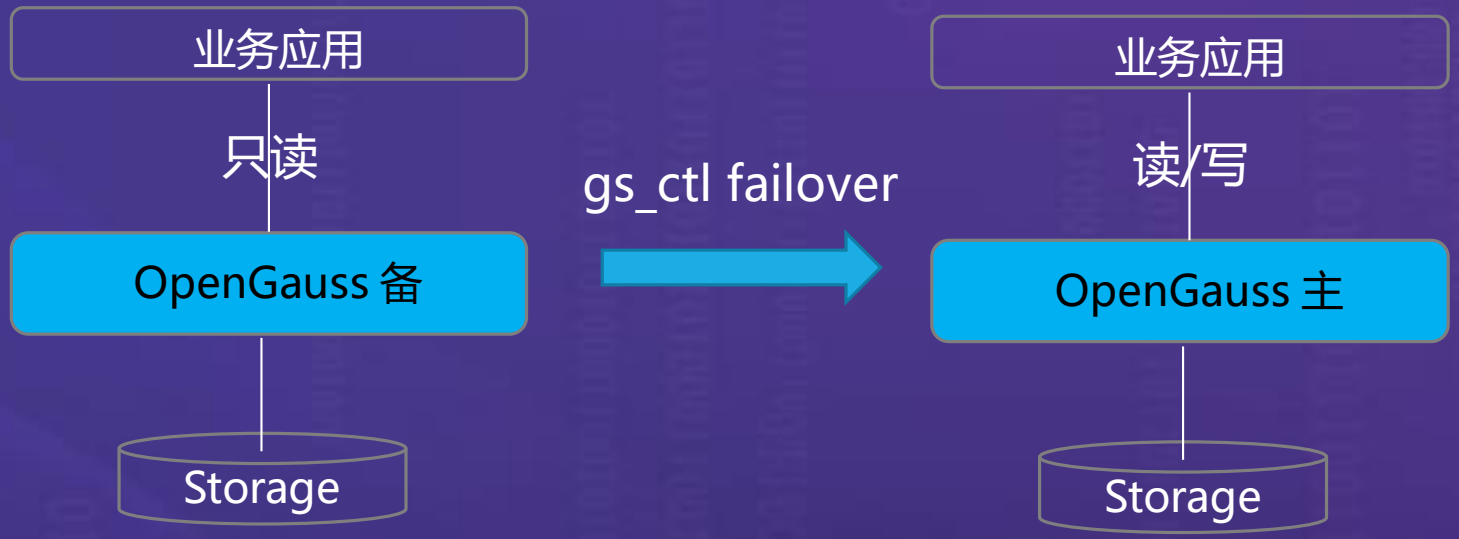
1. 备机日志实时落盘，同时将日志流**并行解码、读取和回放事务日志**；
2. 保证极大化日志回放吞吐，**功能切分服务化、流水线协同**，减少串行度。
3. 批处理化回放，**消除单条回放反复获取/释放锁**的并发控制和IO开销。
4. **页级物理并行**，去除按表切割并行机制，locality更好、并行度更高。

openGauss

https://opengauss.org

openGauss 数据库RTO时间

在60%负载、70+万tpmC下可达**RTO<10s（主备切换指令后10秒内，备机接管业务）**。



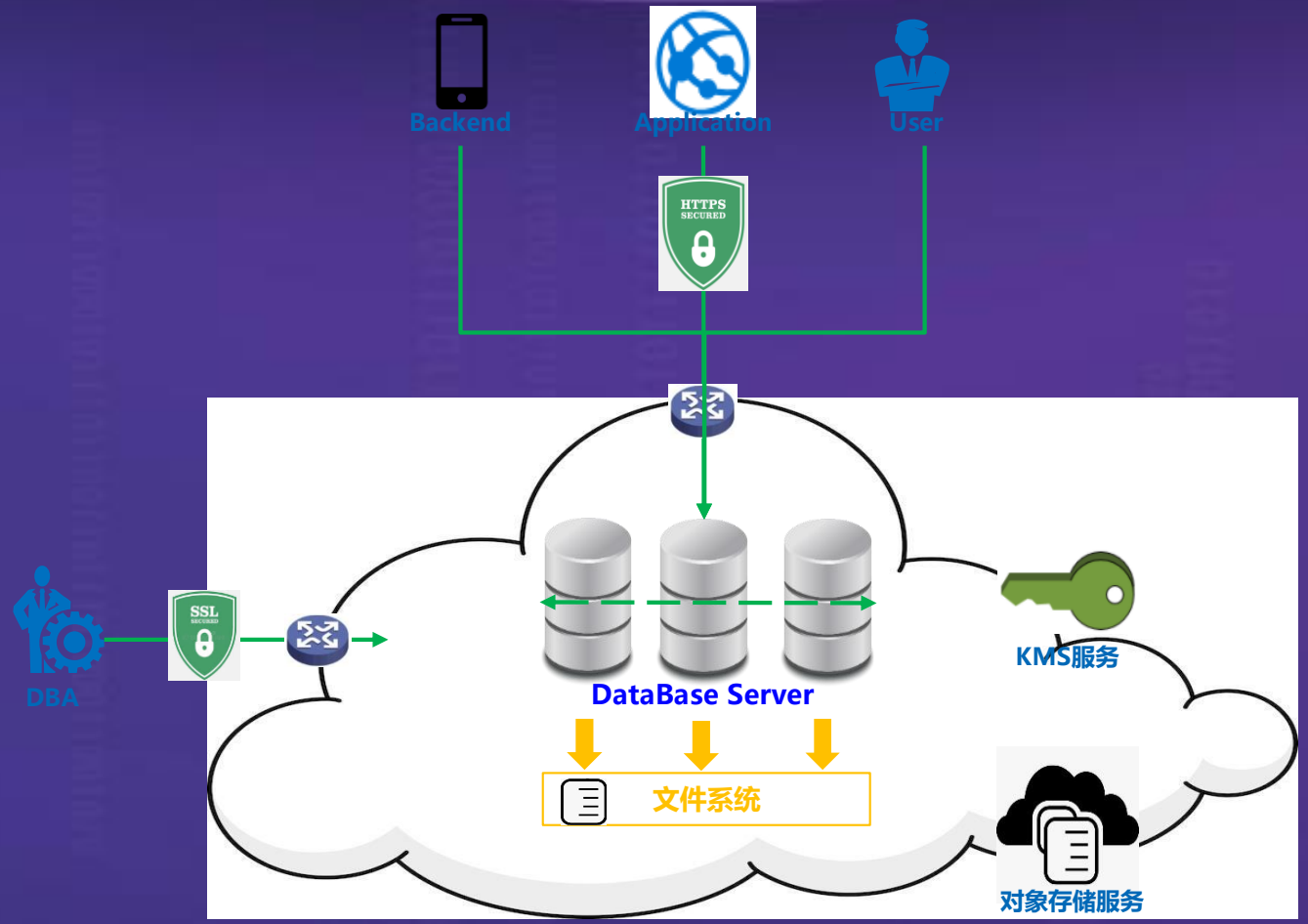
openGauss

https://opengauss.org

openGauss 目录

- openGauss 开源社区介绍
- openGauss 架构
- **openGauss 核心技术及实践**
  - 高性能
    - NUMA多核优化
    - 行列混合引擎
    - 线程池原理
    - MOT内存表原理
    - 增量检查点原理
  - 可用性
    - 极致RTO优化
  - **安全性**
    - 全密态等值查询原理
  - 易运维
    - AI4DB和DB4AI
- openGauss 未来技术发展方向

openGauss部署在云上时的安全问题



数据库在云上运行时数据存储和访问的安全问题--窃取数据

openGauss 全密态等值查询

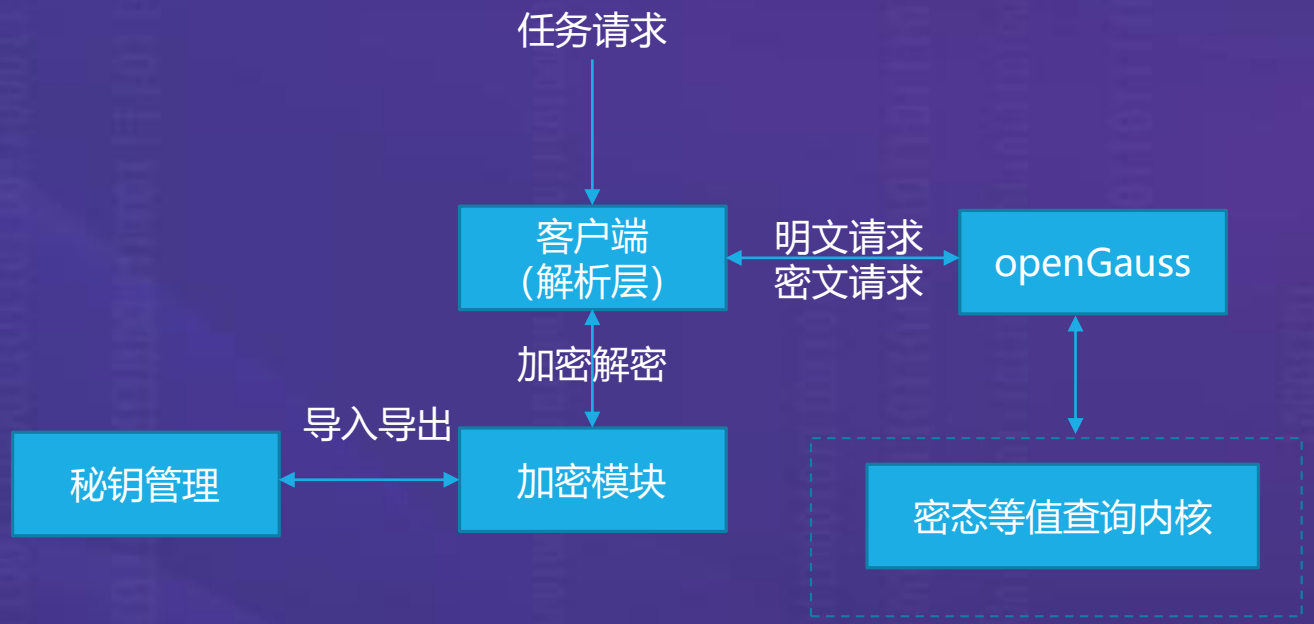
- 数据以密文形式存在，无法解密
- 不存储密钥明文信息

- 提供与密文环境隔离的明文环境
- 提供SQL查询和计算能力

openGauss

https://opengauss.org

openGauss 全密态等值查询流程

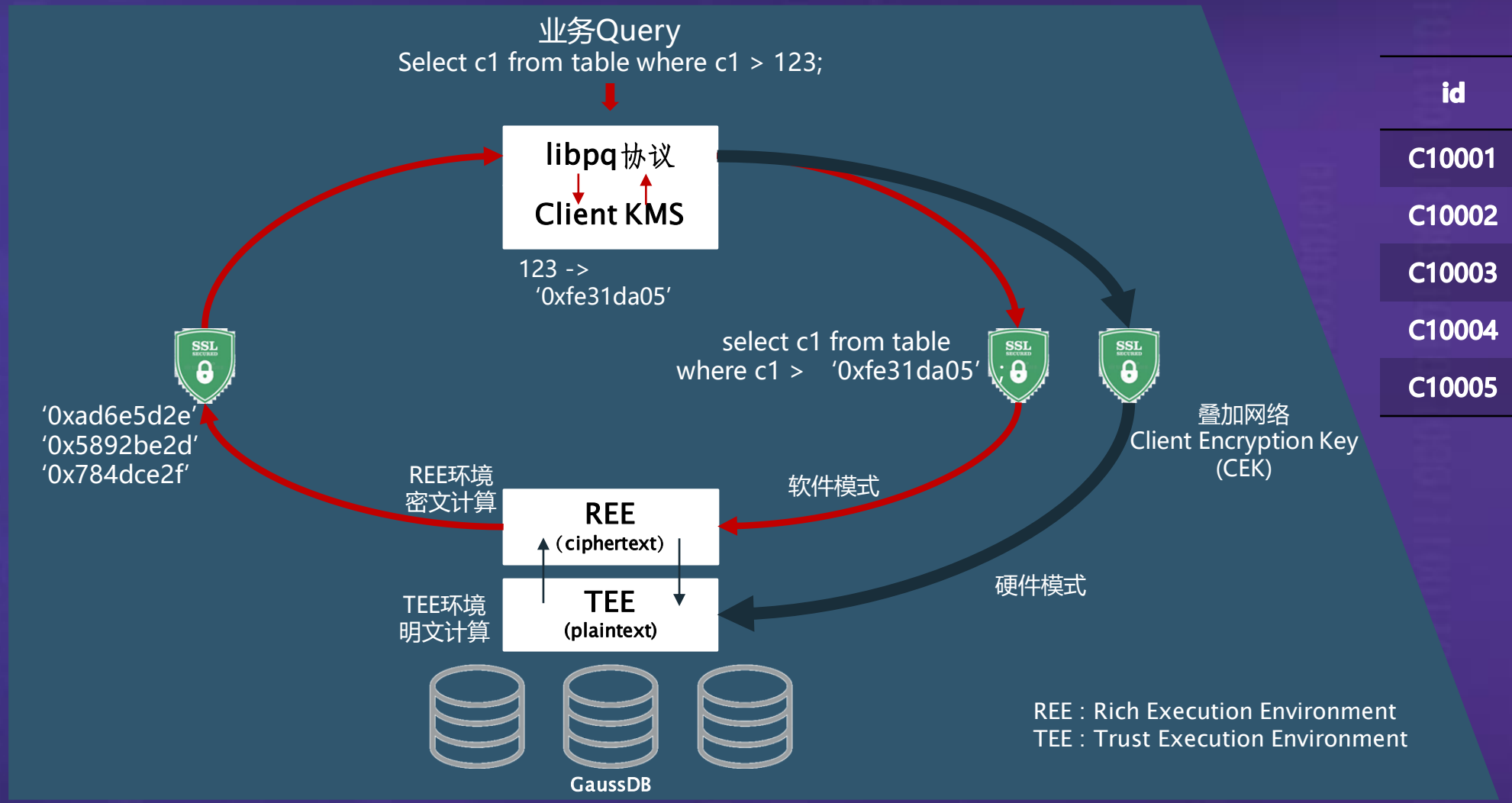


1、三层秘钥管理机制：根秘钥，主秘钥和列加密秘钥。

2、客户端完成数据的加密和解密，服务器完成密态数据计算。

3、不需要加密的字段仍然是明文处理。

openGauss 全密态加密等值数据流程



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

openGauss

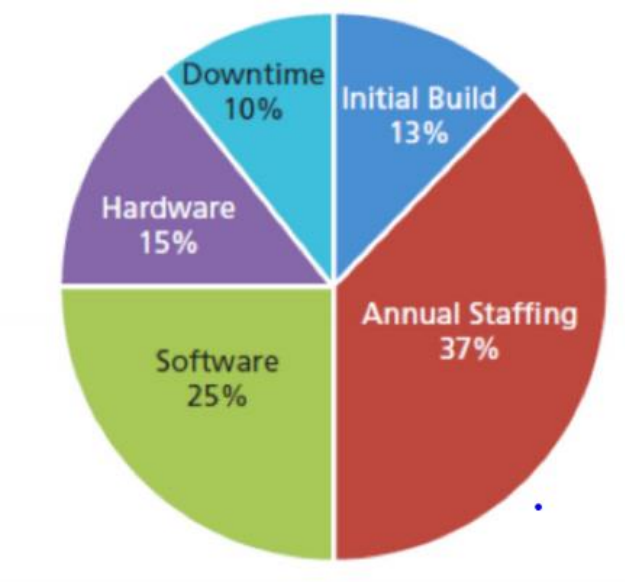
https://opengauss.org

openGauss 目录

- openGauss 开源社区介绍
- openGauss 架构
- **openGauss 核心技术及实践**
  - 高性能
    - NUMA多核优化
    - 行列混合引擎
    - 线程池原理
    - MOT内存表原理
    - 增量检查点原理
  - 可用性
    - 极致RTO优化
  - 安全性
    - 全密态等值查询原理
  - **易运维**
    - AI4DB和DB4AI
- openGauss 未来技术发展方向

企业数据库运维问题

典型数据库应用系统TCO分布：数据库运维占三分之一左右

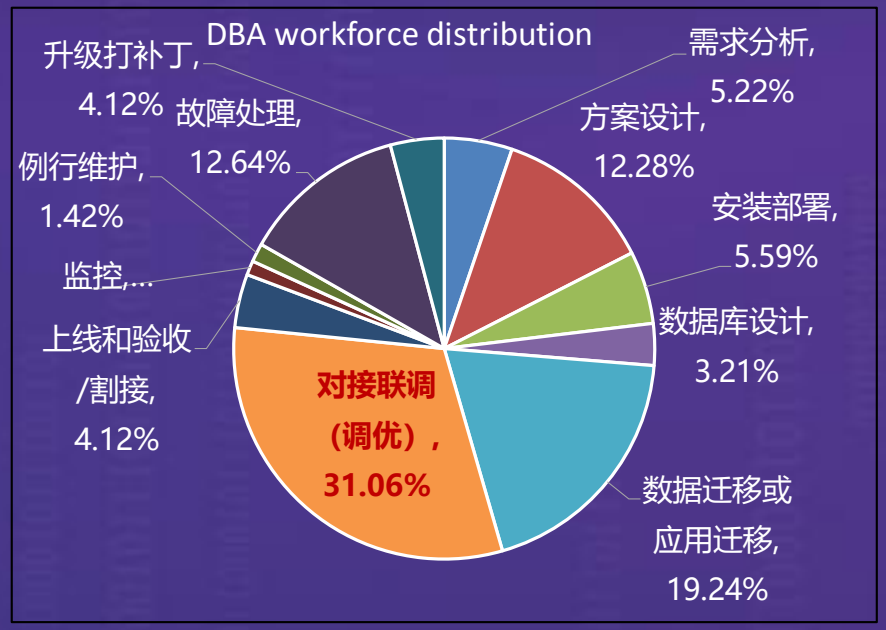


DB maintenance cost is biggest part of TCO

Source: IDC 2007.
"Three Year Server TCO. Based on more than 300 interviews conducted across numerous platforms"

日产运维工作占DBA50%以上工作内容

Tuning/upgrading/debugging...



Source: DBA survey from CSDN.

大部分DBA日常工作可以自动化

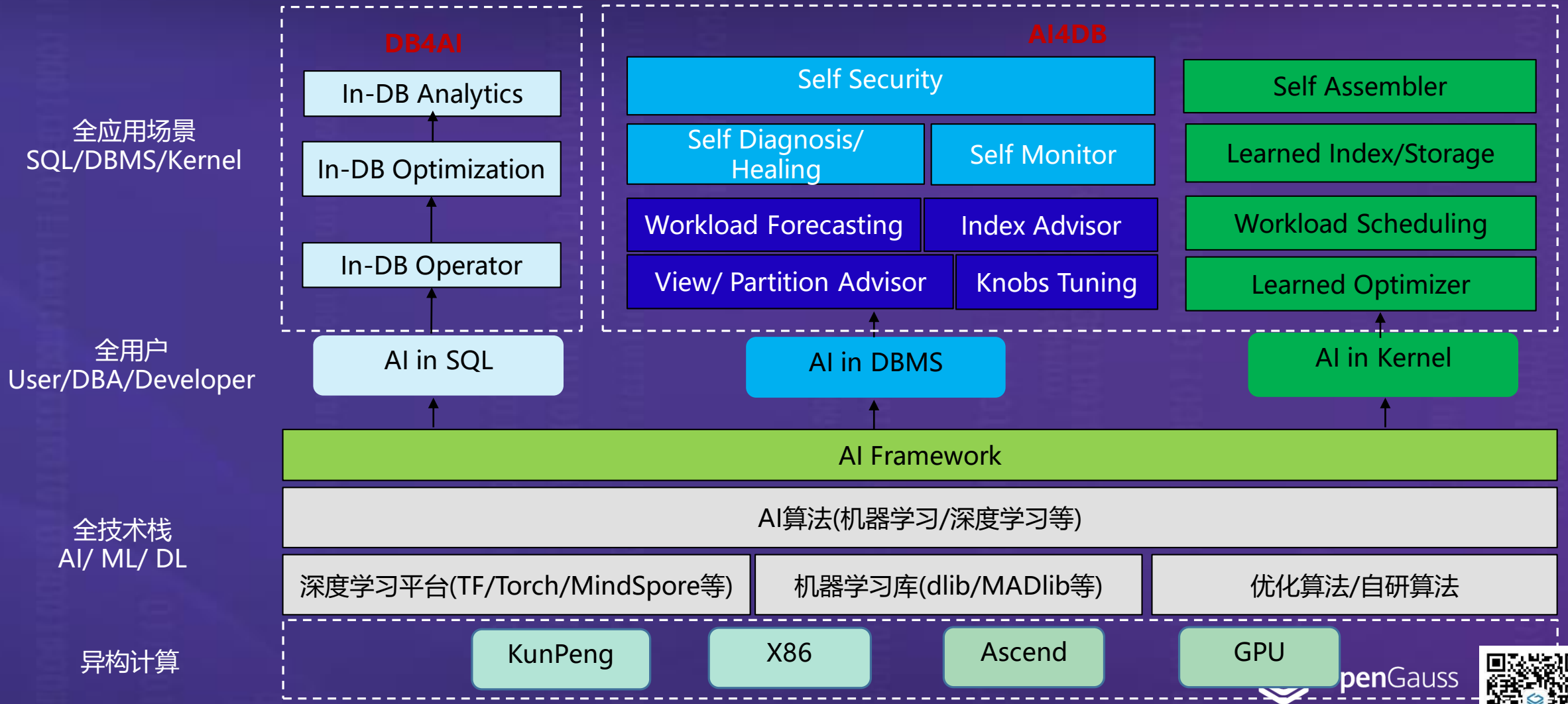
- schema 设计
- 应用查询开发
- SQL查询优化
- 故障恢复和诊断
- 数据库备份和恢复
- 安全管理
- 数据库扩容和数据迁移
- 数据库部署、补丁、升级
- 监控与故障排查
- OOS部署、补丁、升级
- 服务器、网络、存储日常运维
- 基础设施和资产管理

业务相关　　可自动化

openGauss

https://opengauss.org

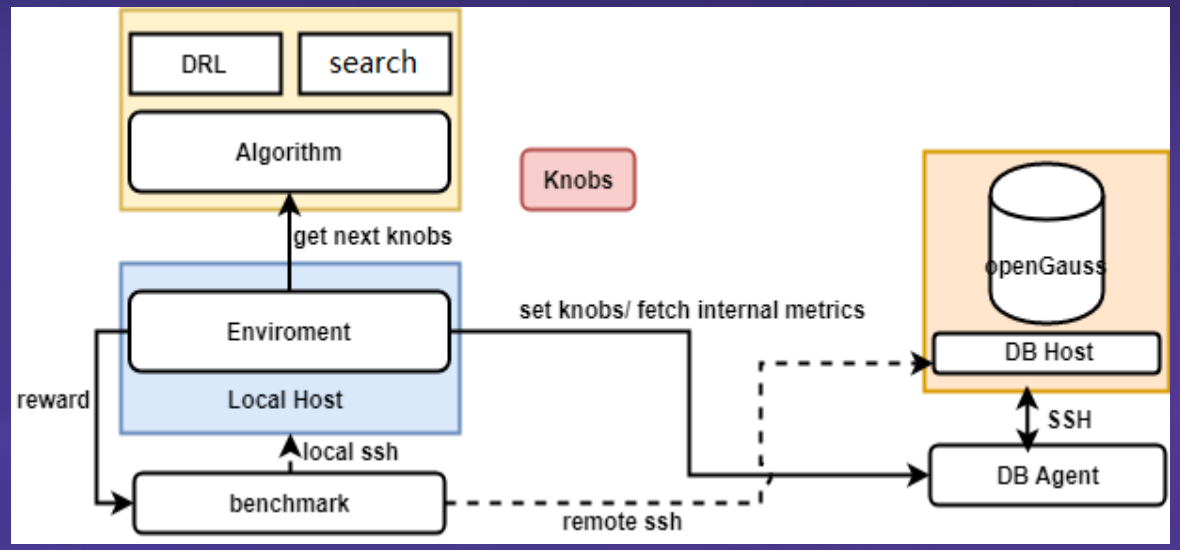
openGauss AI 全景



openGauss

https://opengauss.org

openGauss参数调优与诊断能力



离线参数调优流程概述:

1. 利用长期参数调优总结出的先验规则进行参数配置诊断与生成数据库workload报告；
2. 根据系统的workload、环境信息推荐初始参数配置，包括推荐参数值、建议最大值和最小值（用以保证稳定性，供用户结合自身经验进行选择）；
3. 利用训练好的强化学习模型进行调优，或者使用全局优化算法在给定的参数空间内进行搜索；
4. 常规评价调优效果好坏的方法是跑benchmark获得反馈，调优框架除支持常规benchmark如TPC-C、TPC-H等，还为用户提供了自定义benchmark的框架，用户只需要进行少量工作进行适配即可；目前还在演进支持Performance Model的参数调优，将更进一步加快调优速度；.

在线参数调优流程概述:

1. 采集用户系统内的统计信息和workload，根据训练好的监督学习模型和先验规则，推荐给用户对应的参数配置。

```
Start to recommend knobs. Just a moment, please.
************************* Knob Recommendation Report *************************
INFO:
+-----------------------------------+----------------------+
|               Metric              |        Value         |
+-----------------------------------+----------------------+
|             workload_type         |          tp          |
|        average_connection_age     |          0           |
|       dirty_background_bytes      |          0           |
|            temp_file_size         |          0           |
|         current_connections       |         0.0          |
|         current_locks_count       |         0.0          |
|    current_prepared_xacts_count   |         0.0          |
|        rollback_commit_ratio      | 0.09168372786632421  |
|              uptime               |  0.122942879722222   |
| checkpoint_proactive_triggering_ratio | 0.488598416181662 |
|        fetched_returned_ratio     | 0.9915911452033203   |
|           cache_hit_rate          | 0.9979742937232552   |
|          read_write_ratio         | 123.86665549830312   |
|         all_database_size         | 134154882.48046875   |
|        search_modify_ratio        | 187.59523392981777   |
|              ap_index             | 2.3759498376861847   |
|          current_free_mem         |       31161892       |
|           os_mem_total            |       32779460       |
|       checkpoint_avg_sync_time    |  381.359603091308    |
| checkpoint_dirty_writing_time_window |       450.0       |
|           max_processes           |          46          |
|        track_activity_size        |         46.0         |
|         write_tup_speed           |  6810.36309197048    |
|             used_mem              |     73988850.25      |
|           os_cpu_count            |          8           |
|            block_size             |         8.0          |
|          read_tup_speed           |  845237.440716804    |
|   shared_buffer_toast_hit_rate    |  98.16007359705611   |
|   shared_buffer_tidx_hit_rate     |  99.11667280088332   |
|    shared_buffer_idx_hit_rate     |  99.74473859023848   |
|   shared_buffer_heap_hit_rate     |  99.81099543813004   |
|         enable_autovacuum         |         True         |
|              is_64bit             |         True         |
|              is_hdd               |         True         |
|           load_average            |  [1.89, 3.175, 3.005]  |
+-----------------------------------+----------------------+
p.s: The unit of storage is kB.
WARN:
[0]. The number of CPU cores is a little small. Please do not run too high concu
rrency. You are recommended to set max_connections based on the number of CPU co
res. If your job does not consume much CPU, you can also increase it.
[1]. The value of wal_buffers is a bit high. Generally, an excessively large val
ue does not bring better performance. You can also set this parameter to -1. The
 database automatically performs adaptation.
BAD:
[0]. The database runs for a short period of time, and the database description
may not be accumulated. The recommendation result may be inaccurate.
************************* Recommended Knob Settings **************************
+---------------------------+-----------+---------+----------+---------+
|            name           | recommend |   min   |   max    | restart |
+---------------------------+-----------+---------+----------+---------+
|       shared_buffers      |  1638973  |  614614 |  1884818 |   True  |
|     max_connections       |     43    |    24   |    500   |   True  |
|   effective_cache_size    |  1638973  | 1638973 | 24584595 |  False  |
|       wal_buffers         |   51217   |   2048  |   51217  |   True  |
|     random_page_cost      |    3.0    |    2.0  |    3.0   |  False  |
| default_statistics_target |    100    |    10   |    150   |  False  |
+---------------------------+-----------+---------+----------+---------+
```

openGauss

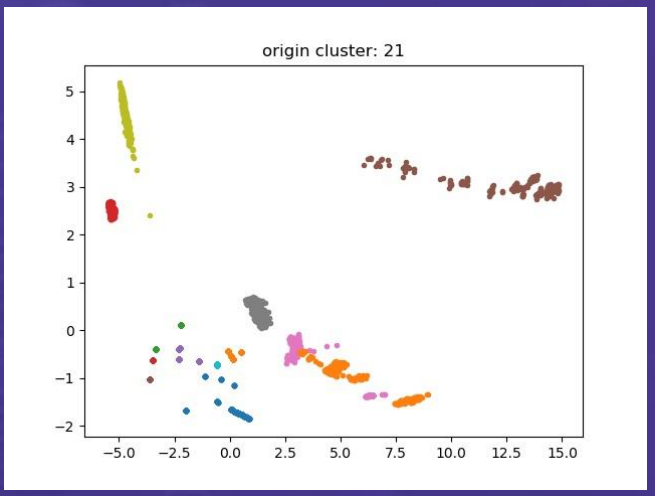
https://opengauss.org

openGauss慢SQL发现能力

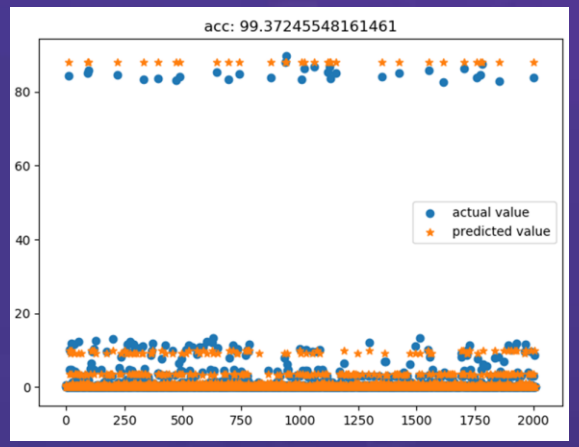
使用场景介绍：

1. 上线业务预检测：上线一批新业务前，使用SQL诊断功能评估此次上线业务的预估执行时长，便于用户参考是否应该修改上线业务。
2. Workload分析：对现有workload进行分析，自动分为若干类别，并依次分析此类别SQL语句执行代价，以及各个类别之间的相似程度；
3. SQL透视：能够将workload进行可视化，通过不同颜色和距离远近判断SQL语句之间的相似性，从而供用户直观地分析SQL语句的特点。

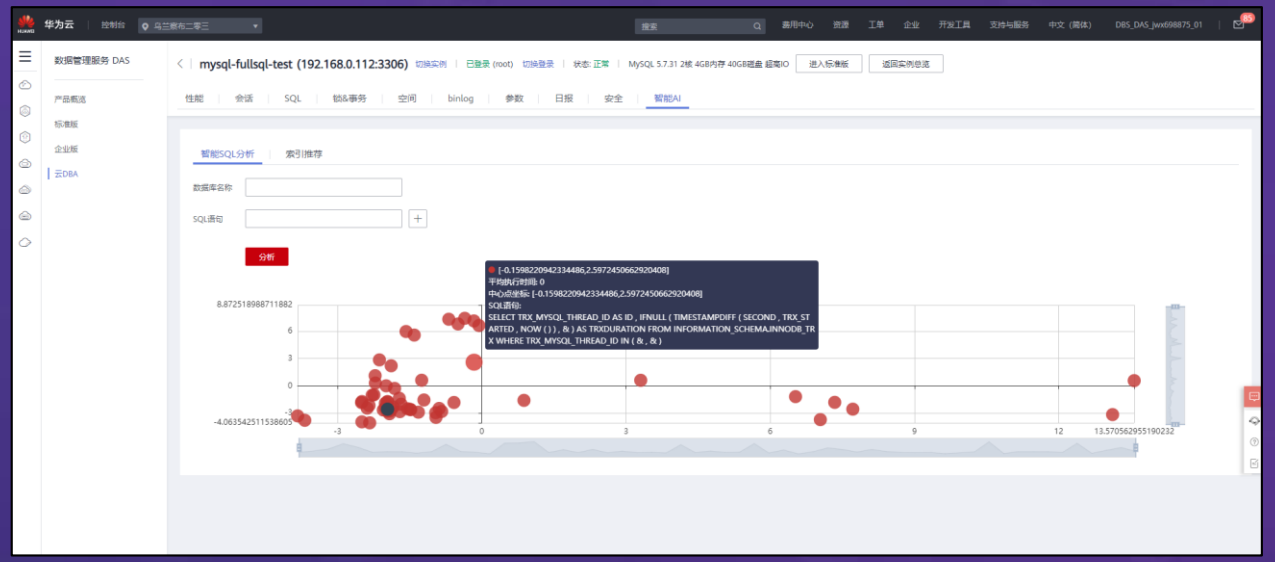
SQL透视效果



具备较高的预测准确率



华为云DAS上基于海量数据训练后的SQL透视效果



关键功能：

- 预估SQL语句的执行时间
- 根据SQL语句类型进行类别区分
- SQL语句分布情况的可视化

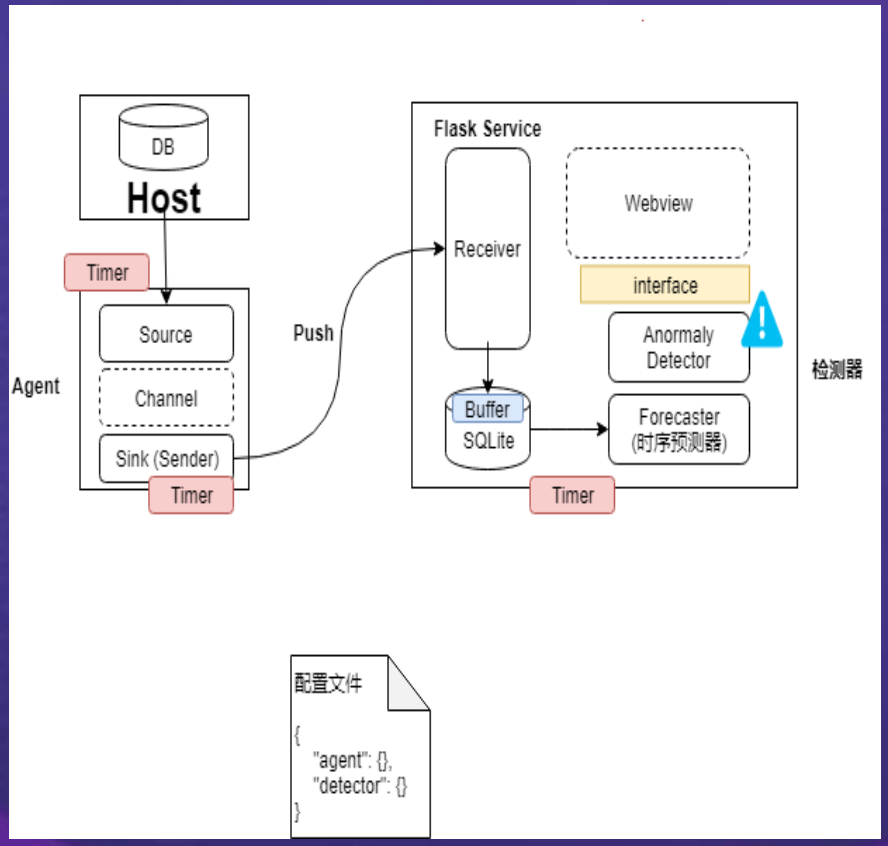
openGauss

https://opengauss.org

openGauss数据库监控与异常检测

数据库指标是数据库与用户行为健康的重要标志，数据库中的异常行为可能导致数据库指标产生异常，因此对指标进行有效的监控显得十分必要。
数据库状态监控，指对数据库运行指标进行全方位实时监控。系统能够发现和识别数据库异常以及潜在的性能问题，并及时将数据库异常报告给用户，通过针对各项运行指标的统计分析报告，帮助管理员、运维人员、决策者多视角了解数据库的运行状态，从而更好的应对数据库的需求及规划。

Detection 架构示意图



各个子模块

Agent由Source、Channel以及Sink组成。部署在数据库环境上的，用于采集数据库中的性能指标，并通过网络，将其传送给远端检测器模块。

Detector由数据收集Agent推送的数据，并将数据存储在本地磁盘，同时Detector基于时序预测和异常检测等算法对数据库指标进行监控和异常检测。

Receiver：数据接收器，接受Agent.HttpSink数据。同时将接受的数据推送到储存模块。

储存模块：对Receiver中的数据进行储存，目前主要支持数据库方式储存，默认是SQLite数据库。

算法模块：该模块是Detector的核心模块，实现了时序预测、异常检测、告警推送等功能。同时用户可以按照接口要求实现自定义的算法。

```
[l  .n@linux173 anomaly_detection]$ python main.py forecast --metric-name disk_space --forecast-periods 30M
+-------------+-----------------------------------------+--------------------+--------------------+--------------------+
| Metric name |               Date range                |      Minimum       |      Maximum       |      Average       |
+-------------+-----------------------------------------+--------------------+--------------------+--------------------+
|  disk_space | 2020-11-24 11:41:53~2020-11-24 12:11:52 | 102.35646415444381 | 105.60445111891163 | 103.83065385316687 |
+-------------+-----------------------------------------+--------------------+--------------------+--------------------+
[l   .n@linux173 anomaly_detection]$ python main.py forecast --metric-name disk_space --forecast-periods 1H
+-------------+-----------------------------------------+--------------------+--------------------+--------------------+
| Metric name |               Date range                |      Minimum       |      Maximum       |      Average       |
+-------------+-----------------------------------------+--------------------+--------------------+--------------------+
|  disk_space | 2020-11-24 11:41:53~2020-11-24 12:41:52 | 102.35646415444381 | 110.55916605998928 | 105.89202214450336 |
+-------------+-----------------------------------------+--------------------+--------------------+--------------------+
[likun@linux173 anomaly_detection]$ python main.py forecast --metric-name disk_space --forecast-periods 1D
+-------------+-----------------------------------------+--------------------+--------------------+--------------------+
| Metric name |               Date range                |      Minimum       |      Maximum       |      Average       |
+-------------+-----------------------------------------+--------------------+--------------------+--------------------+
|  disk_space | 2020-11-24 11:41:53~2020-11-25 11:41:52 | 102.35646415444381 | 306.9882691991314  | 211.11922600898393 |
+-------------+-----------------------------------------+--------------------+--------------------+--------------------+
[l.   .@linux173 anomaly_detection]$
```

openGauss单Query/ Workload级别索引推荐

使用场景介绍：

根据用户的Workload整体信息，为用户推荐需要创建的索引。

使用示例：使用gs_index_advise() 推荐索引，性能提高10000倍！

```
tpch=# select gs_index_advise('select * from lineitem where l_orderkey < 100 and l_suppkey > 50;');
       gs_index_advise
-----------------------------
 (lineitem,"(l_orderkey)")
(1 row)
```

← 建议在lineitem表的l_orderkey列上创建索引。

```
tpch=# explain analyze select * from lineitem where l_orderkey < 100 and l_suppkey > 50;
                                                        QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------------
 Gather  (cost=1000.00..898324.11 rows=5946 width=129) (actual time=37831.112..37834.006 rows=105 loops=1)
   Number of Workers: 2
   ->  Parallel Seq Scan on lineitem  (cost=0.00..896729.51 rows=2478 width=129) (actual time=113448.603..113456.964 rows=105 loops=3)
         Filter: ((l_orderkey < 100) AND (l_suppkey > 50))
         Rows Removed by Filter: 59985947
 Total runtime: 37888.647 ms
(6 rows)

tpch=# create index idx_orderkey on lineitem(l_orderkey);
CREATE INDEX
tpch=# explain analyze select * from lineitem where l_orderkey < 100 and l_suppkey > 50;
                                                     QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------
 Index Scan using idx_orderkey on lineitem  (cost=0.00..18.79 rows=97 width=129) (actual time=0.022..0.095 rows=105 loops=1)
   Index Cond: (l_orderkey < 100)
   Filter: (l_suppkey > 50)
 Total runtime: 1.850 ms
(4 rows)

tpch=#
```

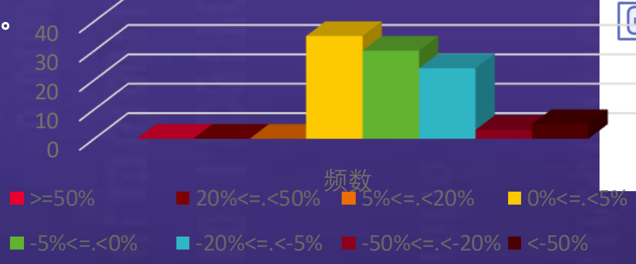
对索引推荐效果的整体评估：

TPC-DS: 除40%左右无明显变化外，其余语句均有不同程度的执行时间缩短；

TPC-C: 与原生索引基本持平，比无索引有巨大性能提升。

	无索引	原索引	算法一	算法二
tpmC	3.22	135.64	134.27	134.49
tpmTOTAL	8.06	299.64	299.24	298.69
Transaction Count	120.67	4495.00	4489.00	4480.67

TPC-C



TPC-DS

单Query索引推荐的核心方法：

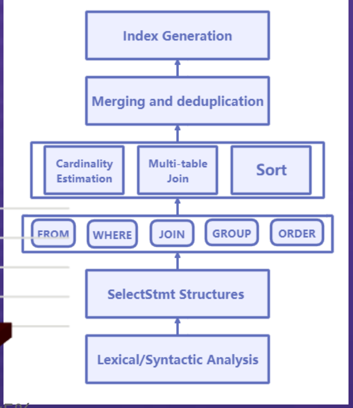
采用索引设计和优化的相关理论，基于原生的词法和语法解析，对查询语句中的子句和谓词进行分析和处理，再结合字段选择度、聚合条件、多表Join关系等输出最终建议。

索引性能验证的方法：

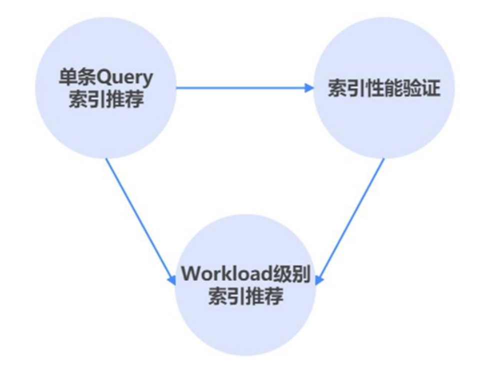
通过修改优化器相应的数据结构，利用**优化器评估**，进而判断创建该索引后，对优化器生成执行计划的影响。该过程可以不用真正创建索引，即业内所谓的“**假设索引**”，业内也多采用此种方法。

Workload级别索引推荐的核心方法：

通过用户输入（或自主采集）得到的workload信息，根据预设模型，进一步评估创建索引**对整体Workload的影响**，从而从候选索引中筛选出若核心索引。



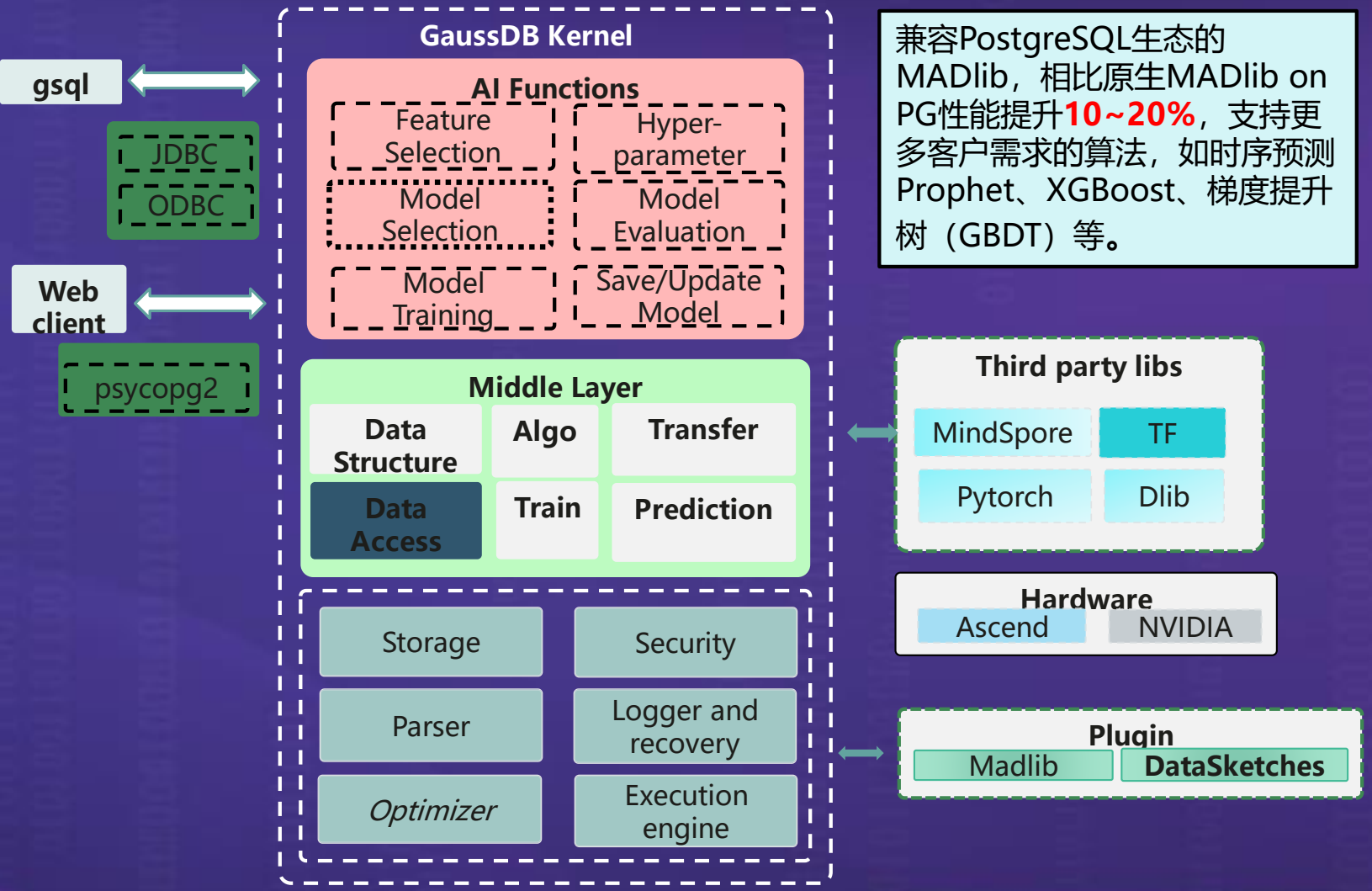
从单条到workload



openGauss

https://opengauss.org

openGauss DB4AI



组件说明:

1. **自动特征工程**: 从相关的数据表中自动提取有用且有意义的特征，减少了特征工程所需的时间。将周级的处理时间缩短至天级。
2. **模型最优选择**: 在多个模型中，选择出最适合本数据的模型。
3. **超参数优化**: 选出合适模型后，并且能够设定好它的最优参数。
4. **优化算法选择**: 自动地选择出一个优化算法（如SGD、L-BFGS、GD等），以便能够达到效率和精度的平衡。
5. **支持深度学习**: 支持主流深度学习框架。并自动地选择出一个优化算法，以便能够达到效率和精度的平衡。
6. **模型管理**: 记录模型准确度等相关信息。增加模型生命周期管理，支持增量训练，模型更新，支持模型导入和导出。

openGauss

https://opengauss.org

openGauss 核心技术总结

1、在CPU　NUMA 多核的硬件发展趋势下，openGauss通过线程绑核，NUMA化数据结构改造，数据分区和原子指令优化实现150W tpmc。

2、企业可用性指标为RPO和RTO，openGauss支持双机同步保证RPO=0，通过极致RTO技术保证RTO<10s。

3、企业的性能指标为吞吐量和时延，openGauss通过服务器线程池支持企业的高并发，通过增量检查点保证IO性能的稳定性。

4、企业的业务场景为OLTP和OLAP，openGauss通过行列混合引擎同时支持行存和列存，适应企业混合场景。

5、在风控，计费等极端性能企业场景下，openGauss通过免锁内存表，内存索引算法保证高吞吐，低时延，满足企业场景要求。

6、在云化的发展趋势下，openGauss通过全密态实现端到端加密，解决企业上云安全顾虑。

7、通过DB4AI和AI4DB，实现openGauss自运维和调优，减少企业应用开发和维护的TCO。

openGauss 核心技术及实践

- openGauss 开源社区介绍
- openGauss 架构
- openGauss 核心技术及实践
- **openGauss 未来技术发展方向**

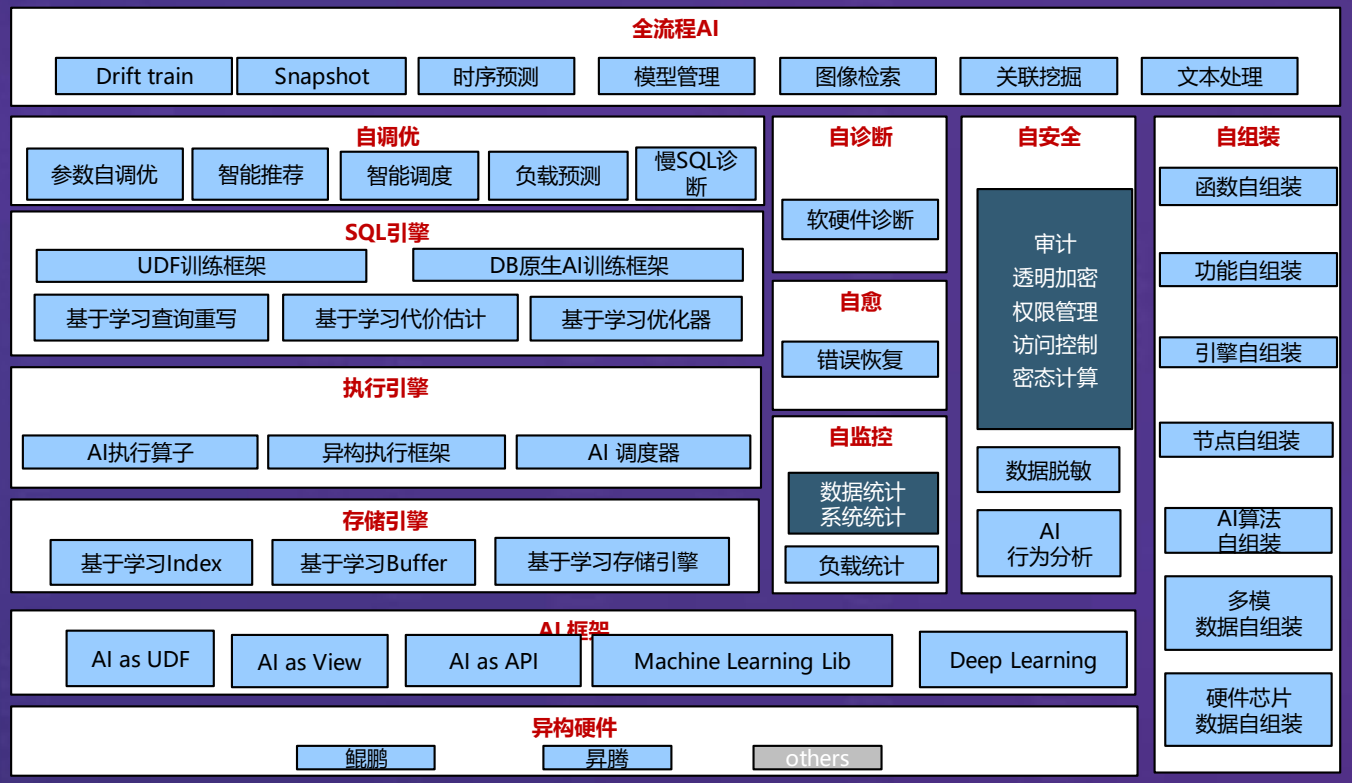
openGauss

https://opengauss.org

openGauss未来技术方向：**更智能**、更安全、更高效

- **AI算法加持，解决传统数据库问题**
  - 索引/视图推荐，如Index Advisor
  - 学习型数据库组件，如Learned Index等
  - 数据库自治运维，如AI-Based Optimizer
- **数据库系统的软硬件故障自诊断与自定位技术**
- **数据库内AI技术**

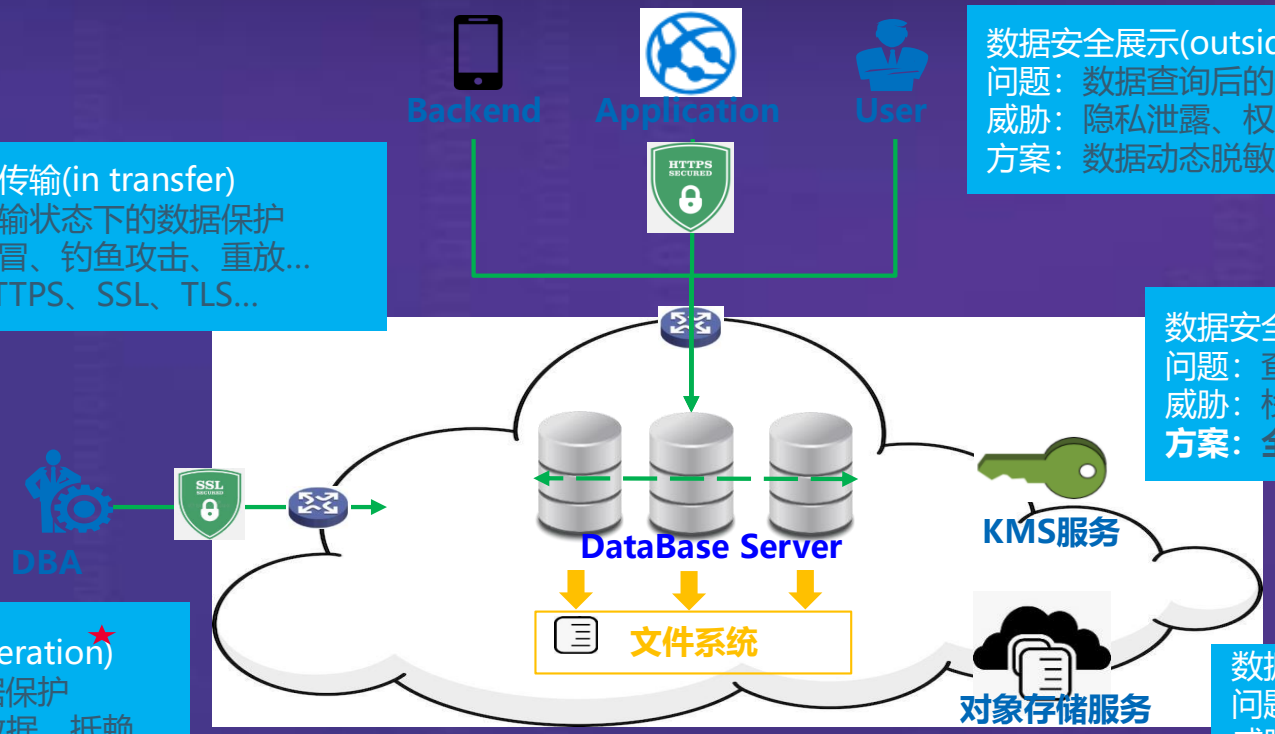
DBMind功能全景图



openGauss

https://opengauss.org

openGauss未来技术方向：更智能、更安全、更高效



数据安全传输(in transfer)
问题：传输状态下的数据保护
威胁：仿冒、钓鱼攻击、重放...
方案：HTTPS、SSL、TLS...

数据安全展示(outside system)
问题：数据查询后的数据保护
威胁：隐私泄露、权限提升...
方案：数据动态脱敏、Row Level Security

数据安全计算(in calculation)
问题：查询计算中的数据保护
威胁：栈溢出信息泄露
方案：全密态数据管理

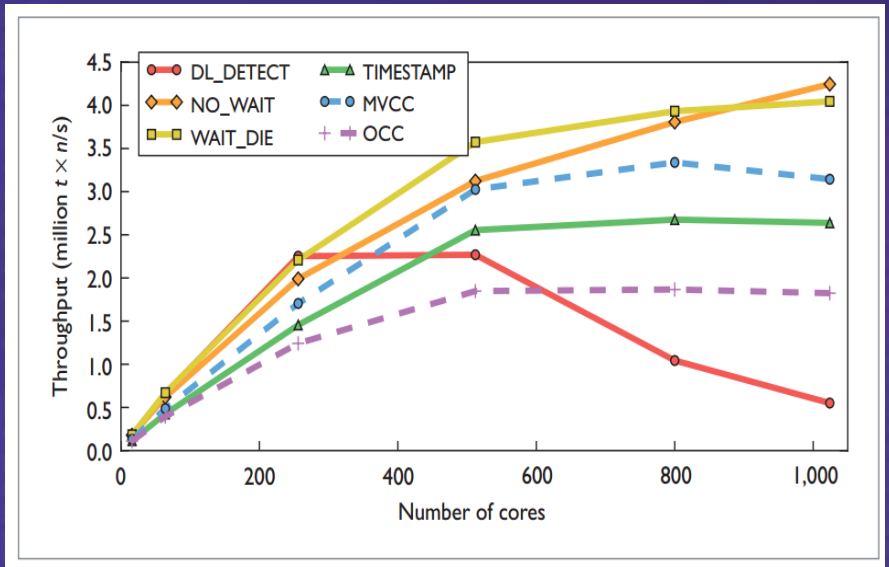
数据安全运维(under operation)
问题：运维形态下的数据保护
威胁：权限提升、篡改数据、抵赖...
方案：防篡改数据管理

数据安全存储(at rest)
问题：存储状态下的数据保护
威胁：拖库、信息提取
方案：透明加密(Transparent Data Encryption)、数据存储加密(Encryption API)

openGauss

https://opengauss.org

openGauss未来技术方向：更智能、更安全、更高效



众核系统线性scale-up

- 竞争冲突→Latch-free/wait-free；
- 注： Stonebraker 2014 VLDB: Staring into the abyss: an evaluation of concurrency control with one thousand cores

	DRAM	PCM	RRAM	MRAM	SSD	HDD
Read latency	60 ns	50 ns	100 ns	20 ns	25 μs	10 ms
Write latency	60 ns	150 ns	100 ns	20 ns	300 μs	10 ms
Addressability	Byte	Byte	Byte	Byte	Block	Block
Volatile	Yes	No	No	No	No	No
Energy/ bit access	2 pJ	2 pJ	100 pJ	0.02 pJ	10 nJ	0.1 J
Endurance	$>10^{16}$	10^{10}	10^{8}	10^{15}	10^{5}	$>10^{16}$

DB结合新硬件整合优化，解决传统架构挑战

- 持久化内存：秒级故障恢复→毫秒级故障恢复
- 注：Joy Arulraj, Andrew Pavlo ,Subramanya R. Dulloor 2015 SIGMOD Let’ s Talk About Storage & Recovery Methods for Non-Volatile Memory Database Systems

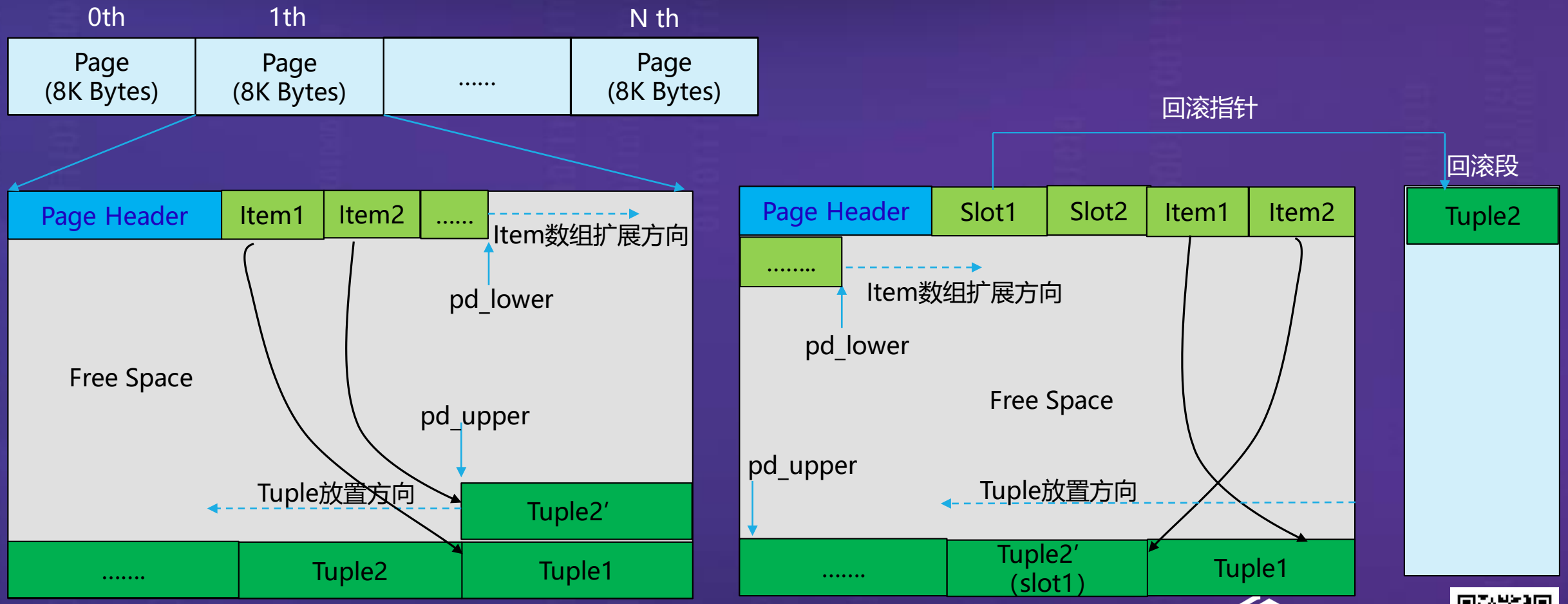
openGauss

https://opengauss.org

openGauss 2021年主要规划特性：性能，可靠，安全

DFX维度	关键特性	特性描述
性能	2P鲲鹏提升20%，4P鲲鹏230Wtpmc。	2P 提升20%，4P 230W tpmc，2P到4P线性度1.5。 In-place Update引擎开源，24小时性能弱抖动。 基于libnet和DPDK的网络加速。
可靠	支持Paxos协议。 备机能力增强。	主备支持Paxos协议，自选主。 备机能力增强：备机备份，逻辑复制，延迟回放，备机IO优化。
安全	全密态支持不等于查询。 支持国密加密算法。	全密态支持不等于运算符。 支持国密加密算法（SM3/SM4）。 敏感数据发现和保护。 存储透明加密。 防篡改数据库(单中心账本能力)。
AI	根因分析和AI算子扩展	AI4DB：根因分析（慢SQL为什么慢） DB4AI：库内AI算子，通过SQL语法进行训练和预测。
实时分析	支持混合负载，数据交易与报表分析实时处理。 支持IoT场景边缘节点实时分析。	一个实例交易，另一个实例支持分析。 行存并行查询优化。 列存插入性能优化（delta store）。 向量化查询+CodeGen+列存，查询性能提升50%。
未来探索	架构演进	基于Memory Fabric的架构探索。 基于4P Pro的UCE的故障感知与修复。 基于1620网卡的性能优化。

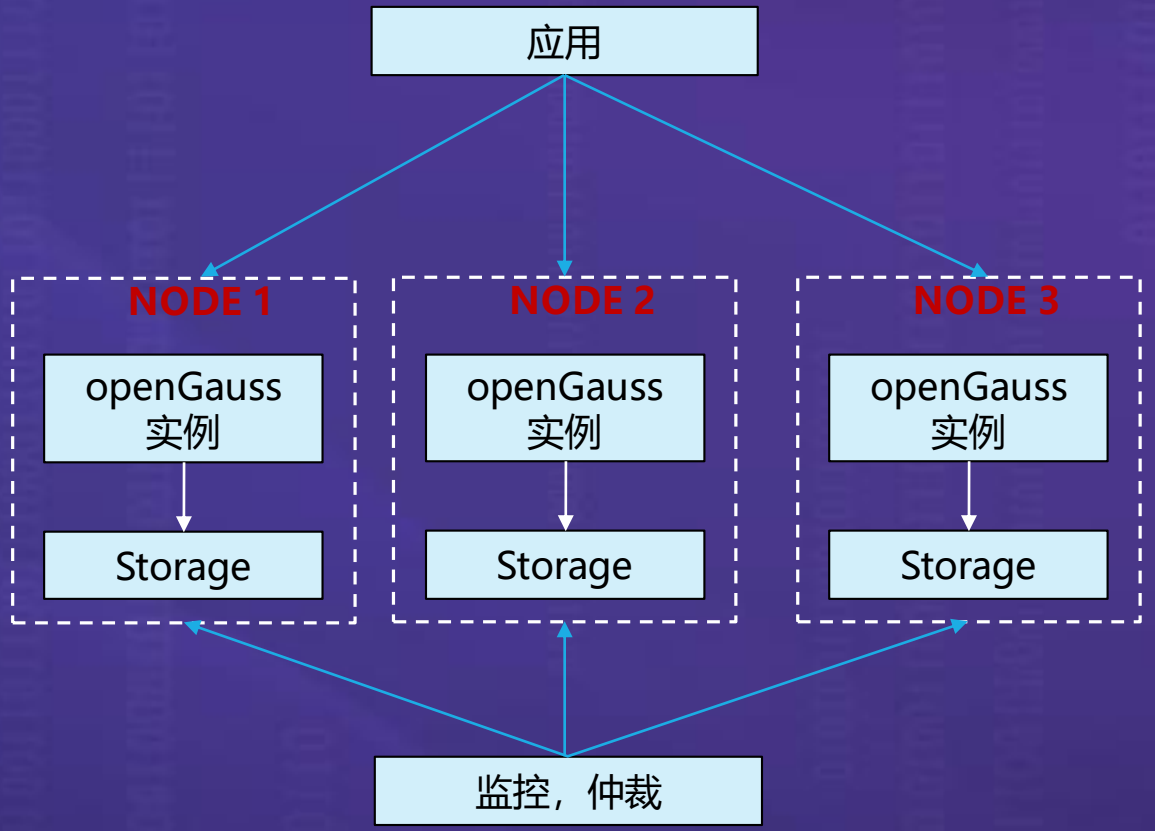
openGauss即将发布技术特性：In-place Update

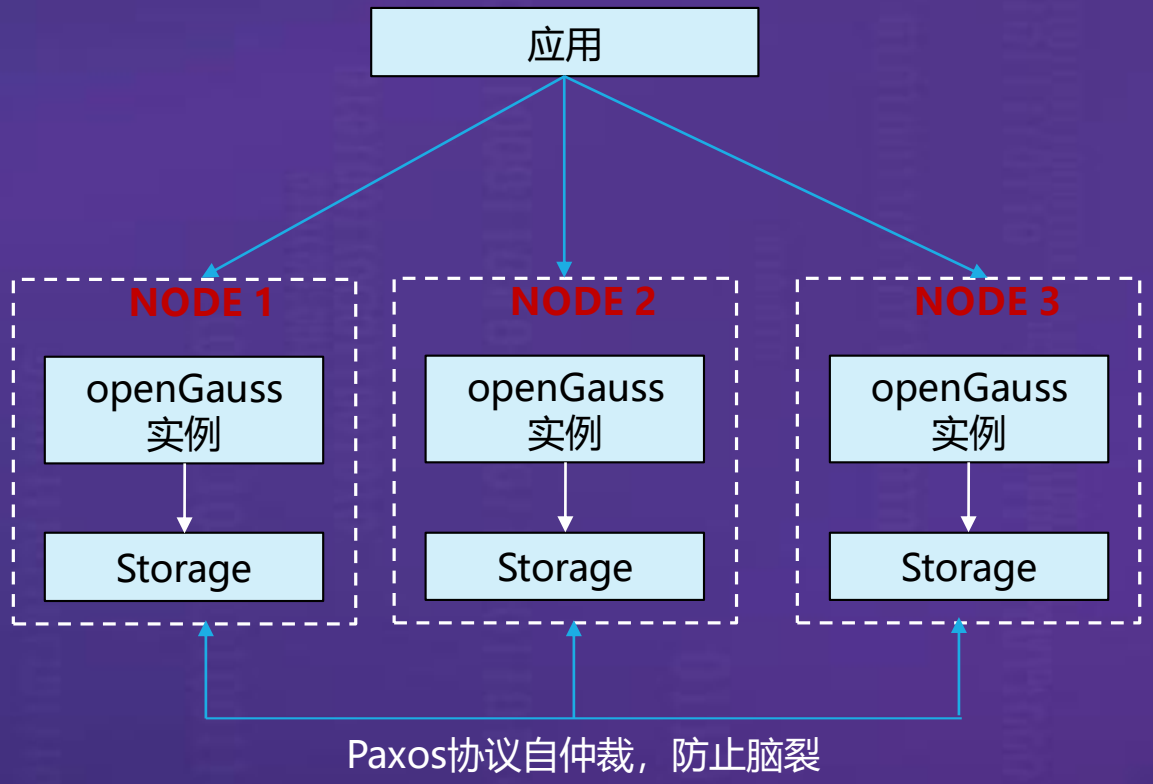


openGauss

https://opengauss.org

openGauss即将发布技术特性：paxos协议自选主





openGauss

https://opengauss.org

openGauss 社区交流

我们希望与各位一起，

共同打造更优秀的openGauss数据库！

共同构筑良好的openGauss数据库生态！

微信公众号

添加微信小助手
即可加入官方社群交流

官网：https://opengauss.org

邮箱：contact@opengauss.org

openGauss

https://opengauss.org

Thank you!

openGauss

https://opengauss.org