

openGauss 技术主打胶片



<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

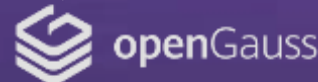
硬件开放

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



<https://opengauss.org>



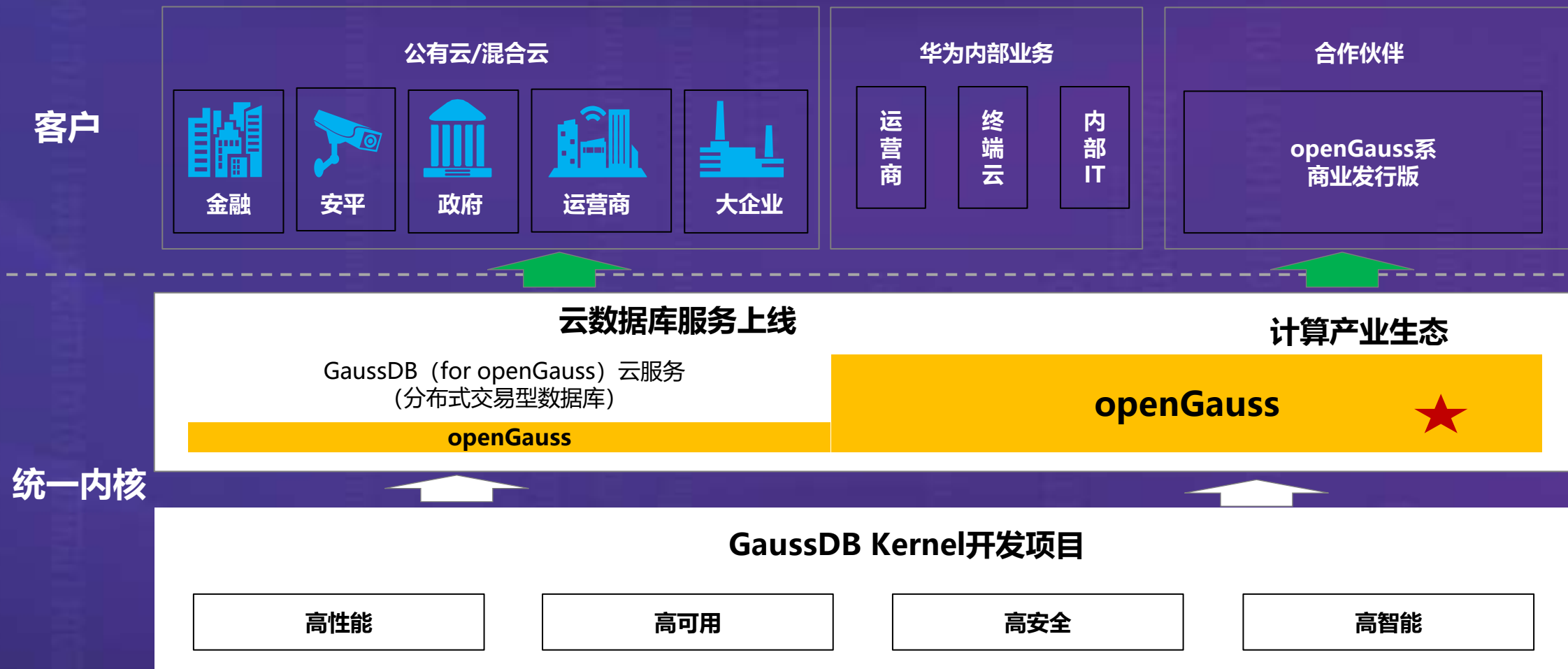
| 华为GaussDB演进历程

内部自用孵化阶段 → 联创产品化阶段 → 共建生态



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



| openGauss社区：持续开放社区治理架构，共建、共享、共治



<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；
- Inplace-Update引擎

高可用 & 高安全

- 业务无忧，故障切换时间**RTO<10s**；
- ③ 基于Paxos协议的高可用
- 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&
- ⑥ 传输&导出加密、动态脱敏、全密态计算、防篡改；

易运维

- AI自治运维：参数推荐、慢SQL诊断、索引推荐、趋势预测及异常检测等；
- WDR报告：多维性能自监控视图，实时掌控系统性能表现；
- 库内AI引擎：通过简易SQL接口提供库内机器学习算法的训练和推理；

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



<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



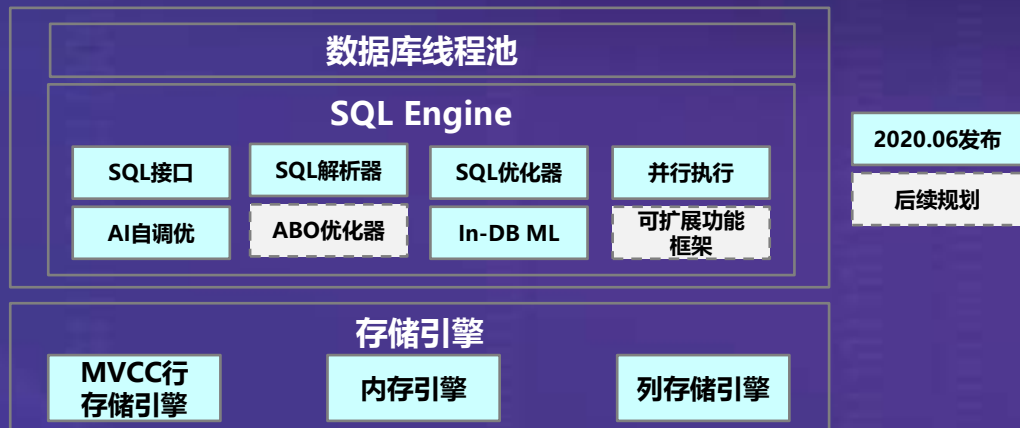
openGauss

<https://opengauss.org>

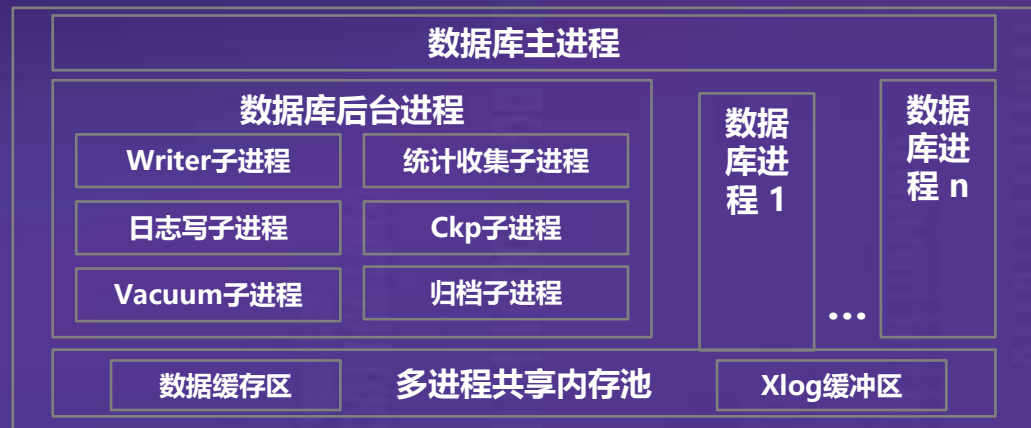


openGauss VS. PG架构和关键技术对比

openGauss架构



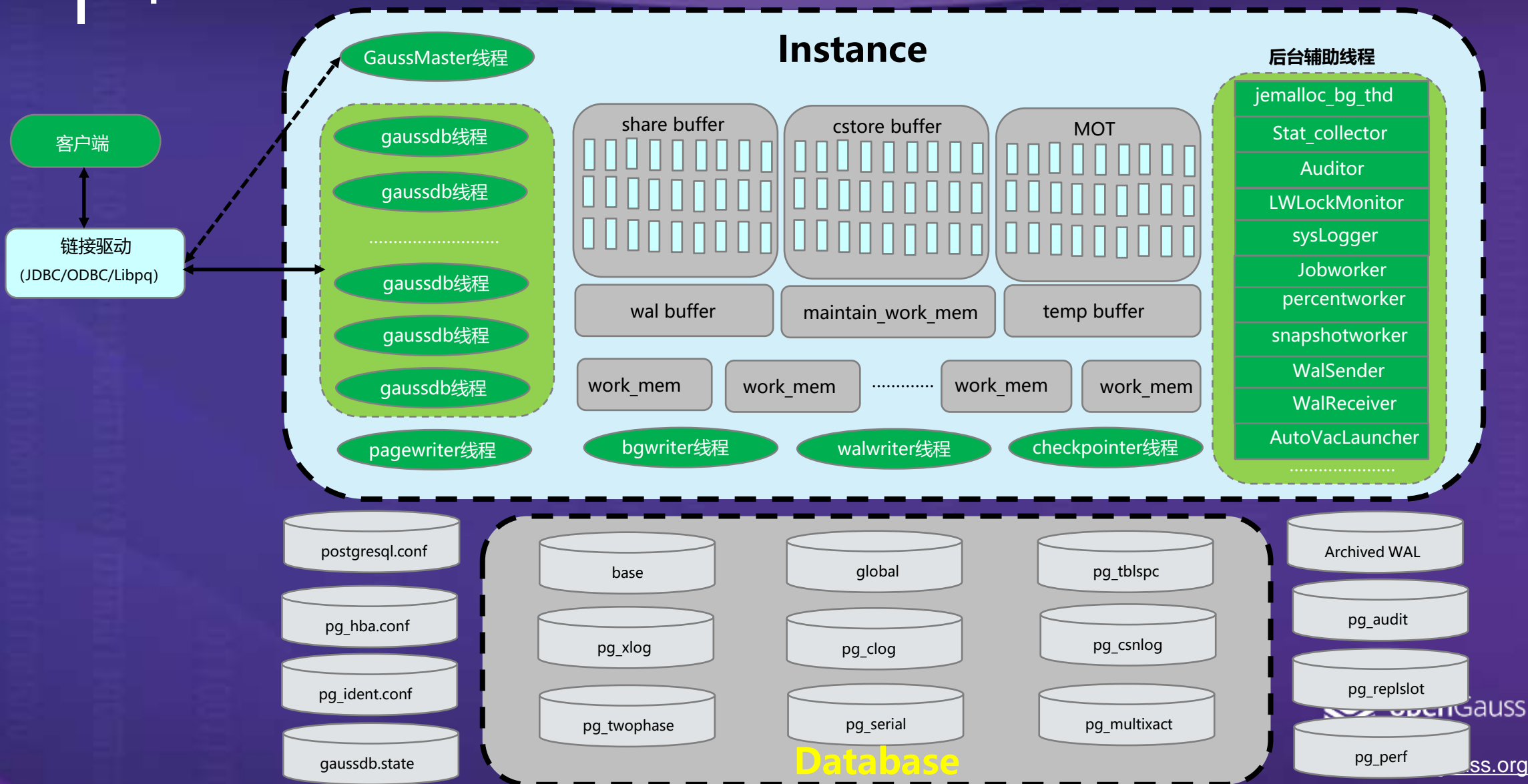
PostgreSQL架构



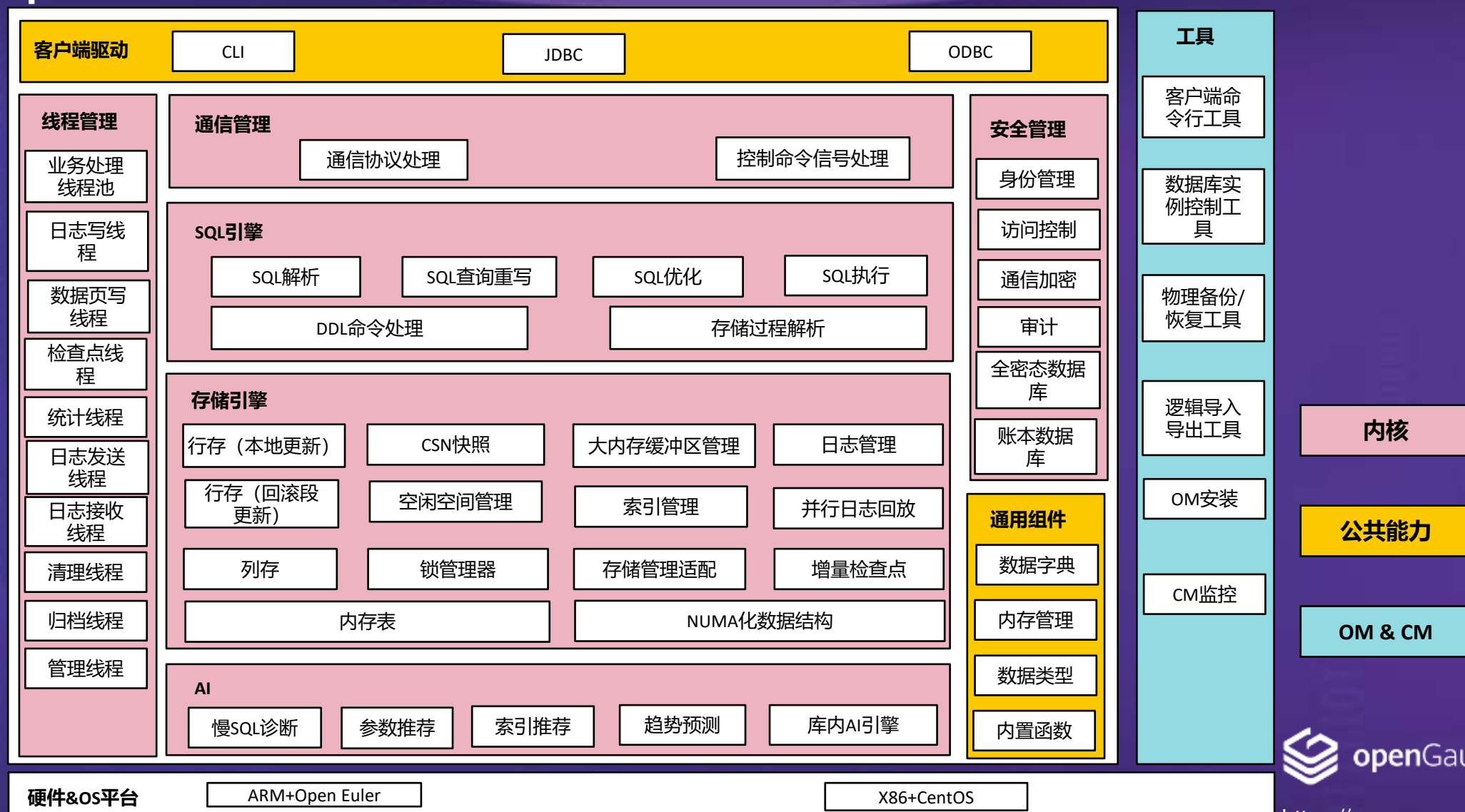
关键差异化因素		openGauss	PostgreSQL
运行时模型	执行模型	线程池模型，高并发连接切换代价小、内存损耗小，执行效率高，一万并发连接比最优性能损耗<5%	进程模型，数据库进程通过共享内存实现通讯和数据共享。每个进程对应一个并发连接，存在切换性能损耗，导致多核扩展性问题。
事务处理机制	并发控制	64位事务ID，使用CSN解决动态快照膨胀问题；NUMA-Aware引擎优化改造解决“五把大锁”	事务ID回卷，长期运行性能因为ID回收周期大幅波动；存在“五把大锁”的问题，导致事务执行效率和多处理器多核扩展性存在瓶颈
	日志和检查点	增量Checkpoint机制，实现性能波动<5%	全量checkpoint，性能短期波动>15%
	鲲鹏NUMA	NUMA改造、cache-line padding、原生spin-lock	NUMA多核能力弱，单机两路性能TPMC <60w
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	仅支持行存
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	支持CBO，复杂场景优化能力一般
	SQL解析	ANSI/ISO标准SQL92、SQL99和SQL2003和企业扩展包	ANSI/ISO标准SQL92、SQL99和SQL2003



openGauss 体系结构



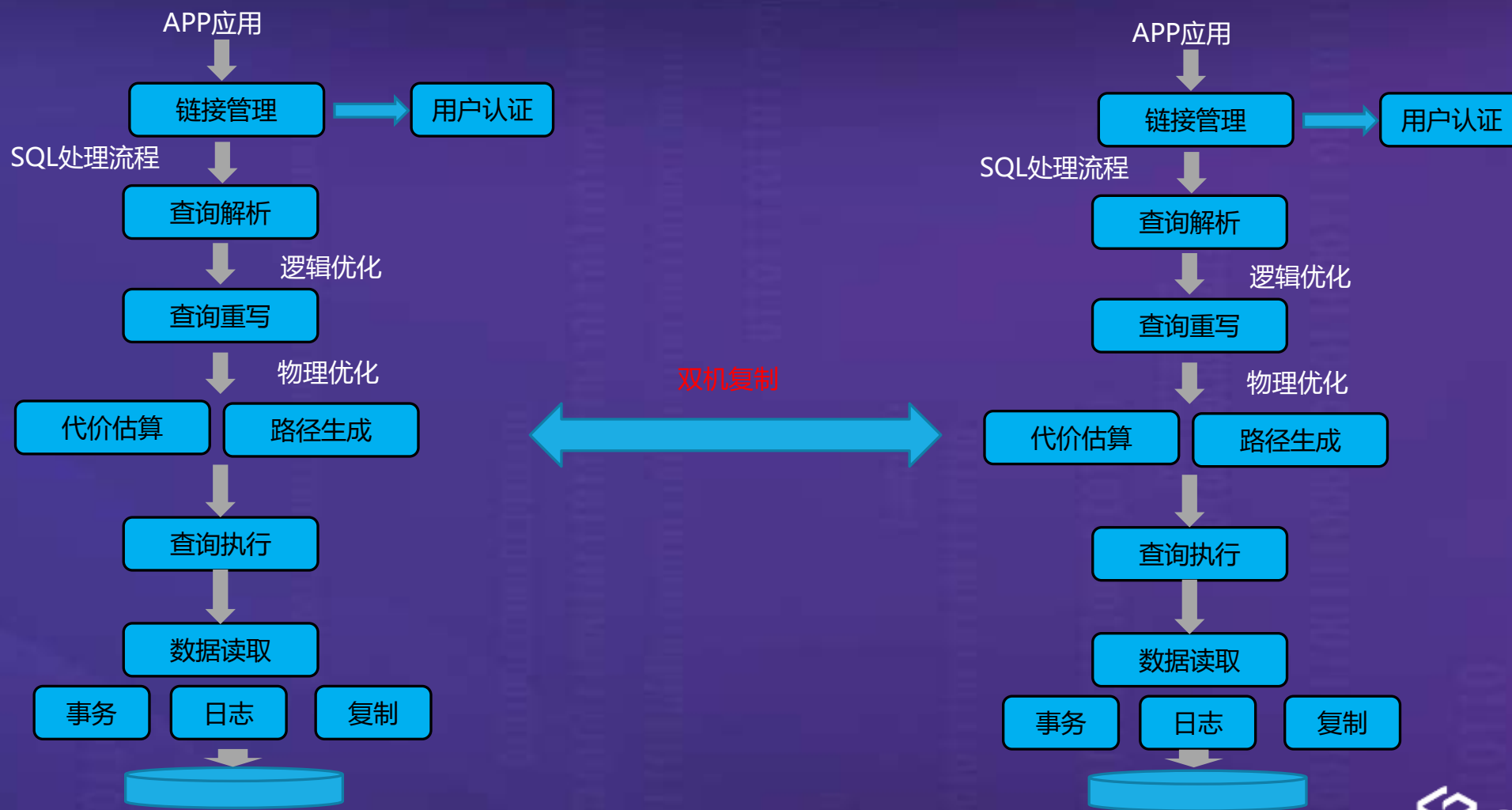
openGauss 逻辑模块



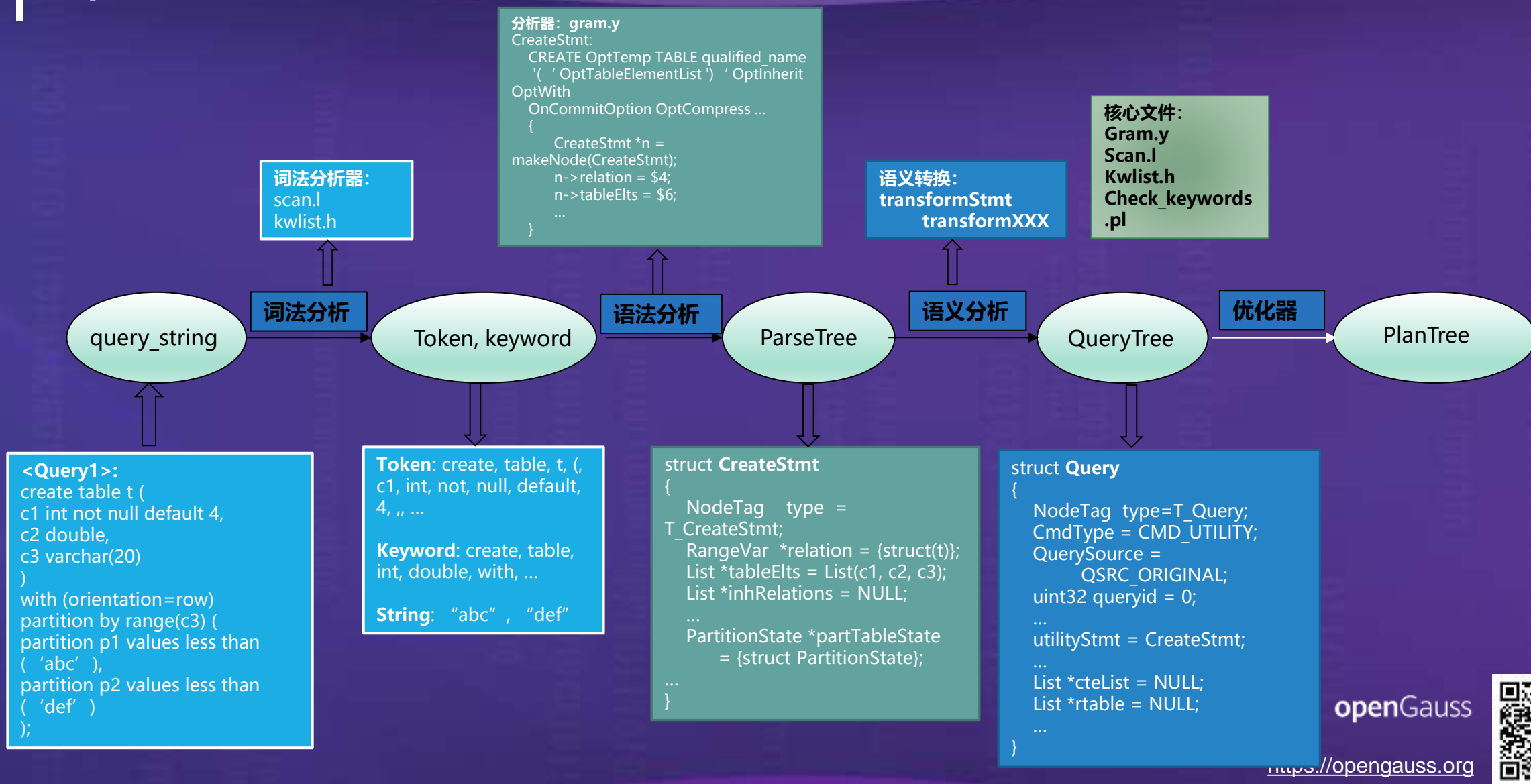
<https://opengauss.org>



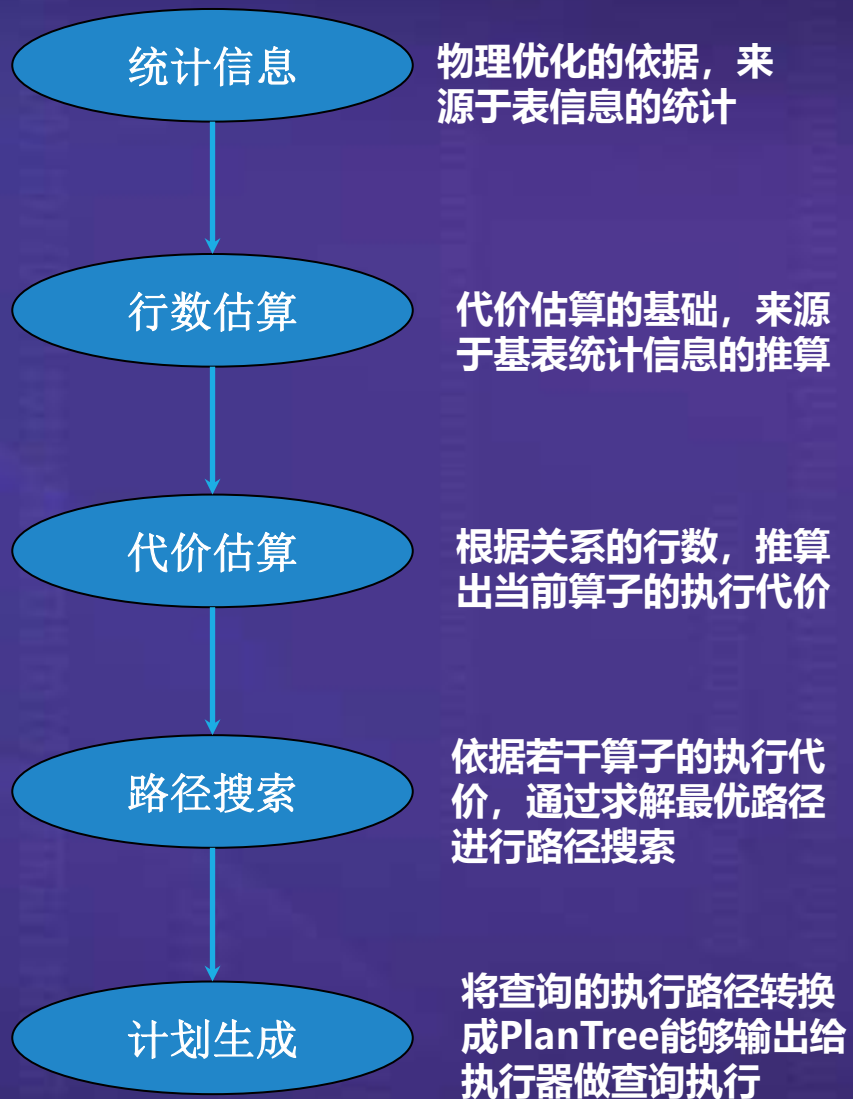
openGauss SQL命令处理流程



openGauss SQL引擎



openGauss 查询优化



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

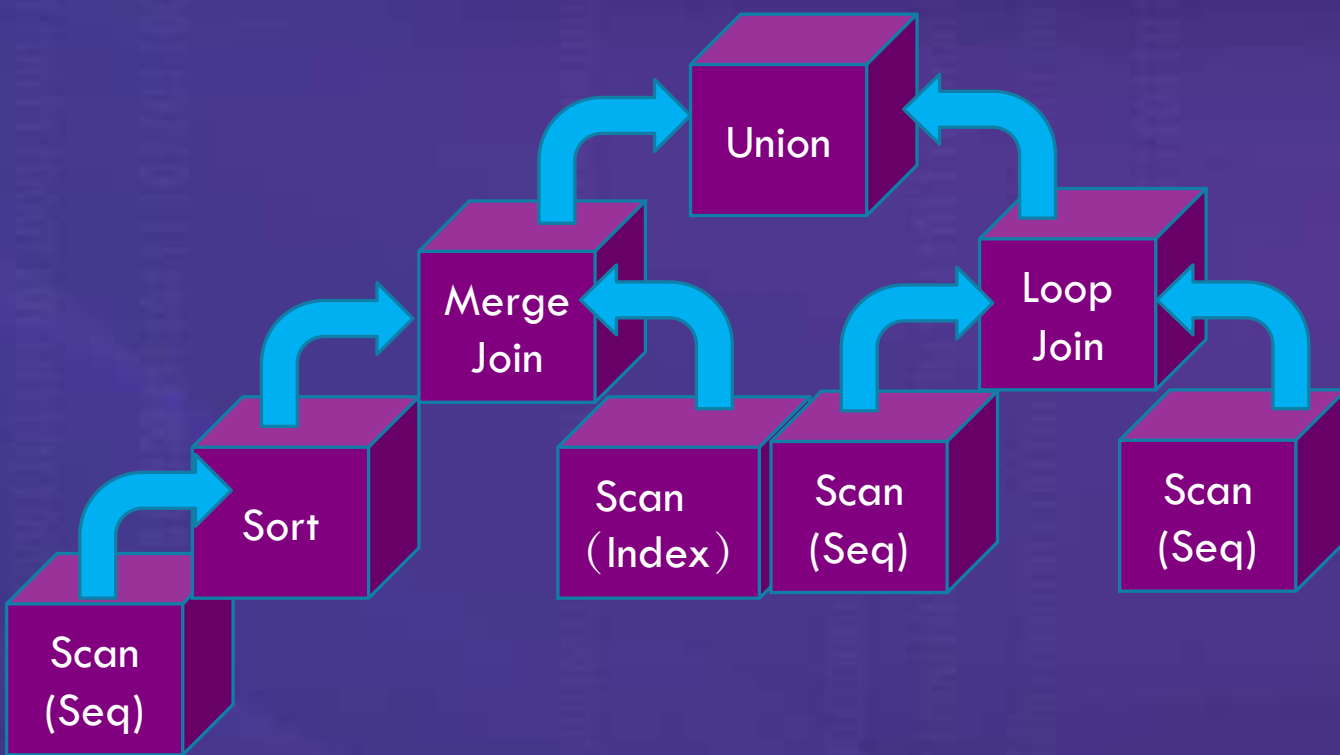
通过求解路径最优算法（e.g. 动态规划、遗传算法）处理连接路径搜索过程，以最小搜索空间找到最优连接路径



<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。

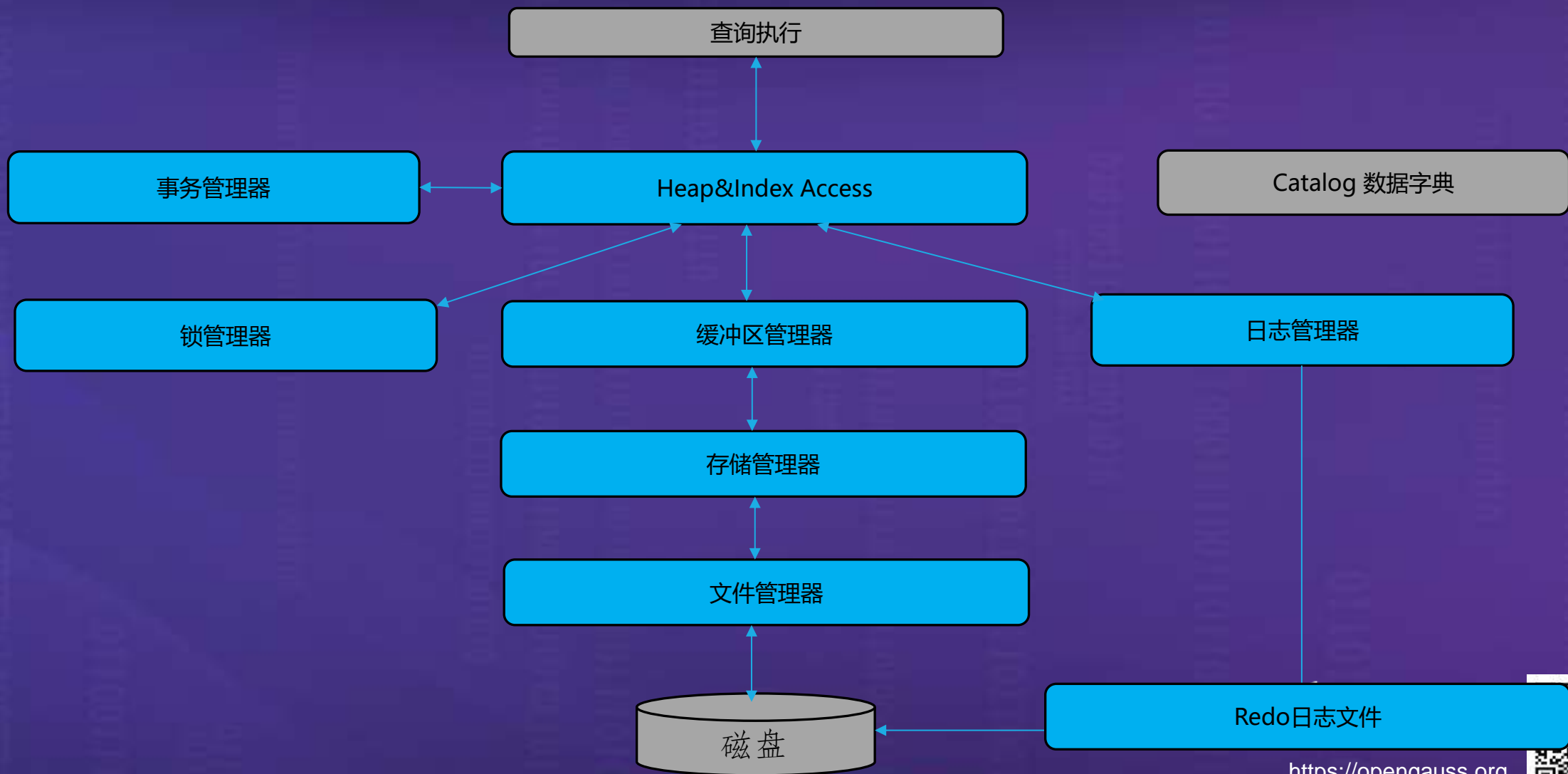
关键特征：多个输入



<https://opengauss.org>



| openGauss 存储引擎



| 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
 - 线程池原理
 - 增量检查点原理
 - 行列混合引擎
 - MOT内存表原理
 - Inplace Update引擎
- ◆ 可用性
 - 极致RTO优化
 - Paxos协议自选主
- ◆ 安全性
 - 全密态等值查询原理
 - 账本数据库
- ◆ 易运维
 - AI4DB自治数据库
 - DB4AI库内AI引擎

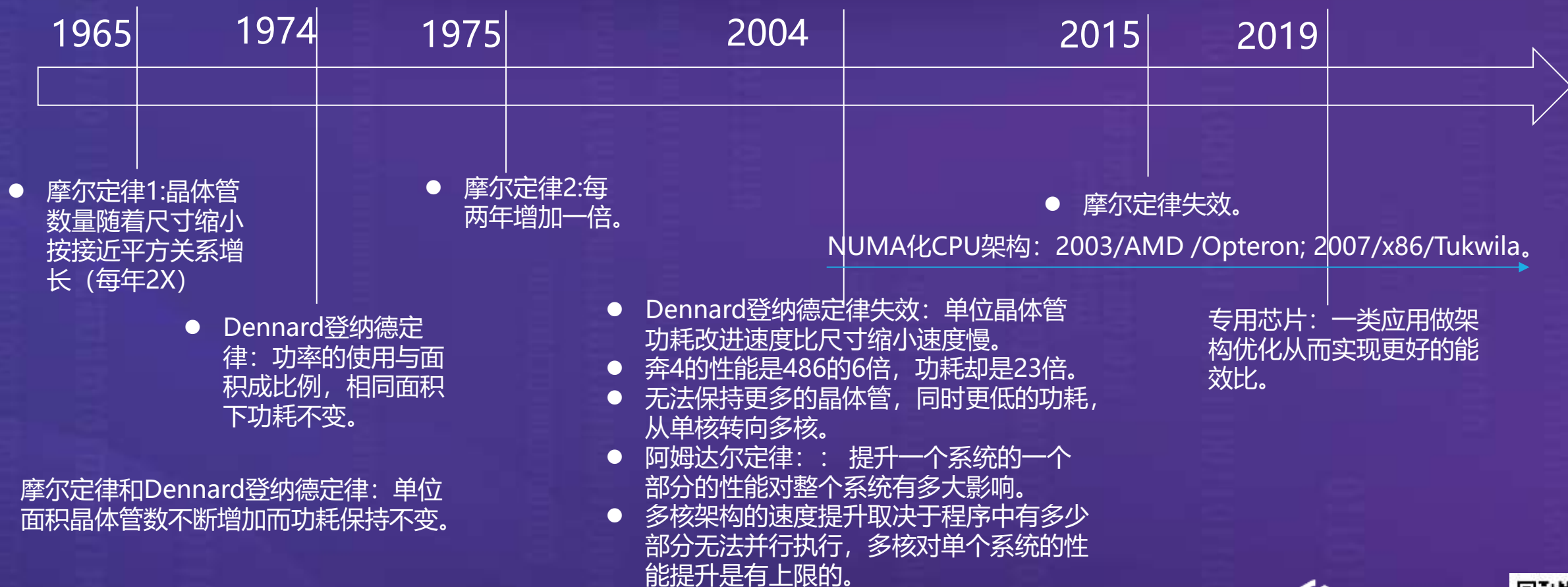
■ openGauss 未来技术发展方向



<https://opengauss.org>



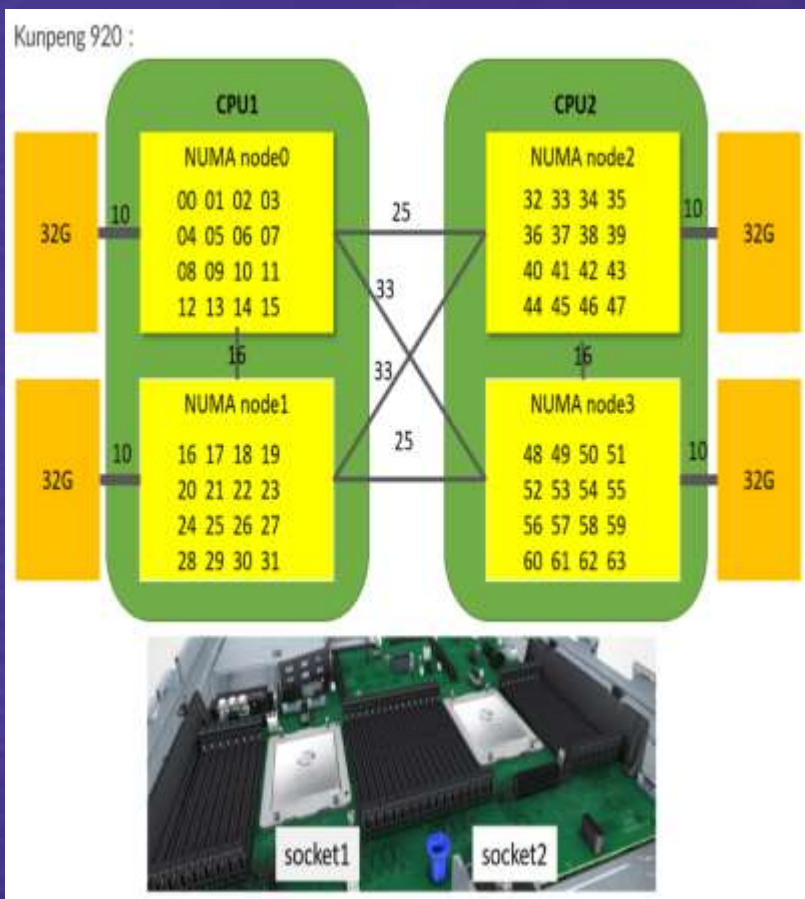
| openGauss CPU体系架构发展历史



摩尔定律和Dennard登纳德定律: 单位面积晶体管数不断增加而功耗保持不变。



| openGauss NUMA CPU结构

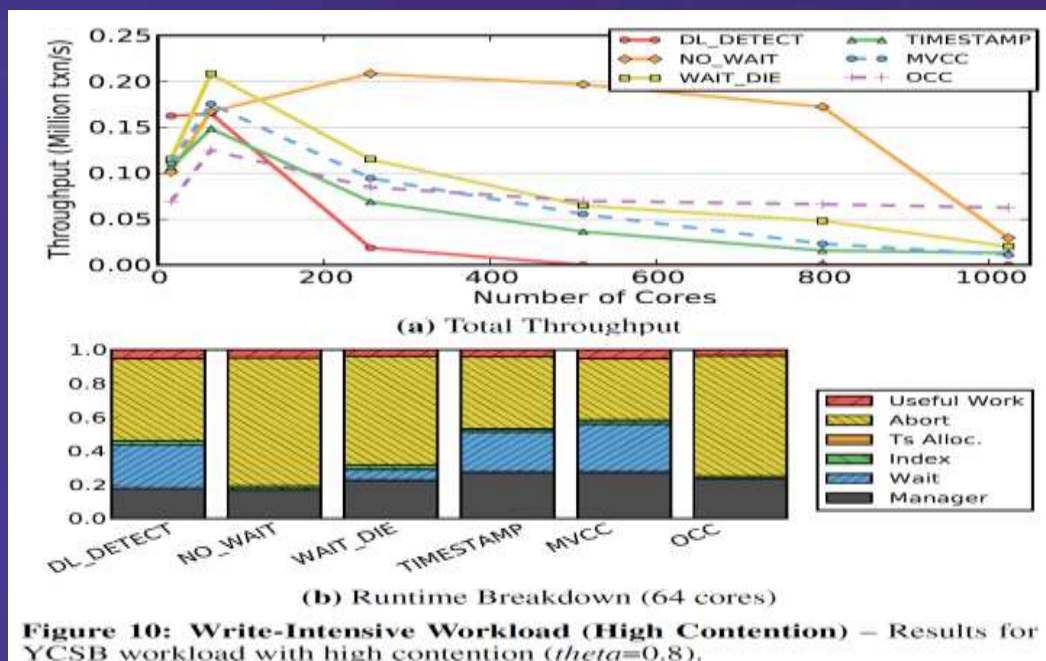


- 1、NUMA: Non Uniform Memory Access.
- 2、NUMA距离: NUMA node的处理器和内存块的物理距离。通过numactl工具可以查询到CPU访问的距离信息。

node/ node distances	0	1	2	3
0	10	16	32	33
1	16	10	25	32
2	32	25	10	16
3	33	32	16	10



openGauss NUMA CPU架构数据库的并发问题



```
[minqi@localhost dTrace]$ ./lwlock_wait.d
```

Lock Id	Mode	Count
CLogControlLock	Exclusive	2085
CLogControlLock	Shared	2336
WALInsertLock	Exclusive	3993
ProcArrayLock	Shared	7400
ProcArrayLock	Exclusive	34405
WALWriteLock	Exclusive	208354
XidGenLock	Exclusive	275126

Lock Id	Mode	Combined Time (ns)
CLogControlLock	Exclusive	28956662
CLogControlLock	Shared	29554120
WALInsertLock	Exclusive	58710394
ProcArrayLock	Shared	125250951
ProcArrayLock	Exclusive	446470964
WALWriteLock	Exclusive	19448626395
XidGenLock	Exclusive	450481068249

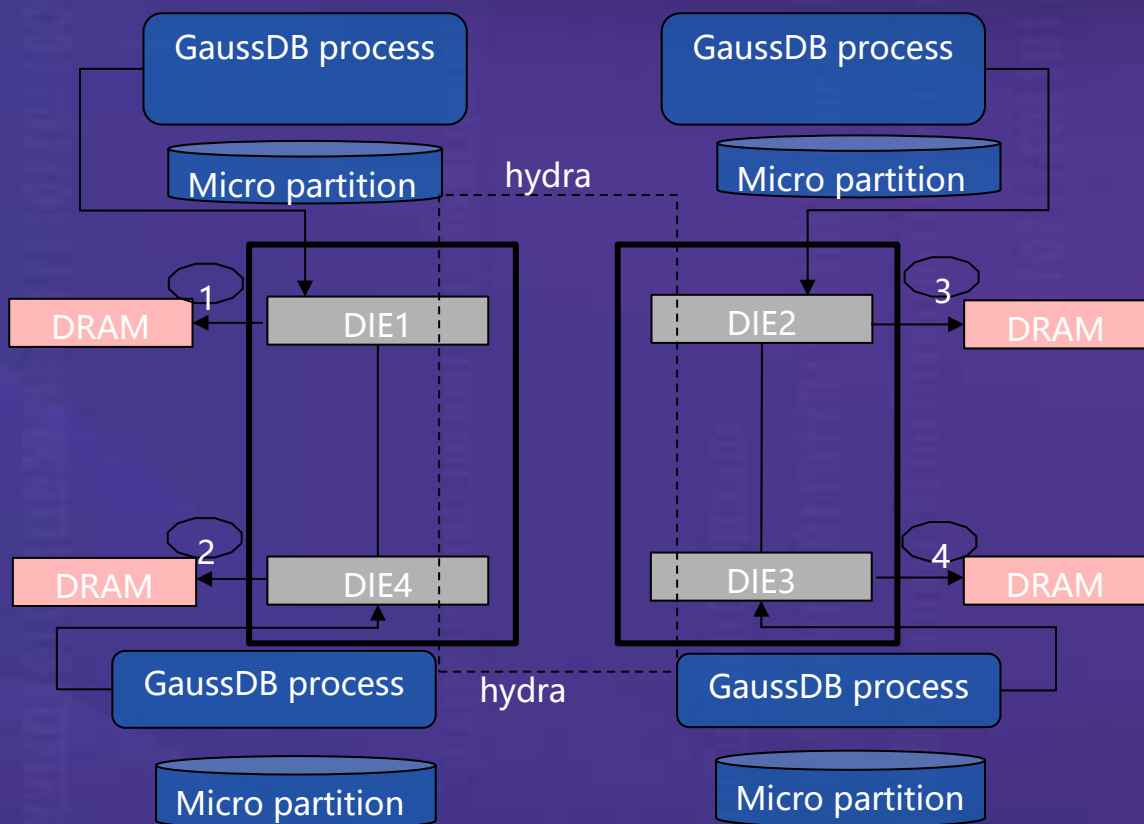
- 1、数据库是高并发，数据访问冲突严重的一些软件系统。数据库领域图灵奖获得者Stonebraker等人于2014年在 VLDB发表的论文：Staring into the abyss: an evaluation of concurrency control with one thousand cores，传统数据库的事务处理机制无法有效利用数十到上百个核处理能力。
- 2、数据库中存在很多锁，这些锁是性能瓶颈点：Clog、WALInsert、WALWrite、ProcArray、XidGen。



<https://opengauss.org>



openGauss NUMA 内核数据结构

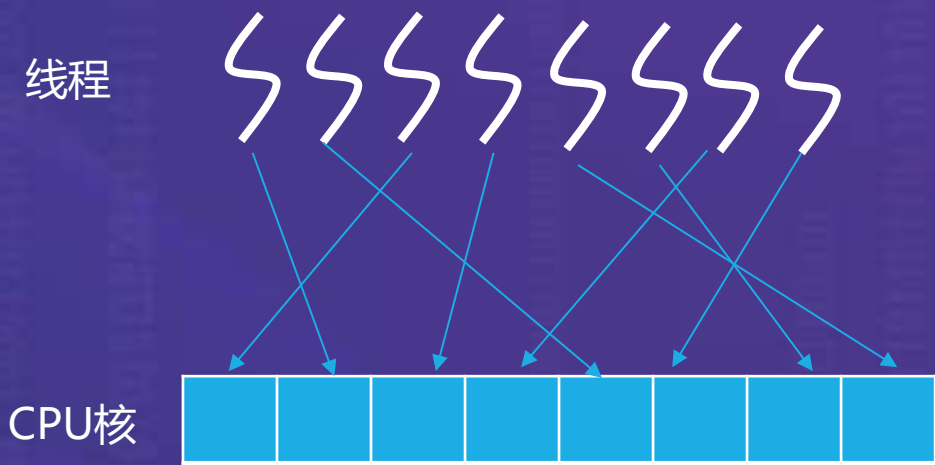


- 1、线程绑核，避免线程在核间偏移。
- 2、NUMA化数据结构改造，减少跨核访问。
- 3、数据分区，减少线程访问冲突。
- 4、算法调整，减少单点瓶颈。
- 5、借助ARM原子指令，减少计算开销。



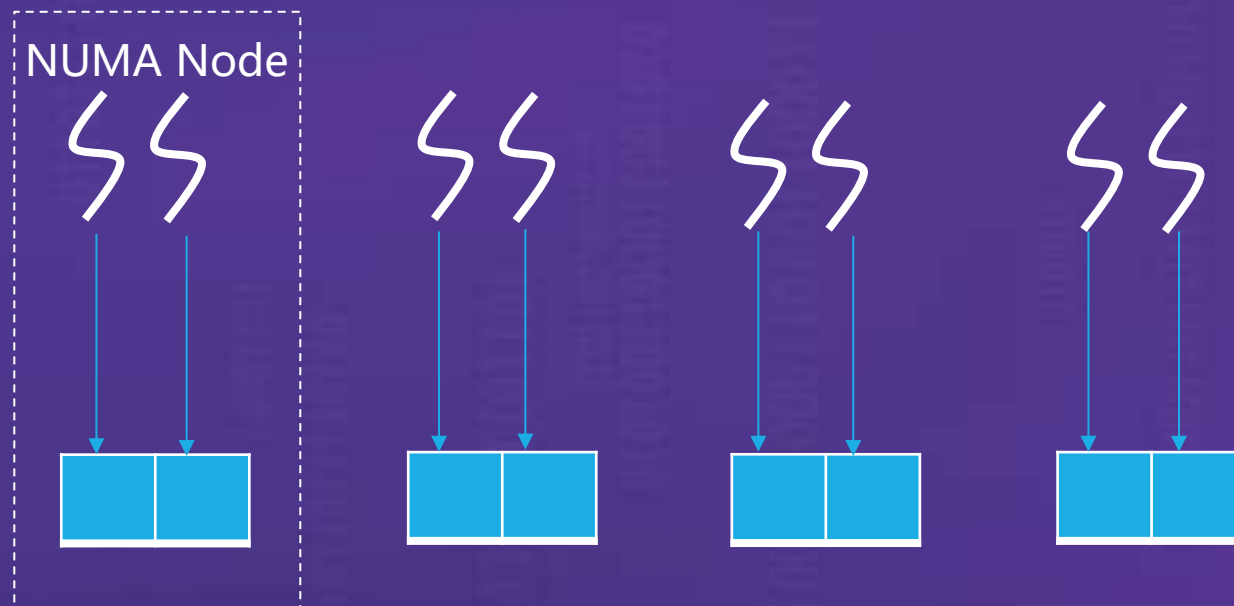
| openGauss NUMA 线程绑核

整改前：线程随机调度



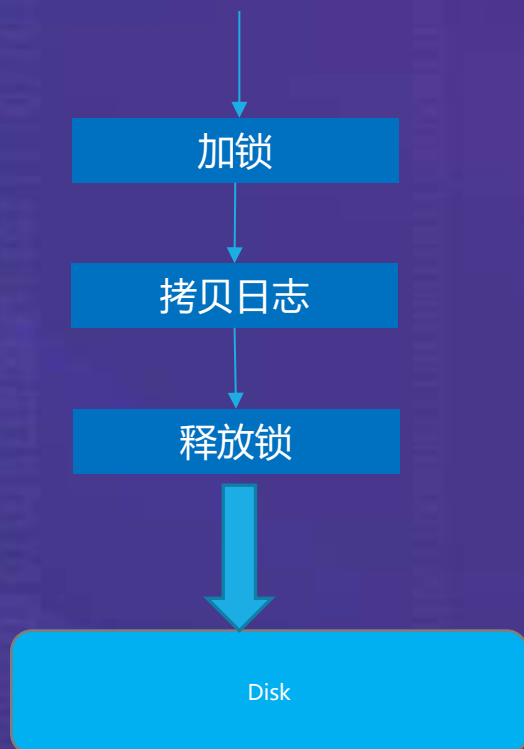
- 1、网络中断和后台业务线程分开绑核。
- 2、不同业务线程分开绑核。

整改后：绑核调度

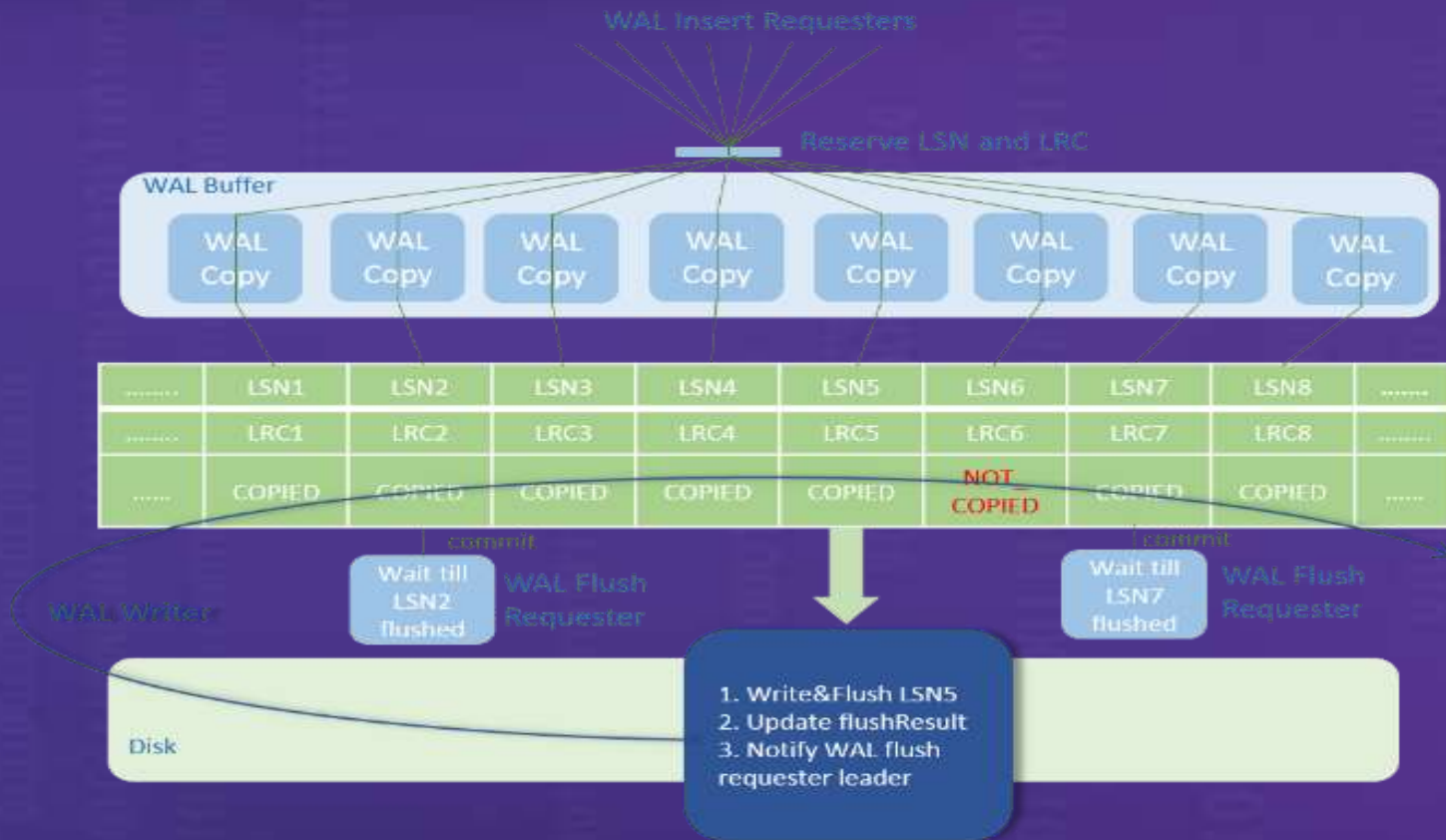


openGauss NUMA 化数据结构

整改前：串行加锁拷贝日志



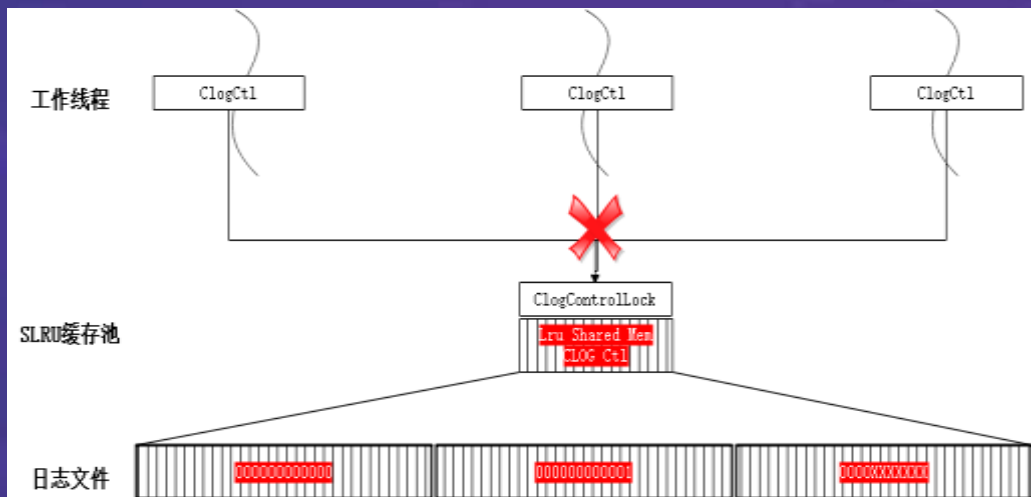
整改后：并行异步拷贝日志



openGauss NUMA数据分区

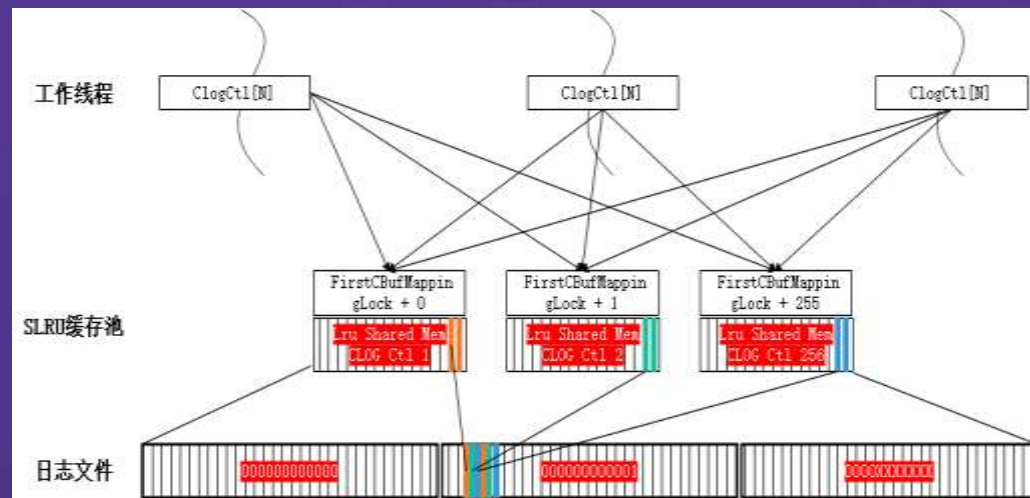
CLOG 日志即事务提交日志，用来记录事务的最终状态，是XLOG日志的辅助，用来加速通过日志判断事务状态的过程，存在四种事务状态，即：IN_PROGRESS、COMMITTED、ABORTED、SUB_COMMITTED，每条日志占2 bit，clog 需要存储在磁盘上，一个页面（8k），按照页面分partition加锁。

整改前：全局唯一的CLOG缓冲池



原方案：CLOG的日志缓冲池在共享内存中且全局唯一，名称为“CLOG Ctl”，各工作线程使用由线程局部变量 ClogCtl 来指向该资源，在高并发的场景下，该资源的竞争成为性能瓶颈

整改后：分区的CLOG缓冲池



整改后：按pageno将日志均分到多个共享内存的缓冲池中，由线程局部对象的数组 ClogCtlData来记录，名称为“CLOG Ctl i”，同步增加共享内存中的缓冲池对象及对应的全局锁。

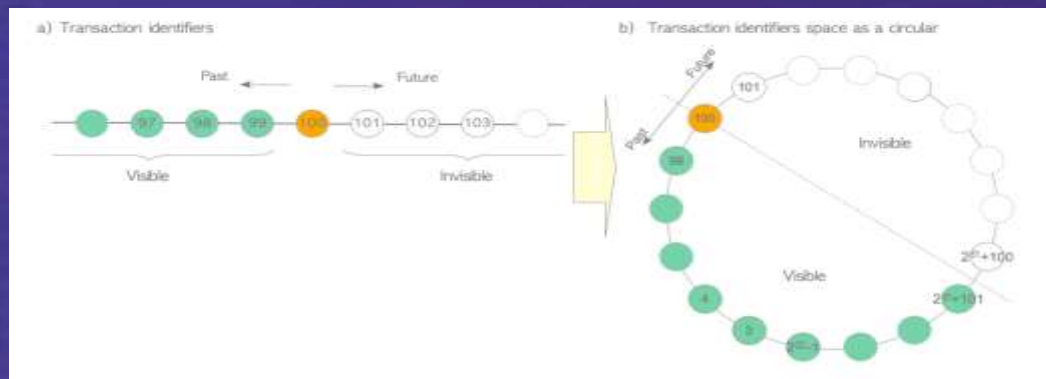


<https://opengauss.org>

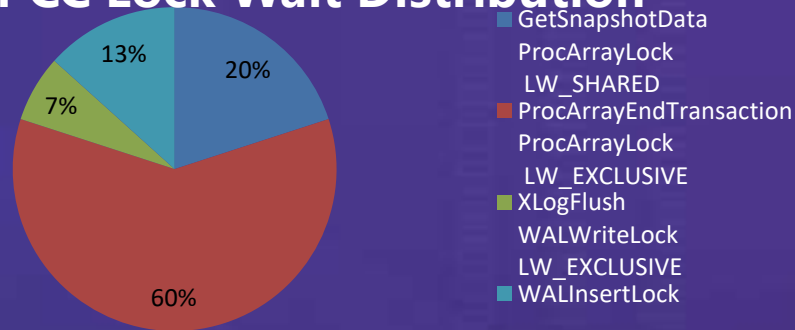


openGauss CSN快照

整改前：活跃事务ID快照

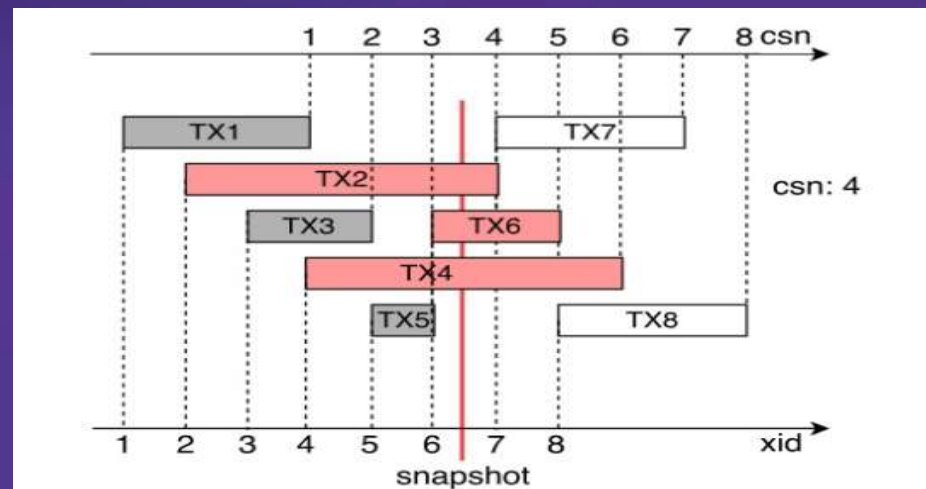


TPCC Lock Wait Distribution



- 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

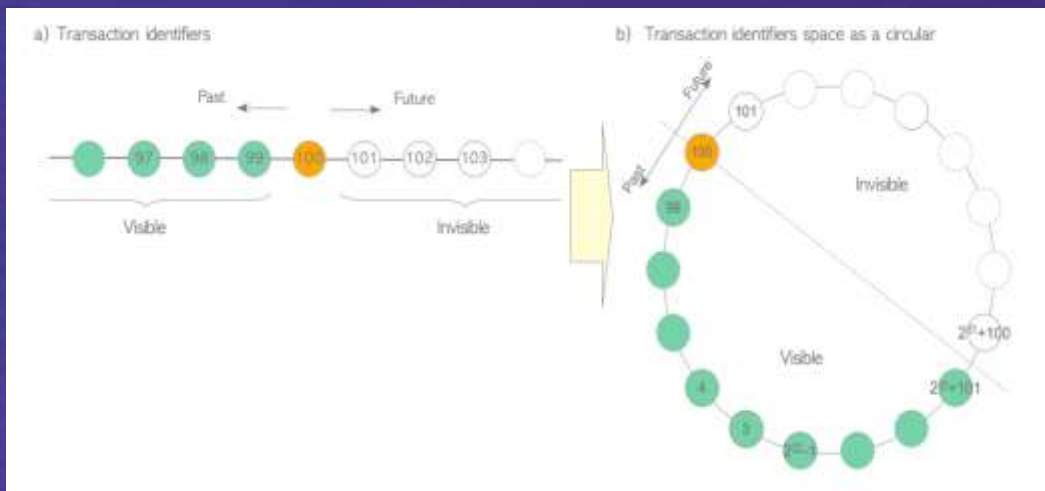


- 1、上图中每个非只读事务在运行过程中会取得一个XID，在事务提交时会推进CSN，同时会将当前CSN与事务的XID映射关系保存起来。
- 2、上图中，棕色竖线表示取snapshot时刻，如果不使用CSN方案，那么棕色竖线对应snapshot的集合应该是{2,4,6}。如果采用CSN方案，会获取当前的CSN值，也就是3，事务TX2、TX4、TX6、TX7、TX8的CSN分别为4、6、5、7、8，对于该snapshot而言，这几个事务的修改都不可见。



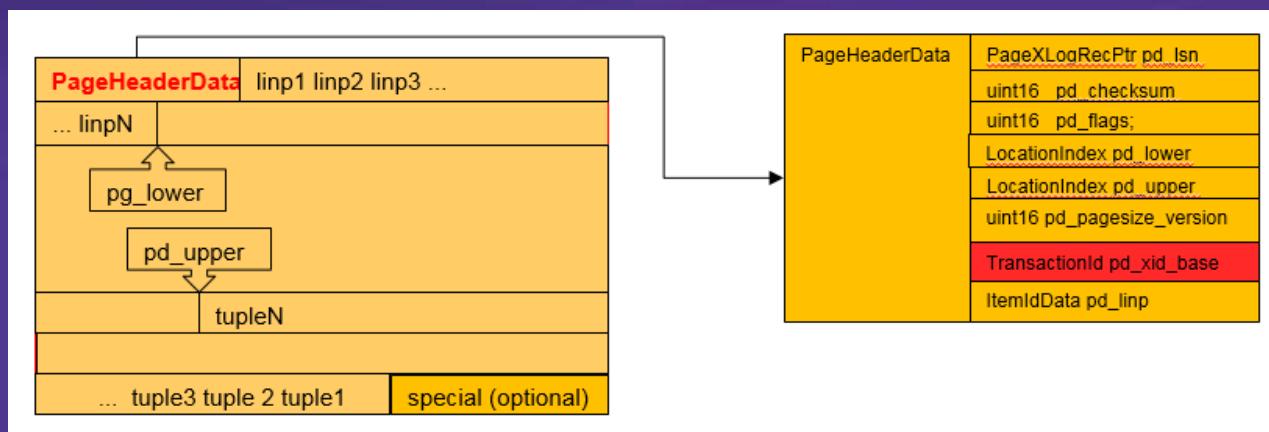
openGauss 64位事务ID

整改前：32位事务ID

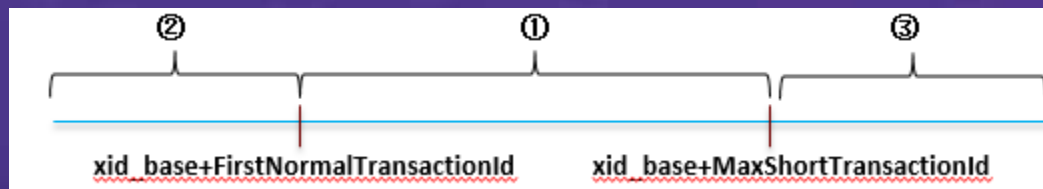


- 1、32位事务ID机制中可以将事务ID理解为一个循环可重用的序列串。对其中的任一普通XID来说，都有20亿个相对它来说过去的事务，和20亿个未来的事务。可以看出同一个数据库中存在的最旧和最新两个事务之间的年龄是最多是 2^{31} 。为了避免这种问题，需要Vacuum freeze定期的对最旧事务进行回收，以推进该值；当事务ID的差值超过 2^{31} 会造成业务中断，强制全库回收事务ID。
- 2、TP场景事务ID增长迅速，导致32bit事务ID会频繁发生回卷
- 3、Vacuum机制来回收过老事务ID，效率低，IO开销大。
- 4、事务ID回收不及时，会导致停库对业务造成严重影响。

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。



<https://opengauss.org>



openGauss NUMA ARM原子指令

CAS (Compare and Swap) , 原子操作, 用于多线程操作下, 确保数据写入的一致性;

在多线程编程中实现不被打断的数据交换操作, 避免同时改写某一数据时由于执行顺序不确定性及中断的不可预知性导致数据不一致问题。

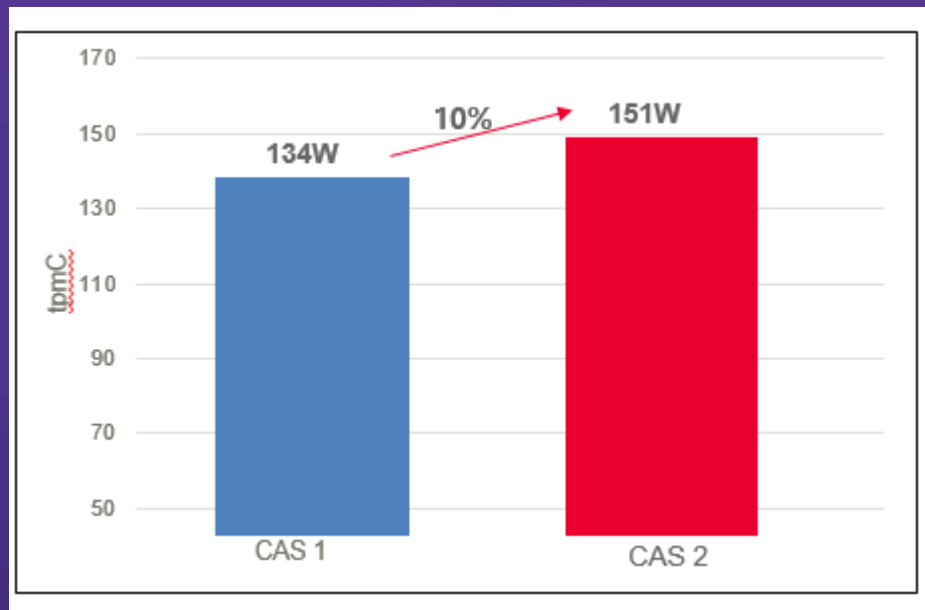
CAS实现方式一：由ldxr和stlxr指令来实现

```
20      os_compare_and_swap_uint(&old,old,new);
0x0000000000004005bc <+24>:    ldr     w0, [x29,#24]
0x0000000000004005c0 <+28>:    mov     w2, w0
0x0000000000004005c4 <+32>:    ldr     w1, [x29,#28]
0x0000000000004005c8 <+36>:    add     x0, x29, #0x18
0x0000000000004005cc <+40>:    ldaxr   w3, [x0]
0x0000000000004005d0 <+44>:    cmp     w3, w2
0x0000000000004005d4 <+48>:    b.ne    0x4005e0 <main+60>
0x0000000000004005d8 <+52>:    stlxr   w4, w1, [x0]
0x0000000000004005dc <+56>:    cbnz    w4, 0x4005cc <main+40>
0x0000000000004005e0 <+60>:    dmb     ish
```

CAS实现方式二：由casal一条指令来实现

```
4005b0:      b9001ba0      stl     w0, [x29,#24]
// int *ptr=100;
os_compare_and_swap_uint(&old,old,new);
4005bc:      b9401ba0      ldr     w0, [x29,#24]
4005c0:      2a0003e2      mov     w2, w0
4005c4:      b9401fa1      ldr     w1, [x29,#28]
4005c8:      910063a0      add     x0, x29, #0x18
4005cc:      2a0203e3      mov     w3, w2
4005d0:      88e3fc01      casal   w3, w1, [x0]
4005d4:      6b02007f      cmp     w3, w2
```

四个指令→1个指令, 提升执行效率;



<https://opengauss.org>



openGauss NUMA 多核优化结果

```
3.192.168.34.111 (root) 5.192.168.34.111 (root) 7.192.168.34.121 (root) 6.192.168.34.121 (root) 4.192.168.34.121 (root)
1 195.5 33 192.5 55 194.2 97 196.1
2 194.2 34 193.5 56 194.9 98 194.9
3 194.9 35 193.4 57 194.3 99 194.9
4 194.2 36 193.4 58 194.2 100 194.8
5 194.9 37 193.5 59 194.2 101 194.1
6 194.2 38 193.5 60 194.2 102 194.2
7 194.0 39 193.5 61 194.9 103 195.0
8 194.8 40 193.5 62 194.9 104 195.1
9 194.2 41 193.5 63 194.2 105 194.9
10 194.2 42 193.5 64 194.9 106 195.5
11 194.2 43 194.2 65 194.2 107 195.5
12 194.2 44 193.5 66 194.2 108 196.2
13 194.2 45 194.1 67 194.8 109 194.2
14 194.2 46 193.5 68 194.8 110 194.2
15 194.2 47 193.6 69 194.2 111 195.4
16 194.2 48 193.5 70 194.9 112 195.3
17 194.8 49 193.6 71 194.1 113 196.1
18 194.8 50 194.1 72 194.9 114 196.1
19 194.8 51 193.4 73 194.2 115 195.5
20 194.3 52 193.7 74 194.2 116 195.3
21 194.2 53 194.2 75 194.2 117 195.3
22 194.2 54 194.2 76 194.9 118 195.4
23 194.2 55 194.2 77 194.8 119 196.2
24 194.2 56 194.2 78 194.9 120 195.3
25 194.9 57 193.5 79 194.2 121 195.5
26 194.2 58 193.5 80 194.9 122 196.3
27 194.2 59 193.5 81 194.2 123 194.2
28 194.8 60 194.2 82 194.2 124 195.5
29 194.2 61 194.2 83 194.2 125 195.5
30 194.2 62 194.2 84 194.2 126 195.5
31 194.2 63 194.2 85 194.2 127 195.5
32 194.2 64 194.2 86 194.2 128 195.5
Mem: 16 Tasks: 51, 877 Vm: 120 running
Swap: Load average: 357.59 321.83 203.95
Uptime: 1 day, 02:16:36
PID USER PRU NI KIBI RSS SARA S CPU% MEM% TRES Command
28112 oom 20 0 8440 1536 1390 S 10437 36.2 23h23:47 /data/huawei/install/app/bin/guassdb --single_node -D /data/huawei/install/data/dn
29873 20 0 8440 1536 1390 R 54.9 36.2 7:57.30 /data/huawei/install/app/bin/guassdb --single_node -D /data/huawei/install/data/dn
29874 20 0 8440 1536 1390 S 25.2 36.2 3:28.32 /data/huawei/install/app/bin/guassdb --single_node -D /data/huawei/install/data/dn
28970 20 0 8440 1536 1390 S 13.6 36.2 1:42.57 /data/huawei/install/app/bin/guassdb --single_node -D /data/huawei/install/data/dn
28827 20 0 8440 1536 1390 S 13.6 36.2 1:41.40 /data/huawei/install/app/bin/guassdb --single_node -D /data/huawei/install/data/dn
28905 20 0 8440 1536 1390 S 13.6 36.2 1:43.88 /data/huawei/install/app/bin/guassdb --single_node -D /data/huawei/install/data/dn
28740 20 0 8440 1536 1390 S 12.6 36.2 1:42.14 /data/huawei/install/app/bin/guassdb --single_node -D /data/huawei/install/data/dn
28770 20 0 8440 1536 1390 R 11.8 36.2 1:42.36 /data/huawei/install/app/bin/guassdb --single_node -D /data/huawei/install/data/dn
28534 20 0 8440 1536 1390 S 11.6 36.2 1:41.34 /data/huawei/install/app/bin/guassdb --single_node -D /data/huawei/install/data/dn
28769 20 0 8440 1536 1390 S 13.6 36.2 1:42.72 /data/huawei/install/app/bin/guassdb --single_node -D /data/huawei/install/data/dn
29254 20 0 8440 1536 1390 S 21.1 36.2 1:07.12 /data/huawei/install/app/bin/guassdb --single_node -D /data/huawei/install/data/dn
28853 20 0 8440 1536 1390 S 12.3 36.2 1:43.17 /data/huawei/install/app/bin/guassdb --single_node -D /data/huawei/install/data/dn
28949 20 0 8440 1536 1390 S 13.6 36.2 1:43.40 /data/huawei/install/app/bin/guassdb --single_node -D /data/huawei/install/data/dn
28859 20 0 8440 1536 1390 R 11.6 36.2 1:42.59 /data/huawei/install/app/bin/guassdb --single_node -D /data/huawei/install/data/dn
28940 20 0 8440 1536 1390 S 12.9 36.2 1:41.38 /data/huawei/install/app/bin/guassdb --single_node -D /data/huawei/install/data/dn
28742 20 0 8440 1536 1390 R 11.6 36.2 1:41.85 /data/huawei/install/app/bin/guassdb --single_node -D /data/huawei/install/data/dn
28780 20 0 8440 1536 1390 R 12.3 36.2 1:42.31 /data/huawei/install/app/bin/guassdb --single_node -D /data/huawei/install/data/dn
28858 20 0 8440 1536 1390 R 12.3 36.2 1:42.56 /data/huawei/install/app/bin/guassdb --single_node -D /data/huawei/install/data/dn
File Edit Setup Search Filter Tree Sort By Filter Nice + Kill Quit
```

**基于鲲鹏920，TPCC测试单机性能
达到150W tpmC。
CPU运行效率接近95%。**

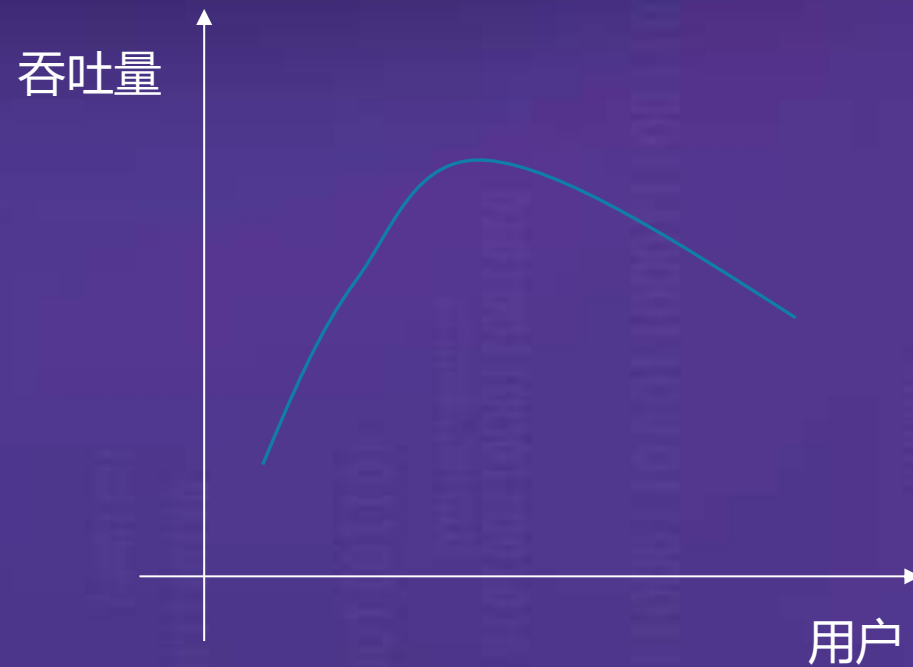
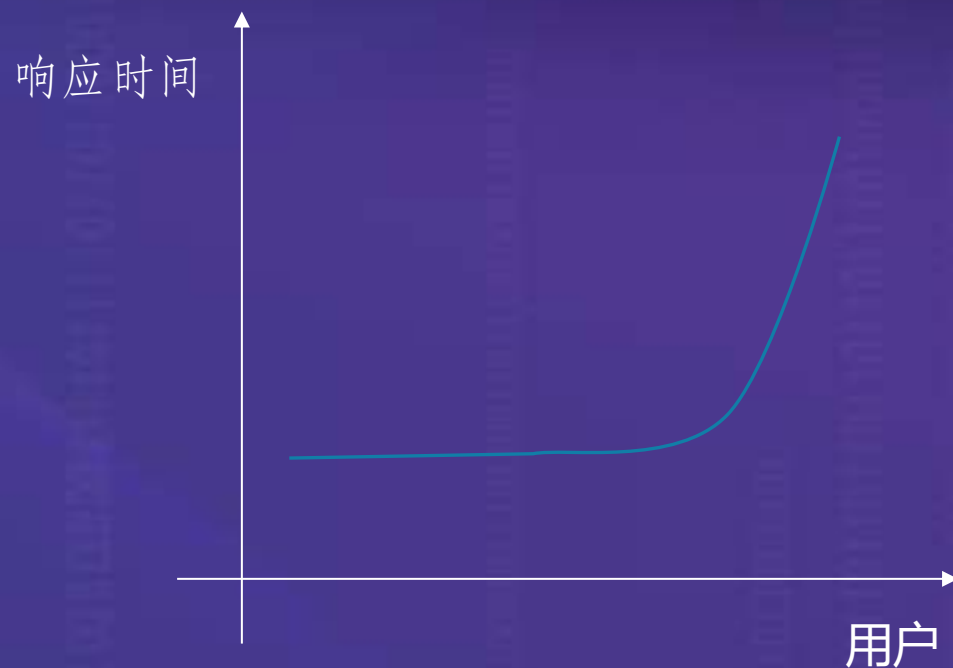
服务器	项目	型号	数量
Taishan 2280 V2	CPU	Kunpeng 920/64cores/64threads	2(每个cpu包含64cores,共计128cores)
	主频	2.6Ghz	---
	内存	DDR4,2933MT/s	512G
	网络	Hi1822	
	硬盘	NVME;3.2T	4
	OS	openEuler-20.03-LTS	1



<https://opengauss.org>



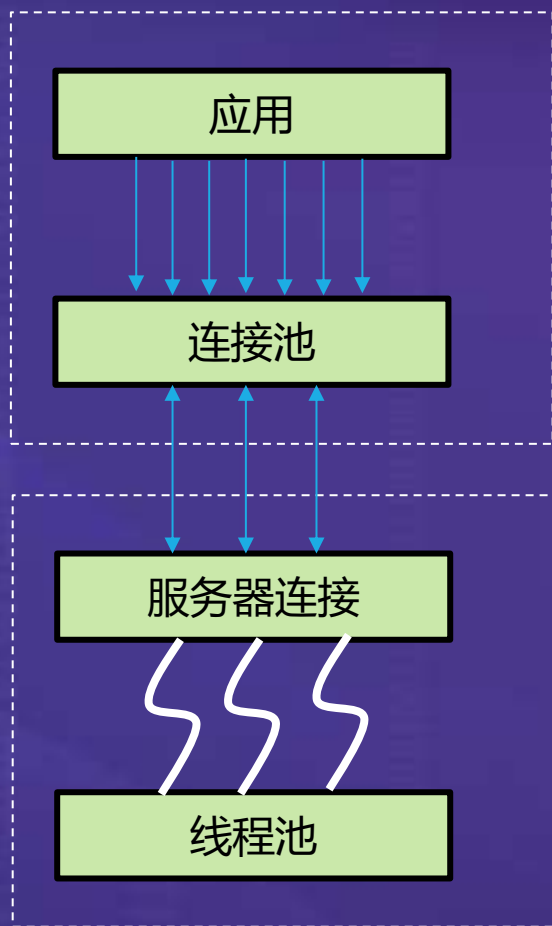
| openGauss 大并发业务场景问题



- 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。



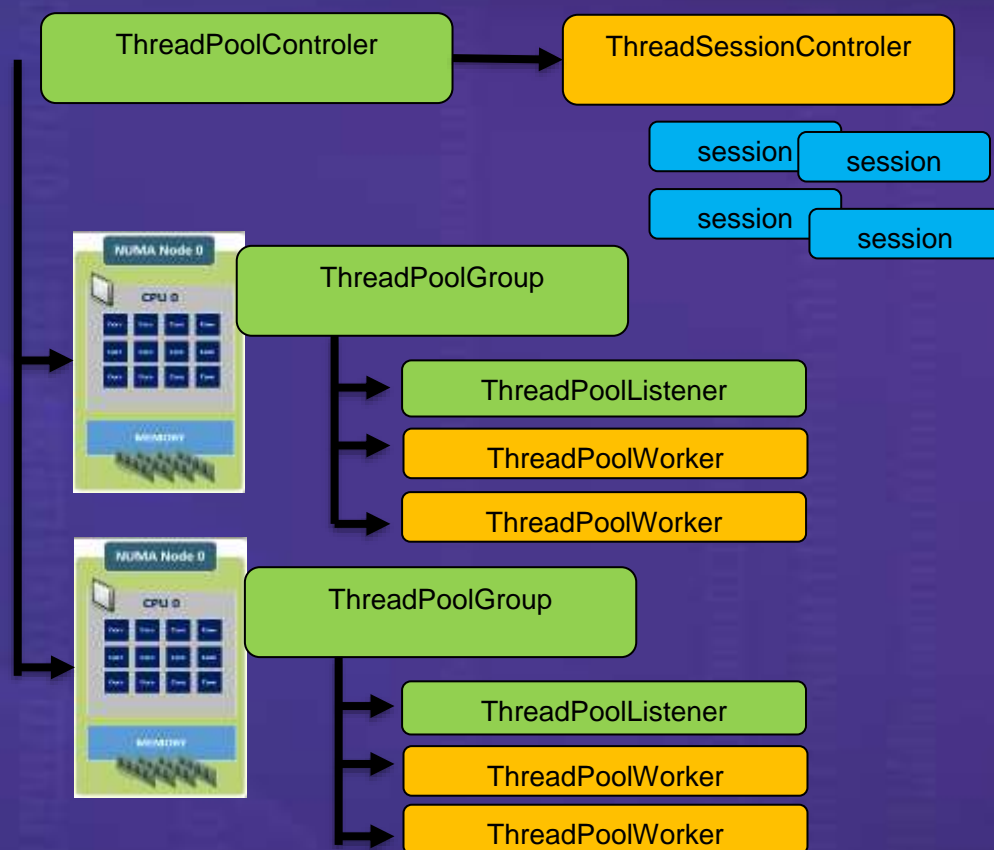
| openGauss 大并发问题解决方案



- 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。



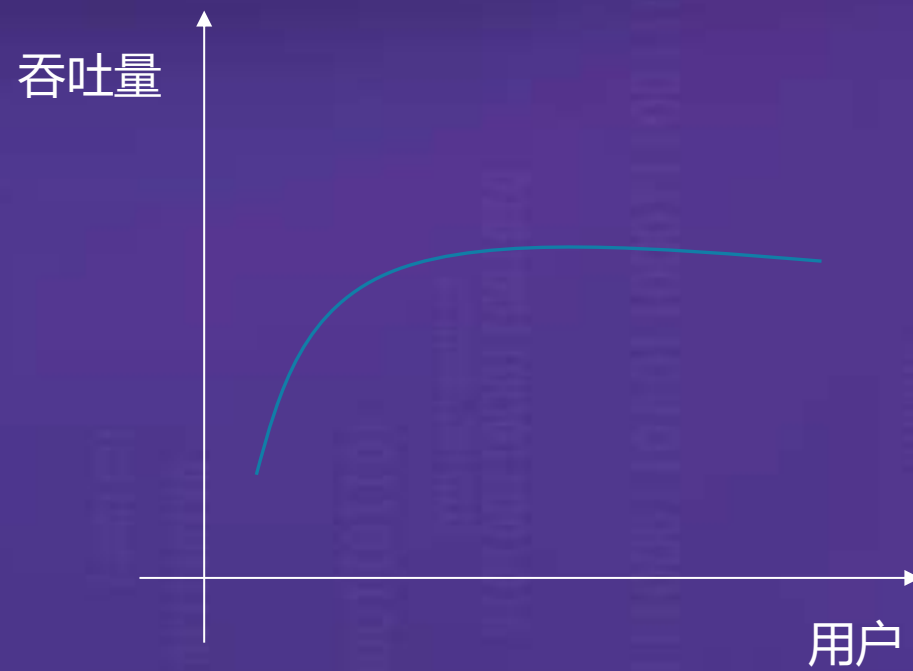
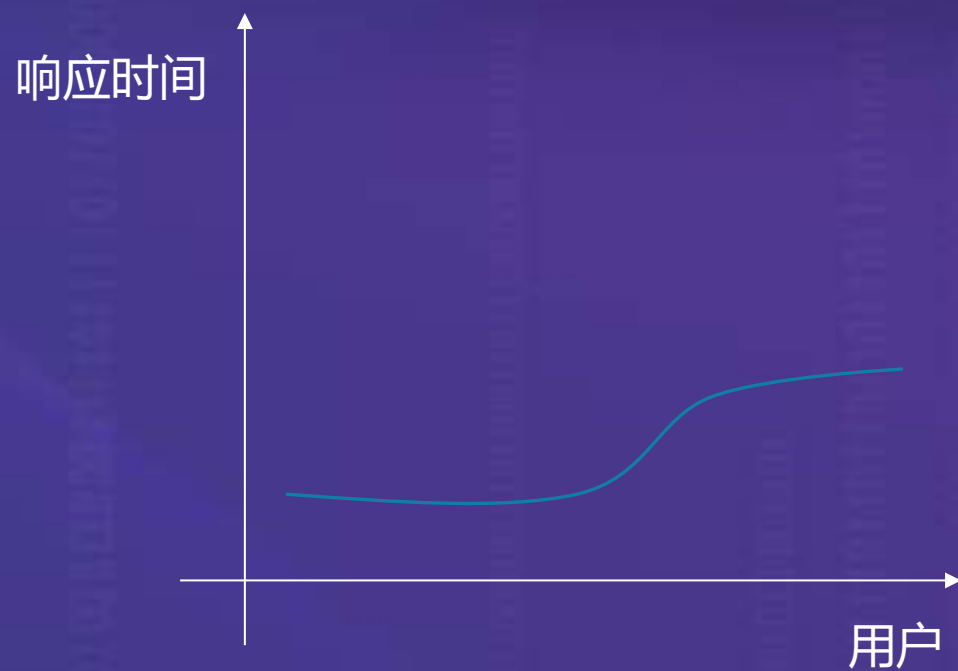
openGauss 线程池实现原理



- 1) 客户端向数据库发起连接请求，PostMaster线程接收到连接请求并被唤醒。PostMaster线程创建该连接对应的socket，以及创建session数据结构。遍历当前所有的Thread Group，找到当前活跃session数量最少的Thread Group，并把最新的session分发给该Thread Group，加入到该Thread Group 的epoll列表之中。
- 2) Thread Group的listener线程负责监听epoll列表中所有的客户连接。
- 3) 客户端发起任务请求，listener线程被唤醒。Listener线程检查当前的Thread Group 是否有空闲worker线程，如果有，则把当前session分配给该worker线程，并唤醒该worker线程；如果没有，则把该session放在等待队列之中。
- 4) worker线程被唤醒后，读取客户端连接上的请求，执行相应请求，并返还请求结果。在一次事务结束（提交、回滚），或者事务超时退出的时候，worker线程的一次任务完成。worker线程将session返回给listener线程，listener线程继续等待该session的下一次请求。worker线程返还session后，检查session等待队列，如果存在等待响应请求的session，则直接从该队列中取出新的session并继续工作；如果没有等待响应的session，则将自身标记为free状态，等待listener线程唤醒。
- 5) 客户端断开连接时，listener线程被唤醒，关闭连接，同时清理session相关结构，释放内存和fd等资源。



| openGauss 线程池下大并发性能结果



1、高并发下保持性能的稳定。

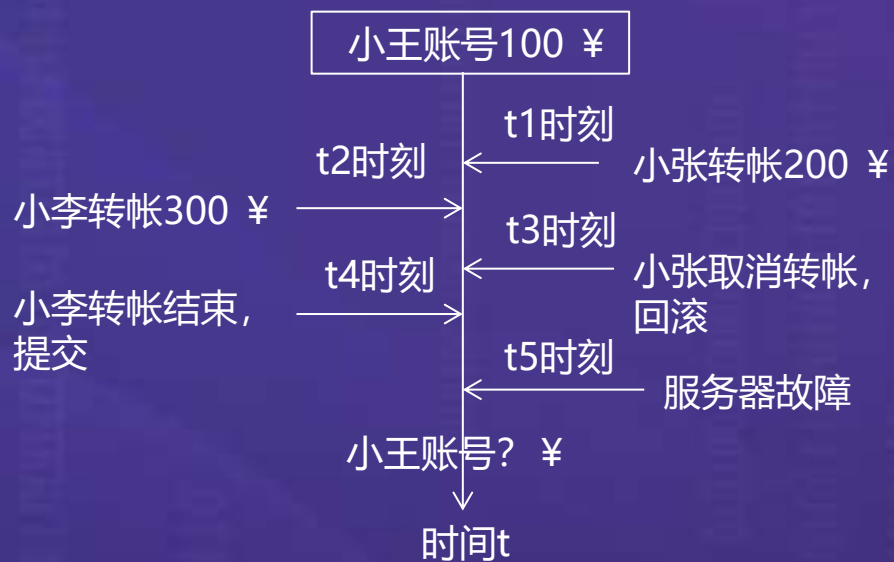


openGauss 数据库事务

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。

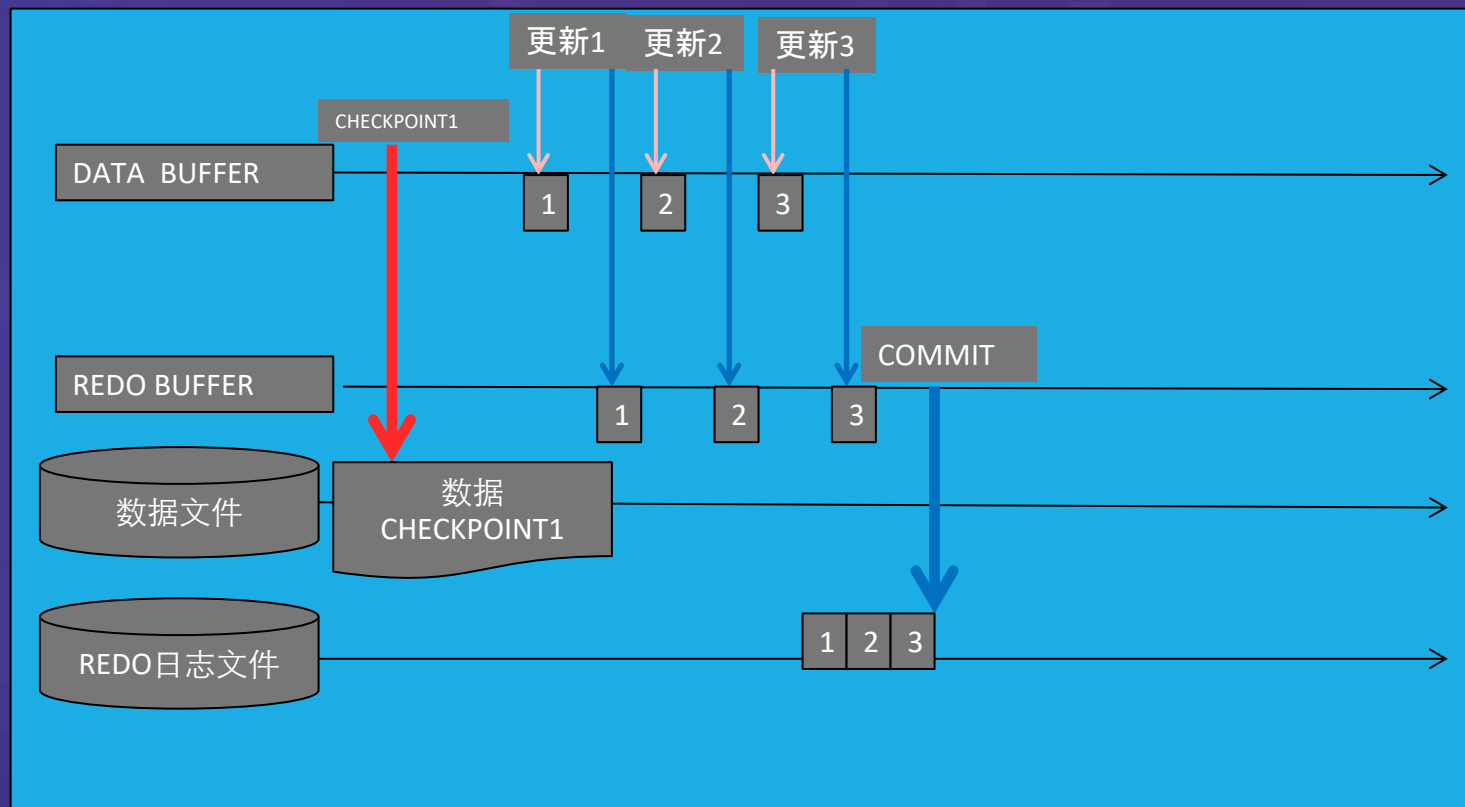
■ 范例：银行转账（小张，小李都给小王转账）



- 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, 小李的转账仍然是成功的。
- 小王的账号始终为400 ¥。



| openGauss 数据库事务机制



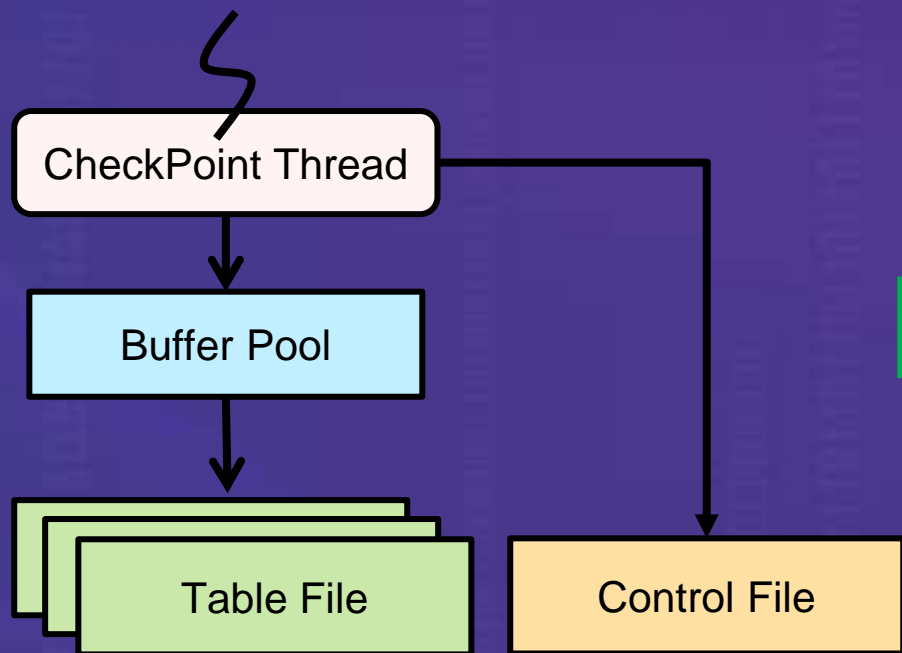
1、事务的持久性 (Durability) 主要通过预写式日志WAL算法实现。在事务提交时，采用预写式日志方式，把REDO日志写到磁盘。

2、检查点：将脏缓冲队列上的全部数据写出到数据文件。

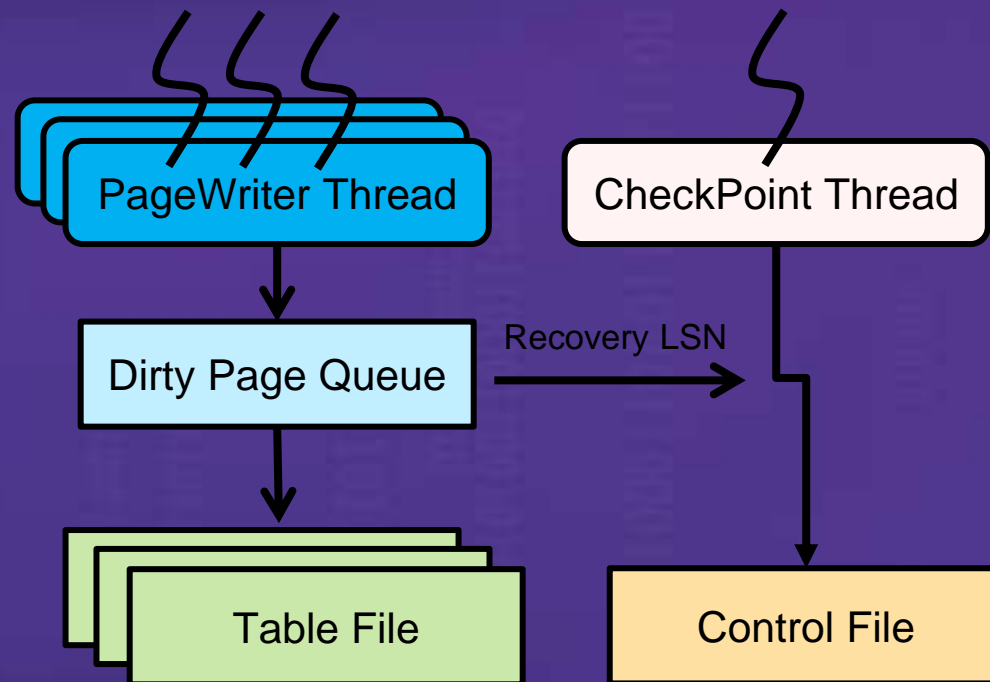


| openGauss 增量检查点技术原理

全量检查点



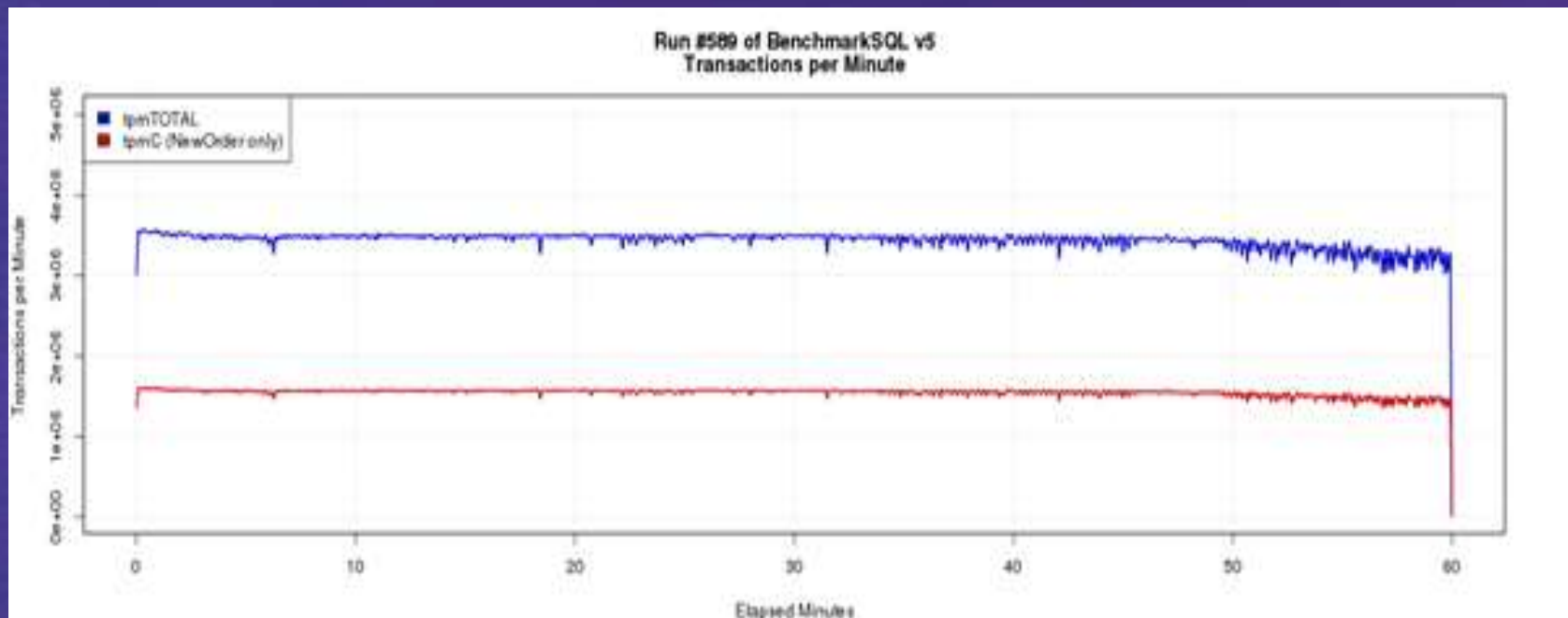
增量检查点



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。



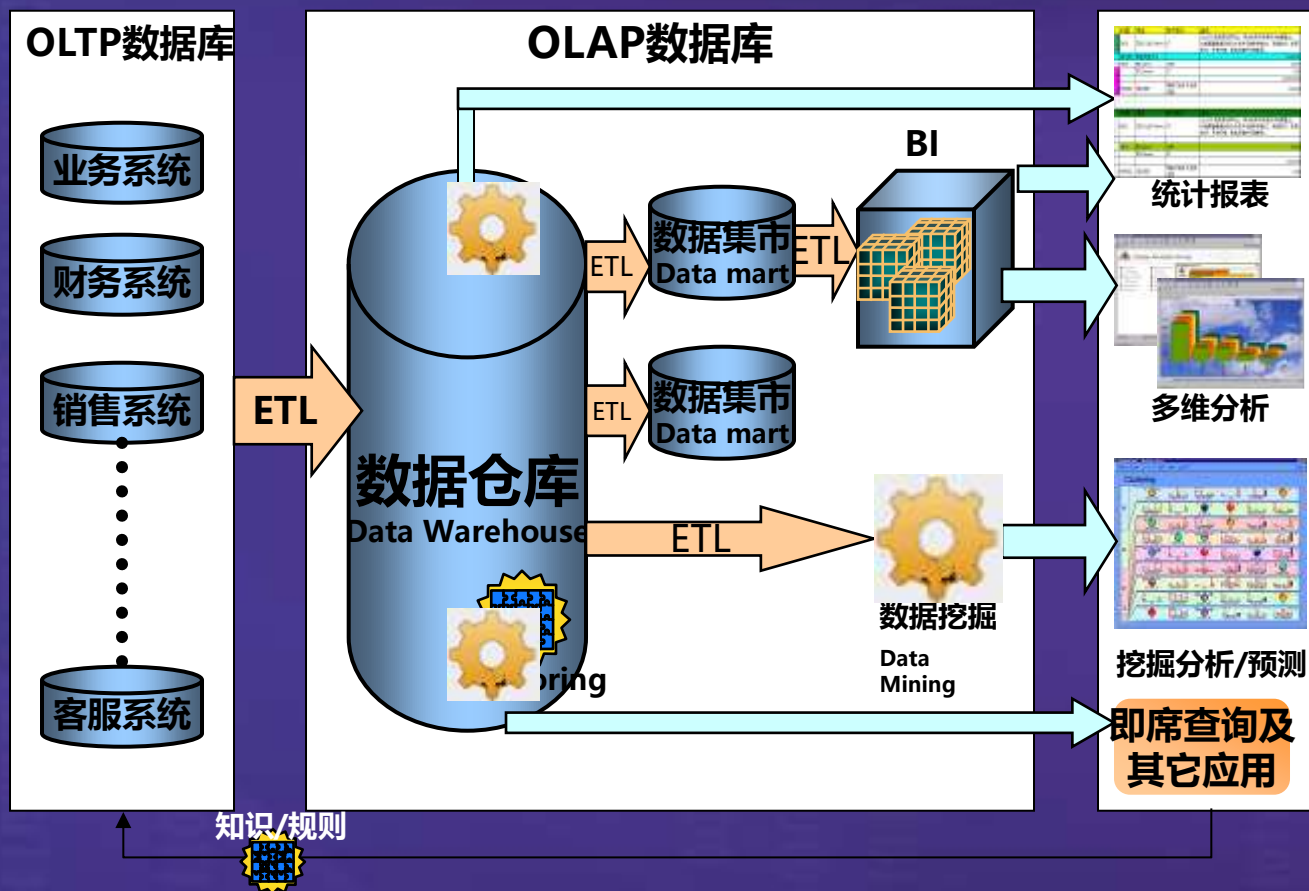
| openGauss 增量检查点IO平稳



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 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 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

行存表

1	1	2001-01-01
2	1	2002-02-01
3	1	2002-05-01

列存表

1	2	3
1	1	1
2001-01-01	2002-02-01	2002-05-01

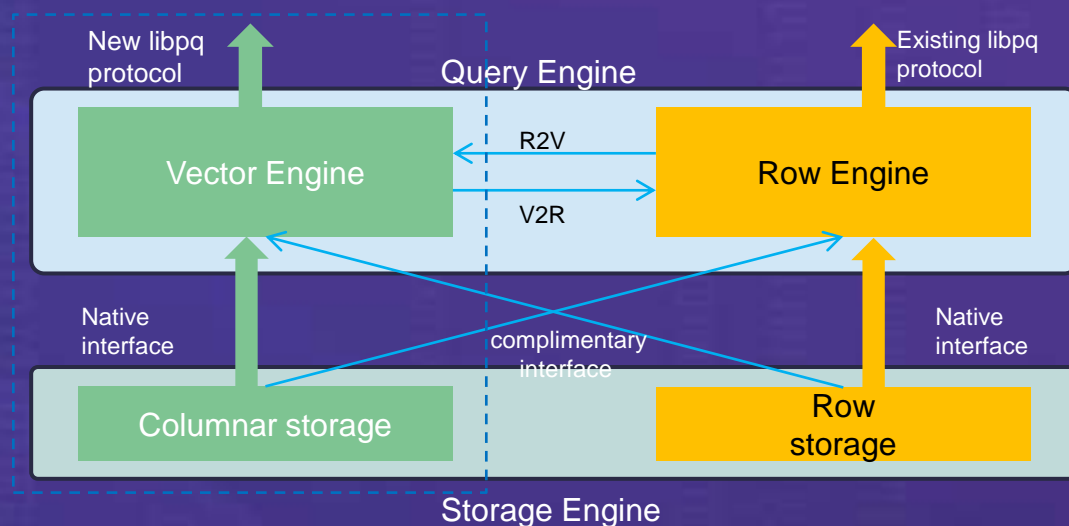
行存 (Row Store)	列存 (Column Store)
容易修改记录	修改记录代价高, 写一条记录时需要访问多次IO
读取数据代价高, 可能读入不需要的列	可以只读取相关的数据
不同记录重复率低, 压缩率低	同一列重复值高, 压缩率高



openGauss 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。



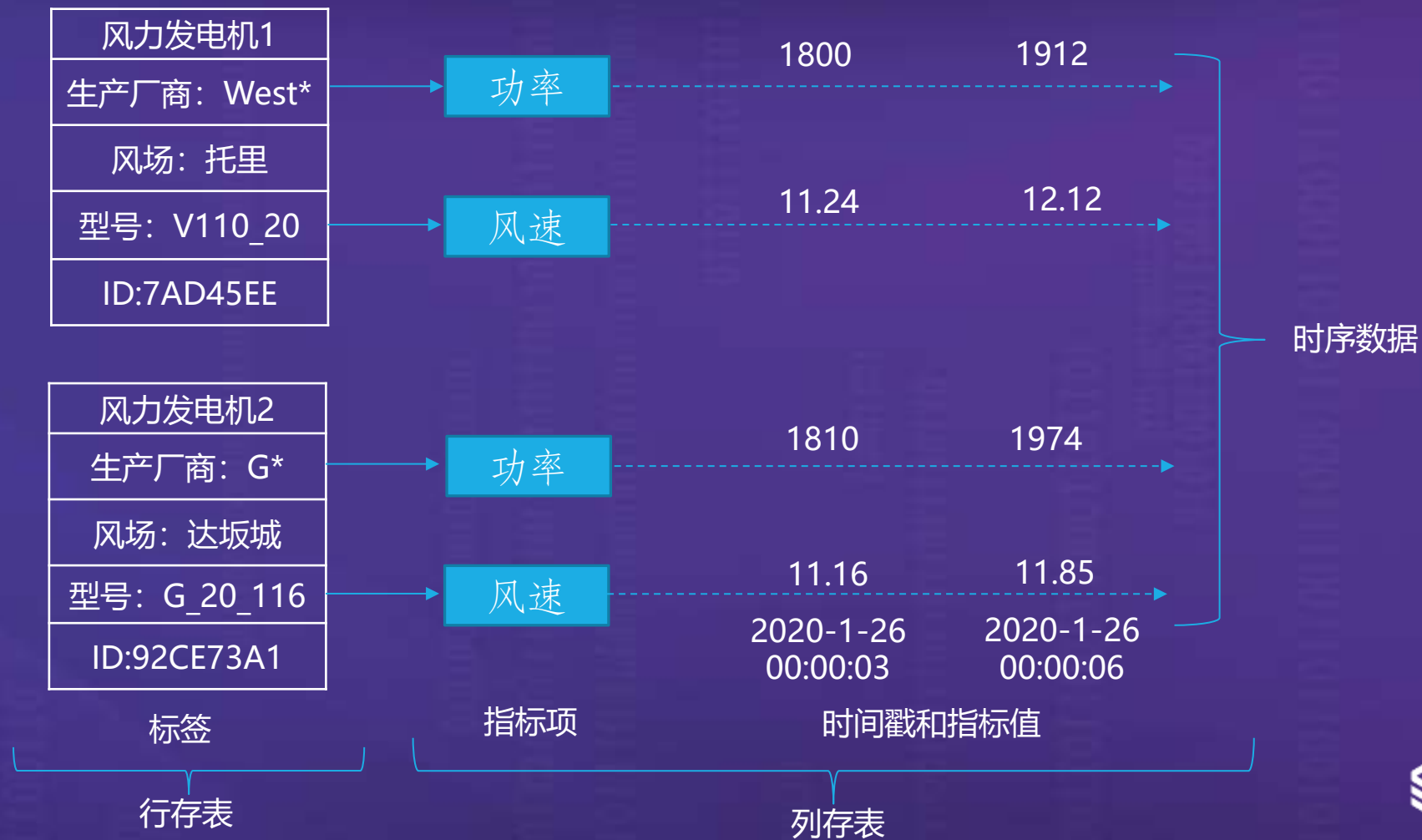
场景	行存	列存	优选
点查询	B+树索引，直接定位到行（页）	粗粒度索引，定位到CU	行存
数据更新	<ul style="list-style-type: none">支持行级别锁支持CU级别并发更新	支持CU级别锁，支持CU级别并发更新	行存
统计分析	Pipeline执行	天然和向量化引擎对接，降低CPU Cache Miss和指令Miss，效率成倍提升	列存
批量加载	并行批量加载	压缩率高，IO量更小	列存



<https://opengauss.org>



openGauss 行列混合IoT场景



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。



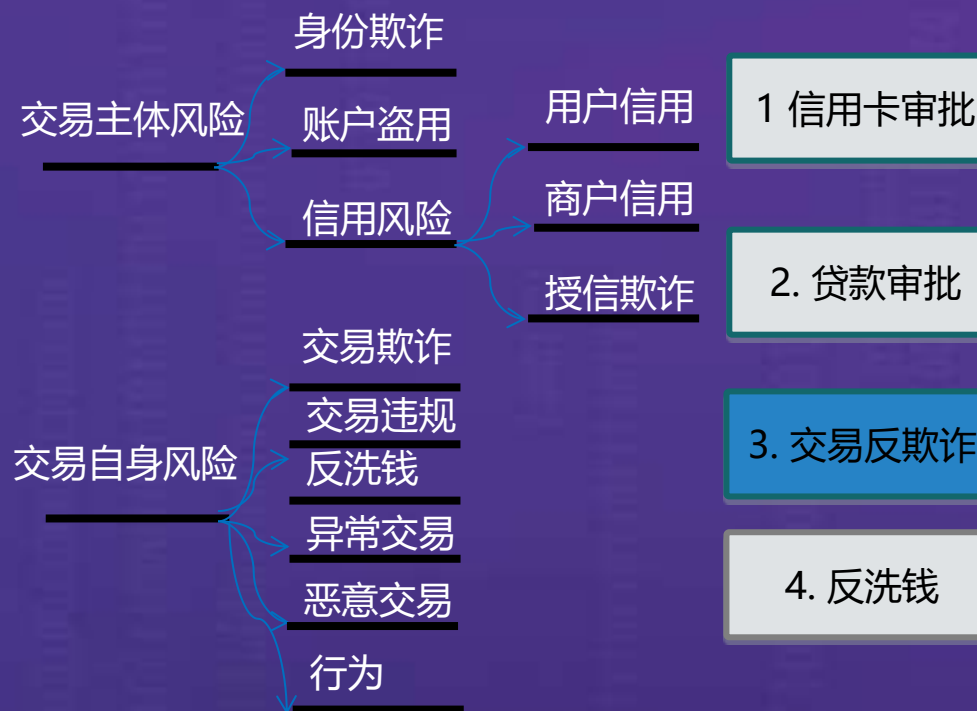
移动支付占比70%，
网络伪冒成上升趋势



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



风控
需求

**多业务、多
维度、多渠
道、事中风
控**

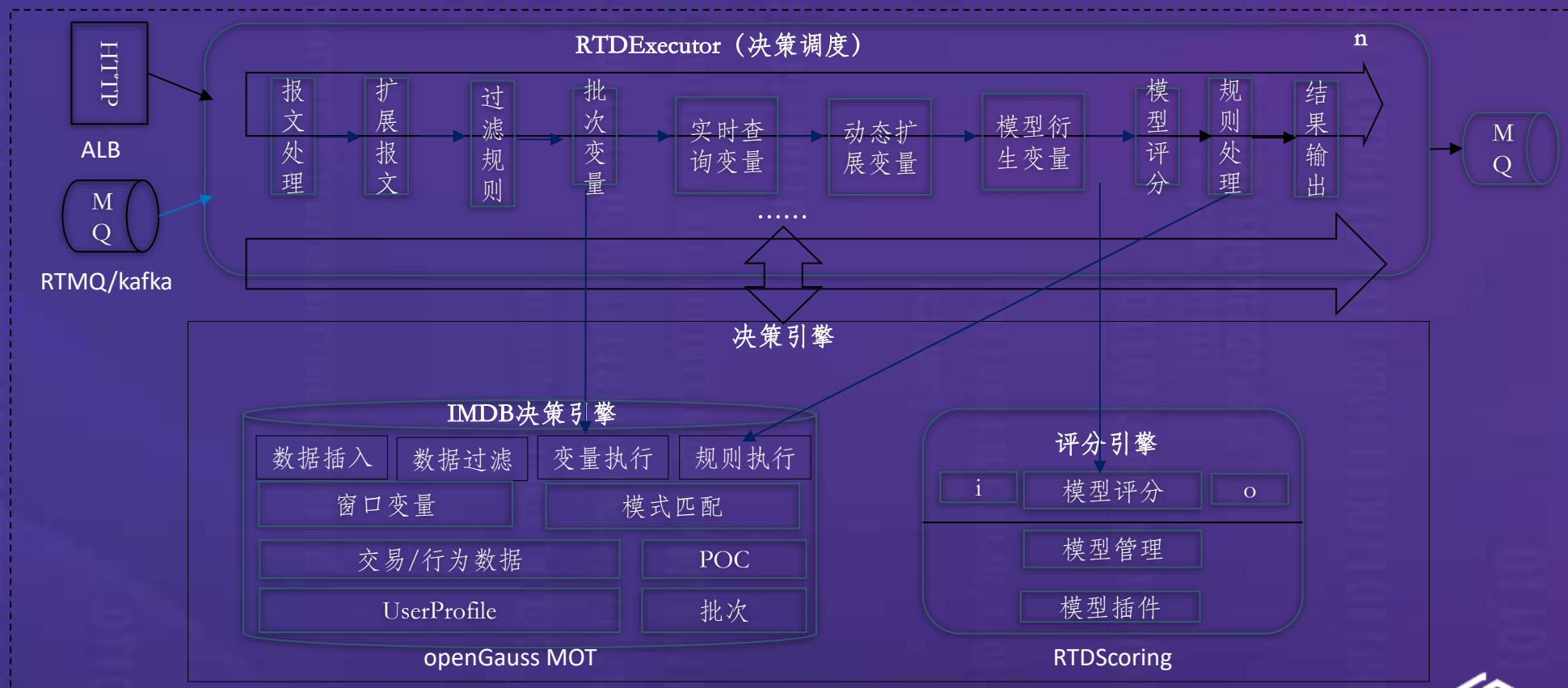


<https://opengauss.org>



openGauss MOT 风控实时处理

“规则+模型”的实时决策



openGauss 磁盘引擎问题

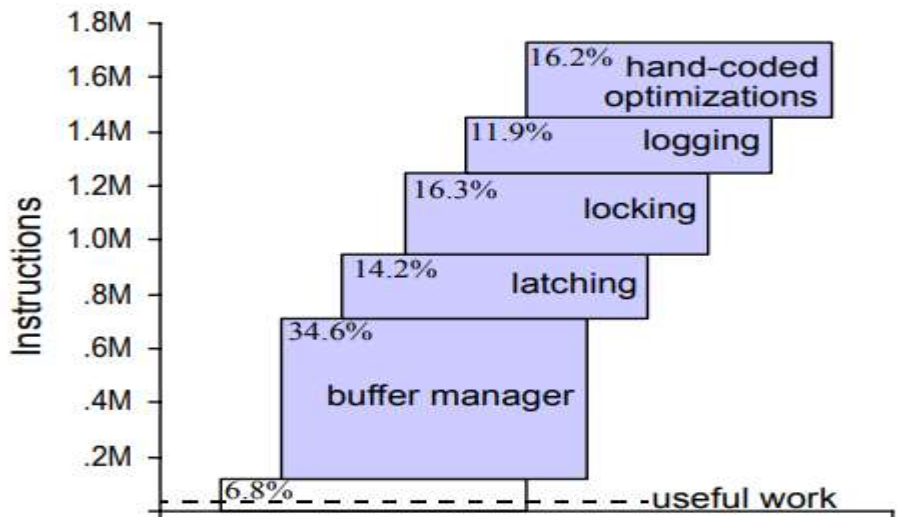


Figure 1. Breakdown of instruction count for various DBMS components for the New Order transaction from TPC-C. The top of the bar-graph is the original Shore performance with a main memory resident database and no thread contention. The bottom dashed line is the useful work, measured by executing the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

1、图灵奖获得者Micheal Stonebraker：磁盘架构70%左右指令在等待锁/闕，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/闕等待，事务处理延迟低，满足实时业务；

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

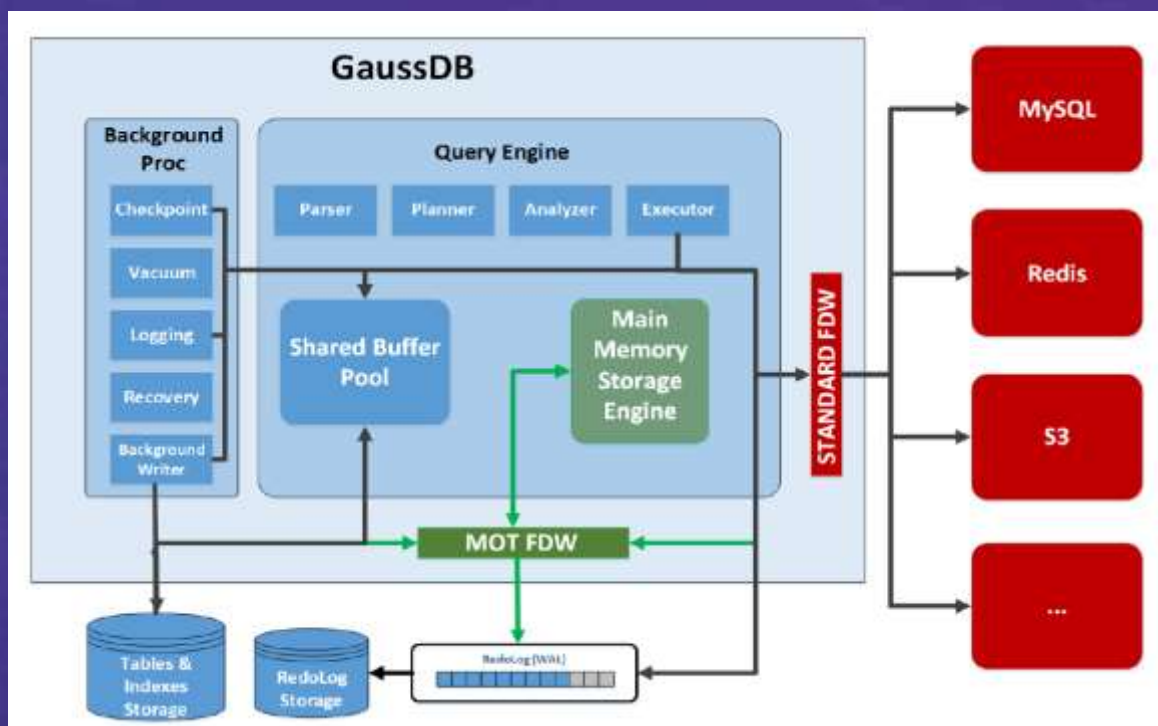


<https://opengauss.org>



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

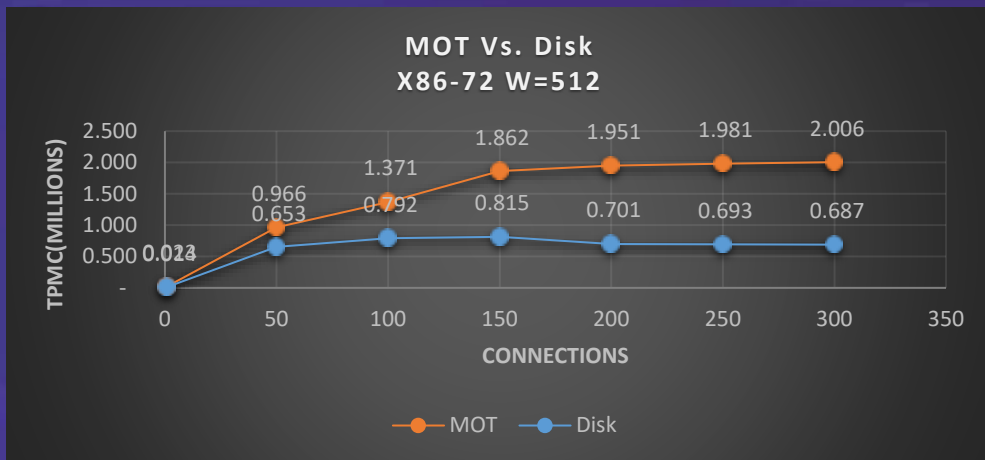


<https://opengauss.org>

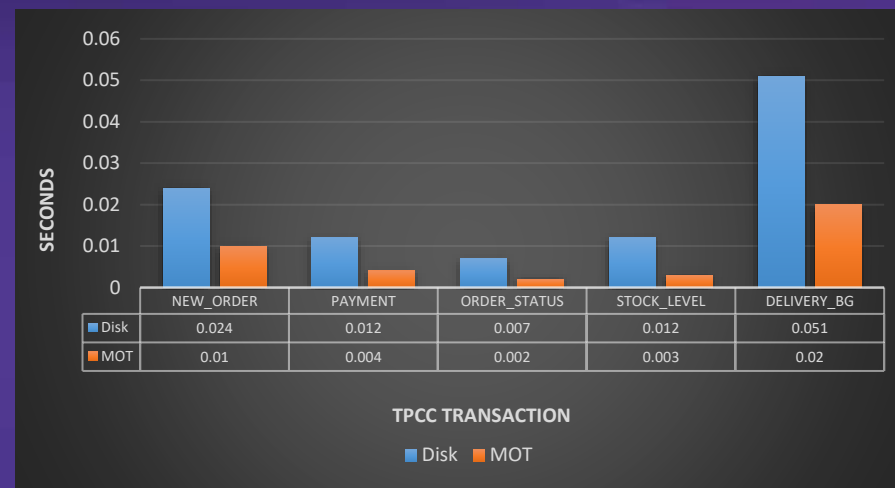


openGauss MOT性能

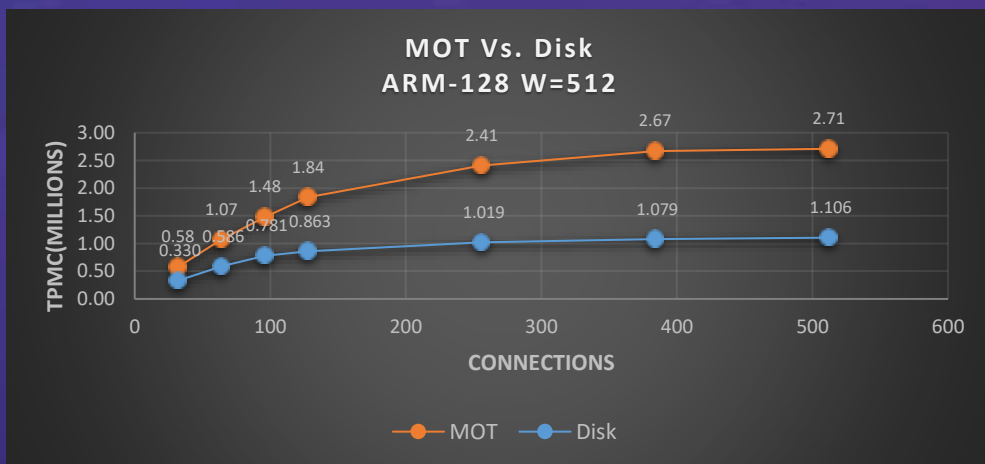
TPCC Throughput on X86-72 vCores



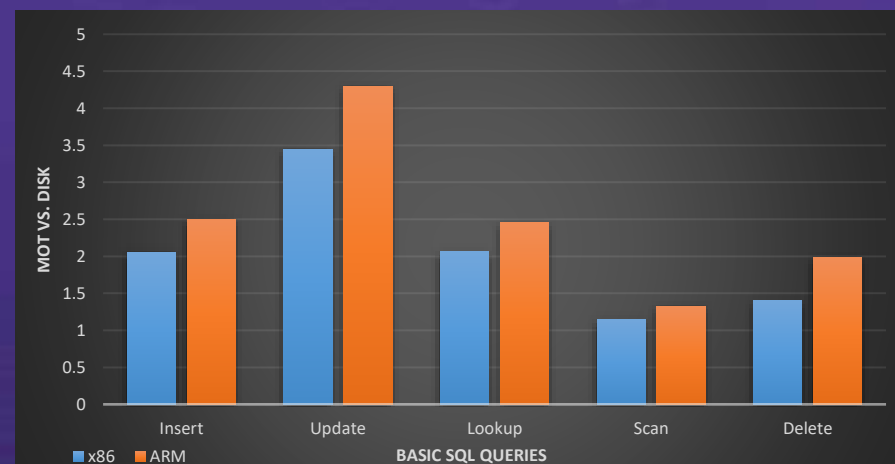
TPCC Latency



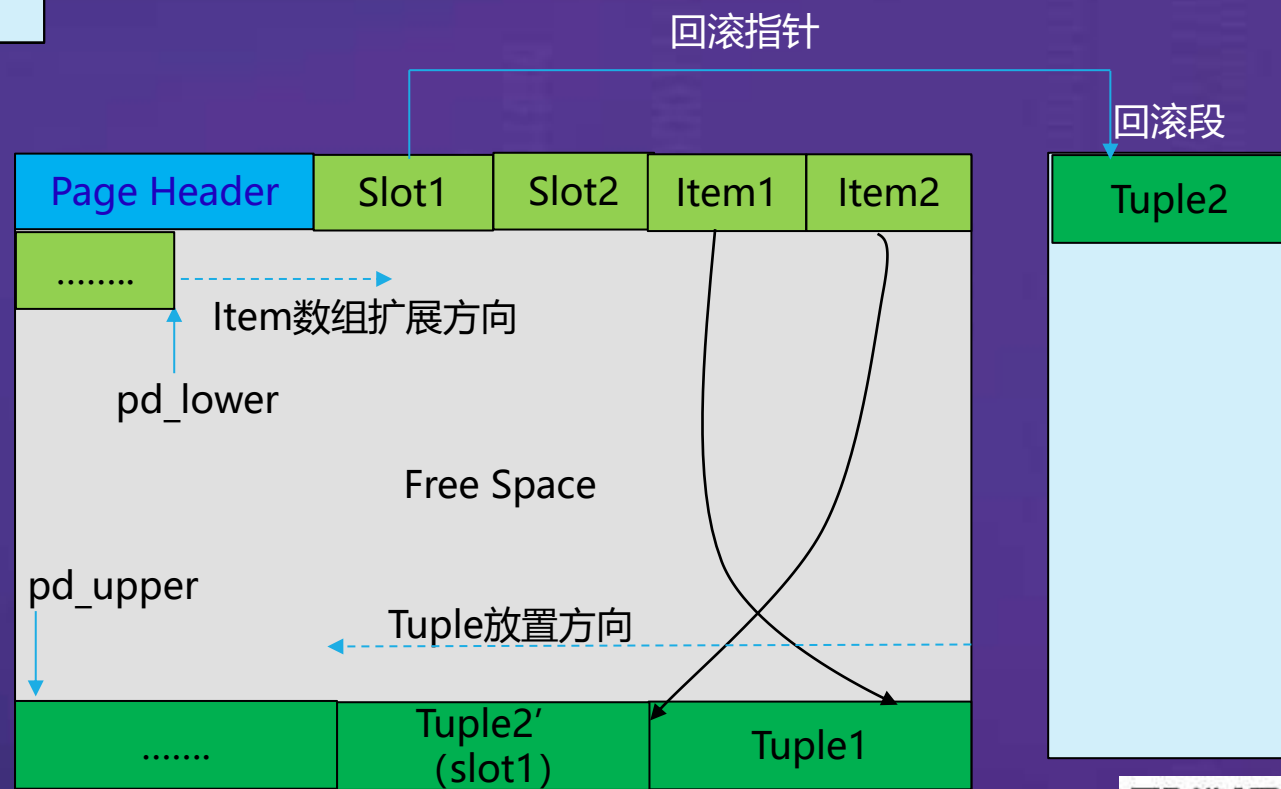
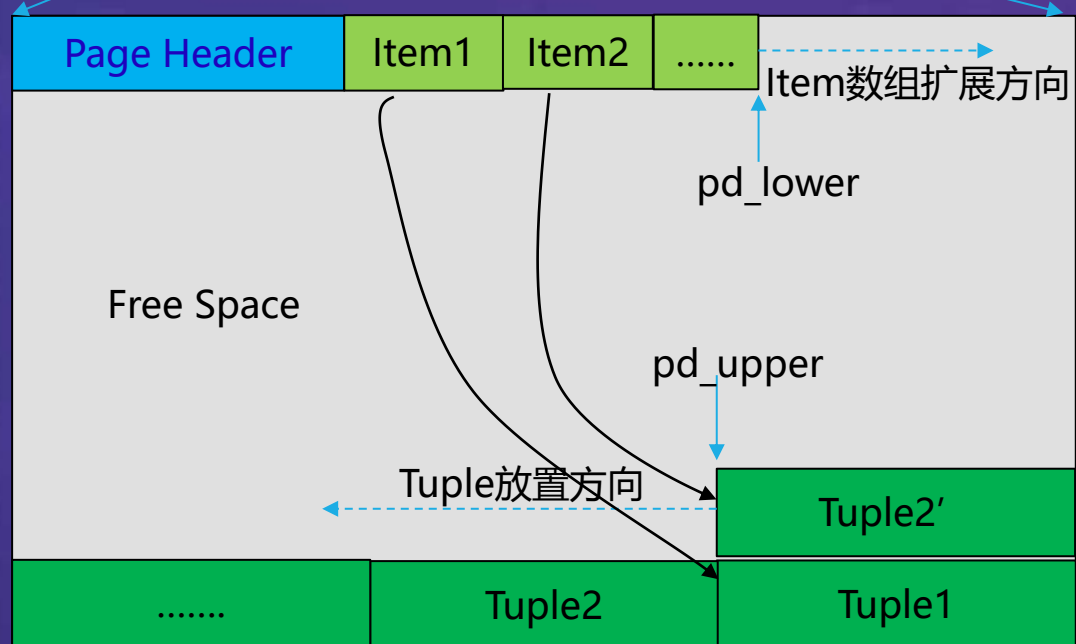
TPCC Throughput on ARM-128 Cores



Basic SQL Speed Up



In-place Update引擎



| 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
 - 线程池原理
 - 增量检查点原理
 - 行列混合引擎
 - MOT内存表原理
 - Inplace Update引擎
- ◆ 可用性
 - 极致RTO优化
 - Paxos协议自选主
- ◆ 安全性
 - 全密态等值查询原理
 - 账本数据库
- ◆ 易运维
 - AI4DB自治数据库
 - DB4AI库内AI引擎

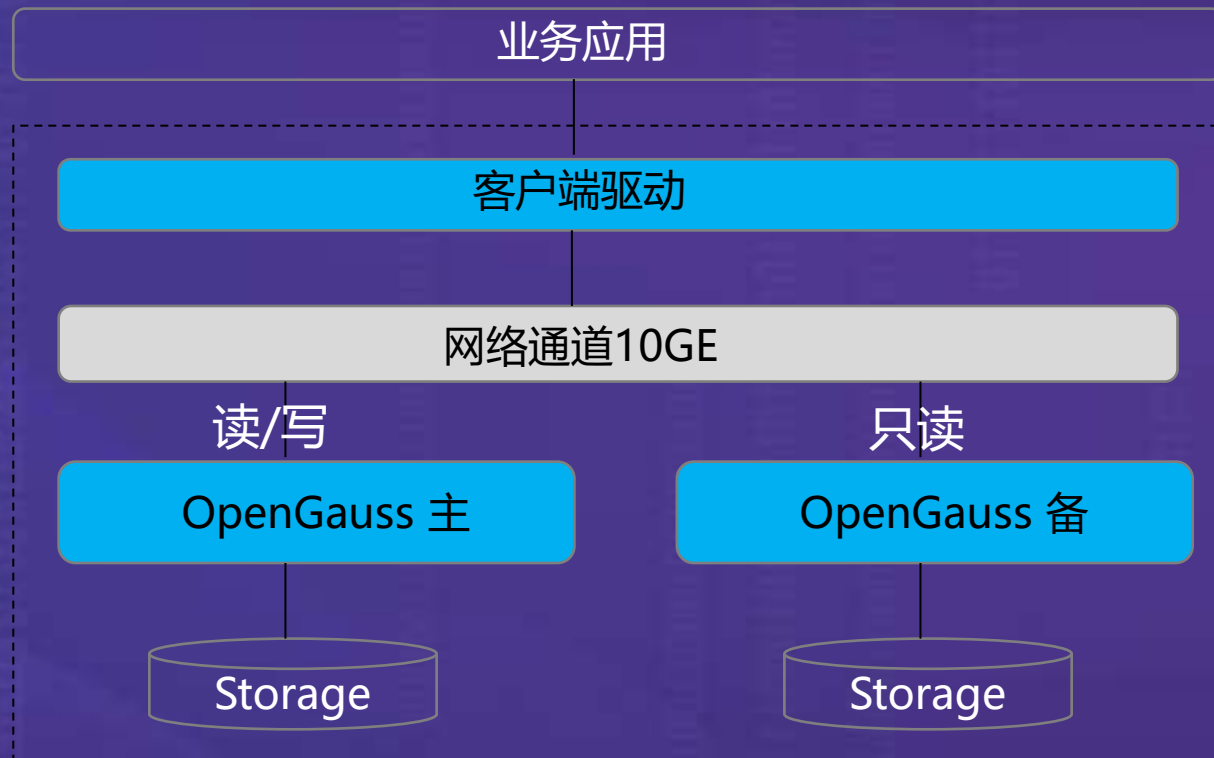
■ openGauss 未来技术发展方向



<https://opengauss.org>



| 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。

。



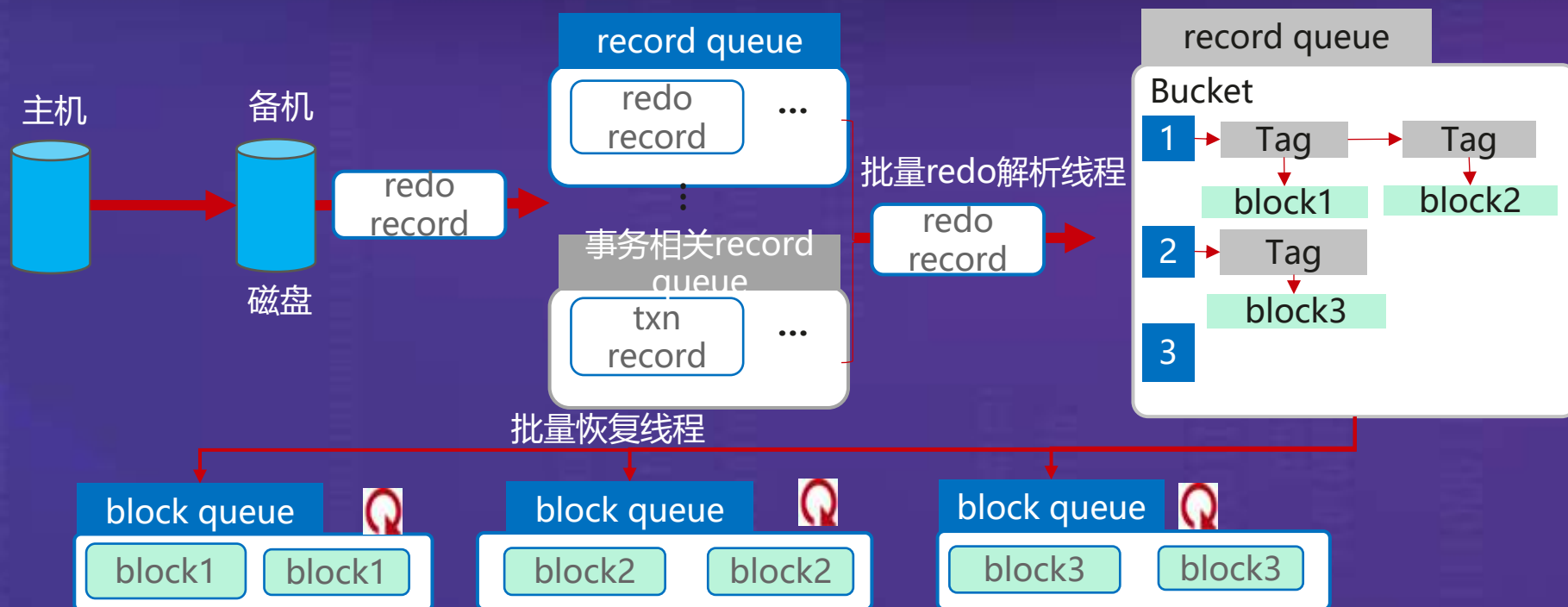
<https://opengauss.org>



openGauss 数据库双机原理



openGauss 数据库极致RTO

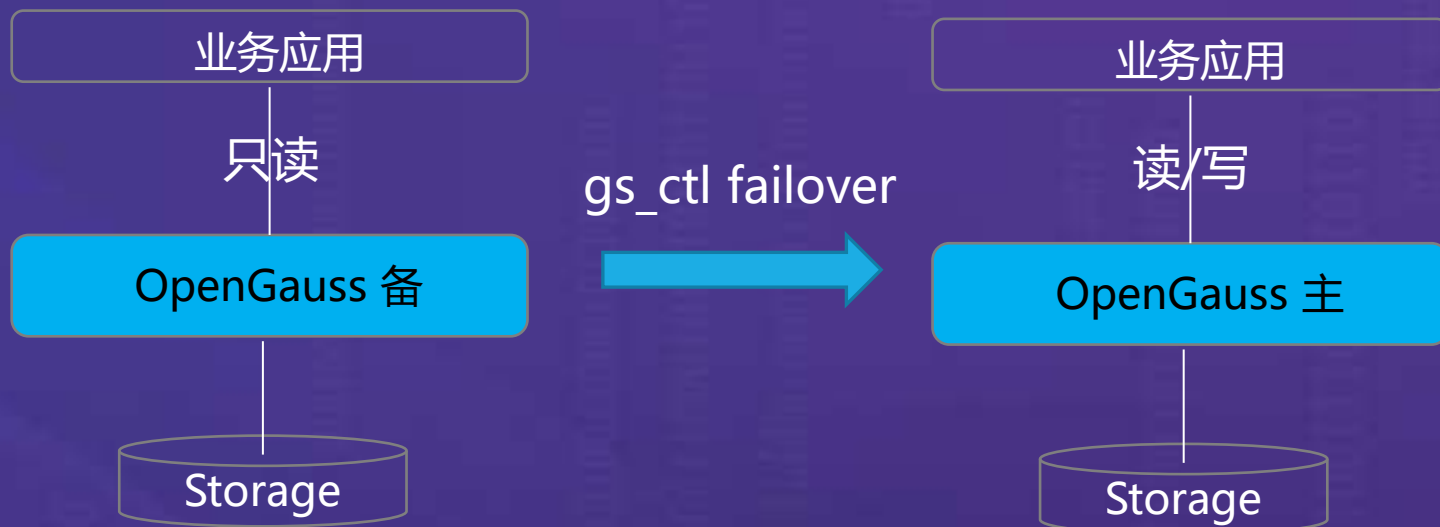


1. 备机日志实时落盘，同时将日志流**并行解码、读取和回放事务日志**；
2. 保证极大化日志回放吞吐，**功能切分服务化、流水线协同**，减少串行度。
3. 批处理化回放，**消除单条回放反复获取/释放锁**的并发控制和IO开销。
4. **页级物理并行**，去除按表切割并行机制，locality更好、并行度更高。



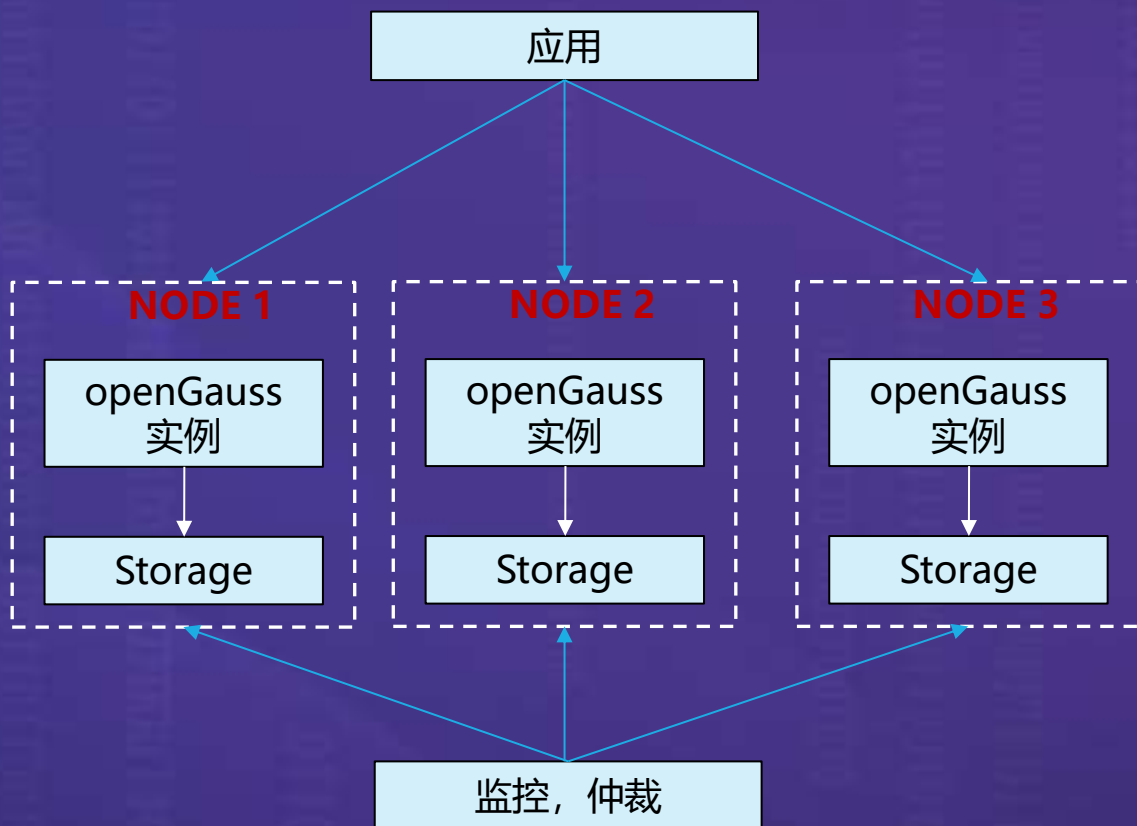
| openGauss 数据库RTO时间

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）。

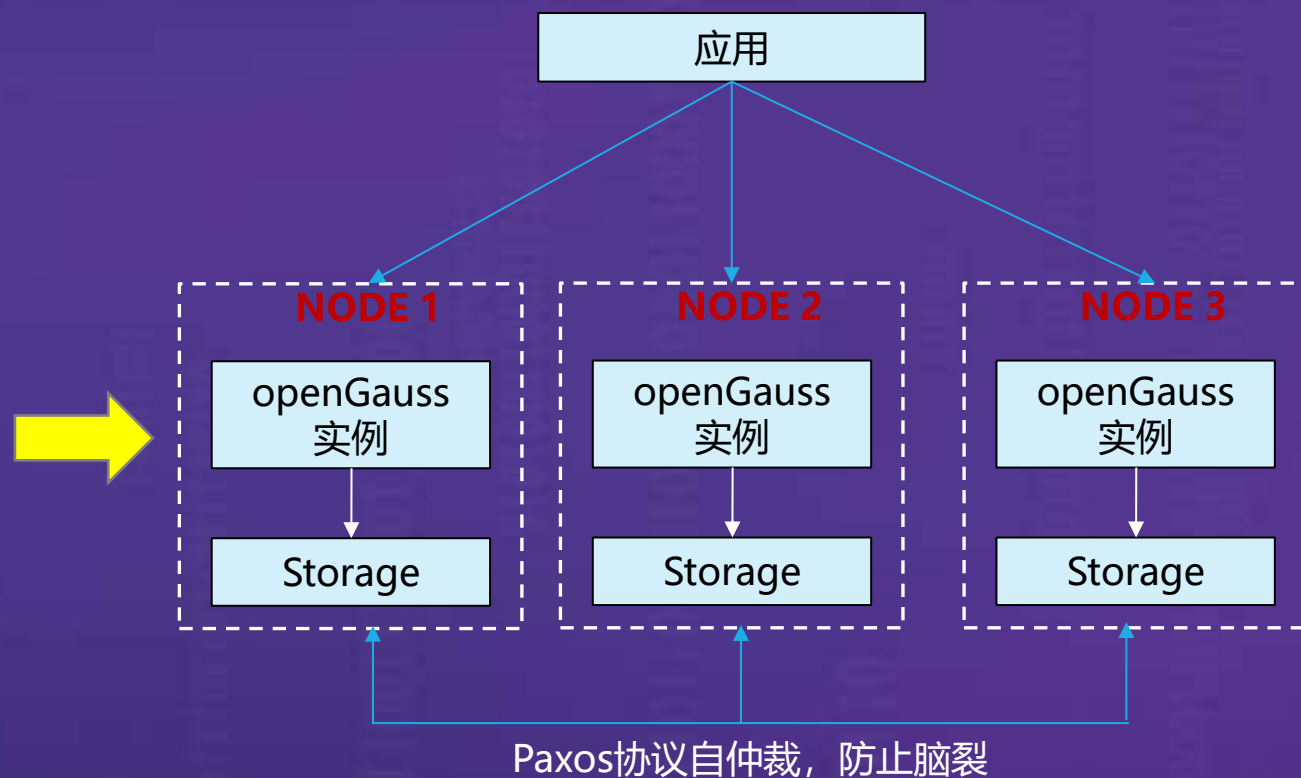


paxos协议自选主介绍

传统主备+集群管理



DCF自仲裁



<https://opengauss.org>

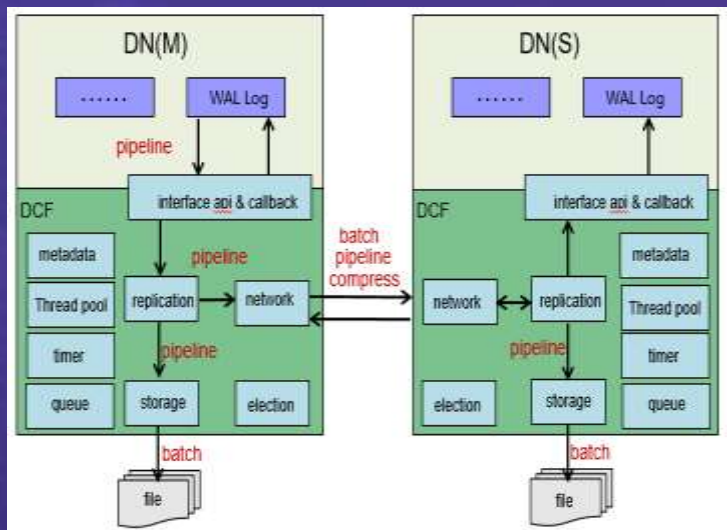


paxos协议自选主实现

openGauss部署示意图



运行架构



openGauss引入基于Paxos协议的DCF组件，能够保证数据一致性的同时，在高可用方面得到极大增强，主要包括：

- ✓ 通过自仲裁、多数派选主能力摆脱第三方仲裁组件，极大缩短RTO时间，且可预防任何故障下的脑裂双主
- ✓ 基于Paxos协议拓展的多样化节点角色，能够提供节点同步、同异步混合部署等多种集群部署方式
- ✓ 高性能、低时延的复制能力，使得openGauss在高可用方面得到极大增强的同时，还提升了主备节点间日志复制效率，提升系统最大吞吐能力

关键设计：

- ✓ 支持自选主
- ✓ 支持异步多线程框架和无锁cache设计
- ✓ 支持多级batch、pipeline操作，以及日志压缩能力
- ✓ 支持日志多流多复制
- ✓ 不同分区主备副本可以在同一节点，方便实现负载均衡和异地多活
- ✓ 基于网络带宽及网络时延自动探测的流控算法
- ✓ 支持batch size、pipeline并发数自适应调整，提升系统最大吞吐量

下阶段演进：

- ✓ 权重化选主：用户可以指定各节点选主优先级，按照优先级从高到低依次激活选主
- ✓ 支持日志空洞和并行离散日志复制能力
- ✓ 高性能网络通信，支持自定义拥塞控制算法的TCP通信及基于UDP协议的丢包重传能力



<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
 - 线程池原理
 - 增量检查点原理
 - 行列混合引擎
 - MOT内存表原理
 - Inplace Update引擎
- ◆ 可用性
 - 极致RTO优化
 - Paxos协议自选主
- ◆ 安全性
 - 全密态等值查询原理
 - 账本数据库
- ◆ 易运维
 - AI4DB自治数据库
 - DB4AI库内AI引擎

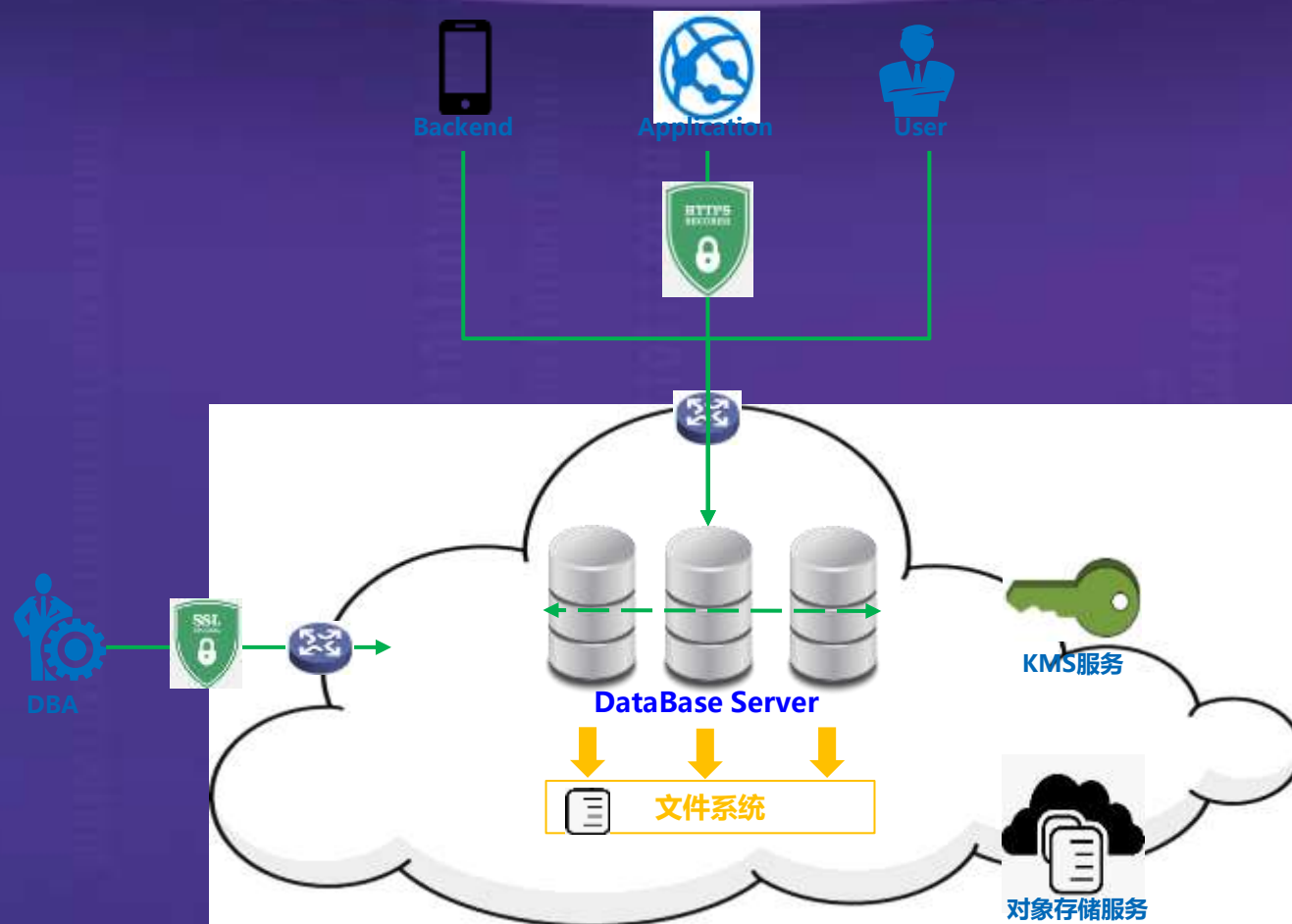
■ openGauss 未来技术发展方向



<https://opengauss.org>



| openGauss部署在云上时的安全问题



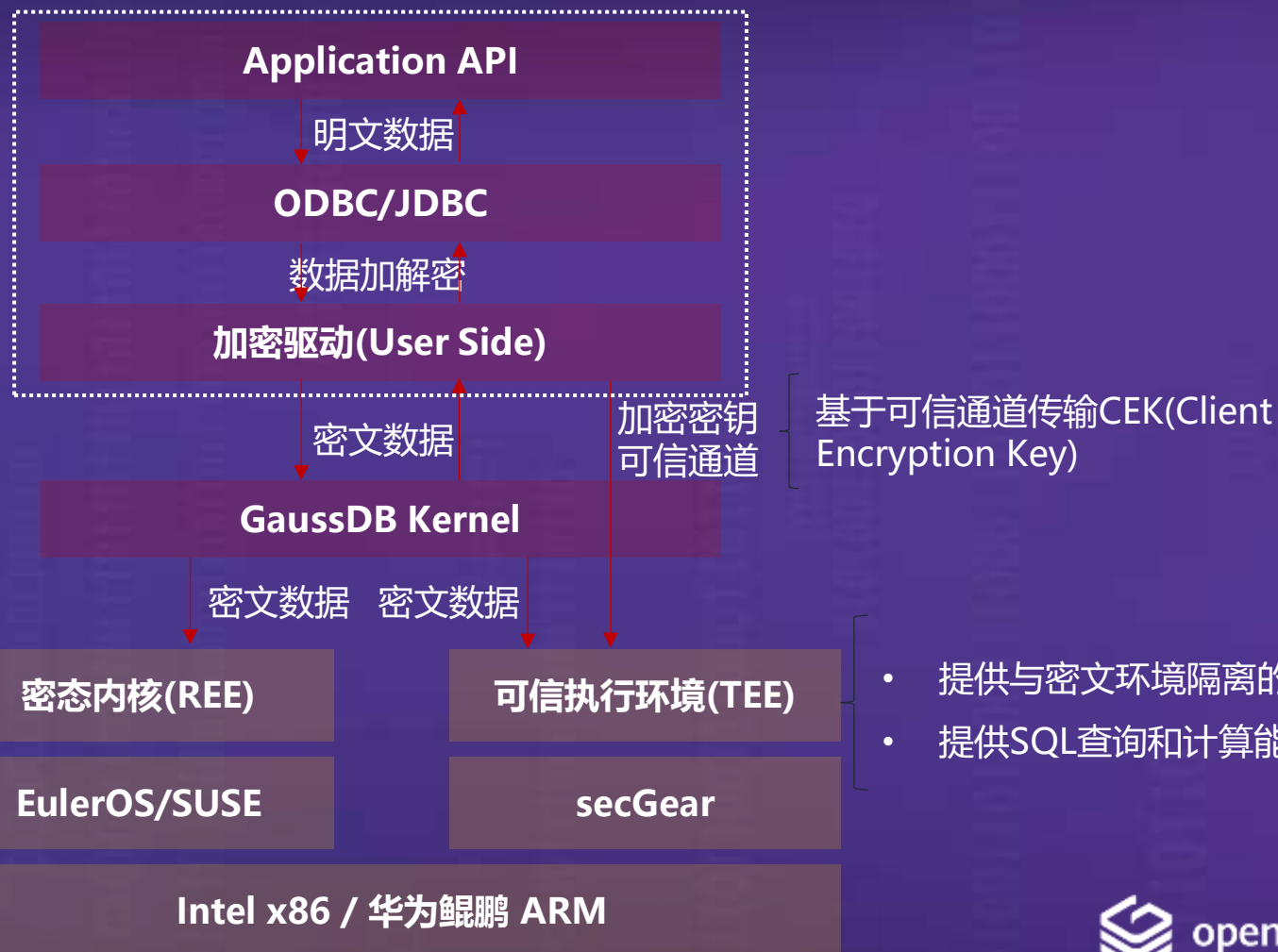
数据库在云上运行时数据存储和访问的安全问题--窃取数据

openGauss

<https://opengauss.org>



| openGauss 全密态等值查询

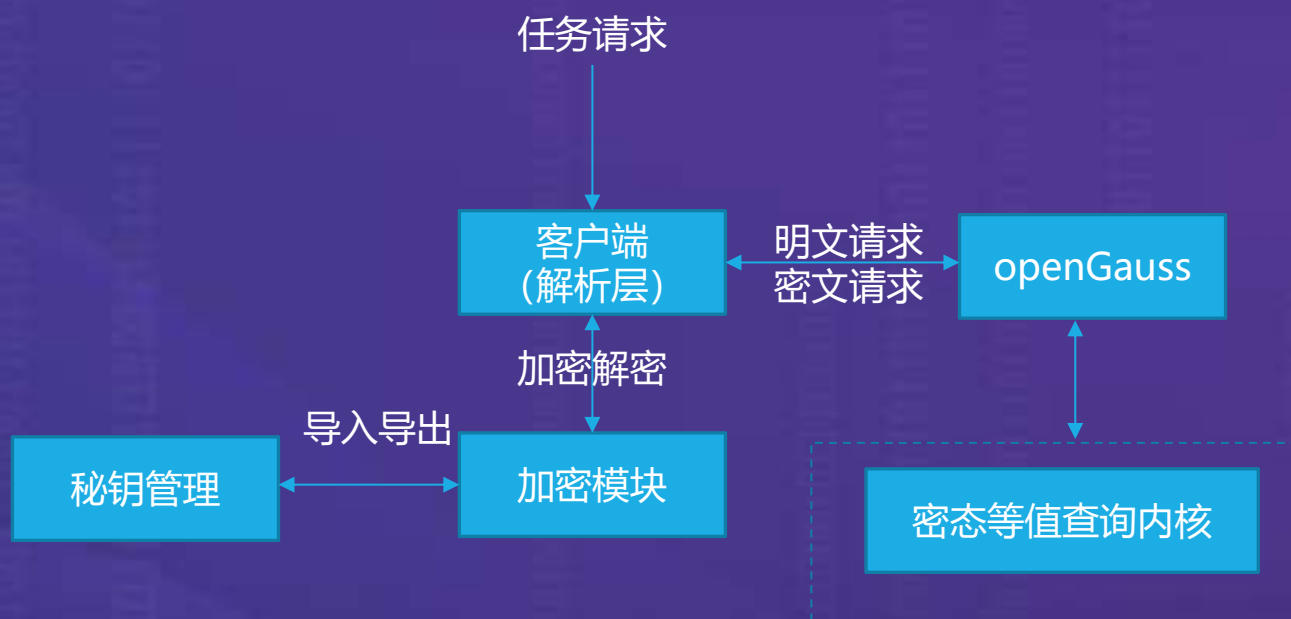


- 数据以密文形式存在，无法解密
- 不存储密钥明文信息

- 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 全密态等值查询流程



1、三层密钥管理机制：根密钥，主密钥和列加密密钥。

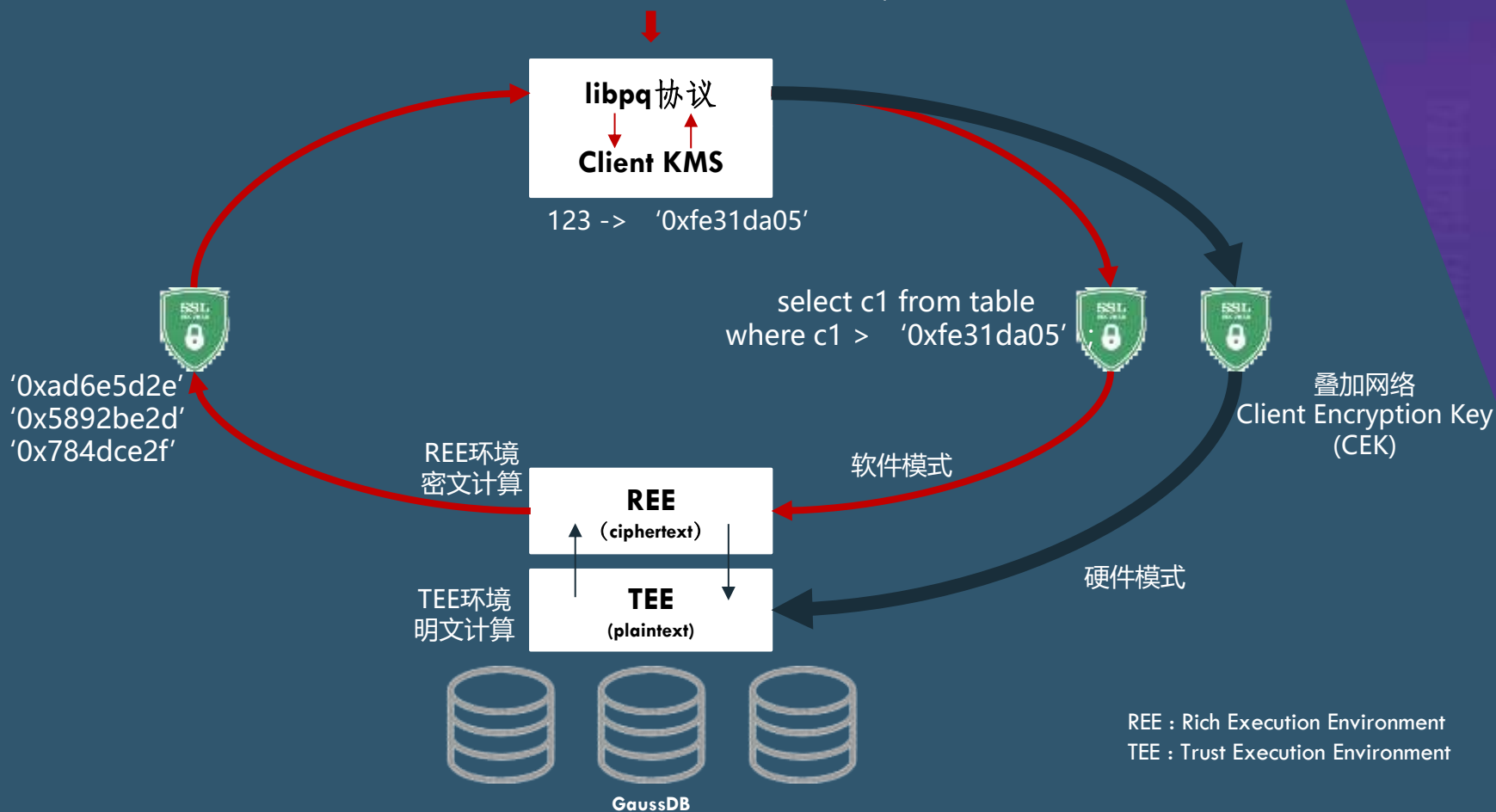
2、客户端完成数据的加密和解密，服务器完成密态数据计算。

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

业务Query
Select c1 from table where c1 > 123;



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

REE : Rich Execution Environment
TEE : Trust Execution Environment

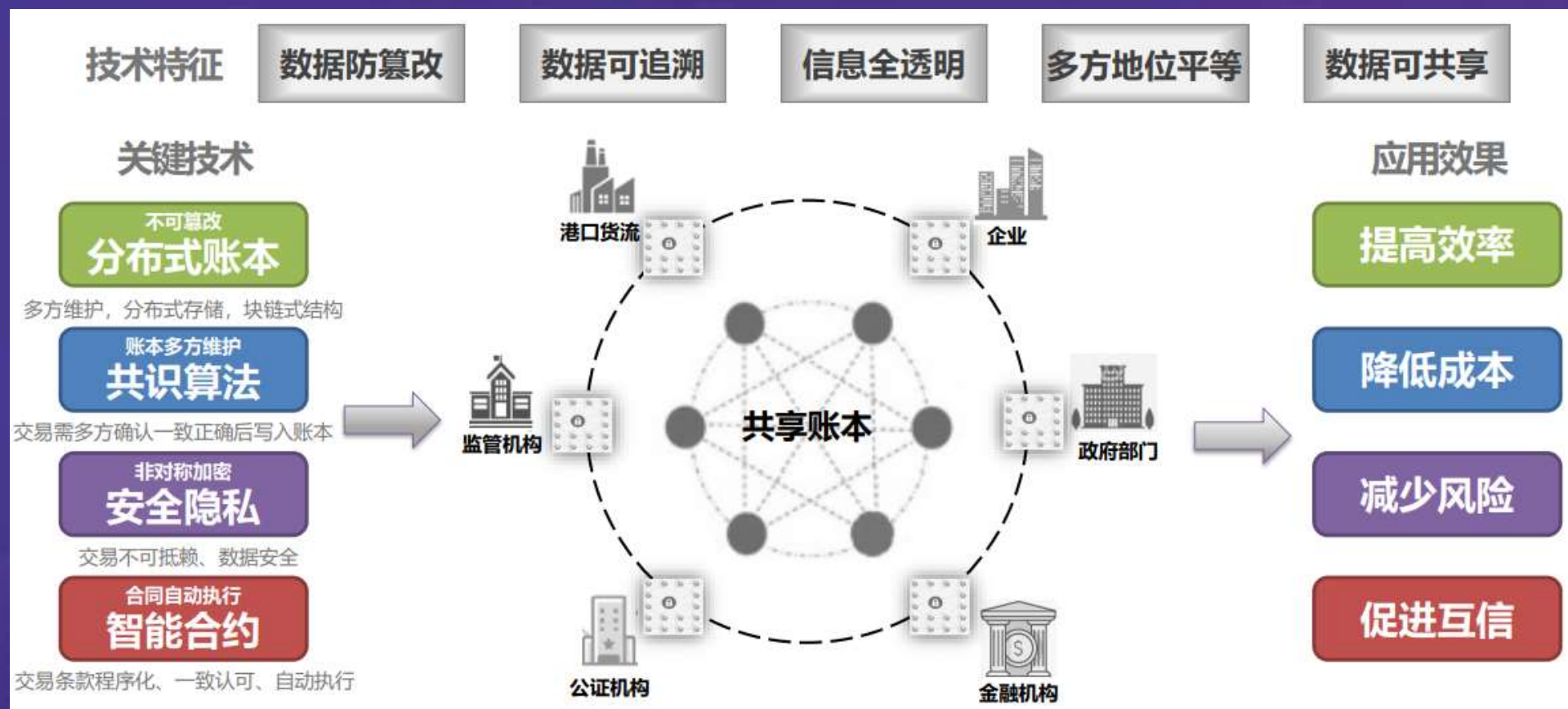


<https://opengauss.org>



| 数字资产可信流动需求

□ 数据追溯：数字货币、政务、食品安全等领域渴望全向追踪管理，防止数据篡改与作弊，实现多方认同，促进数字资产的可信流动；

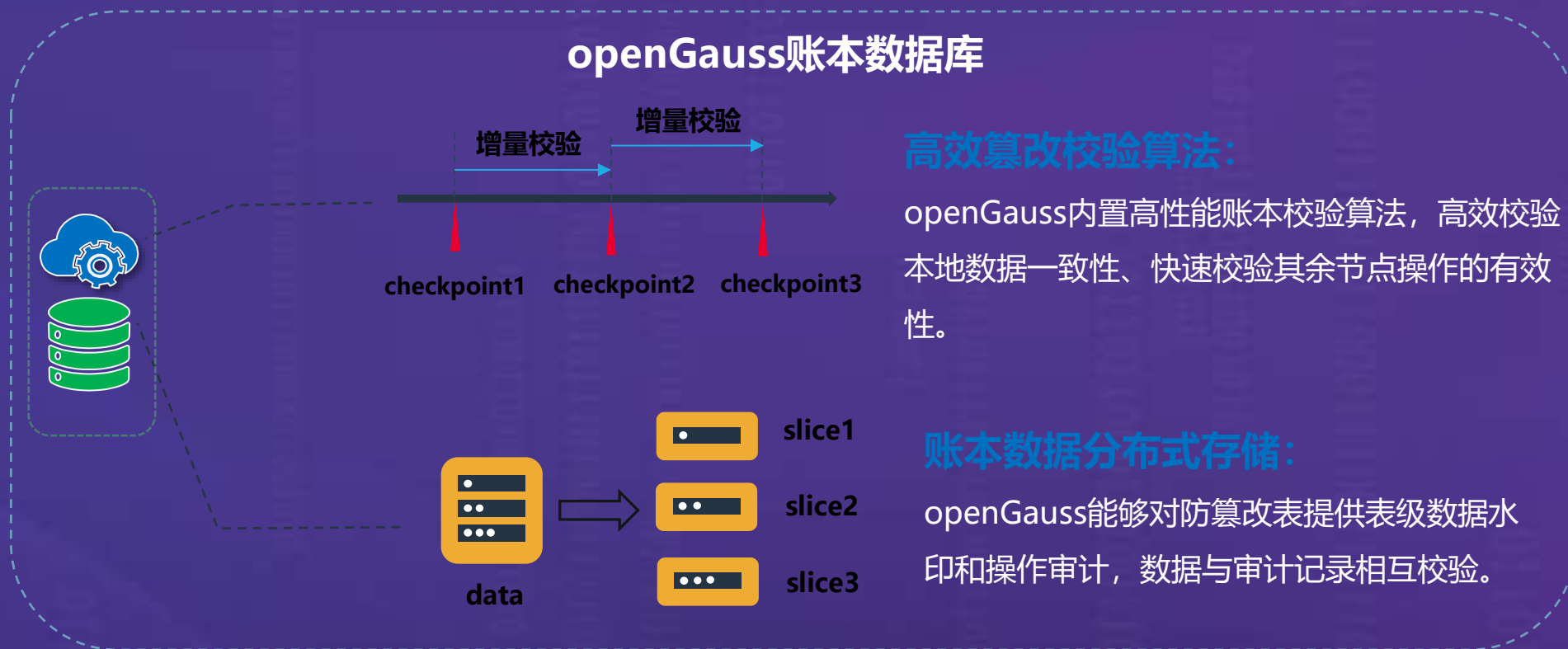


以区块链为代表的数字资产可信流动框架是数字世界不可或缺的根基，是IT领域的新基建



| openGauss防篡改账本数据库

- ❑ **账本数据库部署方案：**防篡改用户表+一致性校验算法



| openGauss防篡改账本数据库

• 账本表的使用

账本表使用与普通表一样，支持全量的增、删、改、查操作，在用户看来，仅仅是普通表多增加了一列hash列。用户可以显示的查询每行数据的hash。

```
db=# create schema ledger with blockchain;

db=# create table ledger.user(id int, name varchar);

db=# insert into ledger.user values(1, 'alex'), (2, 'bob');

db=# select *, hash from ledger.user;
```

id	name	hash
1	alex	1f2e543c580cb8c5
2	bob	8fcd74a8a6a4b484

(2 rows)

账本表操作及记录

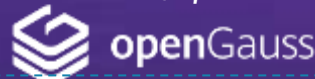
• 校验信息的记录(历史表和全局表)

账本表的数据增删改操作均会被记录到历史表和全局表。记录到历史表的形式为只记录hash摘要，不记录数据；记录到全局表的形式为记录操作的SQL语句和操作造成的摘要更改。

```
db=# select * from blockchain.ledger_user_hist limit 1;
-[ RECORD 1 ]-----
rec_num  | 0
hash_ins | 1f2e543c580cb8c5
hash_del |
pre_hash | b62faf1967d78a3043a2b4ea5cc075d2
```

```
db=# select blocknum, username, starttime, relhash,
txcommand from gs_global_chain limit 1;
-[ RECORD 1 ]-----
```

blocknum	28
username	omm
starttime	2021-11-10 11:20:34.202908+08
relhash	aefbc8e4feb16d49
txcommand	insert into ledger.user values(1, 'alex'), (2, 'bob');



历史表和全局表记录 <https://opengauss.org>



| openGauss防篡改账本数据库

- 数据全流程追踪

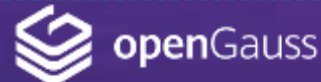
对记录的动作、数据和校验信息进行关联查询，结合数据库物理日志/逻辑日志中的源信息可以对数据的全流程进行追踪审计，知道：

a. 谁执行了哪些命令， b. 命令改动了哪些数据， c. 数据被谁修改过。同时，追踪功能的每一步都能结合校验信息进行篡改校验。

同时，对导出的数据，能够跟库内的数据进行关联校验，保证数据的全生命周期的追踪。

starttime	txcommand	data_ins	data_del
2021-12-02 11:22:22.168519+08	insert into ledger.t1 values(1, 'alex'), (2, 'bob'), (3, 'peter');	1, 'alex'	
2021-12-02 11:22:22.168519+08	insert into ledger.t1 values(1, 'alex'), (2, 'bob'), (3, 'peter');	2, 'bob'	
2021-12-02 11:22:22.168519+08	insert into ledger.t1 values(1, 'alex'), (2, 'bob'), (3, 'peter');	3, 'peter'	
2021-12-02 11:22:22.170443+08	update ledger.t1 set name = 'bob2' where id = 2;	2, 'bob2'	2, 'bob'
2021-12-02 11:22:22.171845+08	delete from ledger.t1 where id = 3;		3, 'peter'
2021-12-02 11:22:22.173231+08	update ledger.t1 set name = 'bob3' where id = 2;	2, 'bob3'	2, 'bob2'

数据全流程追踪



<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
 - 线程池原理
 - 增量检查点原理
 - 行列混合引擎
 - MOT内存表原理
 - Inplace Update引擎
- ◆ 可用性
 - 极致RTO优化
 - Paxos协议自选主
- ◆ 安全性
 - 全密态等值查询原理
 - 账本数据库
- ◆ 易运维
 - AI4DB自治数据库
 - DB4AI库内AI引擎

■ openGauss 未来技术发展方向

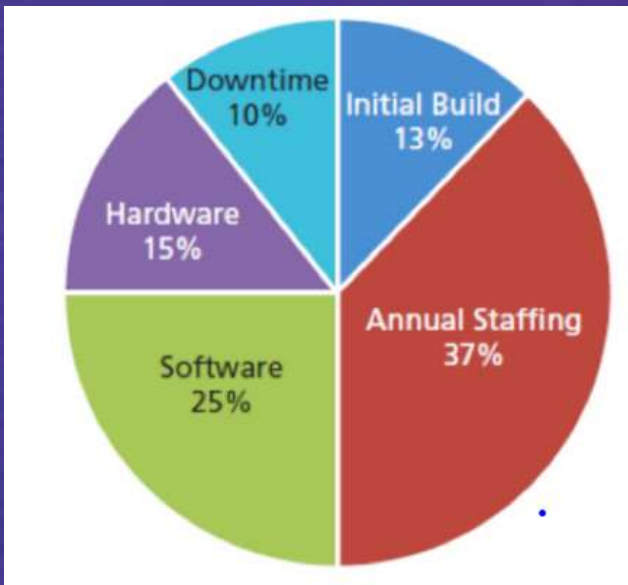


<https://opengauss.org>



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

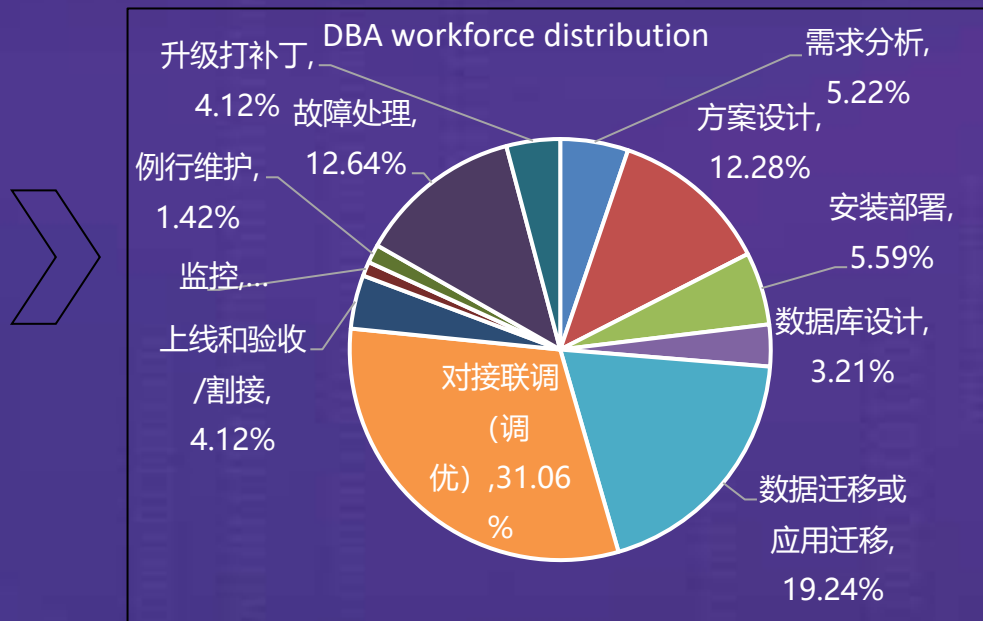


DB maintenance cost is biggest part of TCO

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

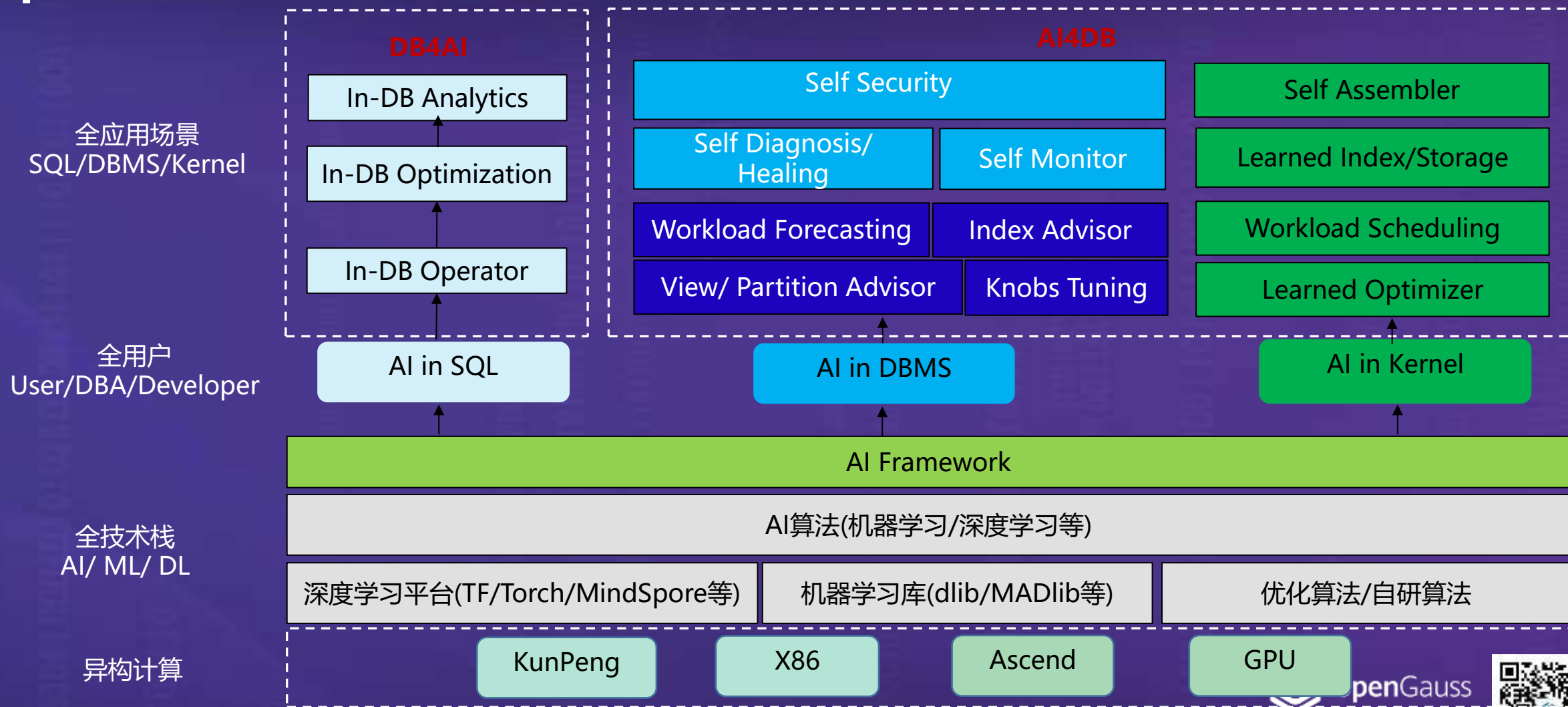
可自动化



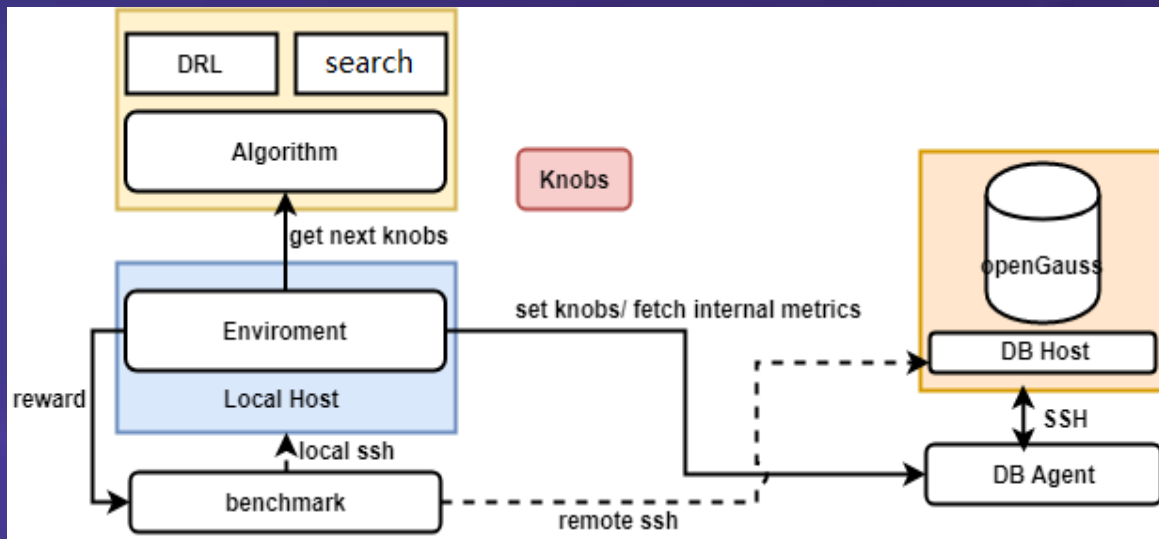
<https://opengauss.org>



openGauss AI 全景



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 | 0.0 |
| 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 | 134154882.48040875 |
| search_modify_ratio | 187.59523392981777 |
| ap_index | 2.3759498376861847 |
| current_free_mem | 31161892 |
| os_mem_total | 32779400 |
| 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 | 845237.440716804 |
| shared_buffers_toast_hit_rate | 98.16007359705611 |
| shared_buffers_tidx_hit_rate | 99.11667280088332 |
| shared_buffers_idx_hit_rate | 99.74473859023848 |
| shared_buffers_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 concurrency. You are recommended to set max_connections based on the number of CPU cores. 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 value 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 |
+-----+-----+-----+-----+-----+
```



<https://opengauss.org>



openGauss慢SQL发现能力

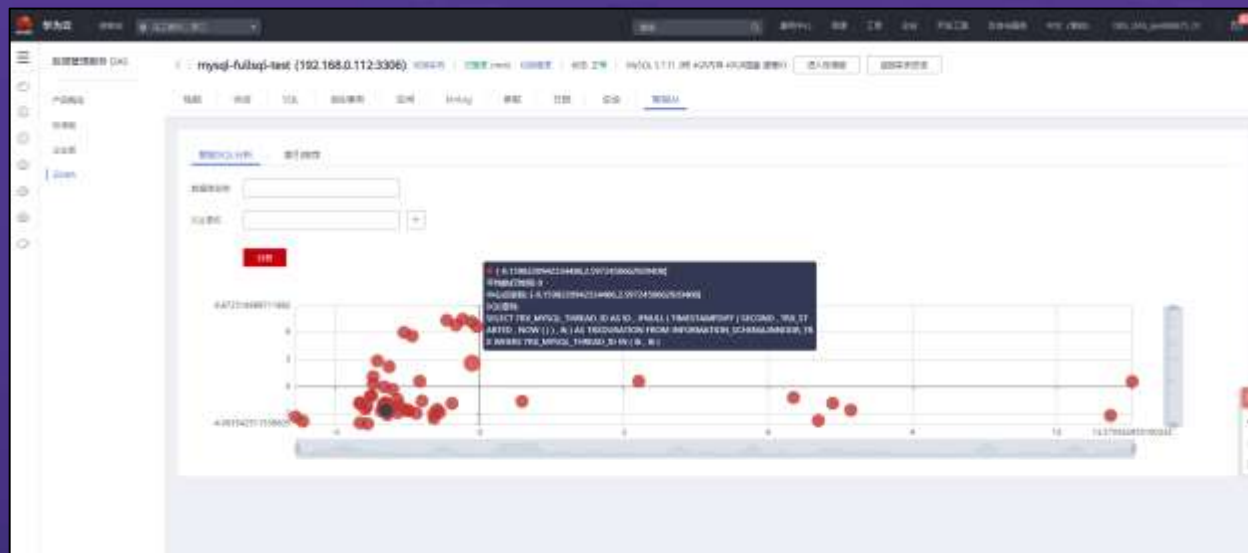
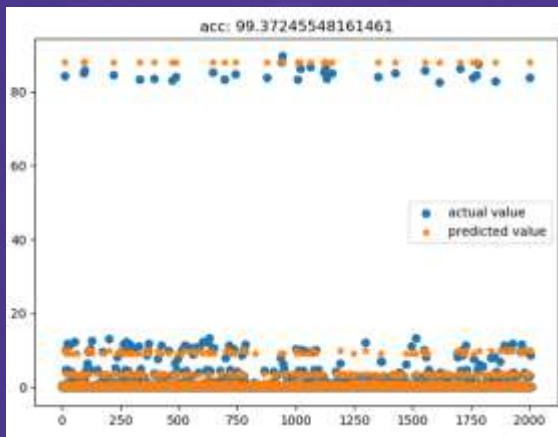
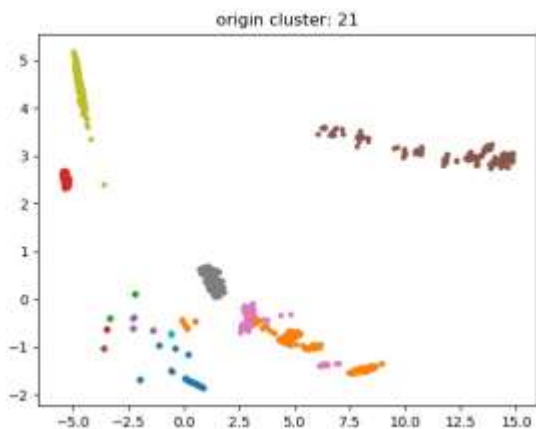
使用场景介绍:

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



<https://opengauss.org>

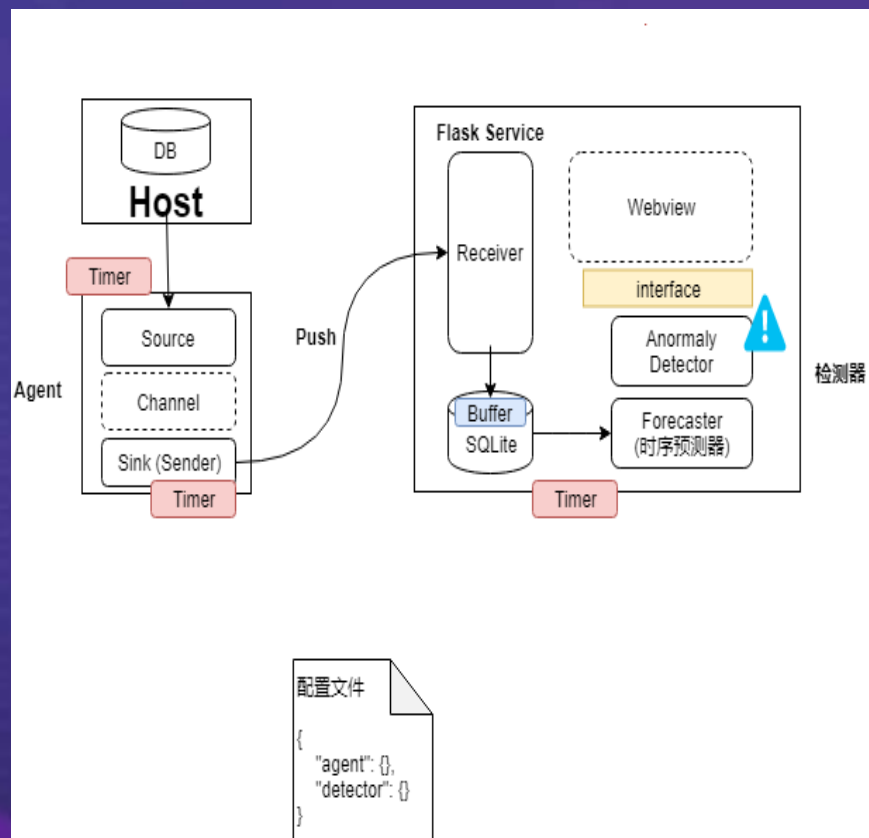


openGauss数据库监控与异常检测

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] kun@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] kun@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 |
+-----+-----+-----+-----+
[l] kun@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] kun@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=37031.112..37034.606 rows=105 loops=1)
  Number of Workers: 2
  => Parallel Seq Scan on lineitem (cost=0.00..898729.51 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: 59085947
  Total runtime: 37000.647 ms
(5 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=0.80..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: 1.050 ms
(4 rows)
```

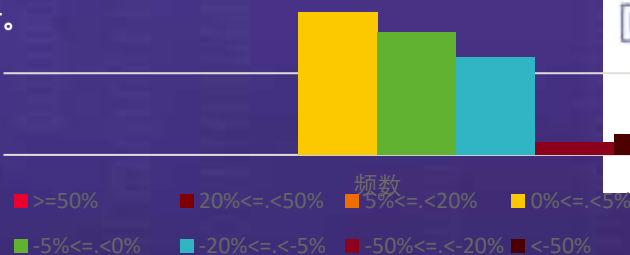
对索引推荐效果的整体评估:

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:

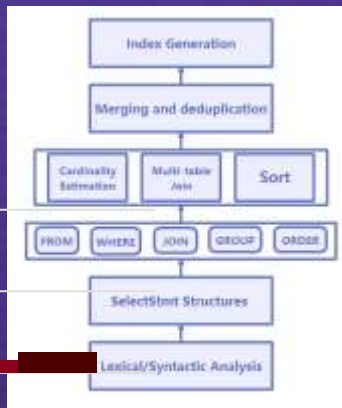
采用索引设计和优化的相关理论, 基于原生的词法和语法解析, 对查询语句中的子句和谓词进行分析和处理, 再结合字段选择度、聚合条件、多表Join关系等输出最终建议。

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。



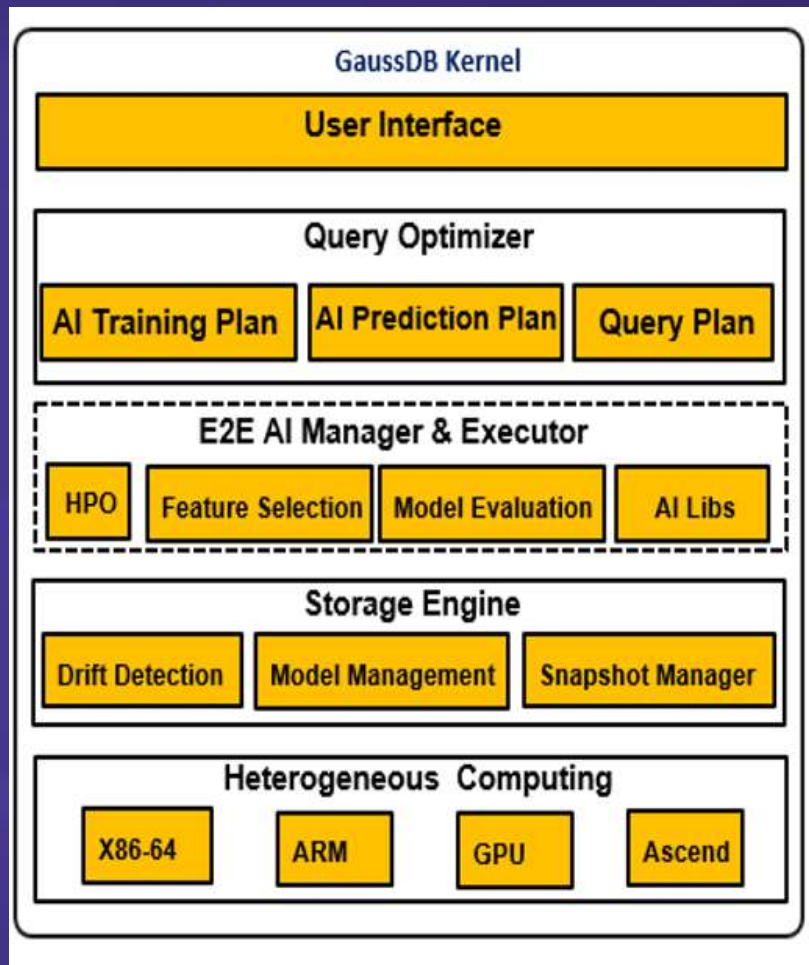
从单条到workload



<https://opengauss.org>



openGauss DB4AI



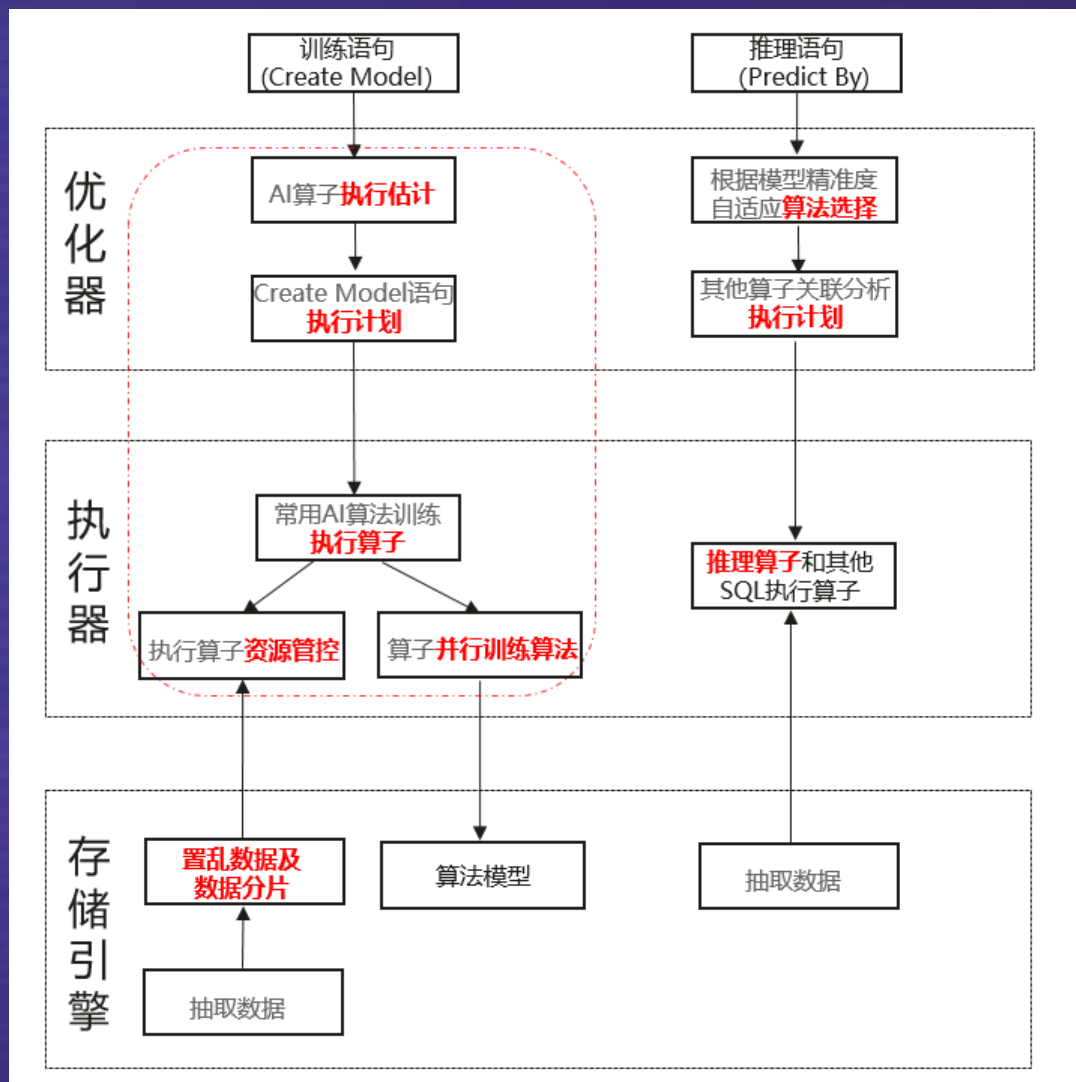
- **用户接口层**: 实现SQL-like 语法, 提供PREDICT、MODEL等关键字, 支持模型的训练、预测以及管理等;
- **语句优化层**: 实现业内首创的原生AI算子, 优化器生成包含原生AI算子的执行计划; 实现代价估计, 支持通过EXPLAIN语句查看详细的执行开销, 并提供可能的路径选择能力 (ShuffleScan 下推) 。
- **AI全流程管理**: 支持对模型的管理、评估, 支持模型的定期、定点更新与 fine-tune等; 支持AutoML能力, 具有超参数调优 (Hyper Parameter Optimization, HPO) 能力; 具备特征选择、特征处理、与数据清洗能力。
- **AI执行器**: 负责执行AI执行计划, 支持算法的并行执行。
- **存储层**: 支持为数据创建快照并进行管理, 负责模型文件的存储与组织。
- **异构计算能力**: 支持多种计算平台, 包括X86架构、ARM架构、以及具备GPU的环境。
- **AI底座**: 提供数据库内部API, 支持数据库内AI能力, 如AI优化器、AI buffer、AI索引等。



<https://opengauss.org>



openGauss DB4AI SQL方式训练和推理的流程



使用Create Model语法实现数据库内AI训练和模型存储:

- 通过数据库并行训练算法, 实现库内训练加速;
- 通过执行代价估计及资源管控, 实现迭代轮次和模型参数最优化;
- 提供可视化执行计划, 实现AI可查、可解释性。

举例: **Create Model** price_model using logistic_regression Features size, lot Target price < 1000 from Houses;

使用Predict By语法实现数据库内AI模型推理:

- 基于数据集和查询特征进行模型归类;
- 优化器通过同类模型的精准度, 自适应选择算法, 实现最优化模型推理
- AI推理算子结合其他执行算子, 实现关联分析

举例: Select address, **Predict By** price_model(Features size, lot), size from ho



<https://opengauss.org>



| openGauss DB4AI 性能测试 - XGBoost

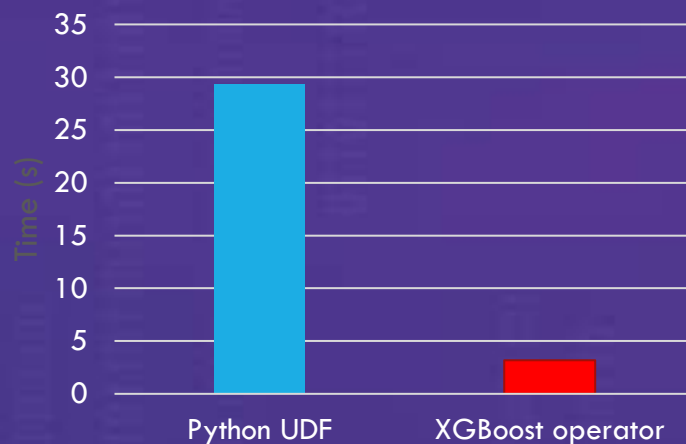
结果分析:

- 1、数据读取(批次读取)
- 2、Python UDF在计算过程中存在大量的数据转写环节
- 3、DB4AI存在指令集加速:(option)

xgboost

with hyperparams: "n_iter=10, max_depth=5, min_child_weight=1, nthread=10, eta=0.1, booster='gbtree'"

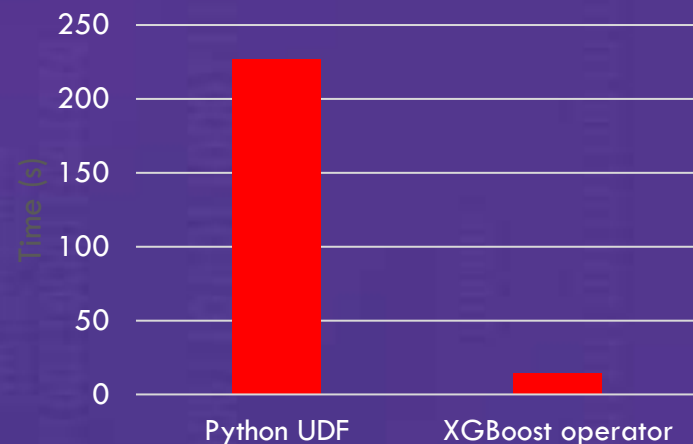
CREATE MODEL - 1M rows



Results for dataset with 1m rows:

	Classification (s)	Prediction (s)
Python UDF	29.29	227.36
In-operator	3.18	14.46
KPI Speedup	9.2x	15.7x

PREDICT BY 1M rows



Results for dataset with 10m rows:

	Classification (s)	Prediction (s)
Python UDF	N/A (Error)	N/A (Error)
In-operator	43.0	204.7



<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。



<https://opengauss.org>



| openGauss 核心技术与实践

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



<https://opengauss.org>



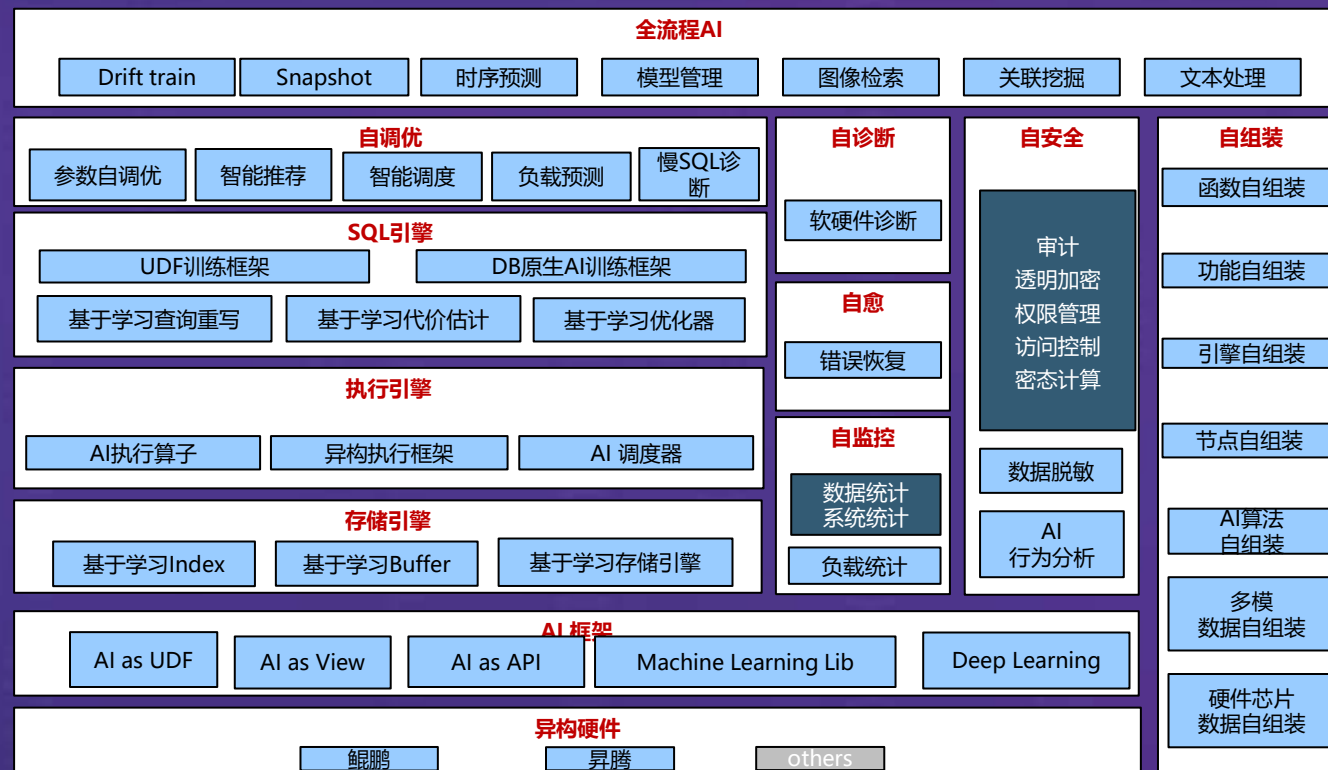
openGauss未来技术方向：更智能、更安全、更高效



We are here

- AI算法加持，解决传统数据库问题
 - 学习型数据库组件，如AI-Based Optimizer
 - SQL自优化：hint推荐、查询重写
- 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：全流程AI

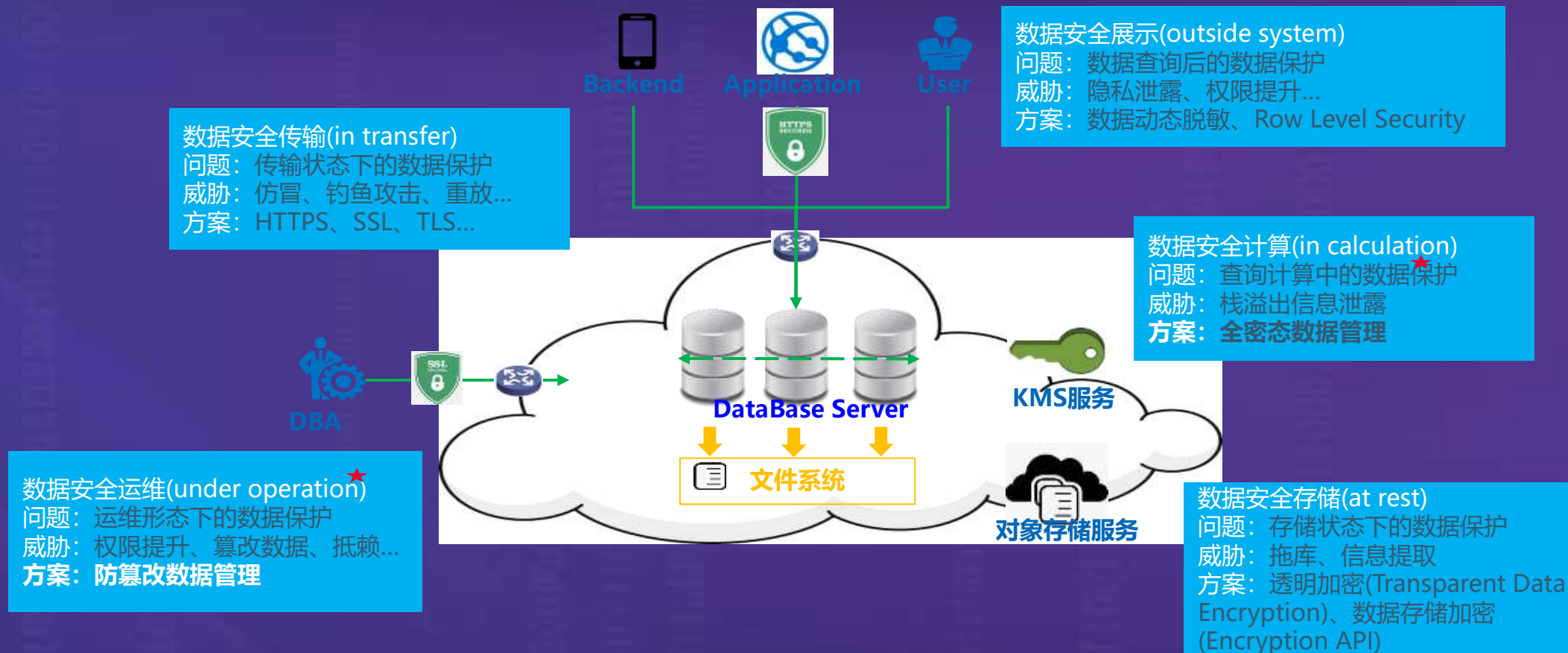
DBMind功能全景图



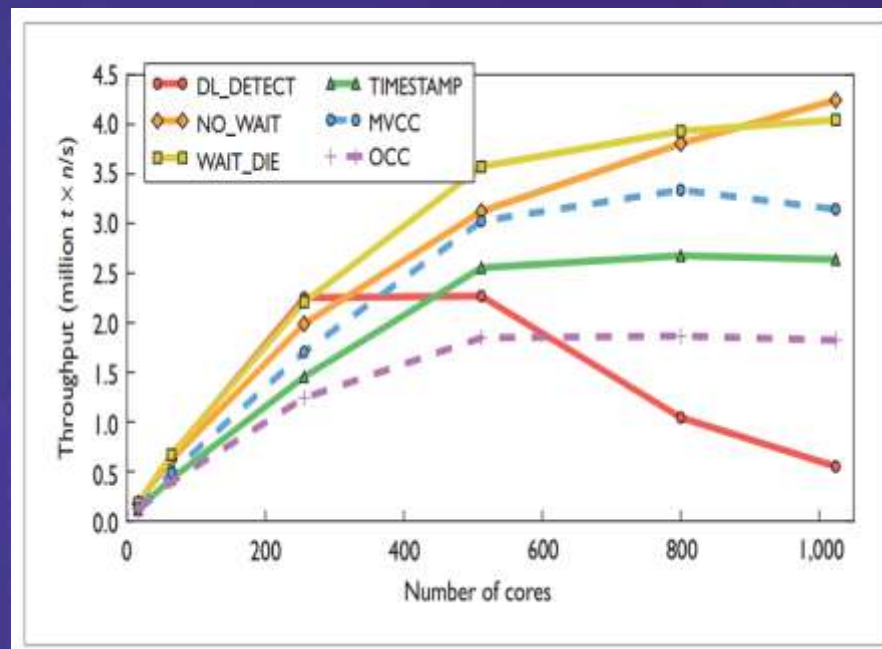
<https://opengauss.org>



| openGauss未来技术方向：更智能、更安全、更高效



| 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;

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

	DRAM	PCM	RRAM	MRAM	SSD	HDD
Read latency	60 ns	50 ns	100 ns	20 ns	25 μ s	10 ms
Write latency	60 ns	150 ns	100 ns	20 ns	300 μ s	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	$>10^{16}$



<https://opengauss.org>



| openGauss 2022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。 I基于libnet和DPDK的网络加速。
可靠	Paxos协议增强。 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。
AI	SQL根因分析、库内AI算子扩展、ABO优化器	AI4DB：SQL根因分析（慢SQL为什么慢）、ABO优化器 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。



<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



<https://opengauss.org>



Thank you!



<https://opengauss.org>

