GaussDB:智能云原生分布式数据库

李国良[1],王 磊[1],张金玉[1],王 江[1],任 阳[1],张 琼[1],申 宇[1],
董亚辉[1],宋 涛[1],乔 典[2]
(1. 清华大学计算机科学与技术系,北京 100084;2. 华为技术有限公司2012高斯实验室,北京 100085)

摘 要: 在互联网技术飞速发展,传统企业面临数字化转型的大背景下,作为数字时代数据基础设施,数据库面临大数据量、高可用、在线弹性伸缩、智能化、安全防护等方面的巨大挑战. 为了应对诸多挑战,GaussDB应运而生,GaussDB提出分布式数据库的分布式查询优化技术提升了数据查询性能,提出分布式数据库的高可用容灾技术提升了企业数据的可用性和可靠性,提出分布式数据库的云原生计算存储分离和弹性伸缩技术提高了对存储等资源的利用率,提出分布式数据库的自治管理技术增强了数据库的智能管理,提出分布式数据库的全方位安全防护技术提高了数据的安全保护能力. GaussDB能支撑关键基础行业核心场景的数字化转型.

关键词: 数据库;高可用;云原生;分布式;智能化;安全
基金项目: 国家重点研发计划(No.2023YFB4503600);国家自然科学基金(No.61925205,No.62232009,No.62102215)
中图分类号: TP315;TP392 **文献标识码:** A **文章编号:** 0372-2112(2025)04-1103-20
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20231026

GaussDB: Intelligent Cloud-Native Distributed Database

LI Guo-liang[1], WANG Lei[1], ZHANG Jin-yu[1], WANG Jiang[1], REN Yang[1], ZHANG Qiong[1], SHEN Yu[1],
DONG Ya-hui[1], SONG Tao[1], QIAO Dian[2]
(1. *Department of Computer Science and Technology*, *Tsinghua University*, *Beijing* 100084, *China*;
2. *Gauss Laboratory* 2012, *Huawei Technologies Co.*, *Ltd.*, *Beijing* 100085, *China*)

Abstract: With the rapid development of Internet technologies and digital transformation of traditional enterprises, databases are facing great challenges in terms of large data volume, high availability, cloud-native elasticity, intelligence, and security protection as data infrastructure in the digital age. To address these challenges, GaussDB came into being. It proposes the distributed query optimization technology of distributed databases to improve data query performance, proposes the HA disaster recovery technology of distributed databases to improve enterprise data availability and reliability, proposes the cloud-native computing and storage separation and elastic scaling technology of distributed databases to improve storage resource utilization, proposes the autonomous management technology of distributed database is proposed to enhance the intelligent management of database, and proposes the all-round security protection technology of distributed database is proposed to improve the security protection ability of data. With all these new technologies, GaussDB supports the digital transformation of core scenarios in key basic industries

Key words: database; high availability; cloud native; distributed; autonomy; security
Foundation Item(s): National Key Research and Development Program of China (No.2023YFB4503600); National Nature Science Foundation of China (No.61925205, No.62232009, No.62102215)

1 引言

随着云计算、互联网技术的高速发展,数据库技术也经历了四个阶段的变革. 在当前传统企业数字化转型,大数据、人工智能等技术发展的大背景下,数据库技术面临着如下五大挑战[1].

挑战一:大数据量挑战. 分布式在应对海量数据处理方面相较于传统的集中式数据库有了较大的突破,然而随着分布式数据库研究的深入,其发展和规模推广也面临着一些关键的技术挑战,如高性能的分布式事务处理,智能数据分布、分布式查询优化等.

收稿日期:2023-11-01;修回日期:2024-12-16;责任编辑:宋甲英

挑战二：高可用挑战. 在企业数字化建设日益繁盛的今天，企业业务的可用性、数据存储的可靠性变得越来越重要，尤其是对于金融、政府等重点行业来说，如何提供不同等级的高可用容灾方案也成为企业数字化转型背景下数据库技术的关键挑战.

挑战三：云原生弹性挑战. 随着云计算技术的发展，云计算架构对数据库的技术演进，产生了非常重要的影响，以往传统数据库基本采用的都是紧耦合设计，主要为了系统的高性能. 云计算架构的理念是通过计算、存储的高度解耦，来实现各个层次资源的高效利用和弹性扩展，如何将数据库与云计算技术相结合，从而实现数据库服务性价比的提升也就成为了云时代数据库技术发展的关键挑战.

挑战四：智能优化挑战. 近年来，人工智能发展如火如荼，AI技术的演进离不开数据，同时数据库本身就是一个复杂的优化与调度系统. 一方面，如何通过AI技术实现数据库管理系统的智能自治，另一方面实现数据库原生AI引擎，对外提供一站式AI能力也成为了智能型数据库发展的关键挑战.

挑战五：安全防护挑战. 在大数据、大计算的时代背景下，数据安全与隐私保护已然成为当今社会关注的热点，尤其是近年来敏感数据泄露案件时有发生，大众保护个人信息意识提升. 如何保证数据的安全和隐私是数据管理系统的核心诉求.

本文围绕数据库技术发展面临的如上挑战和机遇，介绍了一种智能云原生分布式系统GaussDB，其主要技术创新贡献可以总结为5个方面：①分布式查询优化技术及实现，用以解决大规模分布式查询性能与资源最优调度难题；②分布式高可用与容灾方案设计，提供多层级的高可用、高可靠方案，基于底层共享存储解决日志同步复制痛点；③云原生分布式存储、计算分离与弹性伸缩设计及实现，提高系统的动态扩展能力，充分发挥云基础设施灵活性；④分布式智能优化技术创新与实现，通过智能化的技术手段，实现内核的自治优化，提高数据库系统易用性，降低维护及优化成本；⑤全方位安全防护，构建数据全生命周期、全方位的安全防护能力，消减数据库安全威胁.

2 GaussDB关键技术架构设计

GaussDB采用云原生分布式架构作为设计关键底座，如图1所示，分为SQL优化、执行、数据存储管理、数据库安全防护以及AI自治加速等关键模块. SQL引擎主要包含了分布式近数计算、全并行编译执行以及大规模并发处理等关键技术；存储引擎则侧重于数据组织管理，以及高可用容灾设计，包含了故障自感知技术、跨区域容灾解决方案等；安全模块则主要分为事前的行为管控如权限分离、访问控制等，事中的数据安全防护如透明加密、全密态等，以及事后的统一审计及防篡改追踪等；AI主要侧重于自治优化、智能优化以及库内原生AI算法等.

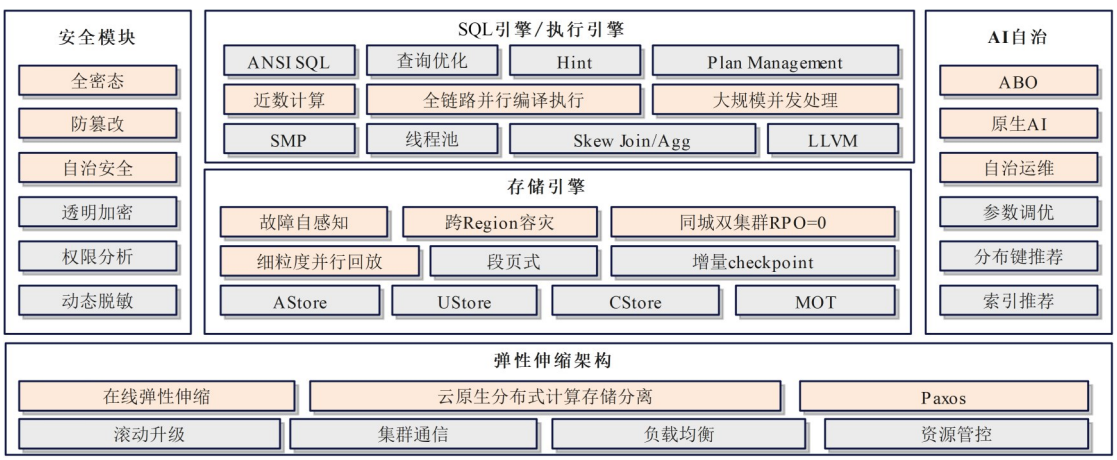

图1 GaussDB关键技术架构图

3 GaussDB关键技术及实现

3.1 分布式查询处理技术

数据库处于IT应用软件栈最核心的位置，是一个对数据访问时延、吞吐极度敏感的系统，其性能决定了整个应用软件对外的服务体验. 因此，对于分布式数据库来说，分布式查询优化的能力至关重要. 当前分布式数据库面临三个挑战[2,3]："如何产生最优的计划？""如何执行速度最快？""如何处理大规模的并发事务？"针对这三大挑战，GaussDB分布式数据库分别在分布式优化、分布式并行执行以及大规模并发事务处理开展技术攻坚工作，消除分布式系统中心瓶颈点，提升多节点协同计算效率[4]. 以下将详细介绍GaussDB分布式数据库在分布式查询优化技术[5]创新上的关键技术点.

3.1.1 近数计算的分布式优化技术

传统分布式数据库产生的分布式计划,都是通过协调节点进行控制. 数据节点提供数据,所有数据汇聚到协调节点上再进行计算. 典型的例子有PGXC计划. 这种架构实现起来简单,但是使得协调节点容易成为瓶颈,单条查询只能在一个协调节点上执行;同时,对于典型的OLTP查询,协调节点、数据节点都需要完成解析、优化、执行一系列操作,会增加大量额外的延迟.

GaussDB实现了支持近数计算的分布式并行优化技术,自动分析查询涉及的数据分布,产生最佳的分布式计划,尽量让计算在协调节点上完成,如图2所示. 首先,对于可以在单个协调节点上执行的查询,提供协调节点剪枝技术,让协调节点退化成网络代理,所有的数据库内核操作如解析、优化、执行都在单个协调节点上执行,性能达到极致;其次,对于涉及多数据节点,但是可以垂直分片(即查询可以在各个数据节点上独立执行,无需交互)的查询,提供数据分布感知的查询转移技术,查询直接下发到各个数据节点上独立执行,协调节点只负责收集所有数据节点的结果,不做额外的计算操作;最后,对于涉及多个数据节点且不能垂直分片的查询,提供数据节点自协同的流式计划生成技术,允许数据节点之间交换数据,协同计算. 所有计算尽可能在数据节点上并行执行,既可以让数据和计算在同一台机器上,减少网络传输,又可以增加并行度,避免协调节点成为单点瓶颈. 上述所有模式的识别,都是自动完成的,不需要用户参与.

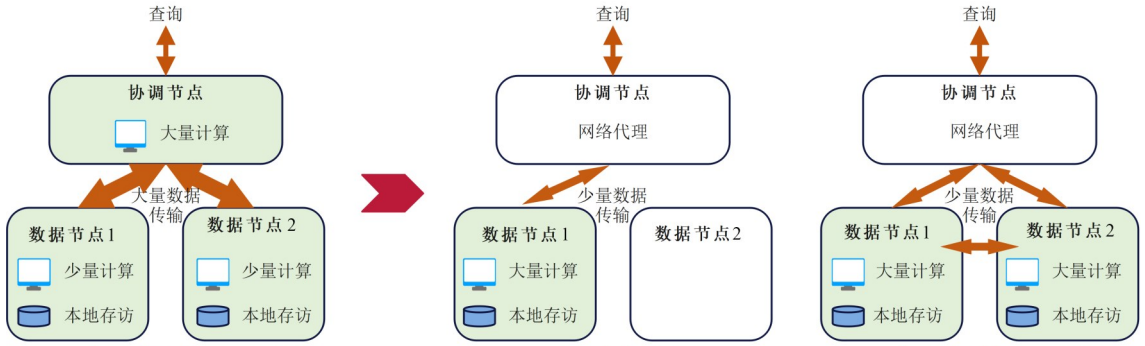

图2　GaussDB分布式优化技术

(1)协调节点旁路技术

协调节点旁路技术指的是当查询所有涉及到的数据都在一个数据节点上时,把整个查询的执行都下放到数据节点上,包括解析、优化、执行,协调节点退化成一个网络代理. 相当于单机数据库的执行流程,把协调节点上的开销降到最低,性能达到理论上限. 典型场景如TPCC中的单点查询、单点插入.

该技术的创新点在于自动识别出满足协调节点剪枝条件的查询. GaussDB提供了一种下推算子识别技术完成单分片特征的识别. 该技术会对语法树进行自底向上递归分析. 例如,对于基表,根据查询条件,来判断是否可以在某几个节点上独立执行;对于多表连接,会根据连接条件、内外表分布键来综合判断,连接操作能否下推到各个数据节点上独立执行. 最终,分析出整条查询是否可以下推到数据节点上独立执行.

该技术第二个创新点在于协调节点动态路由. 现有数据库普遍支持计划缓存,对于格式相同、参数不同的查询,解析和优化只做一次,产生计划并缓存. 然而分布式数据库中,不同的参数会导致数据存在不同的数据节点上. GaussDB通过动态路由技术来支持参数绑定. 当识别出某条查询满足协调节点剪枝的条件时,会自动产生一个轻量级代理对象,并抽象出查询条件中的表达式. 在执行阶段,根据表达式计算出数据所属的数据节点,再把请求路由到对应的数据节点上.

(2)数据分布感知的查询转移

查询转移指的是查询的数据在多个数据节点上,但是可以把计划下推到各个数据节点上执行,然后在协调节点上汇总. 在执行期间,数据节点之间不需要进行数据的交互. 这种技术比较适合完美分片的场景,例如,只查询单表,或者查询多表连接时,分布键和连接条件是一致的.

GaussDB自动识别出满足查询转移条件的查询,用到的技术和协调节点剪枝技术类似,需要用下推算子识别技术来判断是否可以完美分片. 同时也需要动态路由技术支持动态参数绑定.

(3)数据节点自协同的流式计划生成

对于不能完全下推到数据节点上的查询,传统分布式数据库如PGXC,从数据节点上获取原始数据,然后把计算过程放到单个协调节点上. 即PGXC计划以协调节点为中心,只有协调节点-数据节点之间的通信. 这种计算模式很容易使得协调节点成为系统瓶颈,并且引起大量的网络数据传输.

GaussDB为了支持各种场景下的近数计算,引入数据节点自协同的流式计划,支持数据节点-数据节点之间的数据交互,尽可能把计算都在数据节点上完成.即使涉及到跨多个数据节点的算子,也可以在数据节点之间交互数据之后,继续在数据节点上并行执行,提高整个系统的并行度.流式计划在TPC-H、TPCDS等测试集中比传统的PGXC计划快10倍以上.

首先,GaussDB抽象了三个基础算子,如图3所示.

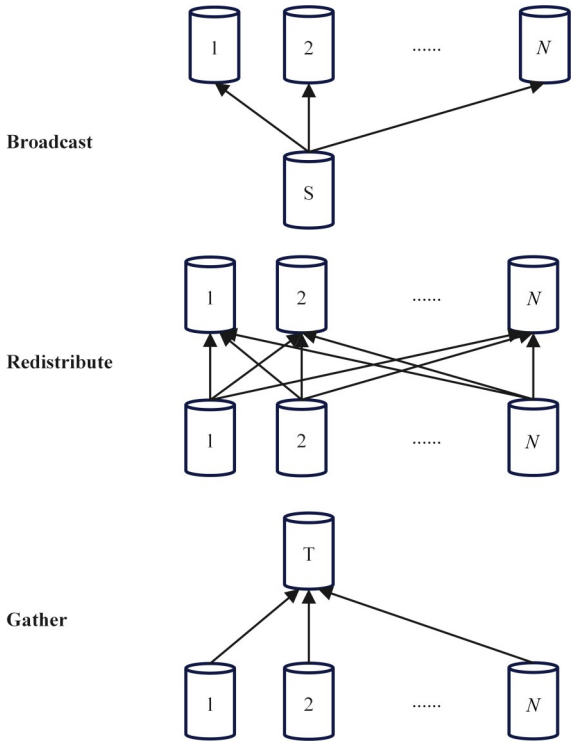

图3 GaussDB分布式优化流式算子抽象

(a)Broadcast:广播算子,用于点查询中将某个数据节点上的数据发送到其他多个节点进行连接.

(b)Redistribute:重分布算子,用于将数据按照键值重新分布到各个数据节点上.

(c)Gather:汇聚算子,一般用于将各DN处理完的数据汇聚到CN进行处理.

通过这三个基础算子,基本可以涵盖绝大部分分布式计划的场景.比如,广播算子产生类似于复制表的效果,某个表经过广播算子之后和其他表的连接就可以下推到各个数据节点上执行.

其次,GaussDB会根据每个查询,产生各种可能的流式计划路径.流式路径是按需产生的.比如,当两个基表做连接的时候,两个表的分布键并不在连接条件上,这时候,需要在某一个基表的扫描路径上,加一个流式算子(广播算子或者重分布算子);经过流式算子的数据转移之后,满足连接条件的数据会被汇聚到相同的数据节点上,那么连接操作可以下推到各个数据节点上执行.

最后,GaussDB构建了基于流式算子的代价模型.流式计划的生成是代价可感知的.通过比较代价选出最优的计划.

3.1.2 全链路并行编译执行技术

面向企业业务的复杂查询,生成最优执行计划是远远不够的.如何根据最优的执行计划,高效地完成复杂查询的执行和计算,也是分布式数据库当前面临的一大挑战.分布式执行的效率直接决定了分布式数据库系统在应对复杂场景的表现.如图4所示,在执行过程,为了更好地发挥多节点算力和资源,构建了一整套分布式并行执行框架,从节点间并行、节点内并行到指令级并行,充分发挥多机、多核硬件优势和特点,提升执行流程效率.为了进一步提升系统性能,采用编译技术,提高CPU指令的利用率,从而进一步提高系统整体面向复杂业务的吞吐.

(1)节点间和节点内全并行执行

一般对于企业的复杂查询而言,往往一个查询包含若干张表的关联,多个聚集操作以及排序操作等查询算子,使用传统的单线程模式意味着一个查询会执行几十分钟,甚至几小时.为了提升这类复杂查询的执行性能,且充分利用计算资源,GaussDB除了实现不同实例节点之间的分布式并行执行框架,还实现了单个节点内的多线程并行执行框架,从而形成完善的全并行执行框架,解决复杂查询的性能瓶颈问题.

GaussDB的并行执行框架主要包含两个部分:并行计划生成和并行计划框架.

并行计划生成的基本原理是在生成每一层路径时,都会增加一个并行路径,最终根据代价选择最优路径.选择的路径可能是节点上的算子全部并行、全部串行或者部分并行部分串行,这是因为在不同情况下对不同算子,有可能串行最佳,也有可能并行最佳.

并行执行框架主要处理不同节点和不同线程间的数据分配.数据分配原理如图5所示.数据以两种方式跨DN节点分配:数据在各个节点所有并行线程之间,按照HASH值进行重新分布,称为分散重分布(split redistribute);数据被广播到所有节点的并行线程之中同时处理,称为分散广播(split broadcast).数据以三种方式在DN内进行分配:在DN节点内,按照HASH值进行数据分布到多线程,称为本地重分布(local redistribute);数据被广播到DN节点内的并行线程之中同时处理,称为本地广播(local broadcast);并行线程中的数据汇集到一个线程中处理,称为本地聚合(local gather).

GaussDB的并行计划目前针对scan、join、append以及agg做了代价评估,其他的算子保持与下层算子相同的并行度.

对于scan算子,仅增加一条并行的路径,计算并行

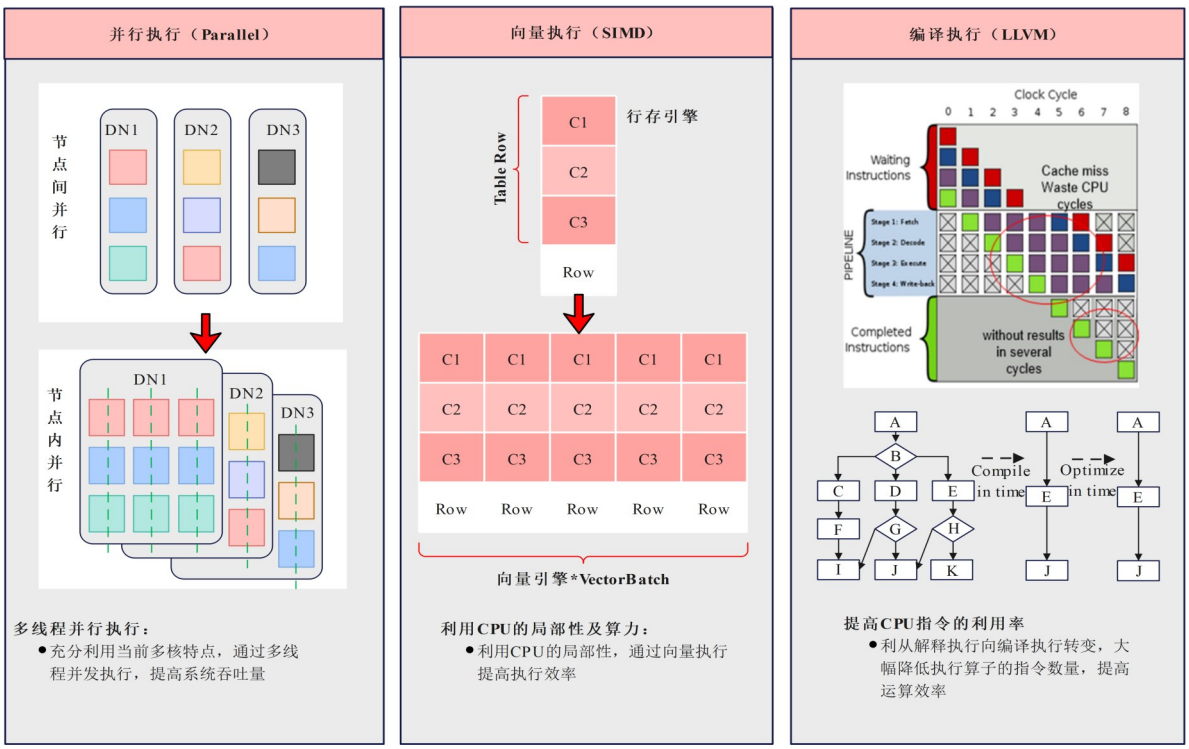

图4　全链路并行编译执行技术架构

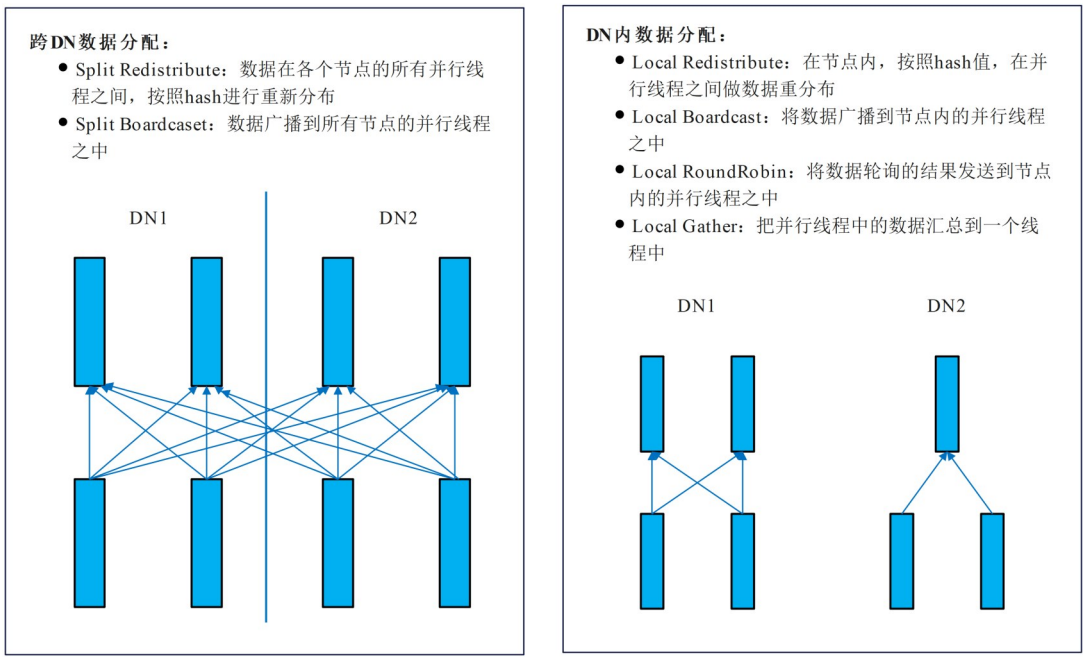

图5　GaussDB数据分配原理

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.

Join算子在打开并行查询的情况下,每次会生成两类路径,一是串行路径,一是并行路径. 串行路径生成比较简单,相比没有并行查询的差异仅在于下层算子可能是并行的,此时需要在中间增加一层本地聚合算子,将并行转换为串行. 并行路径的生成主要分成两个部分,一是判断是否左右子树需要重分布,一是并行路径的生成.

是否重分布主要取决于DistributeKey和Join条件,如图6所示. 如果DistributeKey为NULL,说明下层算子的数据没有做过数据划分,所以需要重分布;而如果DistributeKey和Join条件不相同,说明下层算子的数据分布与需要做Join

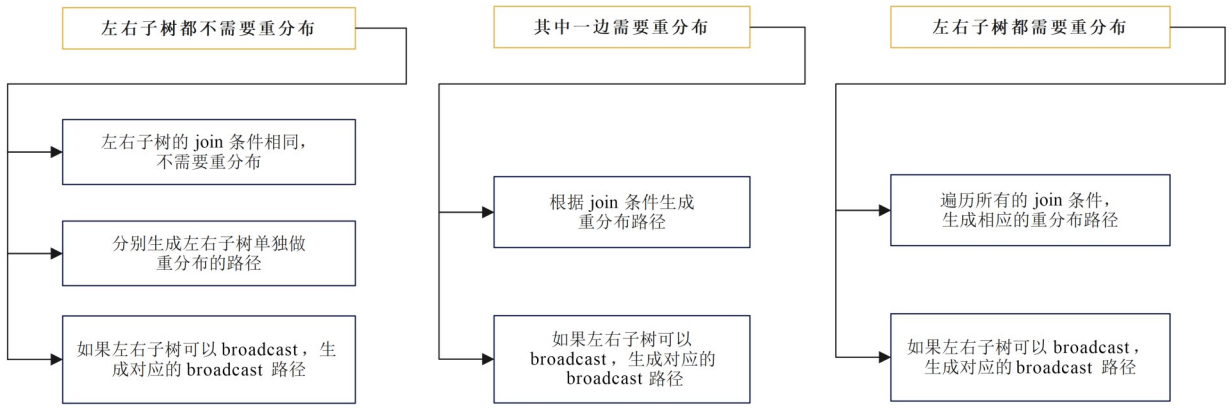

图6 GaussDB查询子树重分布

的列不相同,同样需要做重分布,确保相关的数据都能够集中到一个上层算子线程中,避免结果错误.

Agg算子并行采用双层Agg的方式来实现. 由于Agg算子目前是基于规则来生成的,所以Agg算子的并行依赖于下层算子是否并行,如果下层算子没有并行,则Agg算子也不选择并行. 而当下层算子并行的时候,才对Agg算子做并行操作. Agg算子的并行有两种方式:(a)直接对下层算子返回的结果做重分布,再在各个线程中做Agg;(b)先在各线程做一次Agg,消减数据量,然后对结果做重分布,最后在各个线程中做一次Agg. 具体选择哪种方式,取决于代价的计算. 这两种方式执行完成后,都需要将并行线程的Agg结果做一次数据的汇总,获得最终的Agg结果.

GaussDB将查询计划中Stream算子拆分成多个执行片段,每个片段分配到不同的线程中,并以同时执行的方式实现并行计划执行. 而不同线程间的数据分配通过StreamProducer和StreamComsumer来实现. 其中StreamProducer负责将下层算子的执行结果发送到对应的上层算子线程中,而StreamComsumer负责接收下层算子的执行结果,给本线程的算子进行计算.

对于节点间的数据分配,通过网络通信的方式进行数据传输;而对于节点内的数据分配,则直接通过内存进行传输. 下层算子的线程会将一批执行结果填充到一个消息队列发送到上层算子,而上层算子接收到这批数据后会执行自身的逻辑,构成一个消费者和生产者模型,各自执行自身的逻辑.

GaussDB在并行执行框架的基础上,还实现了自适应的SMP框架. 通过读取服务器的CPU及内存情况,自动选择合适的并行度. 如图7所示,SMP自适应分为两个阶段,第一阶段确定初始dop,第二阶段对基于初始dop生成的计划进行优化. 在第一阶段考虑CPU资源、串行还是并发. 在第二阶段考虑计划复杂程度.

(2)LLVM动态编译Codegen技术

在纯软件方面,提升软件执行效率有如下方法:

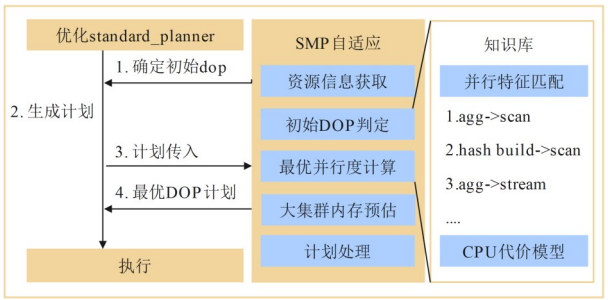

图7 GaussDB并行执行SMP框架

(a)改进并重新实现对数据库执行效率起重要作用的算法;(b)改进并重新实现热点函数;(c)LLVM动态编译技术等. 前两种优化手段如果应用得当,可以取得重要成效,但一般都按个案处理,难以形成可批量复制方法. LLVM动态编译技术则与之不同,属于一种通用的性能优化手段. GaussDB引进LLVM动态编译技术,对执行流程中的热点函数做动态编译,消减查询执行过程中的冗余逻辑、函数调用.

GaussDB的Codegen分为三个层次:LLVM Lib库、LLVM API层、LLVM实现层,如图8所示. 其具体表现如下.

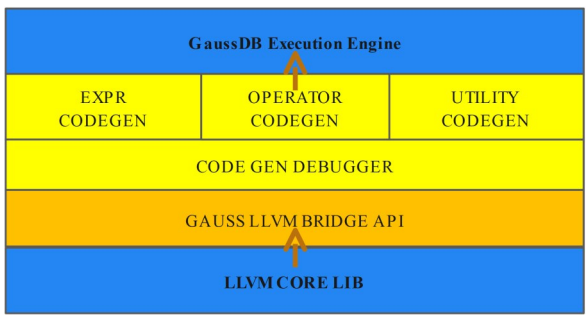

图8 GaussDB编译执行框架

LLVM Lib库:LLVM第三方开源软件静态库,提供各种代码生成所需的功能接口.

LLVM API层:负责封装部分LLVM Lib库函数,实

现优化功能供LLVM实现层调用.

LLVM实现层:根据查询完成对执行计划中所涉及的指定函数的LLVM优化并返回IR函数.LLVM实现层中实现的功能与原有执行器中的功能相对应,即选择LLVM优化或不选择同一查询所获得的结果是一致的.

以图9所示"SELECT id, name FROM stu WHERE age < 18 OR age > 60;"为例,在计算where表达式"age < 18 OR age > 60"时,会将表达式分解为布尔判断(BoolExpr)、操作符(OpExpr)、列引用(Var)、常量(Const)等子节点,而OR类型的布尔判断及">"、"<"操作符都包含两个参数,参数通过函数指针引用.并且为了通用性,每个节点的通用处理函数需要能够处理该节点的所有情况,例如列引用(Var)需要处理引用左表的列、引用右表的列、引用Scanned表的列等各种情况,但是当一个Var节点生成以后,其具体引用的是哪个表的哪一列是固定的,在执行过程当中并不会发生变化.因此在每个节点的通用处理函数内部,都存在一些执行过程当中不会发生改变的冗余条件逻辑.

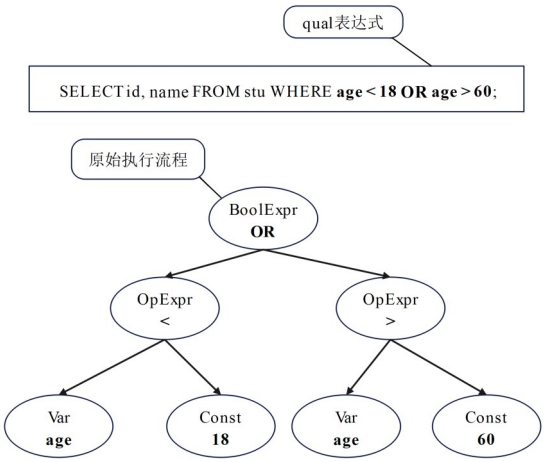

图9　GaussDB解释执行流程

在编写每个表达式节点的通用处理函数时,因为无法预知每个节点的参数类型及调用关系,因此只能在运行时使用函数指针这种灵活的方式来指定调用关系.而Codegen技术提供了在运行时重新编写代码的机会,当要执行一个具体的表达式时,每个表达式节点的参数类型及调用关系都已经固定.还以where表达式"age < 18 OR age > 60"为例,如图10所示.当执行布尔判断OR时,它的左参数是操作符"<"的计算结果,右操作符是">"的计算结果,因此编写Codegen代码时,可以固定这些调用关系,不再需要使用函数指针.因为每个表达式节点的执行顺序也不会发生改变,因此可以把节点间的调用写成inline来减少调用栈的深度,并在编译生成定制化机器码时,对整个表达式的执行过程做一个整体优化,从而提升表达式的执行性能.

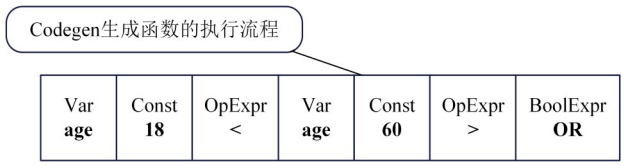

图10　GaussDB编译执行流程

3.1.3　大规模分布式事务并发处理技术

对于一个分布式数据库来说,数据采用分片技术打散到不同的数据节点,因此不可避免地会出现业务跨节点的数据处理操作.如何确保不同分片间的数据操作维持数据库的ACID属性,保证跨分片数据一致性,成为了分布式数据库的一大挑战.解决这一问题的通常做法是采用分布式事务处理技术,而当前的分布式数据库系统的分布式事务处理普遍存在中心点瓶颈,原因是分布式事务处理需要一个全局的快照,一般而言,目前大部分的数据库都采用一个单独的事务管理器组件来解决全局事务快照问题.由此,事务管理器(GTM)的性能好坏,就直接决定了整个分布式数据库的规模大小,以及分布式数据库可达到的性能上限.本文创新提出了GTM-LITE技术,简化分布式事务快照,解放GTM,使得GTM的快照生成和发放能力,呈倍数提升,进而达成分布式数据库性能可随着集群规模近线性扩展的目标.尽管GTM-LITE技术已经把快照内容减少到了极致,但依然是一个中心点瓶颈,无法应对大规模数据库集群的部署要求.为了全局快照的全局可见性和去中心化,提出了跨地域的高精度时钟技术,通过物理时钟的对齐,来解决快照的去中心化问题.进一步地,通过跨地域时钟的技术突破,实现了分布式数据库的全球化部署.

(1)GTM-LITE分布式事务处理

GTM全称为全局事务管理器,主要作用是分发XID(事务编号)、Snapshot(事务快照)、Sequence(序列)等信息辅助分布式事务的执行.为了保证事务标识的一致性和全局唯一性,在集群中只有一个主GTM提供服务.一般情况下,CN会在事务执行阶段连接GTM获取XID和Snapshot.GTM在架构中的关系如图11所示.在一个事务的执行流程中,CN会与GTM进行多次交互,如事务开始时在GTM注册槽位、获取快照时从GTM获取全局事务ID、事务结束时向GTM提交并移除槽位等.GTM同时会向DN节点同步全局事务ID,处理来自DN节点的VACUUM请求.

现有分布式数据库在大并发TP场景下,控制一致性的中心节点全局事务管理器GTM容易成为单点瓶颈,影响集群整体吞吐,增加数据库对客户端的响应时延,影响客户整体体验.

GTM-LITE方案在保证原有高可用,强一致性,支持事务ACID特性的基础之上,最大限度地优化GTM的处理

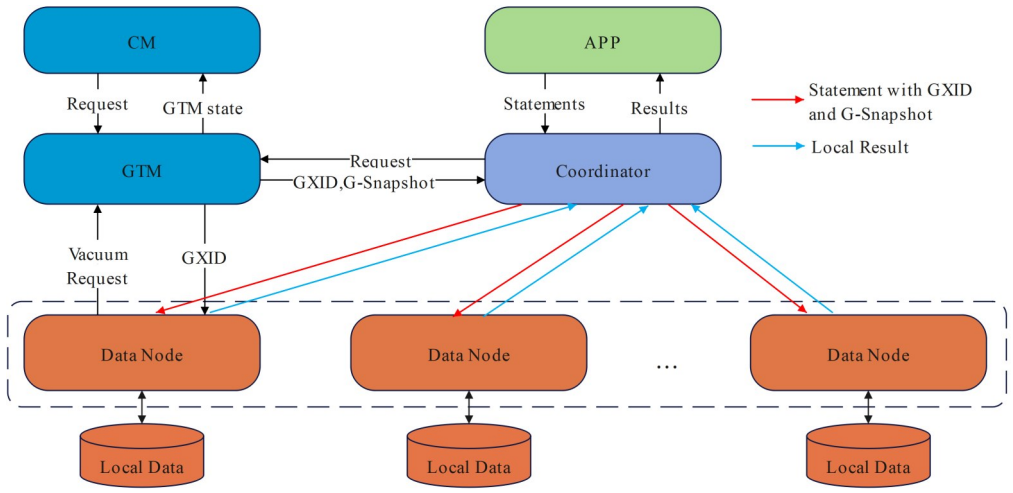

图11 传统分布式数据库架构

能力,支撑大并发连接,在百节点大集群环境下,GTM-LITE能实现集群性能的近线性扩展,同时解决相关的高可靠、高可用问题.

相对于GTM,GTM-LITE技术有如下优化:

(a)网络带宽的优化,取消系统的集群快照,改为逻辑时钟来判断事务的集群可见性,大幅减少对GTM的网络带宽的占用,同时还降低了GTM的CPU占用.

(b)CPU使用率的优化,通过线程资源复用的方式大大减少GTM的线程数据,减少系统调度CPU占用率,大幅地提升GTM的处理效率.

(c)系统锁的优化,在系统吞吐量达到百万级时GTM原来使用的系统互斥锁占用了绝大多数的CPU,编写了用户态的互斥锁,使得CPU使用率只有原来的十分之一,提升了系统的处理能力上限.

当前GTM架构下,GTM面对如下两种场景会成为瓶颈.

①由于当前每有一个活跃的事务,就会在GTM的全局活跃事务数组中注册一个槽位,直到事务结束,该槽位才可以释放,目前该槽位的最大值设为16 384,当业务有大量并发的时候,GTM容易成为单点瓶颈,并发超过16 384时候,还会报出槽位数不足,影响业务.

②在一个事务执行的流程中,CN同GTM会有多次交互,其中包括:(i)BEGIN,向GTM注册槽位;(ii)GetSnapshot,向GTM获取事务ID;(iii)GetNewTransactionId,向GTM获取全局事务ID;(iv)END,向GTM提交事务并移除槽位.

这些交互带来的大量网络通讯和等待也会成为瓶颈.

GTM-LITE目的是缓解GTM带来的瓶颈问题,设计思路就是通过采用本地XID、全局CSN来消除GTM中的槽位,避免事务执行过程中CN同GTM的大量交互请求. 图12展示GTM-LITE的架构设计,其主要设计思路如下.

(a)本地事务ID取代全局事务ID. GTM不再分配全局唯一的事务ID,每个CN/DN节点用本地产生的事务ID,保证节点内事务ID不会重复;对于跨节点的事务,由全局唯一的GID标识符前缀来保证写一致性,由全局唯一的CSN号来保证事务读的一致性.

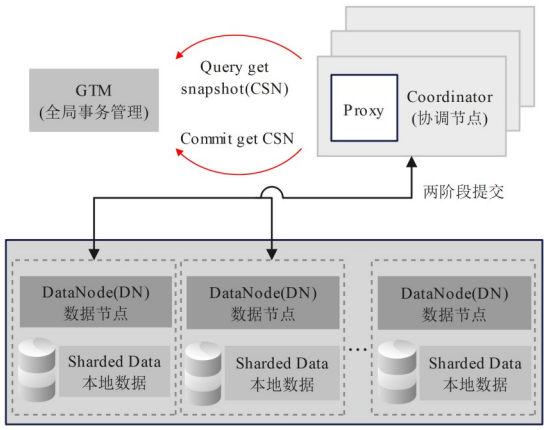

图12 GTM-LITE架构图

(b)GTM不再维护槽位信息,仅在事务提交时下发全局唯一的CSN序列号. GTM下发的全局CSN是一个递增的uint64值. 这一设计消除了BEGIN与GetNewTransactionId同GTM的交互. 进一步,如果事务在GTM提交时失败,可以retry重新获取最新的CSN,减少网络故障对系统造成的影响.

(c)本地维护多版本过期脏元组的回收,并引入Snapshot Invalid机制,保证全局事务的一致性. 由于舍弃全局事务ID,无法直接根据RecentGlobalXmin确定需要清理的脏元组,所以需要利用全局CSN来计算RecentGlobalXmin,从而实现GTM架构的有效复用.

(2)基于高精度时钟的分布式事务处理技术

GTM-LITE属于集中式授时,由于性能原因没有全球化部署的解决方案,否则与GTM节点的通信会消耗大量的时间从而导致性能下跌严重. 将高精度时钟同步技术融合到GaussDB中可以实现分布式授时,从而使GaussDB全球化部署成为可能. 全球化部署的数据库有

更强的数据可用性,在区域故障后不会丢失数据,也可以根据访问频率动态分配数据以搭配用户,并在用户切换位置时出于性能或监管原因重新分配数据,更好地支持不同地点的大量用户. GaussDB的全球化部署方案如图13所示,以三个城市为例,其中每个城市都有一个主分片和其他城市的副本分片. 例如,对于分片1,主分片在北京同时两个副本分片分别在西安和深圳. 对于分片2,主分片在西安同时两个副本分片分别在北京和深圳.

GTM-LITE的集中式授时是每个节点都需要访问GTM节点才能获取事务的全局一致序列号(即CSN),相比GTM方案缓解了GTM节点的瓶颈问题,但是并不能彻底消除. 分布式授时则可以彻底消除中心节点的瓶颈,主要设计思路如下.

(a)每个GaussDB节点需要开启一个后台线程,用来与DC内的Time Server节点定时通信来同步本地时钟.

(b)每个GaussDB节点都可以直接读取本节点的64位时间戳作为事务的全局一致序列号,不再需要与GTM节点进行通信.

3.2　分布式数据库高可用容灾技术

数据库服务的高可用、高可靠乃是企业业务平稳运行的核心要素之一,对于企业核心系统来说至关重要. 一般来说,分布式高可用容灾设计的核心逻辑就是充分利用分布式多副本的一致性协议原理,结合各类故障场景和意外的情况,设计和构建适应于不同场景的高可用解决方案,以满足不同场景下企业核心业务对于数据库高可用容灾的关键诉求.

GaussDB从实际重点行业核心系统诉求角度出发,创新设计了多副本高可用、两地三中心、双集群高可用、跨区域多活等技术,全面提升分布式数据库的高可用能力. 关键技术、场景如表1所示.

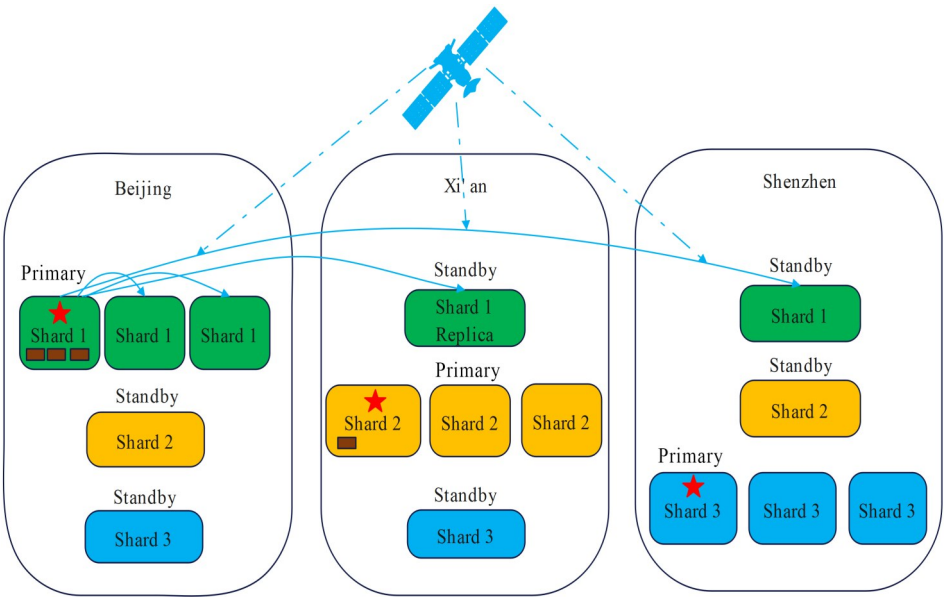

图13　全球数据库的部署

表1　GaussDB高可用容灾关键技术

容灾级别	容灾方案	故障场景	关键技术
同城	同城3AZ	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	同城双集群RPO=0
异地	两地三中心	节点、机房级、城市级故障	双集群流式复制技术
	异地双活	节点、机房级、城市级故障	并行逻辑复制技术

3.2.1　故障自感知的多副本高可用技术

传统的主备架构大多依赖于第三方仲裁组件,故障判断链路长,无法快速对故障做出响应和仲裁. GaussDB提出了基于Paxos协议的自感知副本一致性共识复制以及仲裁技术,如图14所示,实现了在故障模式下数据节点自选主. 提出了策略化多数派以及节点优先级选主技术以便于用户在不同的部署模式下选择合适的选主仲裁策略. 为了应对常见的网络亚健康状态,提出了基于网络流量和网络时延自适应的流控算法,根据日志复制的数据流量特征信息,实现对日志流量的精细控制,保证数据库业务的稳定性.

3.2.2　同城双集群RPO=0技术

GaussDB提出同城双集群系统保障业务连续性的解决方案. 存储设备采用了华为自研的OceanStor Do-

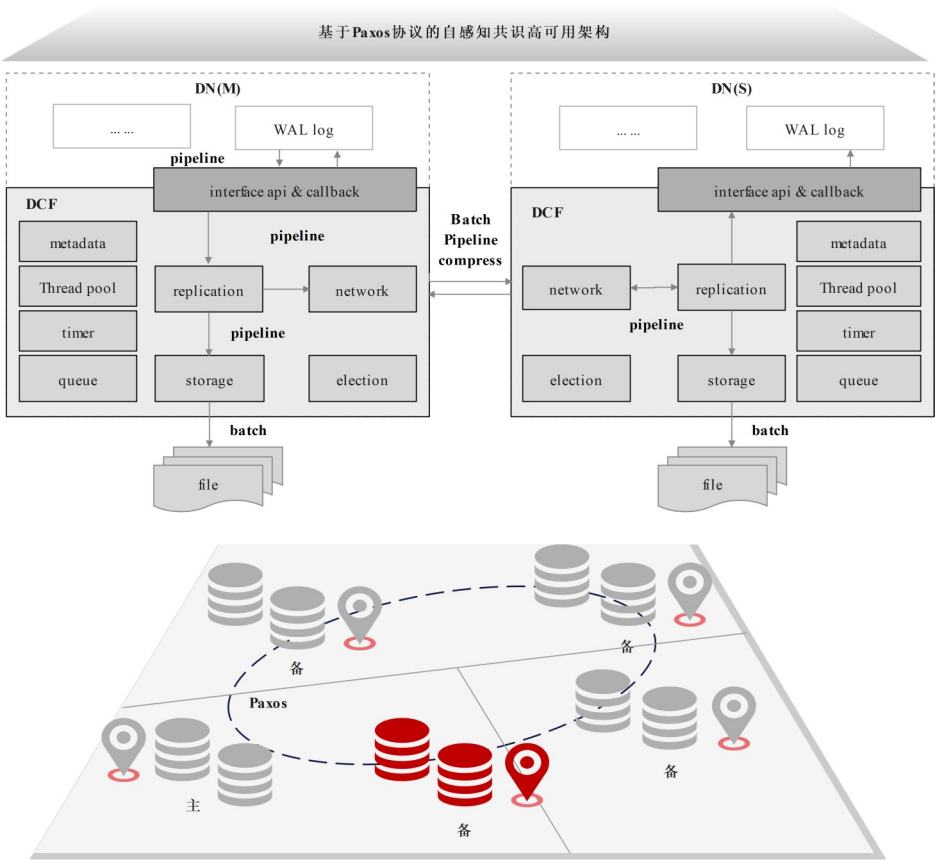

图 14 基于 Paxos 协议的自感知共识高可用架构

rado V6 全闪存存储系统，具有远程并行复制数据的功能，利用 Dorado 独有的网络协议（密集波分），极大降低网络通信时延，可以在异地的存储系统之间实现同步或异步的数据复制，提供端到端低时延、大吞吐量的稳定高性能能力．支撑存储应用构建同城的高可用解决方案．因此，本方案基于 Dorado 存储设备，来保证数据的同步和一致性，构建支持 RPO=0 的双集群高可用系统．

主备集群间通过日志来做数据同步，主集群的主节点将日志写入主集群侧的 Dorado 盘上，利用 Dorado 的远程复制功能复制到备集群的 Dorado 盘上，备集群再从这一侧的 Dorado 盘上读取日志进行恢复．

主集群的主节点产生日志，由日志写线程将日志从本地拷贝到 Dorado 盘上．主节点不再发送日志到备节点．备节点从 Dorado 盘上读取日志到本地．

对 Dorado 裸设备以直接 IO 方式读写，提升读写性能；日志存储区域循环进行读写，节省存储空间．主机将日志写入存储设备成功后，即可认为日志已复制到备集群，提交事务．防止共享存储设备异常场景下发生快照回滚等现象，备机在从存储设备上读取日志时，会复制到本地一份并进行日志回放．

日志共享存储解决了主备节点间数据的一致性问题，从而保证了 RPO=0，RTO < 60 s 的高效可靠的集群切换目标．

3.2.3 细粒度无锁并行回放技术

数据库通过流复制，将主机业务中生成的日志不断地从主机发送给备机，用于同步相应的数据，并在备机回放每个日志．主机对外提供读写服务，备机可以对外提供读服务．主机故障后，备机需要进行故障倒换，故障倒换后备机升主机，开始对外提供服务；主机未发生故障，因为负载均衡等原因进行主备倒换，主备倒换后备升主，开始接管业务．

故障倒换/主备倒换流程：①备机回放日志；②备机升为主机．RTO 取决于待回放的日志量以及 I/O 能力；如果日志量很大，可能需要几分钟至几十分钟．RTO 时间过长是由于主机运行速度过快，产生日志过多，导致给备机发送的日志来不及回放．在系统长时间的运行后，备机上会出现日志累积．备机升级成主机，也需要很长时间进行日志回放，这段时间系统是不能对外提供服务的，会影响系统可用性．

当数据库运行一段时间后，关闭数据库再重新拉起，需要将数据库中已存在的日志回放完才能对外提供服务．若日志回放速度慢，数据恢复需要很长时间，导致数据库很久不能拉起．因此，需要提供极致的日志回放，减少日志积累．

基于现有的并行回放方法,进一步提高日志回放并发度,解除同步瓶颈,最大地加快并行回放速度,整体架构如图15所示.该方案利用流水线分级处理日志,通过解析成页面为粒度的日志,减少线程间同步,通过批量回放提高日志回放并发度.能够缩短主备机切换时间,在备机升主机时秒级内完成日志回放接管主机业务,保证业务连续性.图15中页面为粒度的日志被批量处理,页面1~页面3的回放形成流水线.

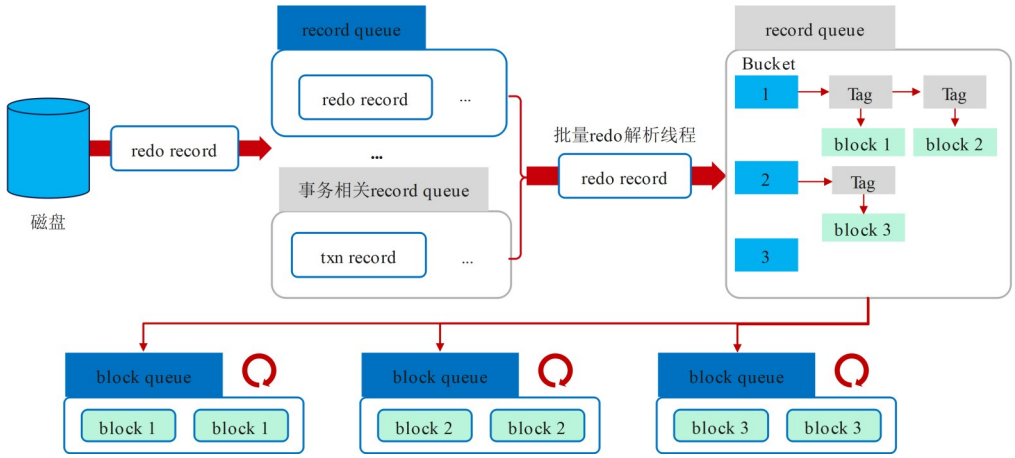

图15　细粒度无锁并行回放技术架构

3.3　云原生计算存储分离和弹性伸缩技术

数据库技术发展多年,分布式数据库逐渐成为绝大部分应用的业务数据存储解决方案.目前市场上绝大多数数据库(OceanBase[6]、TiDB[7]等)均使用无共享架构,但随着互联网大数据时代的来临,越来越多的业务需要应对流量突然激增的挑战,数据库扩容的需求越来越频繁.当前主流的架构需要在扩容时对计算和存储资源同时扩展,用户数据需要通过网络进行数据同步,对资源和时间的消耗巨大.围绕当前主流分布式数据库面临的成本压力和弹性挑战,结合云计算基础设施的特点,创新提出了云原生分布式的计算存储分离架构,实现了计算资源、内存资源[8]以及存储资源[9]的分层管理,通过高速网络链接计算资源池和存储资源池.分层管理的架构使得成本得到了更加精准的控制,以及各层资源得到更加充分均衡的利用.通过哈希桶聚簇存储实现了文件的细粒度分布,因为存储共享,因此在过程中并不需要真实转移数据,只需要搬迁映射元信息表即可,而因为元信息本身可以实现快速迁移,因此可以实现快速高效弹性扩展.为了进一步利用池化资源,提出云原生的分布式多写多读,构筑分布式备机一致性读能力,使得读写负载得到均匀分布,资源得到合理化、最大化利用,节省数据库的使用成本.

3.3.1　云原生分布式计算存储分离技术

传统的数据库不仅写日志,还写数据以及备份数据文件,这导致出现写放大问题.日志即数据技术将日志看作主要数据源存放在共享存储层,计算节点只将日志写到存储层,不刷脏页;存储层异步地回放页面数据用于计算节点读取访问最新数据.另外,由于日志一直在存储层异步地被回放最新数据,以ARIES算法为主的恢复过程不再需要特定的恢复过程,因此日志即数据技术也简化了数据库的恢复过程.当前主流的云原生计算分离架构主要可以分为三类:①计算-存储分离;②计算-日志-数据分离;③计算-内存-存储分离.其计算层的节点不做数据持久化(不写脏页与建立检查点),只写日志,然后日志的持久化与数据同步操作全部被卸载到存储层处理.第1类系统(如Aurora[10]、AlloyDB[11])采取两层分离架构,数据写入路径是主节点接收数据更新查询并生成对应的重做日志,将重做日志写到多数派存储节点.存储节点内同时存储重做日志以及记录页面,并异步完成日志数据的回放.第2类系统(如GaussDB)同样采取两层分离架构,但其在存储层进一步将管理日志的节点从存储节点中分离,优化了写入路径.其日志节点记录事务产生的数据变化,存储节点则保留最新页面数据,服务于读操作.第3类系统(如Socrates[12]、PolarDB[13]等)采取三层的分离架构,其在计算层和存储层之间增加了基于高速网络技术(如远程直接内存访问)的共享缓存层,实现了计算节点之间的缓存共享机制,因此计算节点可快速从共享缓存里读取最新页面.

除了计算存储分离方式存在差别以外,GaussDB创新提出了云原生分布式计算存储分离架构,如图16所示.该创新架构相比于业界其他分布式数据库(如Aurora[10]、PolarDB[13])有三大创新优势:①分布式数据库结合分布式存储,实现数据预分片功能,具备秒级扩容能力;②计算与存储解耦,实现计算池和存储池独立扩容功能;③存储节点实现日志回放功能,大幅减少计算节

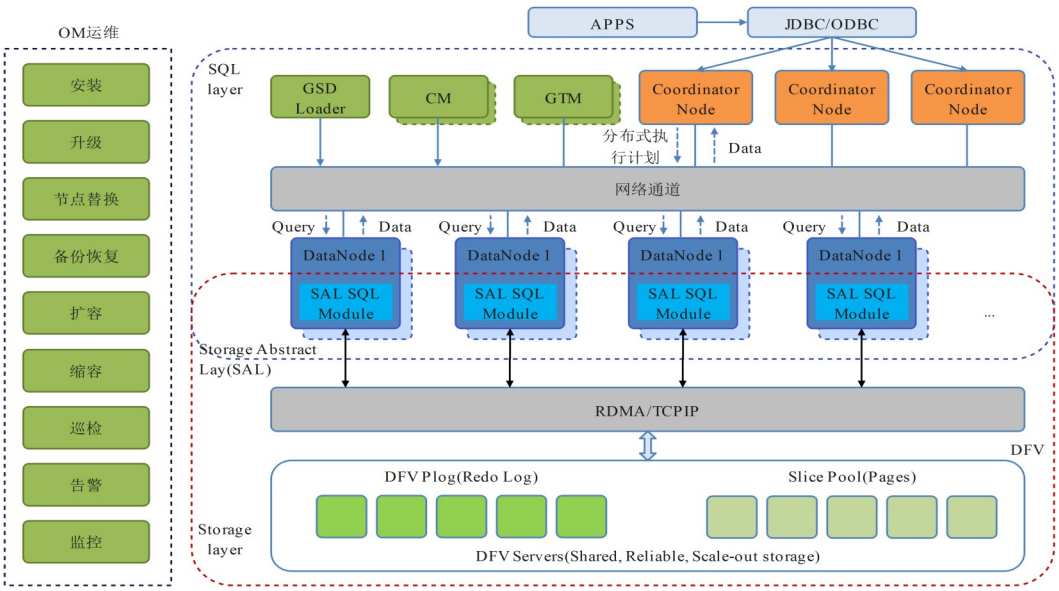

图 16 GaussDB 云原生分布式架构

点和存储节点直接带宽需求,同时减轻计算节点刷脏压力.

3.3.2 在线弹性扩展技术

系统扩容过程的本质是集群内众多本地表的节点组之间的数据搬迁,扩容过程指较小的节点组向较大的节点组进行数据搬迁,反之则看成是缩容过程. 对于每个扩容重分布的表而言可以看成是基线数据扩容重分布、增量数据重分布、表切换三个过程.

GaussDB 通过引入哈希聚簇存储,降低数据重分布过程中数据的扫描量和搬迁量,以及数据重建的规模. 从而缩短重分布流程的整体时间,同时降低对用户在线业务影响的时间窗口.

为了支持高性能扩容,对目前表(普通表、range 分区表)存储形式改为哈希桶聚簇存储. 即把当前表拆分为多个分片的形式存储——每个表切分为 BUCKETLEN 个分片,每个数据节点存储 BUCKETLEN/数据节点个分片.

如图 17 所示,假设该集群有 2 个数据节点,分别为数据节点 0、数据节点 1,在配置集群时选定 BUCKETLEN=8(为了示意方便,选定了较小的 BUCKETLEN,实际 BUCKETLEN=16 384),那么表 t1 将会打散为 8 个分片,分别为 0~7. 数据节点 0 存储分片 0、2、4、6,数据节点 1 存储分片为 1、3、5、7. 一致性哈希算法生成的哈希桶映射为[0, 1, 0, 1, 0, 1, 0, 1],即可把待插入数据映射到数据节点 0/数据节点 1 上对应的分片.

分片的优势在于扩容之后,只需要重分布少量的文件即可实现扩容. 如图 18 所示,在新加一个节点数据节点 2 之后,一致性哈希算法生成[2, 2, 0, 1, 0, 1, 0, 1]的映射,表示下标 0,1 的数据存储在数据节点 2 上. 这时只需搬动 P0、P1 两个分片即可完成集群扩容.

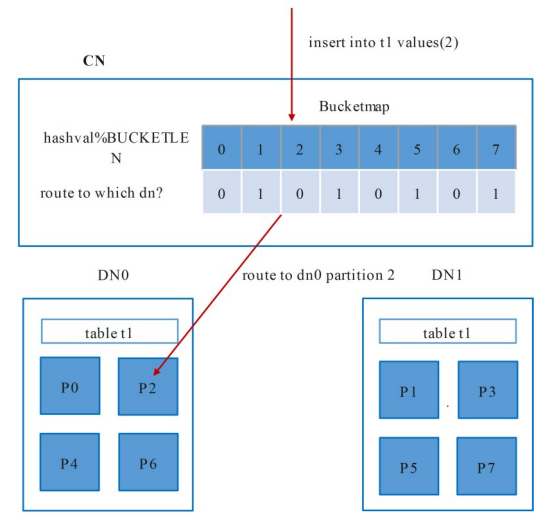

图 17 GaussDB 哈希桶聚簇存储示意图

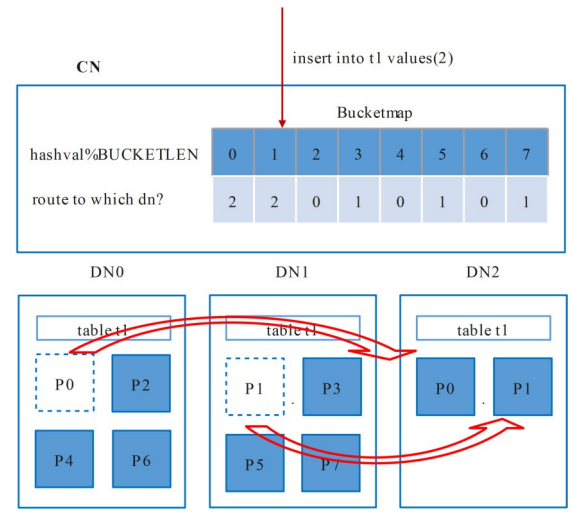

图 18 GaussDB 集群扩容后分片变化情况

通过映射表(元信息)的并行化迁移,实现扩缩动作的高效完成.

3.3.3 云原生分布式多写多读技术

分布式数据库写请求多CN同时接收处理,各数据节点处理写请求. 同时数据节点的备机也存有数据的分片可以用来支持多读. 分布式数据库由于数据存储在不同的备机节点上,每个节点回放速度不一致,为保证数据读取一致性,GaussDB借助组件全局事务管理器GTM保证数据的读一致性. 具体的实现流程如图19所示.

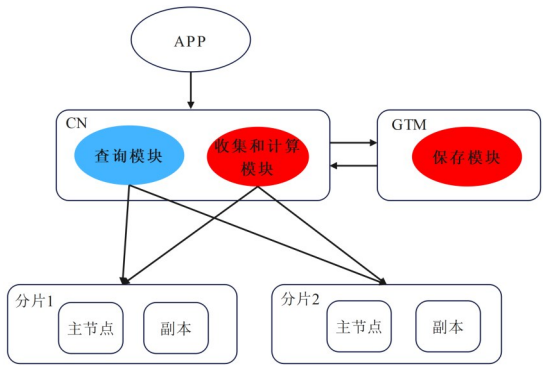

图19　GaussDB分布式备机可读技术原理

(1)首先通过句柄扩充与连接串中加入备机读字段,保证协调节点能与备节点(即图中副本)进行通信. 协调节点(即图中CN)通过收集和计算模块定时收集所有节点的回放进度并计算全局一致性点,然后发送到全局事务管理器.

(2)全局事务管理器通过保存模块负责保存全局一致性点,以及达到该点的备节点. 全局事务管理器收到协调节点发来的全局一致性点信息后,与当前的全局一致性点比较,比当前全局一致性点大才会进行更新,保证全局一致性点的递增推进.

(3)有读请求时,查询模块从全局事务管理器获得全局一致性点,以及达到该点的备节点,选择合适的节点进行查询.

分布式备机读收集集群所有节点的回放进度信息,并根据一定的算法计算全局一致性点,在时延较小的情况下充分利用冗余资源. 利用现有组件,保存全局一致性点的状态信息,并选择合适节点下发查询,保证读一致性. 通过将读请求下发到同一分片的不同备节点,实现会话级的读写分离,同时保证集群故障时RTO可控.

3.4 智能自治优化技术与实现

围绕数据库系统调优难、运维成本高的难题,GaussDB打造了智能自治管理体系[14]架构如图20所示,实现数据库自监控、自诊断、自调优[15]、自恢复、自安全等能力,大大提升运维效率,助力数据库用户迈向智能运维时代. 首先打造数据库内AI训练和推理,实现数据库智能化,设计库内轻量级原生AI引擎,在数据库侧提供AI算子,提供库内原生的AI算法,推动数据库与AI技术进一步融合. 其次,以优化器技术为突破口,打造生产级AI优化器[16~18],实现智能基数估计[19]、计划自适应选择[20]等技术,解决传统优化器代价估不准、计划选不对等痛点问题,构筑GaussDB自学习内核. 第三,打造DBMind自治运维平台,设计参数自动调优、索引自推荐[19,20]、慢查询自诊断、分布键自推荐等技术,实现数据库的自监控、自诊断、自恢复能力.

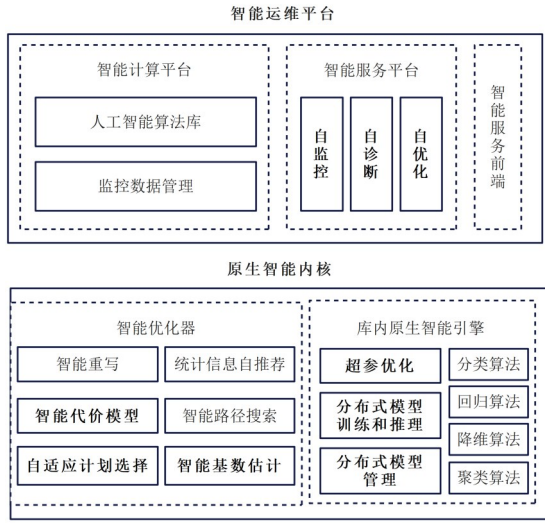

图20　智能自治管理体系架构图

3.4.1 AI优化器技术

AI优化器技术目标是在GaussDB中构建一套基于机器学习的代价引擎和计划管理框架,解决当前传统优化器面临的代价估算误差大以及自适应能力低下的问题. AI优化器利用GaussDB库内AI算法平台提供高效直接的机器学习能力. AI优化器核心技术点包括智能基数估计和自适应计划选择. 前者解决传统基数估计不考虑多列和多表相关性导致估计不准的问题,后者解决计划选择不准的问题.

(1)智能基数估计

数据库依赖基数估计来选择合适的执行计划,是数据库核心的模块. 但是传统基数估计往往依赖于不合理的独立性假设,导致基数估计不准. GaussDB内核中的智能基数估计方案是将轻量概率图模型融合进GaussDB的传统统计信息模块,在统计收集阶段进行模型训练,并且将模型保存在系统表中供优化器使用. 此方案相比于现有数据库的外挂模型训练方案[21,22],具有高安全和高性能的优势,高安全在于数据不需要导出数据库系统安全边界,高性能在于原生的数据获取以及模型训练和推理十分高效. 贝叶斯网络适用于数据列之间存在强相关性,数据分布不均,高频值数量较多的场景;当前的MCV适用

于高频值数量较少,低频值分布均匀的场景;函数依赖适用于需要利用列之间相关性进行多列选择率矫正,对于准确率要求不高的场景.

GaussDB智能基数估计架构如图21所示,主要涉及到统计信息分析器、库内贝叶斯网络算子以及查询优化器三个模块子系统.其中统计信息分析器首先针对数据表分布式水塘采样,利用采样得到的数据进行贝叶斯网络拓扑搜索和模型创建;模型创建阶段调用库内贝叶斯网络算子进行,包括模型训练、模型序列化和模型存储;查询优化阶段中优化器能够识别出多列谓词信息,并且利用库内贝叶斯算子进行谓词选择率估计.

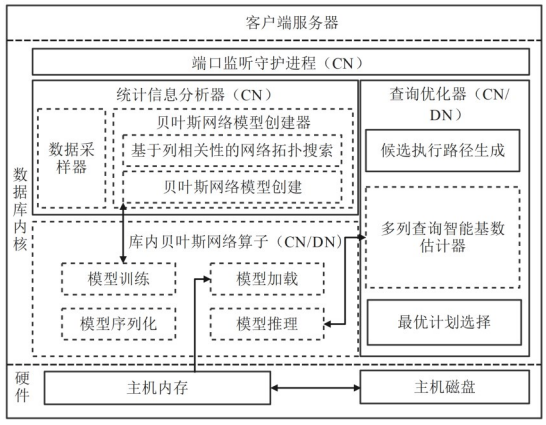

图21 智能基数估计系统架构图

(2)自适应缓存计划选择

当前事务型数据库中广泛使用的技术包括通用计划缓存技术,该技术使用默认参数进行查询优化并且将执行计划固化,避免了后续查询优化的开销,但是这种方式存在的问题是缺乏自适应的能力,针对倾斜数据和查询可能会导致慢查询出现.例如,默认计划使用A索引,而某带参数y的查询的最优计划应使用B索引.

自适应计划选择主要解决由缓存计划与查询不匹配导致的性能下降问题.不匹配的原因主要是当前缓存计划生成仅使用默认值,与查询参数无关.自适应计划选择基于实际查询参数生成多个候选计划,并通过计划选择为相似的查询参数选择相匹配的计划.自适应计划选择首先计算出查询涉及到的基表选择率,然后利用选择率相似距离选择缓存的计划.如果输入的查询基数特征超出缓存中已探测计划的最大超方体范围,那么就进行计划探测,利用优化器生成参数有关的执行计划并且存在计划缓存中.

友商也存在一些计划选择的算法,比如Oracle采用了Recost算法进行计划代价的估计,并且选择代价最小的执行计划.本文使用的算法在解决涉及不同扫描算子选择准确性方面和Oracle相当,并且效率高于Oracle.

如图22所示,自适应计划选择架构涉及四个不同组件,计划选择、计划探索、选择率估计以及计划生成.

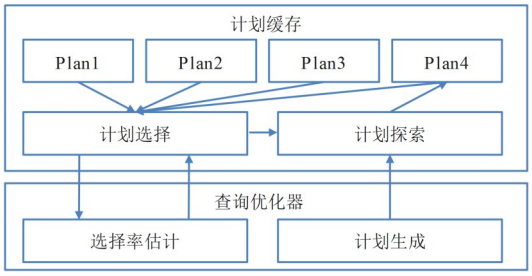

图22 计划选择整体架构

计划选择模块首先将输入查询的基表子查询条件输入选择率估计器进行估计选择率,并且和部分索引上的查询条件,查询获取行数(limit)等信息结合成一个数值型向量.然后计划选择从缓存中找一个向量距离最近的执行计划作为通用计划进行执行.如果该向量落在目前探索的超方体范围之外,那么就触发计划探索.

计划探索阶段完成两个任务,首先计划探索将带参数的查询输入优化器进行计划生成,如果获得的计划和缓存中计划不同,就将获得的执行计划存入计划缓存.最后在内存中扩展探索超方体边界.

选择率估计模块利用智能基数估计基数准确估计查询的基表选择率.

计划生成模块利用数据库内原生路径优化进行查询优化,选择最优执行计划.

3.4.2 库内原生AI引擎

为了实现数据库内数据库的学习,GaussDB支持数据库内原生AI引擎,支持库内机器学习、超参数优化和迁移学习,普惠AI(会SQL就能用AI).数据库内机器学习相比于外置机器学习方案提供更高的数据访问效率,并且有效避免数据搬移带来的安全性问题.数据库中加入了物化算子以及洗牌算子来加速训练,利用洗牌算子加速收敛,相比于外置AI引擎,节省了5~10倍的训练时间.分布式模型训练利用数据节点同步迭代机制,在每个节点进行本地模型训练,然后每次迭代针对模型进行合并,训练性能提升10倍.

3.4.3 自治运维技术

GaussDB设计自治运维平台如图23所示,实现自动参数调优、自动索引推荐、系统自诊断、慢SQL自诊断,以及自动分布键推荐,降低了依赖人力的监控运维带来的成本,并且实现了人工所无法取得的运维效果.自动调参系统利用DS-DDPG算法[23],高效自动搜索最优的数据库配置参数,提高数据库性能1个数量级.相比较现有基于学习的自动参数调优[21~23],DS-DDPG算法不依赖大量高质量训练数据同时提供细粒度调控.智能索引推荐技术利用MCTS算法[24].相较于现有如基于what-if的推荐技术[25],MCTS算法能够有效识别不同索引的收益和维护代

价,并且能够高效搜索性价比最高的索引,智能索引推荐能够推荐出优于人工挑选的索引,并且推荐效率远高于人工. 智能SQL诊断充分考虑数据库的查询特征、数据库性能指标以及数据库等待事件,将机器学习模型和人工经验进行融合,利用反馈修正诊断结果. 实验显示,诊断准确率达95%,诊断效率高于人工3个数量级.

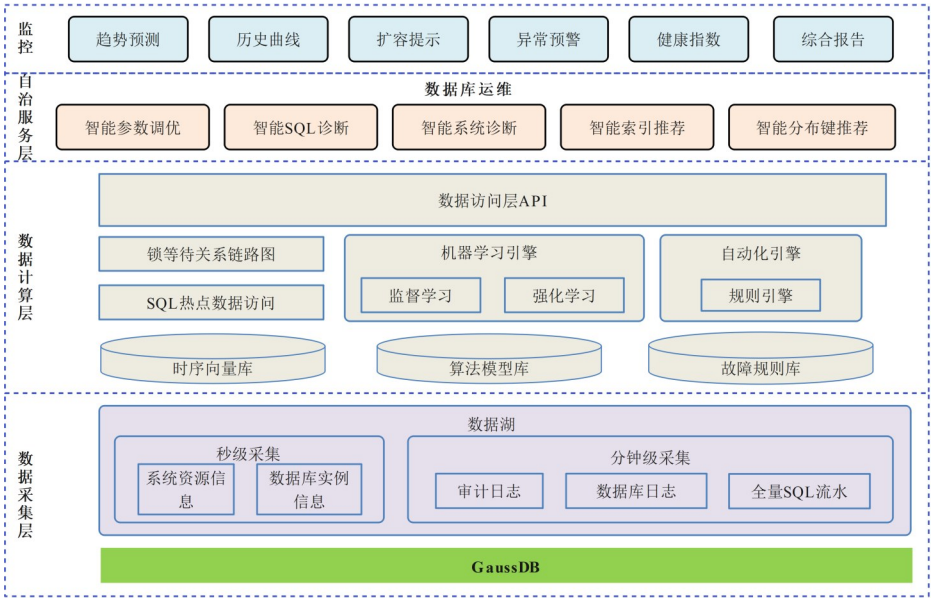

图23　GaussDB AI自治系统架构图

3.5　全方位安全技术

随着企业数字化转型的加速,敏感数据泄露案件时有发生,数据安全防护变得越来越重要. GaussDB构筑了全栈国密算法体系,设计并实现了细粒度权限管理、统一审计、访问控制、透明加密以及动态数据脱敏等功能,保护用户权限、操作和数据的安全,通过了国际EAL 4+安全认证. 从接入、认证、访问控制、数据脱敏与加密到用户行为的操作审计,构建了完善的安全纵深防护体系,全方位安全防护技术架构图如图24所示.

在构建数据库全方位安全技术体系过程中,数据库安全技术更加注重从客户内部的角度确保安全,即信息安全. 其内涵包括了保密性、完整性和可用性,即所谓的CIA(Confidentiality,Integrity,Availability)三个方面.

为了保护数据的机密性,本文提出了全密态数据处理技术. 该技术将密码学与数据存储、数据检索流程结合,提供了端到端的数据密文存储、密文运算、密文检索的能力. 纯软密态查询方案创新地将数据加密内核、密钥管理模块内置于数据库驱动,充分地降低了业务使用密态数据库的门槛,同时使用密钥缓存、查询语句预处理等加速技术,提升密态查询的性能. 同时充分利用可信硬件执行环境的计算能力,在保护数据机密性的同时,将密文运算算子作为最小可信基内置于硬件中,使用硬件为数据库服务侧提供丰富的算子以支持比较查询、范围查询和模糊查询等常用查询类型,同时支持视图、存储过程等数据库常用功能,极大地丰富了全密态数据库的密文检索功能.

针对数据完整性保护,设计了防篡改数据处理技术. 该技术利用Multi-Set哈希算法,将用户的操作和数据的更改映射到同一个哈希摘要空间. 基于这一设计,可以为用户的数据、数据的变迁历史以及用户完整的操作记录增加校验信息并采用链式存储,通过校验三者的一致性,即可保证用户对数据的每一次更改都被真实记录在数据库系统中,保证操作的可追溯、防抵赖. 同时,受益于哈希算法的使用,可以设计快速、并行的哈希校验算法,利用多线程优势并发进行校验. 同时可采用校验信息缓存、定期增量校验的方式,保证校验速度不随数据膨胀而增加.

针对可用性,可以使用AI引擎来实现自治安全技术,充分利用数据库运行过程中产生的大量实时状态数据、日志信息,以及GaussDB创新性的库内AI引擎技术. 基于查询请求的解析流程,结合AI算法提供了SQL注入智能检测功能;利用AI算法学习数据库运行日志、安全日志中的用户行为,可以提供异常行为检测能力. 将数据库内AI框架和库内的处理流程、日志结合,做到数据库内产生、AI持续训练、异常自动拦截,保障了数据库的自治安全能力.

3.5.1　全密态数据处理

GaussDB提出全密态数据处理技术[26],设计纯软密态查询和软硬融合全密态技术[27],保证数据客户端到服务端自动加密,客户端到用户界面自动解密,同时

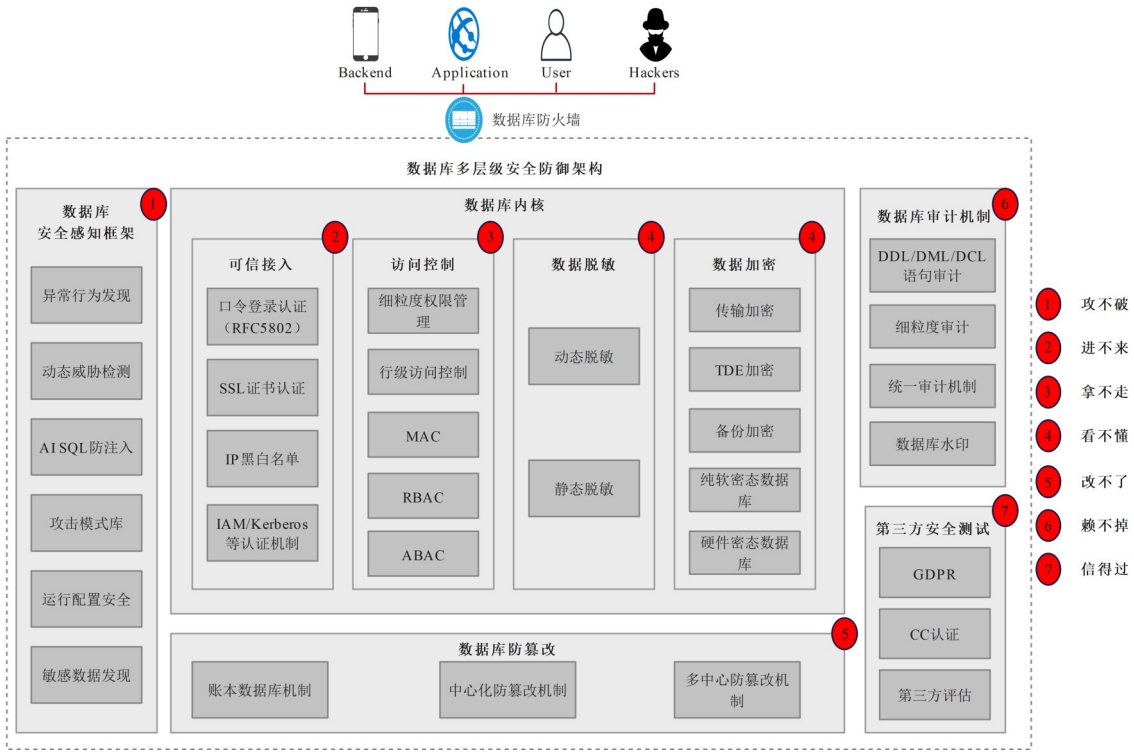

图24 GaussDB全方位安全技术架构

支持对密文进行复杂查询. 如图25所示,纯软全密态查询技术使用客户端侧密文查询预解析、密钥高效缓存服务,支撑客户端“无感”操作密文数据. 通过设计并实现基于区间分桶的密文索引结构来支持范围查询,通过基于trigram的索引结构支持密文的模糊检索功能. 纯软全密态查询具有良好的通用性和可迁移性,同时在查询性能上劣化5%(优于国外友商35%). 软硬融合全密态方案基于可信执行环境,将密文的运算抽象成基础算子内置于数据库服务端的可信执行环境中,设计并实现了依托于处理器可信执行环境(TEE)的密文计算框架. 在TEE中实现复杂SQL处理,对密文进行数值运算、聚集操作、关联查询、密文存储过程等复杂处理. 通过密文算子实现密态索引,提升检索效率. 软硬融合密态能力以及硬件的支持度领先于业界.

3.5.2 防篡改数据处理

提出防篡改数据处理技术,使用集合哈希的校验数据生成方式,支持数据修改的同时并发生成校验数据,提高了防篡改数据的并发写入和更新能力. 数据写入性能高于友商30%. 设计摘要算法将操作和数据的更改映射到同一哈希摘要空间,实现了数据内容与操作之间的关联校验能力,针对数据变更提供了完整、全面的追溯能力. 防篡改数据能力成为首家通过信通院防篡改标准测试的厂商.

3.5.3 自治安全管控技术

发展了自治安全管控技术[28],让数据库可以持续、实

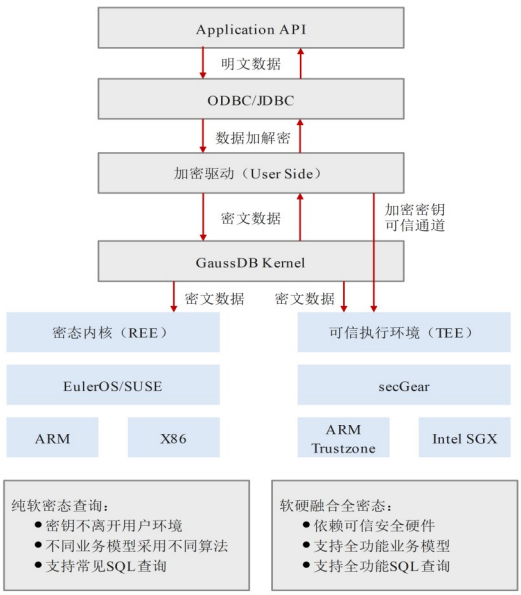

图25 全密态数据库整体架构

时学习用户操作习惯,感知并拦截异常操作和恶意攻击. 针对注入攻击,发展了数据库优化器结合AI方法的SQL注入检测技术,精准识别语句执行过程中的查询失效时刻,结合库内AI算法对永真式SQL进行甄别,避免了传统基于查询字符串解析带来的性能不高的缺陷. 利用数据库运行态的全量状态数据和运行日志、审计日志等数据源,在库内训练AI模型,提高了模型的训练精度,同时避

免了模型学习过程中大量数据的导入、导出操作以及可能存在的信息泄露问题.

4　实验结果与分析

本节将系统介绍GaussDB五个关键技术测试与对比结果. 本节中除特别声明外,实验环境为由一个数据中心和三个分别分布于西安、廊坊、东莞的集群组成. 数据中心集群包含3个TaiShan服务器,每个实验机器配置如表2所示. 位于三地的集群各自包含3个Intel x86服务器,每个服务器配有2个 Intel Xeon E5-2680 v2 CPU和188 GB的DRAM. 本节介绍的所有性能相关测试均采用数据库通用标准测试集.

表2　实验机器配置

机型	CPU型号	CPU核数	架构	内存	磁盘	操作系统
TaiShan 200 (Model 2280)	鲲鹏920	128	ARM	2 048 GB	NVME 7.3 T*4	Euler.9

4.1　分布式查询处理技术测试与分析结果

为了验证GaussDB分布式查询优化能力,本节测试了传统分布式计划与优化后的性能对比结果.

协调节点旁路技术可以极大消减协调节点上的开销,在典型的OLTP场景sysbench测试集中,比传统分布式计划快2倍. 高性能是因为GaussDB推算子识别技术对单分片特征识别,查询实现下推到数据节点上独立执行,对协调节点的性能资源消耗低,如表3所示.

表3　Sysbench性能对比　　单位:K

性能参数	传统计划	协调节点旁路
sysbench性能TPS	6.8	1.2

数据节点自协同的流式计划,可以减少网络传输,提高并行度,在典型的OLAP场景下,TPC-H比传统分布式计划快25倍. 自协同的流式计划允许数据节点之间交换数据,协同计算. 所有计算尽可能在数据节点上并行执行,缩短执行时间,如表4所示.

表4　TPC-H性能对比　　单位:s

性能参数	传统计划	流式计划
TPC-H执行时间	12 430	498

为了验证全并行编译执行效果,本节测试了3节点TPC-H性能,并与业界代表性厂商进行了对比,性能领先82%以上. 高性能是因为GaussDB全并行编译实现从节点间并行、节点内并行到指令级并行,以及LLVM动态编译Codegen技术,充分发挥多机多核的硬件资源,如图26所示.

特别地,GaussDB的分布式查询处理数据能力与业界代表性厂商对比,在TPC-H的Q6、Q10、Q14分别实现1.49、4.15和5.50倍性能提升. Q6、Q10、Q14涉及多表JOIN和子查询,GaussDB分布式查询处理对复杂查询有明显提升,如图27所示.

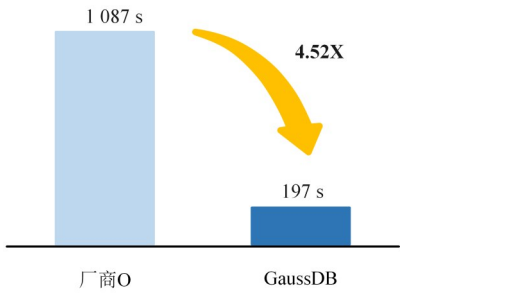

图26　TPC-H性能测试对比结果

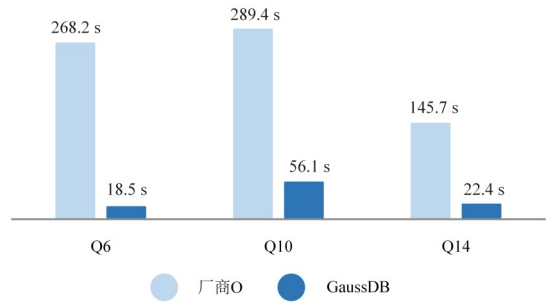

图27　TPC-H性能典型查询对比结果

在大规模分布式事务处理方面,本文测试了32节点分布式事务处理性能,以及全球数据库的性能对比. GaussDB的性能优势主要来自于GTM-LITE简化分布式事务快照,使得GTM的快照生成和发放能力呈倍数提升,以及高精度时钟同步技术,如图28和图29所示.

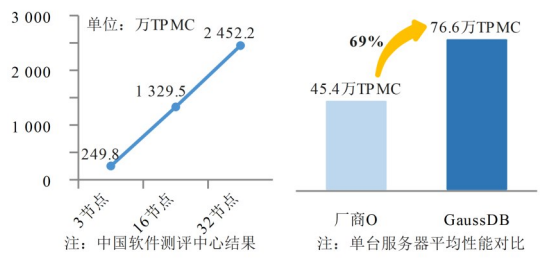

图28　32节点TPC-C事务处理性能

4.2　分布式高可用以及容灾的测试与分析结果

对于数据库系统而言,备节点的日志回放能力深深影响着故障场景下的切换时间,即RTO. 本文详细地测试了细粒度并行回放技术的性能,如表5所示. 表5的结果是在128核CPU,512 GB内存鲲鹏服务器,NVMe SSD盘,25GE的网络上进行测试得出的. 在两台物理机上分别搭建一主机一同步备机,在主机上运行TPCC

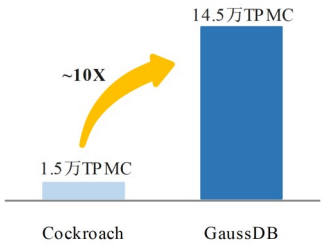

图29 全球数据库TPC-C事务处理性能

benchmark,并发送给备机日志,备机通过日志回放与主机同步. 在使用原始的并行回放,主机在到达50万TPMC总交易数,23万TPMC的交易数以上,备机就会产生日志积累,导致RTO逐渐增大. 在运用本发明的并行回放后,能保证主机在跑到150万TPMC总交易数,70万TPMC的交易数都不会产生日志积累.

表5 并行回放性能对比 单位:万TPMC

	能保证备机追上主机的TPMC总交易数	能保证备机追上主机的TPMC交易数
原始的并行回放	50	23
本方案的并行回放	150	70

从故障恢复端到端的消耗时间,本文测试了20余项故障测试,包含断网、下电等注入故障,GaussDB能够在10 s以内完成切换,实现业务查询性能恢复. GaussDB中基于Paxos协议的自感知副本一致性共识复制以及仲裁技术实现数据节点高效自选主、细粒度无锁并行回放实现快速日志回放,如图30所示.

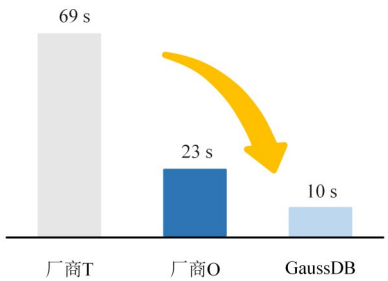

图30 故障测试RTO对比

4.3 云原生弹性伸缩技术测试与分析结果

为了验证云原生弹性伸缩技术效果,采用1 024个分片,使用3 000 Warehouse的数据量进行TPCC测试,单节点业务中断时间在1 s以内,如表6所示.

相比于传统分布式数据库的普通扩容技术,GaussDB在线弹性伸缩技术的性能有万倍提升. 如图31所示,普通扩容技术在1 TB数据量场景下,扩容耗时在3 h以上. 而GaussDB在线弹性伸缩技术[29]可以实现秒级扩容.

表6 GaussDB云原生快速平滑扩容测试

节点数	并发度	CPU压力/%	业务中断DN最长时间/ms
3扩6	200	40~50	207
3扩9	200	40~50	219

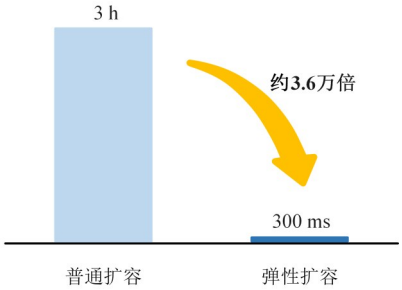

图31 GaussDB弹性扩容与传统分布式数据库普通扩容技术性能对比

4.4 智能自治优化技术测试与分析结果

本节测试了智能优化器在索引选择场景下的性能表现,相对于传统优化器,智能优化器计划生成执行时间提升1倍以上. GaussDB高性能主要因为基于AI的优化器核心智能基数估计更加准确高效,同时自适应缓存计划选择基于实际查询参数生成多个候选计划,并通过计划选择为相似的查询参数选择相匹配的计划,如图32所示.

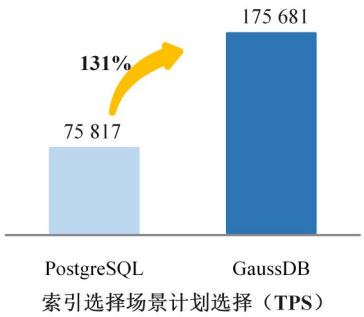

图32 GaussDB智能优化器性能测试

同时,本节也测试了库内原生AI引擎的性能,如图33所示. 相较于PostgreSQL Madlib,性能领先13倍~174倍. 高性能是因为GaussDB的物化算子、洗牌算子、节点同步迭代机制显著加速了训练过程[30].

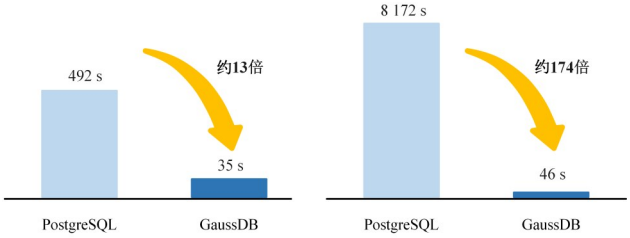

图33 GaussDB库内原生AI引擎性能测试对比

4.5 全方位安全测试与分析结果

GaussDB纯软全密态数据库技术性能领先微软

SQL Server 35%以上. GaussDB实现依托于处理器可信执行环境(TEE)的密文计算框架,其中基于密文算子的密态索引显著提高了查询处理效率,如图34所示.

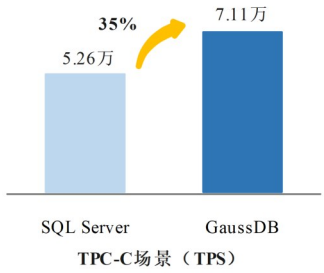

图34　全密态数据库性能对比测试结果

GaussDB防篡改数据库技术性能领先LedgerDB 30%以上. GaussDB防篡改数据库中高性能是因为支持数据修改的同时并发生成校验数据,如图35所示.

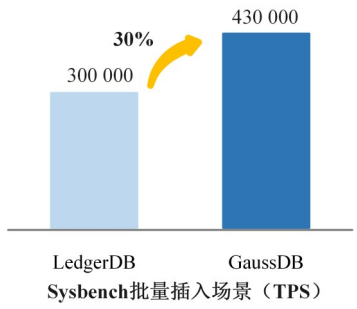

图35　防篡改数据库性能对比测试结果

GaussDB异常行为检测算法相比于业界主流算法iForest/DeepLOG等,在判断精度和召回率表现上更优,F1分数最高可达到0.9+. GaussDB异常处理结合AI方法,感知并拦截异常操作和恶意攻击,如图36所示.

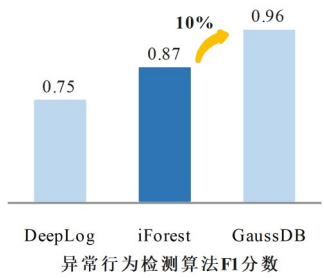

图36　异常行为检测对比测试结果

5　结论与展望

在数据量、应用种类等与日俱增的今天,数据库面临新的挑战,如超大数据量、超高并发,以及业务对数据库的高可用要求. 同时,云计算与人工智能的发展也给数据库带来了新的契机. 本文介绍了GaussDB智能云原生分布式数据库的关键设计与系统实践. 在分布式查询优化、分布式高可用容灾、云原生弹性、智能化、安全防护等方面作出了系列系统性技术突破与创新,使得系统具备了高性能、高可用、高弹性、高智能、高安全等企业级数据库竞争力优势,以应对新时代金融、政府等国计民生关键基础设施行业的核心数据库应用挑战.

展望未来,新型硬件以及异构算力给数据库提供了新的变革条件,而大模型的兴起,也给数据库的开发以及应用带来了新的诉求与挑战. 如何设计一款能适应新型硬件架构如CXL、NPU、NVM等硬件的数据库,同时适应大模型的多模态数据存储及查询要求,将成为大模型与通用人工智能时代,数据库发展的新思路与新方向.

参考文献

[1] CHEN J J, CHEN Y, CHEN Z B, et al. Data management at Huawei: Recent accomplishments and future challenges[C]//2019 IEEE 35th International Conference on Data Engineering (ICDE). Piscataway: IEEE, 2019: 13-24.

[2] CHAUDHURI S, HARITSA J, DENG D, et al. An efficient partition based method for exact set similarity joins[J]. Proceedings of the VLDB Endowment, 2015, 9(4): 360-371.

[3] MA D Z, FENG J H, LI G L, et al. A positional access method for relational databases[C]//Proceedings of the 21st ACM International Conference on Information and Knowledge Management. New York: ACM, 2012: 2164-2168.

[4] NI J C, LI G L, WANG L J, et al. Adaptive database schema design for multi-tenant data management[J]. IEEE Transactions on Knowledge and Data Engineering, 2014, 26(9): 2079-2093.

[5] DENG D, LI G L, HAO S, et al. MassJoin: A mapreduce-based method for scalable string similarity joins[C]//2014 IEEE 30th International Conference on Data Engineering. Piscataway: IEEE, 2014: 340-351.

[6] YANG Z K, YANG C H, HAN F S, et al. OceanBase: A 707 million tpmC distributed relational database system[J]. Proceedings of the VLDB Endowment, 2022, 15(12): 3385-3397.

[7] HUANG D X, LIU Q, CUI Q, et al. TiDB: A raft-based HTAP database[J]. Proceedings of the VLDB Endowment, 2020, 13(12): 3072-3084.

[8] SHANG Z Y, LI G L, BAO Z F, et al. DITA[C]//Proceedings of the 2018 International Conference on Management of Data. New York: ACM, 2018: 725-740.

[9] MA D Z, FENG J H, LI G L, et al. LazyFTL[C]//Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. New York: ACM, 2011: 1-12.

[10] VERBITSKI A, GUPTA A, SAHA D, et al. Amazon aurora: Design considerations for high throughput cloud-native relational databases[C]//Proceedings of the 2017 ACM International Conference on Management of Data. New York: ACM, 2017: 1041-1052.

[11] Cloud Blog Databases. Introducing AlloyDB for PostgreSQL: Free yourself from expensive, legacy databases[EB/OL]. [2023-11-01]. https://id.cloud-ace.com/introducing-alloydb-for-postgresql-free-yourself-from-expensive-legacy-databases/.

[12] ANTONOPOULOS P, BUDOVSKI A, DIACONU C, et al. Socrates: The new SQL server in the cloud[C]//Proceedings of the 2019 International Conference on Management of Data. New York: ACM, 2019: 1743-1756.

[13] CAO W, ZHANG Y Q, YANG X J, et al. PolarDB serverless[C]//Proceedings of the 2021 International Conference on Management of Data. New York: ACM, 2021: 2477-2489.

[14] LI G L, ZHOU X H, SUN J, et al. openGauss: An autonomous database system[J]. Proceedings of the VLDB Endowment, 2021, 14(12): 3028-3042.

[15] LI G L, ZHOU X H, LI S F, et al. QTune: A query-aware database tuning system with deep reinforcement learning[J]. Proceedings of the VLDB Endowment, 2019, 12(12): 2118-2130.

[16] SUN J, LI G L. An end-to-end learning-based cost estimator[J]. Proceedings of the VLDB Endowment, 2019, 13(3): 307-319.

[17] YUAN H T, LI G L, FENG L, et al. Automatic view generation with deep learning and reinforcement learning[C]//2020 IEEE 36th International Conference on Data Engineering (ICDE). Piscataway: IEEE, 2020: 1501-1513.

[18] YU X, LI G L, CHAI C L, et al. Reinforcement learning with tree-LSTM for join order selection[C]//2020 IEEE 36th International Conference on Data Engineering (ICDE). Piscataway: IEEE, 2020: 1297-1308.

[19] SUN J, SHANG Z Y, LI G L, et al. Dima: A distributed in-memory similarity-based query processing system[J]. Proceedings of the VLDB Endowment, 2017, 10(12): 1925-1928.

[20] YU X, CHAI C L, LI G L, et al. Cost-based or learning-based? A hybrid query optimizer for query plan selection[J]. Proceedings of the VLDB Endowment, 2022, 15(13): 3924-3936.

[21] HILPRECHT B, SCHMIDT A, KULESSA M, et al. DeepDB: Learn from data, not from queries![J]. Proceedings of the VLDB Endowment, 2020, 13(7): 992-1005.

[22] ZHU R, WU Z N, HAN Y X, et al. FLAT: Fast, lightweight and accurate method for cardinality estimation[J]. Proceedings of the VLDB Endowment, 2021, 14(9): 1489-1502.

[23] ZHANG J, LIU Y, ZHOU K, et al. An end-to-end automatic cloud database tuning system using deep reinforcement learning[C]//Proceedings of the 2019 International Conference on Management of Data. New York: ACM, 2019: 415-432.

[24] ZHENG C H, DING Z H, HU J L. Self-tuning performance of database systems with neural network[M]//Intelligent Computing Theory. Cham: Springer International Publishing, 2014: 1-12.

[25] BROWNE C B, POWLEY E, WHITEHOUSE D, et al. A survey of monte carlo tree search methods[J]. IEEE Transactions on Computational Intelligence and AI in Games, 2012, 4(1): 1-43.

[26] WU S R, LI Q, LI G L, et al. ServeDB: Secure, verifiable, and efficient range queries on outsourced database[C]//2019 IEEE 35th International Conference on Data Engineering (ICDE). Piscataway: IEEE, 2019: 626-637.

[27] ZHU J W, CHENG K, LIU J Y, et al. Full encryption: An end to end encryption mechanism in GaussDB[J]. Proceedings of the VLDB Endowment, 2021, 14(12): 2811-2814.

[28] LI S N, YIN Q L, LI G L, et al. Unsupervised contextual anomaly detection for database systems[C]//Proceedings of the 2022 International Conference on Management of Data. New York: ACM, 2022: 788-802.

[29] LI G L, TIAN W G, ZHANG J Y, et al. GaussDB: A cloud-native multi-primary database with compute-memory-storage disaggregation[J]. Proceedings of the VLDB Endowment, 2024, 17(12): 3786-3798.

[30] LI G L, SUN J, XU L J, et al. GaussML: An end-to-end in-database machine learning system[C]//2024 IEEE 40th International Conference on Data Engineering (ICDE). Piscataway: IEEE, 2024: 5198-5210.

作者简介

李国良 男,1981年出生,河北唐山人. 2009年在清华大学计算机系获得博士学位. 清华大学教授,计算机系副主任,博士生导师,国家杰出青年科学基金获得者,IEEE Fellow,openGauss社区技术委员会主席. 主要研究方向为数据库系统、智能数据管理. 中国电子学会会员编号:E190015262M.
E-mail: liguoliang@tsinghua.edu.cn