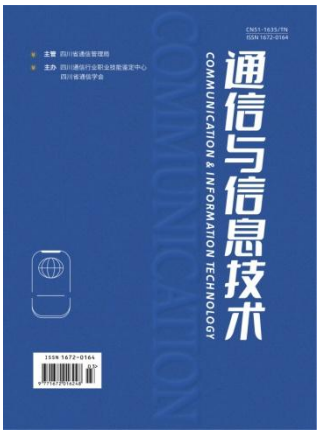

通信与信息技术

Communication & Information Technology

国内统一连续出版物号：CN 51-1635/TN
国际标准出版物号：ISSN 1672-0164
邮发代号：62-166

题目：分布式数据库多芯混合平台技术的应用实践

作者：杨孝平[1]，张旭辉[1]，温健军[1]，杨名[1]，周日明[2]，秦延涛[2]，朱宸希[2]，吕伟初[2]

优先出版日期：2025年4月2日

万方网站：https://w.wanfangdata.com.cn/

分布式数据库多芯混合平台技术的应用实践

杨孝平[1]，张旭辉[1]，温健军[1]，杨名[1]，周日明[2]，秦延涛[2]，朱宸希[2]，吕伟初[2]

1.中国移动通信集团四川有限公司，成都 610097

2. 金篆信科有限责任公司，北京 100176

摘 要：聚焦数据库从传统集中式平台向分布式平台的转型过程，深入分析转型过程中面临的硬件可靠性、软件兼容性及数据迁移等关键挑战。为了解决这些技术挑战，本研究创新性地提出了多项技术解决方案：提出了一种创新业务连续性保障技术，在分布式架构条件下做到生产、容灾库快速切换，保障业务连续性要求；实现了异构芯片混合架构平台，通过创新软件工程方法解决了平台差异和功能兼容性验证难题，通过合理组网避免了性能短板效应，实现数据库在源代码工程上的版本统一；针对数据模型变化带来的数据迁移难题，提出了增量转换的迁移方法，实现了全流程平滑迁移，减少割接风险。本研究成果为分布式架构转型项目提供了完整的解决方案和有力的技术支撑，具有重要的理论意义和实际应用价值。

关键词：数据库；分布式系统；分布式数据库；数据库迁移

中图分类号：TP393.2 **文献标志码**：A

1 引言

四川移动现有CRM/BOSS系统基于垂直架构设计，采用传统集中式数据库建设，经过多年发展，逐渐显露扩容能力弱、新特性投产慢等弊端，难以响应数据量爆发增长和应用快速迭代要求；该系统基于集中式数据库建设，设备和维护成本高昂，增加了企业经营成本。因此对现有系统进行架构升级，建设全栈分布式新平台势在必行。

四川移动联合金篆信科等多方厂商，引入GoldenDB分布式数据库，建设灵活开放、集中管理、融通融智的新一代核心交易系统，存储更大数据量，提升处理效率，降低企业整体运营成本，满足四川移动业务快速发展和应用创新要求，助力公司数智化转型。

2 研究现状

2012年，谷歌（Google）发表《Spanner：Google全球化分布式数据库》[1]，阐述大型互联网公司建设分布式数据库的扩展方法、事务机制以及多副本技术，虽然该成果在处理响应、成本集约化和行业内落地等方面还有待完善，但已为业界呈现了关系型数据库向分布式架构发展的巨大潜能，开启分布式数据库的新时代。

此后国内分布式数据库产品迅猛发展，在高并发处理、横向扩容、事务处理一致性上取得长足进展，并在金融行业发力。2018年信通院发布《金融分布式事务数据库白皮书》[2]，2020年金融标准委员会发布《分布式数据库技术金融应用规范》[3]，为数据库从传统集中式架构向分布式架构发展指明方向，加快了国内分布式数据库建设步伐。其中银行核心以及关键业务系统在如火如荼地建设中，中信银行核心账务系统、建设银行对私核心系统已先后完成了分布式平台投产，标志着分布式数据库建设取得了阶段性成功。

当前分布式数据库正向通信、交通、能源等更多重要行业推广使用，不同行业应用上有其自身特点和技术挑战。以通信行业为例，最突出的挑战有以下3个方面：

硬件可靠性。**分布式**数据库与底层操作系统、芯片和服务器关系紧密，随着国内技术不断成熟和成本需要，通信行业在逐步使用新型操作系统和服务器部署，但上述基础软件和芯片还未经过长期运行检验，性能和稳定性存在不足，需要通过分布式数据库的架构创新来弥补。

软件兼容性。这里包括数据库软件对新型芯片的兼容、对新型操作系统的兼容，以及分布式数据库对存量集中式数据库语法兼容这三个层面，尤其最后一项最为突出。通信业务软件的开发规范约束较为宽泛，存量应用使用了大量的高级数据库对象和语法，这些高级数据库对象和语法在过往项目中提高了软件开发效率，但同时也自设了系统向其他数据库迁移的门槛。而分布式数据库还处于发展初期，数据库对象功能上还亟待完善，与传统数据库在高级语法和性能上存

收稿日期：2024 年 11 月 27 日； **修回日期**：2025 年 3 月 30 日

通信作者：周日明（1980—），男，高级工程师，主要研究方向：分布式系统、数据库。

基金项目：北京市科技计划项目（项目编号：Z231100005923010）

在较大差距，需要通过产品迭代和工具手段保障项目平顺推进。

数据迁移难题。通信系统存储和处理全省近亿规模的用户数据，数据体量达到数TB和数十TB规模，数据迁移耗时长；且均属于重要生产系统，停业影响面广，割接窗口时间短。此外，在数据库迁移项目过程中，一般还伴生应用架构重构、数据模型转换，后者加剧了数据迁移难度。如何在较短的割接窗口内同时完成巨量数据迁移和模型转换是一个重要的技术挑战。

针对上述难题，项目中提出多个创新解决方法，提出了一种创新业务连续性保障技术，在分布式架构条件下做到生产、容灾库快速切换，保障业务连续性要求，实现了异构混合架构平台性能均衡；通过混沌测试工程解决了分布式系统可靠性验证难题；针对数据模型变化带来的数据迁移难题，提出了增量转换的迁移方法，实现了全流程平滑迁移，减少迁移割接风险。

3 系统整体方案

新一代核心交易系统总体由接入层、应用服务层、网络层、分布式数据库和底层软硬件平台构成。以营销交易中心为例，整体架构如图1所示：

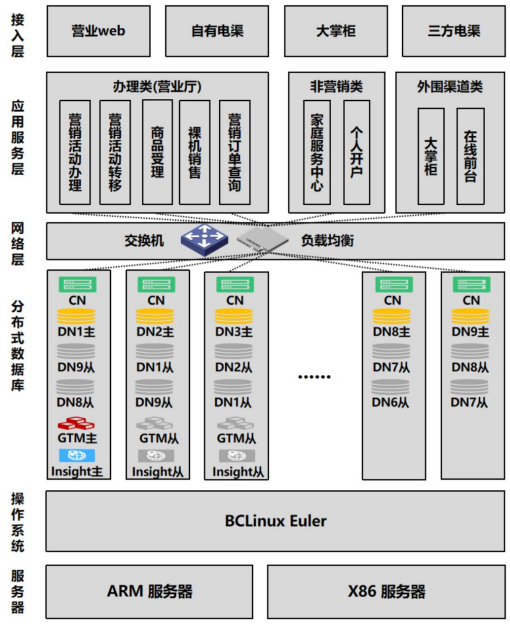

图 1 新一代营销交易系统整体架构图

接入层接收前端请求，将营业厅、自有电渠、大掌柜以及第三方电子渠道的业务请求推送到后端应用服务层。后端应用服务受理营销商品、办理营销活动，提供营销订单查询等功能。应用服务层采用容器化微服务方式开发，具有弹性扩容、灰度发布等架构优势。应用服务层通过JDBC方式访问数据库，请求经过负载均衡分发到GoldenDB分布式数据库。

GoldenDB是金篆信科研发的一款分布式数据库，支持大规模数据存储和高并发访问，整体由计算节点（CN）、数据节点（DN）、全局事务管理器（GTM）、管理节点（Insight）四种核心模块组成：

计算节点（CN），计算节点是分布式数据库的核心处理层，从应用层驱动接收SQL请求，进行逻辑优化和物理优化，生成满足分布式事务一致性的分布式查询计划，调度数据节点执行，并最终完成SQL请求的查询和处理。

数据节点（DN），数据节点是数据的最终存储模块，按照特定策略将数据进行横向分片后存放到多个安全组（Group）中，分片策略包括复制策略、哈希策略、范围策略、列表策略、多级分片等。安全组内部采用多副本技术，由多个数据节点构成的，每个节点拥有相同的数据，其中一个数据节点为主用节点，其他为备用节点，数据在主备节点之间实时复制。

全局事务管理器（GTM），全局事务管理器在分布式数据库中维护全局事务的生命周期，提供申请、释放、查询全局事务的能力。计算节点在全局事务管理器地协作下，对多个数据节点进行跨节点事务控制，虽然数据在物理机上分离存储和计算，但数据库对外保持全局事务的一致性和完整性。

管理节点（Insight），管理节点是GoldenDB分布式数据库产品的统一操作维护入口，负责分布式数据库集群的集中可视化管理。用户在Insight管理台完成用户管理、权限管理、元数据管理、节点扩容、备份恢复以及监控告警等日常维护工作。

GoldenDB和应用均运行在BCLinux Euler操作系统上，服务器有两种类型，包括ARM架构的鲲鹏服务器和X86架构的海光服务器。

4 通过分布式架构提升系统可靠性

在可靠性设计思路上，分布式数据库与集中式数据库存在较大差别。后者采用小型机或大型机等专用服务器，服务器内部有CPU、内存和存储的冗余设计，可靠性高，但价格昂贵；分布式数据库为了降低硬件成本，普遍采用通用PC Server服务器，但其冗余设计单薄，可靠性差，特别是本次项目采用的ARM服务器和海光服务器，两种服务器都是近年推出的新型设备，需要通过分布式架构技术来提升系统可靠性。

假设PC Server服务器可靠性在99.9%左右，如果采用传统HA模式，根据CAP原理[4]，只能在服务可用性和数据可

靠性之间取舍，两者不能兼得。而本次项目中，利用GoldenDB集群的多副本技术（Paxos协议），采用三副本冗余部署，副本之间通过Paxos一致性复制协议进行数据同步[5]。单副本故障时，数据库自动切换，数据零丢失，服务无中断；只有当两个副本同时故障时，才需要人工介入应急处理，可靠性提高到99.9999%。

Paxos协议复制过程复杂，高并发时处理性能不足。本项目在Paxos协议基础上，通过线程池化技术提升同步性能，形成GoldenDB创新的快同步复制技术。

具体做法如图2所示：

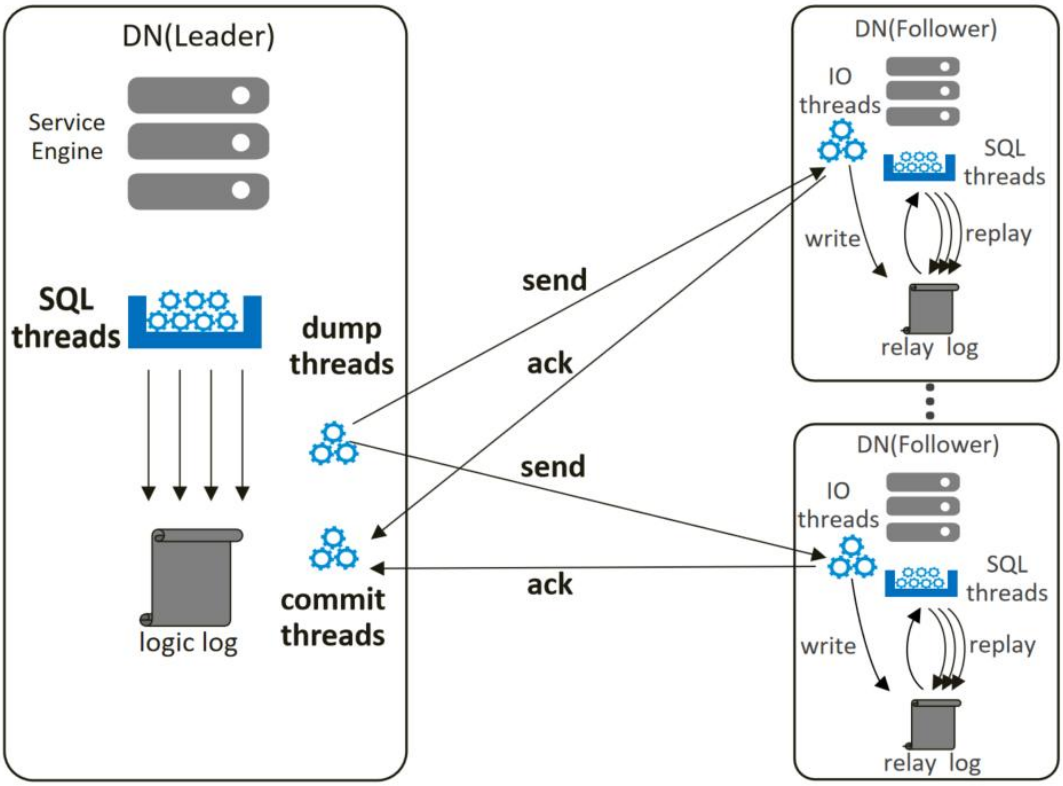

图 2 GoldenDB 快同步复制技术

主节点（Leader）执行线程（SQL threads）在事务提交时产生逻辑日志（logic log），但不负责日志内容的发送和等待响应，而是将当前事务状态保存至会话中，之后就开始掉头执行下一连接上的请求，避免处理线程等待应答而阻塞，从而提升执行线程的处理性能。

主节点常驻一组发送线程（dump threads），负责发送日志，将逻辑日志发送（send）到其他备机（Follower），各备机的IO线程（IO threads）收到逻辑日志后，写入到本地磁盘（relay log），再给主节点应答（ack）。

主节点常驻一组提交线程（commit threads），负责处理应答，收到应答之后找到执行线程保存的对应会话，完成该事务的最终提交。

这种处理线程池化有如下优点：

减少了线程规模。优化前每一个连接需要一个单独的线程来处理，而优化后一个线程可以处理多个连接，这样少量线程即可支持大规模连接，满足四川移动单个数据库并发10万连接的要求。

避免了线程等待。优化前执行线程要阻塞等待备机响应（send->write->ack），优化后执行线程无等待，同步日志和备机响应之后的处理过程交给提交线程专职负责，这种无阻塞机制大大提升了各个线程的处理能力，对比测试性能有20%左右的提升。如图3所示：

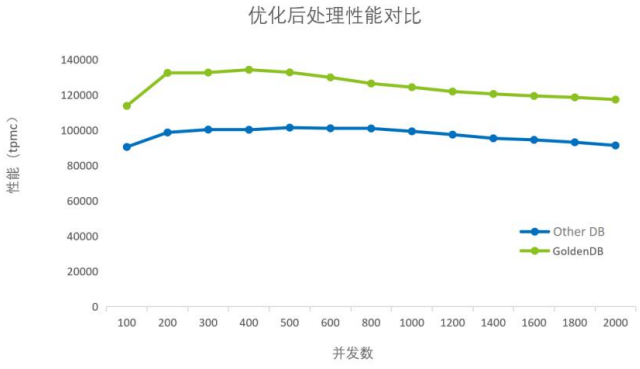

图 3 快同步复制性能效果

本系统在集团一级云部署生产库，省级数据中心部署容灾库。

跨机房高可用架构如图4 所示：

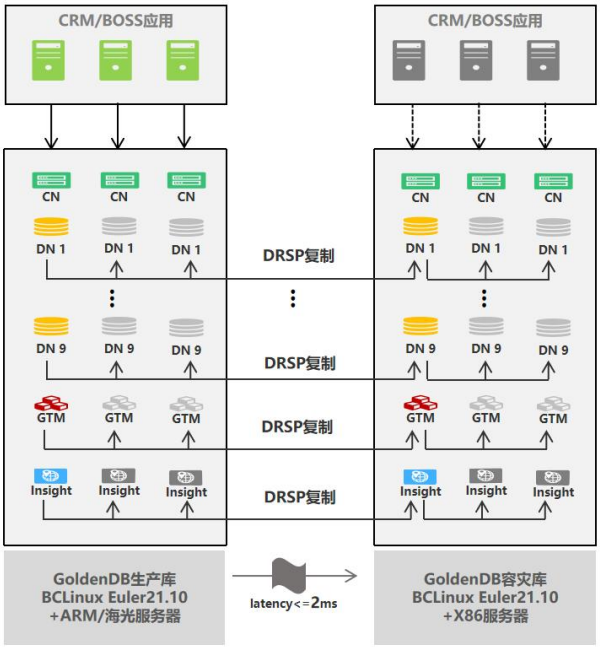

图 4 跨一级云和省级数据中心异构部署（ARM 架构和 X86 架构）

生产库集群和容灾库集群内部各有三个数据副本，三个副本之间采用一致性复制协议，保证各种异常情况下不丢失数据，确保业务高可用和数据高可靠。

通信行业对机房间网络延迟敏感，追求更高的服务可用性，要求机房间网络时延过大或频繁抖动时，不影响生产服务质量，故机房间采用GoldenDB的DRSP机制（Disaster Recovery Switching Platform），应用双集群串联技术，来解决两个集群复杂的复制关系。具体地，容灾库集群与生产库集群是独立部署的两个集群，每个集群内都有多个副本（Replica），DRSP串联关系如下：

Replica1->(Replica2, Replica3,(Replica4->(Replica5,Replica6)) (1)

即容灾库集群内部组成一组主从复制关系，内部投票选举出自己的主节点*(Replica4)*，由该主节点接入到生产库集群任意一个节点*(Replica1, Replica2, Replica3)*追齐数据，如*Replica1->Replica4*，后者再分发给容灾库集群内地从节点*(Replica4->(Replica5，Replica6))*。这样两个集群虽然多达六个副本，但机房间只需要复制一路增量数据，有效减少了带宽占用。容灾库主节点*(Replica4)*只复制日志，不参与生产库日志的投票，机房间网络抖动不影响生产库服务质量。容灾库主节点故障由集群内多个副本自主协商出新主，故障容忍度高。

容灾库首先承担了机房级别的容灾能力，当一级云生产机房出现故障时，部署在省级数据中心的容灾库提升为主节点，对外提供服务，实现同城RPO<5秒，RTO<30秒的业务连续性目标。容灾库同时还承担应急库能力，当生产机房GoldenDB出现集群级故障时，故障不会蔓延到容灾集群。这种情况下机房正常，但生产库内部出现集群问题，如系统级配置引起的整个生产库不可用，通过应急处理将生产流量快速切换到容灾库集群，保障业务连续性不受影响。

ARM服务器和海光服务器属于两种不同的芯片技术路线，在技术方向上各有优势，发展程度也不同，为了避免依赖单一技术路线带来供应链风险，在一套GoldenDB数据库上同时部署两种服务器类型，该方案的实施难点在于不同服务器类型和型号性能存在差异，在混合组网中需要平衡承担处理压力，避免性能短板效应，满足投产系统的性能稳定性要求。

项目过程中对两台服务器进行测试验证，当前不同服务器的极限性能差异较为明显，但每台服务器迭代快速，目前难以给出孰优孰劣的结论。有两种方式进行混合部署，第一种方式是同一个机房使用相同的服务器，如生产库使用ARM服务器、容灾库使用海光服务器，这样按机房粒度拉齐处理性能，更容易做到性能均衡；若某种型号服务器存在故障，另一个机房可以继续提供服务。第二种方式是每个机房都部署两种服务器，主节点采用ARM，从节点采用海光服务器，通过这种方式做到性能均衡和故障替代，也更适应同一个机房内有多种服务器的现实条件。此外，根据测试性能结果，拉平两种服务器的CPU核数、内存、磁盘能力，整机表现上做到对等配置；在系统设计时估计性能差异的梯度，评估从一种服务器切换到另一种服务器上的吞吐性能下降

情况，适当增加冗余量。项目验证两套方案均具有可行性，本次项目采用第一套方案。而后续新项目中，单机房内两种服务器均有部署，为了利用好服务器资源，将使用第二种方案。

5 通过混沌测试验证分布式系统可靠性

在项目实施过程中，将混沌测试引入分布式数据库系统，设计了分布式数据库混沌测试方案[7]，测试和验证混合组网架构下系统可靠性表现，并实现版本自动化迭代发布测试。该过程有效提升分布式系统的健壮性和稳定性，同时大大缩短了多平台架构下版本验证的时间。

混沌测试是一种新型的软件测试方法，通过在系统中引入随机的故障或异常行为[8]，来验证系统在不确定性和混乱情况下的健壮性和恢复能力，该方法主要用于寻找复杂系统可能存在的隐藏问题。分布式系统由大量服务器通过网络协作对外提供整体服务，每台服务器均存在进程异常、磁盘故障、压力过载、机器断电、网络隔离等故障，且故障具有随机性、多发性等特点，因此混沌测试在验证分布式系统中尤为有效。

传统测试围绕用户视角，对产品功能和性能进行测试，验证系统表现是否与用户需求一致，通常情况下，通过尽可能多地编写单元测试，进行足够多的集成测试，可以确保系统在预设的故障场景下能正常工作。但真实生产环境下异常时刻和故障组合是不确定，这些预设的测试条件无法覆盖分布式系统的各种动态组合。而混沌测试侧重系统视角，通过人为注入随机异常并进行多故障组合，更接近生产环境可能出现的各种实际情况，将深层隐藏问题提前暴露出来，从而锤炼产品的健壮性，实现在复杂生产环境下稳态运行。

对分布式数据库来说，混沌测试具有更高一层的意义，其原因在于分布式数据库作为基础服务，对可靠性有更高要求，需要同时确保数据一致性和服务可用性，在多发的故障条件下，需要通过可验证手段获得上述两项指标。

在混沌测试工具方面，业界进行了一些有益尝试。其中Netflix 公司的Chaos Monkey是效果比较好的混沌测试工具，并在GitHub上开源，该工具主要通过对网络、服务器和服务进程随机注入故障，来检验分布式系统的服务质量是否会受到影响。Apache TinkerPop项目下的Gremlin平台提供了一种基于 Web 的 GUI，支持以受控的方式进行混沌实验，执行一系列针对基础设施的故障“攻击”，包括服务器资源过载（CPU、内存、IO、磁盘满）、网络(断网、延迟、包丢失、DNS停服）、状态异常（服务器掉电、时钟异常、进程退出）等，允许工程师观察系统在各种故障情况下的行为。

目前这些工具与目标系统紧密结合，尚不满足通用化的混沌测试场景。业界也在探讨研究分布式数据库下的混沌测试方法，如《 Research on Distributed Database Stability Testing Platform based on Chaos Engineering》研究了混沌工程在可扩展分布式数据库中的应用，提出了一种新的混沌测试方法，并通过实验验证了其有效性[9]。《Testing Resiliency of AWS Services Using the Concept of Chaos Engineering》提出了一种通过混沌工程评估分布式数据库弹性的新方法，并通过实验验证了该方法的有效性[10]。

在本项目中，研发团队在自有的自动化测试平台(SuperMan)基础上，结合混沌测试经验和GoldenDB分布式数据库特点，进行了自研开发。

Superman测试平台和GoldenDB目标系统的整体组网如图5所示：

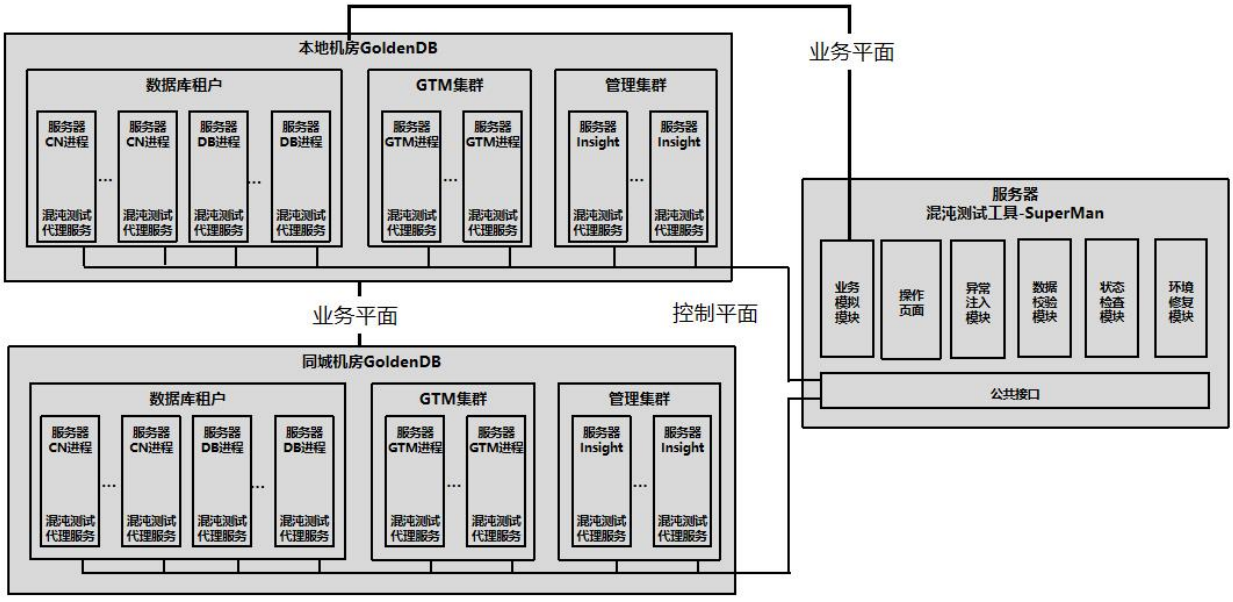

图 5 混沌测试逻辑组网图

其中左侧GoldenDB为混沌测试目标系统，按本地机房和同城机房高可用方式进行部署。右侧是SuperMan混沌测试工具。SuperMan部署在独立服务器上，通过网络控制平面与GoldenDB本地机房、同城机房服务器互联，并在每台服务器上部署混沌测试代理服务，该服务接收SuperMan控制命令，进行异常注入、收集校验数据、状态检查和环境修复操作。SuperMan内部主要由六个模块组成，其中业务模拟模块通过网络业务面向GoldenDB数据库模拟正常业务的背景压力，在压力大的情况下该模块部署在多台服务器上；测试人员在操作页面上配置混沌测试规则，发起混沌测试任务，查看测试结果；异常注入模块按操作页面上配置的策略，向GoldenDB服务器注入异常；注入异常后由数据校验模块和状态检查模块对数据和系统进行检查，并在测试完成后由环境修复模块对系统进行修复。

SuperMan支持两种异常测试模式：遍历执行和混沌执行。一般先进行遍历执行，再进行混沌执行。

遍历执行是向GoldenDB分布式数据库的各个模块逐个执行异常（异常列表见表1）。每执行一个异常后，调度状态检查模块和数据校验模块进行检查，如果符合预期再继续执行下一个异常；如果不符合预期，自动停止测试，并对结果进行记录，由人工介入排查原因，并调度环境修复模块进行修复。

混沌执行在遍历执行的基础上增加了多模块的随机组合，具体是随机选择多个模块发起异常，每个模块执行多个异常组合，并在每个异常之间增加时间随机性（rand_sleep），这种模式更符合

生产实际故障过程。GoldenDB混沌测试异常注入方式如图6所示。

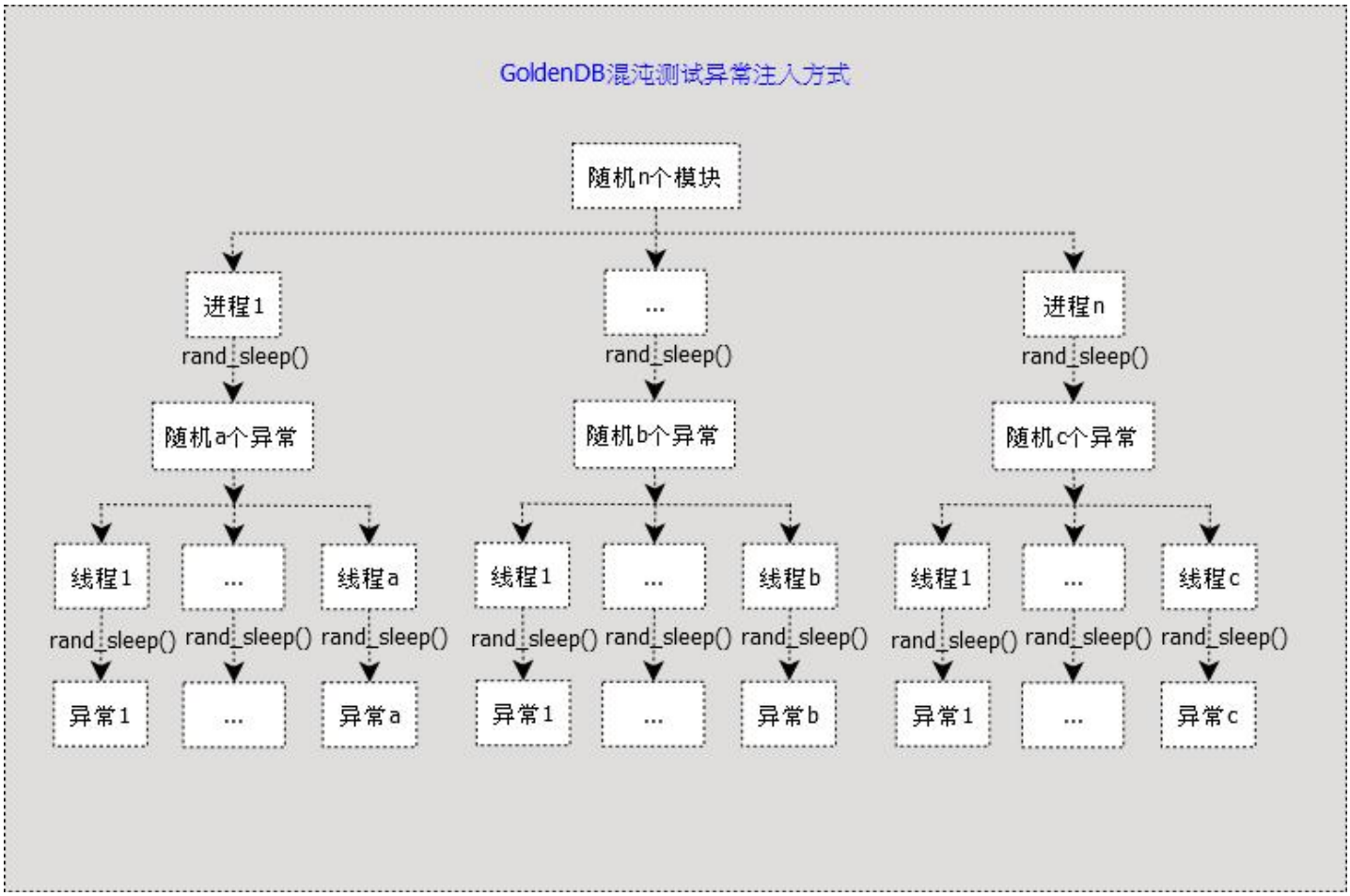

图 6 GoldenDB 混沌测试异常注入方式

GoldenDB研发团队根据混沌工程和分布式数据库特点，整理出八类异常注入规则，可由测试人员在SuperMan操作页面上进行配置和调度。

详细如表1：

表 1 异常注入规则列表

进程异常	网络包异常	运营操作	突发操作
1、dbmoni -stop 2、dbstop 3、server stop 4、gdb attach（夯住）5、kill 6、kill -9 7、kill -stop pid 8、pkill -u 9、pkill -9 -u ……	1、模拟网络延迟 2、模拟网络丢包 3、模拟网络包重复 4、模拟网络包损坏 5、模拟网络包乱序 6、网络闪断 7、网络终断 8、模拟网络异常 9、IP 冲突/IP 被修改 ……	1、在线 DDL 2、主备切换 3、数据备份 4、在线数据重分布 5、数据导入导出 6、密码修改 7、analyze 操作 8、optimize 操作 9、主备数据一致性检查 ……	1、非法 SQL 注入 2、SQL 多样性注入 3、并发慢 SQL 4、大结果集返回 5、锁冲突过高 6、并发客户端过多 7、大事务 ……
系统资源瓶颈	**文件异常**	**服务器硬件异常**	**组网设备异常**
1、CPU:>90%+ 2、内存不足 3、IO 过高：~=100% 4、inode 耗尽 5、磁盘空间满 6、句柄数过多 ……	1、文件只读权限 2、文件只写权限 3、文件无读写权限 4、文件不存在 5、文件损坏 ……	1、硬盘损坏 2、CPU 损坏 3、内存条损坏 4、网卡损坏 5、raid 卡电池不足 6、服务器断电 7、网口线掉落 8、服务器 crash 9、强制关机 ……	1、DNS 异常 2、负载均衡异常 3、交换机断电 4、交换机侧网口线掉落 5、网络风暴 6、系统变为只读 ……

SuperMan完成测试后，生成测试结果。如图7所示：

index_id	group_id	make_time	device_type	ip	port	abnormal_type	abnormal_info	duration	data_consistency	recovery_time
1881	10	2020-09-08 16:36:02	GTM	10.229.42.178	6026	file_abnormal	4:make_file_none_existent()	16	consistency	0
1880	10	2020-09-08 16:35:37	DBPROXY	10.229.42.172	6451	network_abnormal	4:tc_qdisc_netem_corrupt()	18	consistency	0
1879	9	2020-09-08 16:23:48	DB	10.229.42.174	5518	network_abnormal	2:network_flash_break()	11	consistency	0
1878	9	2020-09-08 16:23:06	DBPROXY	10.229.42.172	6451	network_abnormal	3:tc_qdisc_netem_delay()	23	consistency	0
1877	8	2020-09-08 16:11:22	DBPROXY	10.229.42.172	6451	file_abnormal	4:make_file_none_existent()	5	consistency	9
1876	8	2020-09-08 16:11:04	DB	10.229.42.170	5518	network_abnormal	1:network_disconnection()	20	consistency	9
1875	7	2020-09-08 15:57:45	DB	10.229.42.173	5518	file_abnormal	2:make_file_write_only()	54	consistency	4
1874	7	2020-09-08 15:57:01	DBPROXY	10.229.42.175	6451	process_abnormal	1:pkill -u $USER	24	consistency	4
1873	6	2020-09-08 15:45:23	DB	10.229.42.173	5518	file_abnormal	2:make_file_write_only()	7	consistency	10
1872	6	2020-09-08 15:45:12	GTM	10.229.42.179	6026	file_abnormal	4:make_file_none_existent()	3	consistency	10
1871	5	2020-09-08 15:31:12	DB	10.229.42.171	5518	process_abnormal	6:kill $PID	51	consistency	33
1870	5	2020-09-08 15:30:30	GTM	10.229.42.179	6026	process_abnormal	4:gdb attach $PID &	56	consistency	33
1869	4	2020-09-08 15:17:14	DB	10.229.42.174	5518	network_abnormal	2:network_flash_break()	54	consistency	0
1868	4	2020-09-08 15:17:05	DBPROXY	10.229.42.175	6451	network_abnormal	7:tc_qdisc_netem_recorder()	51	consistency	0
1867	3	2020-09-08 15:04:57	DBPROXY	10.229.42.175	6451	process_abnormal	1:pkill -u $USER	40	consistency	20
1866	3	2020-09-08 15:04:33	DB	10.229.42.174	5518	process_abnormal	7:kill -9 $PID	28	consistency	20
1865	2	2020-09-08 14:52:44	DB	10.229.42.174	5518	process_abnormal	3:mysql.server stop	41	consistency	9
1864	2	2020-09-08 14:52:08	GTM	10.229.42.179	6026	file_abnormal	3:make_file_none_permissions()	7	consistency	9
1863	1	2020-09-08 14:40:54	DB	10.229.42.170	5518	network_abnormal	5:tc_qdisc_netem_duplicate()	54	consistency	1
1862	1	2020-09-08 14:39:44	DBPROXY	10.229.42.172	6451	file_abnormal	1:make_file_read_only()	57	consistency	1
1861	11	2020-09-08 11:25:18	DB	10.229.42.174	5518	process_abnormal	5:dbstop	29	consistency	10
1860	11	2020-09-08 11:24:15	DBPROXY	10.229.42.172	6451	process_abnormal	3:mysql.server stop	43	consistency	10
1859	10	2020-09-08 11:12:42	GTM	10.229.42.178	6026	process_abnormal	4:gdb attach $PID &	49	consistency	0
1858	10	2020-09-08 11:12:12	DBPROXY	10.229.42.172	6451	process_abnormal	3:mysql.server stop	41	consistency	0
1857	9	2020-09-08 11:00:56	DB	10.229.42.174	5518	file_abnormal	1:make_file_read_only()	43	consistency	16
1856	9	2020-09-08 11:00:54	DBPROXY	10.229.42.172	6451	network_abnormal	4:tc_qdisc_netem_corrupt()	53	consistency	16
1855	8	2020-09-08 10:48:36	DB	10.229.42.174	5518	process_abnormal	8:jtool -was stop	4	consistency	14
1854	8	2020-09-08 10:48:22	DB	10.229.42.170	5518	network_abnormal	2:network_flash_break()	19	consistency	14
1853	7	2020-09-08 10:37:08	DB	10.229.42.173	5518	file_abnormal	4:make_file_none_existent()	3	consistency	0
1852	7	2020-09-08 10:37:07	DBPROXY	10.229.42.172	6451	network_abnormal	5:tc_qdisc_netem_duplicate()	38	consistency	0

图 7 混沌测试结果

测试结果中记录了被测试的GoldenDB模块类型（device_type）和位置（ip、port），异常类型信息(abnormal_type、abnormal_info），以及异常的持续时长（duration，单位s）、测试完成后数据是否一致性（data_consistency）以及业务恢复时间（recovery_time，单位s），最后两个指标分别用于评估数据一致性和服务可用性，若未达到目标要求，同样需要标注为问题进行分析和迭代优化。

本项目中通过混沌测试工具共发现133处故障缺陷，通过对故障缺陷进行修复，有效提升了GoldenDB分布式系统的健壮性和稳定性。大部分故障场景均发生在极端异常场景下，正常测试用例设计难以覆盖，从测试结果中也验证了混沌测试的价值性。详细如表2所示：

表 2 混沌测试实践效果

序号	故障种类	故障数量
1	进程内异常代码保护	38
2	模块间多副本异常功能保护	28
3	集群间异常故障保护	23
4	网络异常故障保护	12
5	故障下性能优化类	32

此外，Superman 实现自动化测试任务执行，异常用例达数千条，比对工人执行，回归测试执行工时从 800 人时降为 16 人时，版本发布的验证周期缩短 35%。

6 解决数据库软件兼容性难题

本项目通过技术攻关，实现分布式数据库对传统数据库语法高度兼容能力。原CRM/BOSS系统基于传统数据库进行开发，使用了序列（SEQUENCE）、视图（VIEW）、同义

词（SYNONYM）、自定义函数（FUNCTION）、存储过程（PROCEDURE）、自定义类型（TYPE）等高级数据库对象，迁移到分布式数据库后，有较多语法不兼容；此外，由于架构差异，部分SQL性能会显著下降。如果仅依靠应用进行改造适配，项目工程量过大，迁移风险增高。

为了减少迁移过程中的工程量和风险，四川移动联合金篆信科研究和实现分布式数据库与传统集中式数据库语法能力对齐，借助GoldenDB的迁移辅助工具，识别异构数据库之间的兼容性问题，进行大量语法兼容开发，解除存量应用与传统数据库深度绑定关系。

项目中使用了几个重要的辅助工具。

如图8所示：

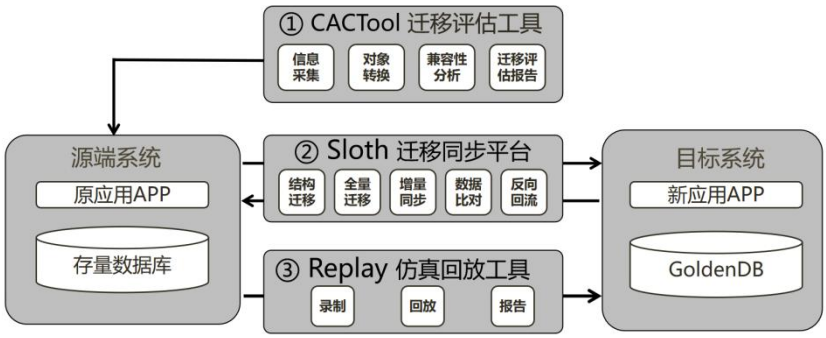

图 8 使用 GoldenDB 工具集辅助项目实施

CACTool迁移评估工具。该工具以只读方式访问现网存量数据库，采集其库表结构、各种数据库对象以及过往SQL语句，导出适配GoldenDB语法对象，在导出过程中自动进行转换，减少了大量人为适配改造工作，同时生成迁移评估报告，给出迁移可行性和兼容性客观翔实的评估数据。

如图9所示：

图 9 CACTool 工具生成兼容分析报告

Sloth迁移同步平台。Sloth用于异构数据库的全量迁移、增量同步和数据比对。将数据从源库以全量或增量方式迁移过来后，Sloth校验两者数据的正确性，可以快速校验、多批次执行。该工具大大减少了人工执行工作量，减少出错。此外，Sloth是一个可视化平台，通过可视化页面进行迁移任务管理，有效降低了迁移复杂度。如图10所示：

图 10 在 Sloth 迁移平台工具上管理迁移任务

Replay仿真回放工具。在GoldenDB正式接替存量数据库投产前，通过实时采集存量数据库上的WCR文件，将其还原为SQL语句，向GoldenDB进行回放。通过这种方式，使用来自生产上真实的请求来验证GoldenDB数据库执行的正确性、稳定性，同时也通过生产压力对GoldenDB处理性能进行充分验证。

在项目启动时使用兼容性评估工具（CACTool）进行兼容性分析，如表3所示。对分析出高频不兼容的条目（属于同一种类型，但出现次数很高），由数据库进行迭代开发，实现该兼容功能。举例year_month是GoldenDB的保留字，作为字段名会报错，共出现15588处；这种情况要求GoldenDB迭代开发，允许作为字段名出现。类似情况包括系统同义词（PUBLIC SYNONYM）、自定义函数、存量数据库私有函数等共有近上万差异点，均按上述方式进行版本迭代，减少了大量的应用修改。

对于其他一些差别则由应用评估后采取折中方案，如decimal小数位数，应用最多只需要保留6位小数，通过调整DDL语句中的精度，满足要求。

表 3 CACTool 兼容分析报告

对象类型	对象数	支持个数	不支持个数	兼容比率	备注
TABLE	100008	99962	46	99.95%	均为 decimal 小数位长度超过总长度，属于应治理的表结构定义问题，等价改写
SEQUENCE	49	49	0	100.00%	
VIEW	156	155	1	99.36%	使用 GoldenDB 保留字 year_month 作为字段名，GoldenDB 改造支持
SYNONYM	73121	72123	998	98.64%	支持公共同义词，GoldenDB 改造支持
FUNCTION	8	4	4	50.00%	自定义函数语法兼容，GoldenDB 改造支持
PROCEDURE	2	2	0	100.00%	
TYPE	2	2	0	100.00%	
Query	790978	775391	15587	98.03%	大部分使用 GoldenDB 保留字 year_month 作为字段名，GoldenDB 改造支持
汇总	964324	947688	16636	98.27%	整体兼容率达到 98%

四川移动已自建一些数据库运维辅助平台，通过访问数据库的系统视图，对数据库进行可视化管理，这部分功能在面向分布式数据库运维时继续保留，这要求GoldenDB兼容原数据库的系统视图功能。

项目中完成的系统视图兼容如表4所示：

表 4 GoldenDB 兼容的系统视图

系统视图功能	系统视图名称	系统视图功能	系统视图名称
查看所有对象	user_objects	查看存储过程	user_procedures
查看表	user_tables	查看表分区	dba_tab_partitions
查看表字段	user_tab_columns/describe	用户	dba_users
查看索引	all_indexs	查看用户权限	dba_sys_privs
查看索引明细	all_ind_columns	查看会话	v$session
查看序列	user_sequences	查看进程	v$process
查看视图	user_views	查看死锁	dba_blockers+dba_waiters
查看同义词	user_synonyms	表使用大小	dba_segments

除了兼容语法外，分布式数据库对存量数据库的性能要保持兼容，否则容易出现性能不达标问题，阻塞项目推进。分布式数据库由于计算和存储分离，数据需要通过网络跨节点传输，特定的SQL性能恶化明显。通过分布式优化器增强优化能力，选择合理执行路径，是解决性能问题的有效手段[11]。项目中实现了两个重要的优化方向：

（1）内置并行执行引擎。报表经分类的 SQL 语句，需要扫描的数据量大，对数据进行聚合处理，执行耗时长。项目中通过 GoldenDB 内置并行执行引擎来优化性能，首先将数据按分片规则打散到多个分片（Sharding）上并行执行，其次根据数据在分片内的分布范围，自动分拆成多个任务（work），每个任务执行一个数据片段，多个任务并行执行。

未分成并行任务需要耗时：

$$totallatency(n) = \sum_{i=1}^{n} worklatencyi \qquad (2)$$

分成为并行任务需要耗时：

$$Max(work\ latency_1, work\ latency_2, \ldots, work\ latency_n) \qquad (3)$$

因此分拆后处理耗时显著降低。

（2）增强分布式优化器。引入基于代价优化（Cost Based Optimizer）等能力，在分布式优化阶段，基于统计直方图（histogram）评估更多的可选路径，并引入分布式数据库节点之间网络传送的成本，从中选择最优的执行路径。

以CRM/BOSS应用典型的三户表为例，如下SQL语句查询某用户订购合同名和账户名：

```
SELECT C.CONTRACT_NAME ,B.ACCOUNT_NAME
FROM CONTRACT_INFO C,     --订购关系表
ACCOUNT_INFO B,           --账户表
USER_INFO A               --用户表
WHERE C.CONTRACT_NO = A.CONTRACT_NO
AND B.ACCOUNT_NO = A.ACCOUNT_NO
AND A.ID_NO = #ID_NO# --#...#为过滤条件参数
```

如果基于规则的优化（Rule Based Optimizer），需要将账户表和订购关系表的所有数据都从数据分片传送到计算节点，进行循环（Loop）处理。

优化前的网络传输开销：

$$transferredrowscost = sum(1, \sum_{i=1}^{m} rows(i), \sum_{j=1}^{n} rows(j)) \quad (4)$$

其中，rows（i）为账户表在每个分片上的记录行数，rows（j）为订购关系表在每个分片上的记录行数，估计的网络传输开销近13500万行(1+4500万+9000万)。

基于统计直方图可选择更多执行路径，先执行用户表查询（估计命中1条记录），然后将该用户对应的CONTRACT_NO、ACCOUNT_NO作为条件，再次下推给DN节点进行条件过滤，网络传输开销：

$$transferredrowscost = sum(1, hitrows(i), hitrows(j)) \quad (5)$$

其中，hit rows（i）为账户表在数据分片上命中的记录行数，hit rows（j）为订购关系表在数据分片上命中的记录行数，估计的网络传输开销只有61行(1+10+50)。优化后的执行树类似如下SQL：

```
1.SELECT CONTRACT_NO,ACCOUNT_NO FROM
    USER_INFO WHERE ID_NO = #ID_NO#
2.SELECT CONTRACT_NAME FROM
CONTRACT_INFO WHERE CONTRACT_NO
IN(#CONTRACT_NO#)
3.SELECT ACCOUNT_NAME FROM
ACCOUNT_INFO WHERE ACCOUNT_NO IN
(#ACCOUNT_NO#)
--其中语句2、3并行执行 #...#为过滤条件输入参数
```

通过上述优化效果，分布式优化器选择小表驱动大表（Small Table Driving Large Table）的执行路径，优化前总耗时2.5分钟，优化后耗时3毫秒。

优化前后的直观原理如图11所示：

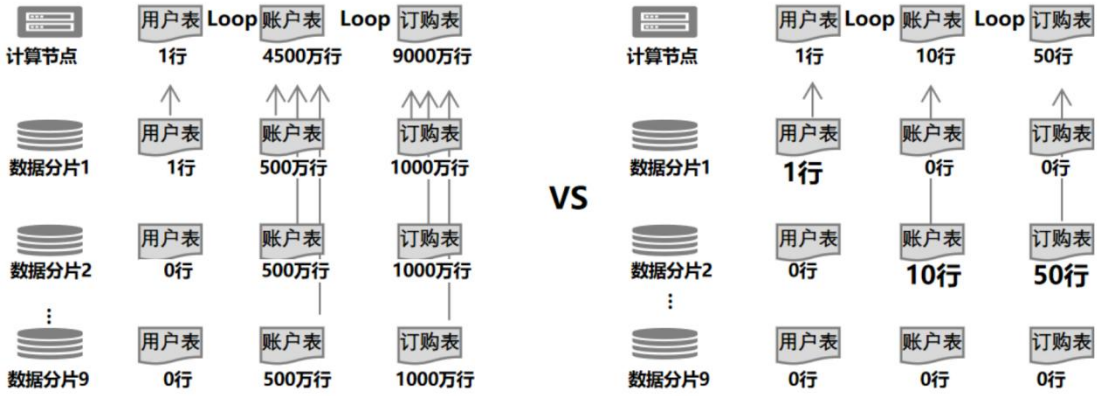

图 11 基于成本的优化减少网络传输开销

7 全线路增量迁移解决割接窗口难题

CRM/BOSS各系统数据量规模大，以营销交易为例，数据量在10TB级别。迁移过程中对数据模型进行转换，为了降低大规模数据下模型重构导致迁移数据时间过长、业务停机过长的风险[12]，创新地进行迁移架构设计，通过全路径增量迁移方法，将主体数据提前迁移，割接当晚仅做增量数据迁移，大大降低停机迁移时间窗口，有效保障迁移实施成功。

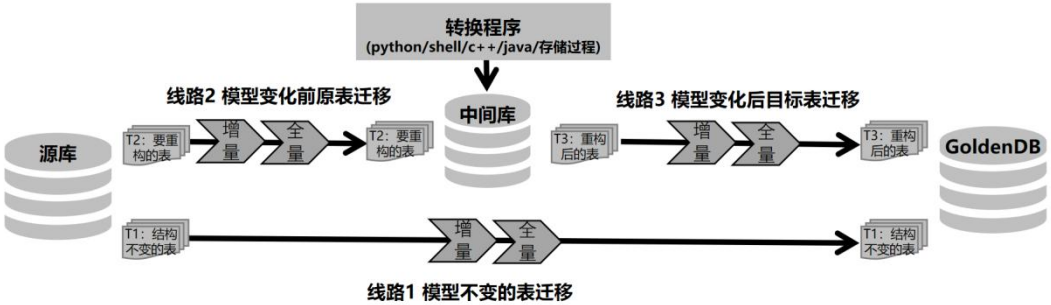

图 12 全流程增量同步过程

为支撑表模型变化，在迁移中实施中间库技术，将模型转换的复杂处理过程在中间库上实施，转换为目标模型后再迁移至目的数据库。如图12所示，整个迁移线路分为以下三条：

线路一：模型不变化的数据表，从源库一比一的同步至GoldenDB目标库。

线路二：模型变化的表，先将源表同步至中间库，在中间库上实施转换，转换模式包括多表合一、单表分拆、表结构调整、错误数据清洗等。这部分可以使用python、shell、c++、java以及存储过程等方式进行数据转换。

线路三：转换后的目标表，从中间库一比一的同步至GoldenDB。

由于迁移数据大，中间数据模型转换复杂，为了降低迁移时间窗口，三条线路都实施先全量再增量的迁移过程。

割接前先将表全量数据从三条线路上迁移到GoldenDB数据库，割接当晚只做增量数据迁移，总时长从7小时缩短至1.5小时，数据迁移时长大大降低，有效保障割接成功。如表5所示：

表 5 全量和增量迁移时间对比

同步线路和阶段	全量	增量
数据量	10+TB/6000+张表	500+万条/2000 张表
线路 1	<5 小时	<30 分钟
线路 2->转换->线路 3	5 小时	1 小时
数据比对	2 小时	30 分钟
合计	7 小时	**1 小时 30 分钟**

使用上文中提到的GoldenDB迁移辅助工具进行迁移，并在迁移过程中进行数据比对。应用团队主要投入在中间库数据转换脚本的开发上，迁移辅助工具加持减少了迁移人员投入和工程量。

8 投资效益分析

该项目完成数据库从集中式向分布式架构转型，全系统采用PC Server服务器建设，不再需要采购小型机和专用存储设备，在相同硬件成本下，业务处理性能提升3倍，综合维护成本下降30%。

本次项目中引入混沌测试工程，帮助数据库软件提升版本发布效率，版本验证周期缩短35%；通过提升分布式数据库语法兼容性能力，减少应用改造成本，比预算降低30%；使用迁移全流程工具，数据迁移投入人力下降40%，项目实施工期减少20%。

此外，得益于分布式架构弹性扩容能力，在业务扩容时无需停止服务，平均每年停服实施时间减少6小时，业务连续性得以增强，客户满意度得到提升。

9 结语

该项目是四川省使用分布式交易型数据库在移动核心系统中规模商用，初步显现分布式架构技术优势。利用分布式数据库的良好扩容和高可用架构技术，通过全新的分布式平台设计，可靠性和性能大幅提升，可靠性超99.9999%，在同等硬件成本下，业务整体处理性能提升三倍。此外本项目：

引入了混沌测试模型，攻克了分布式系统复杂性的软件工程难题。在具体实施过程中设计了分布式数据库混沌测试方案，该方法有效提升分布式数据库的健壮性和稳定性，大大缩短了多平台架构下版本验证的时间。

实现分布式数据库对传统数据库语法高度兼容能力，做到应用平滑迁移。通过推动分布式数据库语法兼容工作，解除存量应用与传统数据库深度绑定关系，有效降低整个项目应用改造成本。

实现异构芯片混合架构，夯实了硬件供应链安全基础。该项目实施多线路服务器和芯片混合组网，在一套数据库上同时部署海光服务器和鲲鹏服务器，避免依赖单一硬件厂商带来供应链风险和技术线路风险。

实现了模型重构下的增量迁移，减少了系统停服时间。通过全路径增量迁移方法，大大降低停机迁移时间窗口，有效保障迁移实施成功。

该项目充分验证了从传统集中式数据库向新型分布式数据库转型已具备成熟条件，在项目过程中积累了大量实施经验和辅助工具，提升了通信行业大规模应用架构升级的效率，降低了项目实施工程量。

项目中为异构数据库语法兼容性难题提出解决方法，降低了语法兼容性的应用改造，但目前仍无法做到对传统数据库所有语法的完全兼容，需要数据库产品在未来项目实施过程中逐步提升。分布式数据库通过分片技术提高了数据存储容量和吞吐率，但也增加了数据在网络间传输的开销，通过分布式优化技术减少复杂查询的网络传输和计算时间，提升分布式数据库整体性能，是分布式数据库未来重要的技术攻关方向。

参考文献

[1] Corbett J C, Dean J, Epstein M, et al. Spanner: Google’s globally distributed database[J]. ACM Transactions on Computer Systems (TOCS),2013,31(3): 1-22.

[2] 中国信息通信研究院云计算与大数据研究所,中国支付清算协会金融科技专业委员会.金融分布式事务数据库白皮书[Z].2018.

[3] 中国人民银行. 分布式数据库技术金融应用规范:JR/T 0204-2020[S]. 北京: 全国金融标准化技术委员会, 2020.

[4] Gilbert S, Lynch N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services[J].ACM Sigact News,2002,33(2): 51–59.

[5] Lamport L. Paxos Made Simple[J].ACM Sigact News (Distributed Computing Column),2001,32(4):51-58.

[6] 李华, 王磊. 分布式数据库一致性协议的性能优化研究[J]. 计算机工程, 2021, 47(12): 40-48.

[7] 王宪刚,孙晓璇.分布式核心系统混沌测试探索与实践[J].金融电子化,2022(02):75-76.

[8] 王蒙,黄永厚.基于混沌工程的数据库故障演练平台研究与实践

[J].网络安全和信息化,2024(10):58-60.

[9] Wang C, Yang J, Han X, et al. Research on Distributed Database Stability Testing Platform based on Chaos Engineering[C]//2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). IEEE, 2023: 2092-2097.

[10] Assudani P J, Kadu R K, Yenurkar G, et al. Testing Resiliency of AWS Services Using the Concept of Chaos Engineering[C]//2024 5th International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV). IEEE, 2024: 352-356.

[11] Hong-fei Z. 分布式数据库系统的查询优化技术研究[J]. 电脑与电信, 2020, 1(3): 55-58.

[12] Mateen A, Ali K. Optimization strategies through big-data migration in distributed cloud databases[C]//2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI). IEEE, 2017: 96-99.

作者简介

杨孝平（1977—）男，硕士，高级工程师，主要研究方向：分布式数据库、互联网技术。

张旭辉（1979—）男，本科，高级工程师，主要研究方向：大数据建模、互联网技术。

温健军（1977—）男，硕士，高级工程师，主要研究方向：云计算、数据库。

杨名（1982—）男，硕士，高级工程师，主要研究方向：云计算、数据库。

秦延涛（1978—），男，本科，高级工程师，主要研究方向：数据库、云计算。

吕伟初（1978—），男，硕士，高级工程师，CCF 高级会员，主要研究方向：数据库、分布式架构。

朱宸希（1991—），男，本科，主要研究方向：4G/5G 无线通信系统，数据库。

Application practices on hybrid platform distributed database technology

YANG Xiaoping[1] , ZHANG Xuhui[1], WEN Jianjun[1], YANG Ming[1], ZHOU Riming[2], QIN Yantao[2], ZHU Chenxi[2],Lǔ Weichu[2]

1. China Mobile Communications Group Sichuan Co., Ltd., Chengdu 610097,China

2. Jinzhuan Xinke Co., Ltd., Beijing 100176,China

Abstract: This study focuses on the transformation process of databases from traditional centralized platforms to distributed platforms, and deeply analyzes the key challenges faced in the transformation process, such as hardware reliability, software compatibility, and data migration. To address these technological challenges, this study innovatively proposes multiple technical solutions: proposing an innovative business continuity assurance technology that enables rapid switching on disaster recovery under distributed architecture conditions, ensuring business continuity requirements; Implemented a heterogeneous chip hybrid architecture platform, solved the problems of chip architecture differences and functional compatibility verification through innovative software engineering methods, avoided performance shortcomings through reasonable networking, and achieved version uniformity of databases in source code engineering; In response to the challenges of data migration caused by changes in data models, an incremental transformation migration method was proposed to achieve smooth migration throughout the entire process and reduce the risk of cutover. This research provides a complete solution and strong technical support for distributed architecture transformation projects, which has important theoretical significance and practical application value.

Keywords: Domestic database , Distributed system , Distributed database , Database migration