# 一种基于GPU加速和关联事务融合的全同态数据库中间件

姓名：关荣鑫

www.huawei.com



HUAWEI

# 专利申请合规Checklist

| 检查项类型 | 检查提示项 | 确认结果<br>[不涉及，是，否] |
| --- | --- | --- |
| **是否直接引用外来非公开信息** | （1）Idea/交底书/专利申请文档不涉及在外部交流、技术合作、产品合作、采购等场景中从他人获取到的外来非公开信息，包括但不限于他人资料、仿真数据、测试结果、模型图、原理图、器件结构图、软件算法和其他非公开技术方案等；尤其注意不能引入带有他人标志或保密标识/水印的图片或资料； | 否 |
| **是否以外非信息为基础进行改进** | （2）Idea/交底书/专利申请文档不存在以外非信息的技术方案或技术构思为基础或主要设计参考而提出的改进方案，尤其注意不能在背景技术中引用外非 | 否 |
| **是否涉及外网引入** | （3）Idea/交底书/专利申请文件不涉及从外部网站下载的有保密义务和/或使用限制的信息，如在线签署保密协议或者最终用户协议中限制分发或商业使用等 | 否 |
| | （4）Idea/交底书/专利申请文件不涉及利用个人注册账号等途径从外部网站下载的非公开信息 | 否 |
| **职务成果归属** | （5）Idea/交底书/专利申请文档所涉及的发明方案100%属于发明人在华为的职务成果，不涉及前雇主/前研究机构/学校的职务成果 | 是 |
| **其他** | （6）Idea/交底书/专利申请文档等使用的技术材料中，不涉及来自于其他部门/其他产品线或关联公司，不包括来源不明的或无法确定是否属于外非信息的图片及资料 | 不涉及 |
| | （7）Idea/交底书/专利申请文档不涉及其他公司的私有协议、私有接口、私有标准等。 | 不涉及 |
| | （8）Idea/交底书/专利申请文档不涉及通过对未上市产品或软件通过拆解、反编译、反汇编等技术手段获取到的信息 | 不涉及 |
| **是否涉及出口管制** | （9）如果Idea含有海外发明人，请确认技术是否达到出口管制规格。如是，是否已获得相应的出口许可? | 否 |
| **发明人署名确认** | （10）确保所有发明人都做出了实质性贡献。（包括：a. idea的初始创造者，关键技术路线设计者； b.《技术交底书》主笔人，技术实施例完整设计者；c.对idea的至少一个创新点的新颖性或创造性提供了有帮助的技术特征者。） | |

# 发明背景 - 云端数据库广泛应用于各行各业

- 在数据规模爆炸增长的时代，越来越多政府、企业，与个人使用云端数据库存储海量私有数据，享受云端数据库例如弹性扩展等诸多优点。

## 案例一：政务数据管理

政府存储海量政务数据（例如财政信息和公民身份信息）于云端。政府查询与更新财政信息和公民身份信息。

图1：陕西省财政厅采用华为云 ServiceStage 重构数字财政，实现政府财政信息的统一化存储与管理。


## 案例二：金融数据分析

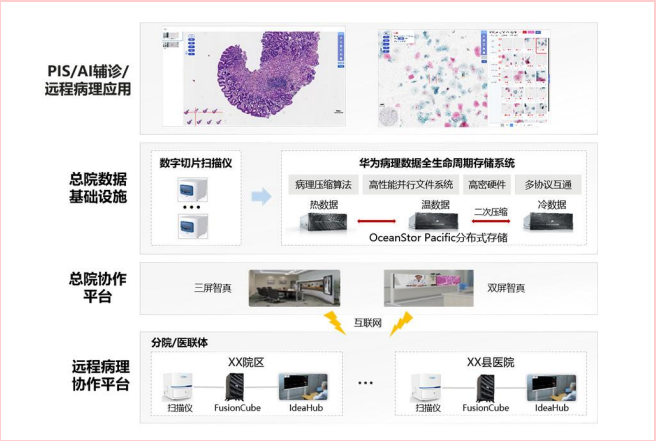
银行存储海量金融数据于云端。银行查询云端数据库进行融合数据分析和异构数据整合管理。

图2：工商银行采用华为 GaussDB (DWS) + Fusioninsight MRS 实现湖仓一体的融合数据分析解决方案。


## 案例三：医疗数据诊断

医院存储海量医疗数据于云端。医院查询病理数据以进行高效的阅片诊断。

图3：上海瑞金医院采用华为 OceanStor Pacific 分布式存储，实现数字化病理高性能调阅和数据缩减



# 发明背景 - 云端密态数据库极具商业价值

- 两大商业动机推动云端密态数据库发展
  - **数据机密性是数据库核心需求**：用户私有数据具有高度机密性。当部署于云端数据库时，一旦数据明文泄漏，恶意攻击者便可攫取机密数据进行牟利。因此，必须保证云端数据库的数据机密性才能释放其商业价值。
  - **法律法规强制保护数据机密性**：政府立法保护机密数据，例如中华人民共和国数据安全法。因此，必须保证云端数据库的数据机密性才能推动其合法商业落地。
- 工业界亟须云端数据库保障数据机密性，投入海量资源研发“密态数据库”。
  - 案例1：华为 GaussDB 迫切需要密态数据库解决方案，在云存储，金融，电子商务等领域实现数据的可算不可见（摘录自 GaussDB Full Encryption [VLDB’21]）。
  - 案例2：微软 Azure SQL AE 依靠 TEE 实现了基于 AES256 算法的数据透明加解密，但具有极大局限性，例如不支持密文层级直接运算，依赖 TEE 硬件环境（摘录自 Azure SQL Database Always Encrypted [SIGMOD’20]）。
  - 案例3：MySQL Enterprise Encryption 提供基于非对称加密算法的列级数据加密能力，但面临和 Azure SQL 数据库类似困境。
- 学术界发表众多关于密态数据库的顶级国际会议论文
  - CryptDB [SOSP’11], Senate [Security’21], Titanium [Security’22], HE3DB (CCS’23) ...
  - 然而，现有研究存在三大痛点：
    - ◆性能低下：低吞吐、高时延，为了完成特定的密态数据库 CRUD 任务须要承受极高的密码学代价。
    - ◆安全易损：执行密文事务需进行密文解密、明文运算与再加密的过程，容易遭受数据泄露。
    - ◆架构单一：绑定单一特定数据库架构（例如 SQL 数据库），无法推广到其他多种数据库架构。

# 发明背景 - 理想的云端密态数据库应实现的目标

**安全性**

- 目标一：数据机密性
  - 在数据存储于**不可信**的云端数据库的全生命周期中，数据明文不被泄露。
- 目标二：数据完整性
  - 在数据传输、存储和计算过程中，数据不被恶意篡改或丢弃。

**功能性**

- 目标三：密文可算性
  - 支持密文层级运算，避免对数据进行密文解密、明文运算、再加密，从而避免明文泄露与优化运算性能。
- 目标四：执行高效性
  - 提供超越现有主流密态数据库的事务处理高吞吐量和低时延，尽可能逼近非密态数据库的性能。
- 目标五：异构支持
  - 支持 SQL 和 NoSQL 等多种数据库架构。

# 发明背景 - 全同态加密

- 全同态加密（Fully Homomorphic Encryption，FHE）是一类强安全的新型非对称加密算法。
- 全同态加密**可抵抗量子攻击**。它基于容错学习问题(LWE/RLWE)或格密码学最短向量问题(CVP)，对比当前主流的非量子安全的非对称加密算法（例如 RSA）更具安全性。
- 可以直接对全同态加密的密文进行**加法**和**乘法**运算，而无需经历传统加密算法的不安全且开销高的运算流程（即先解密，明文运算，再加密）。

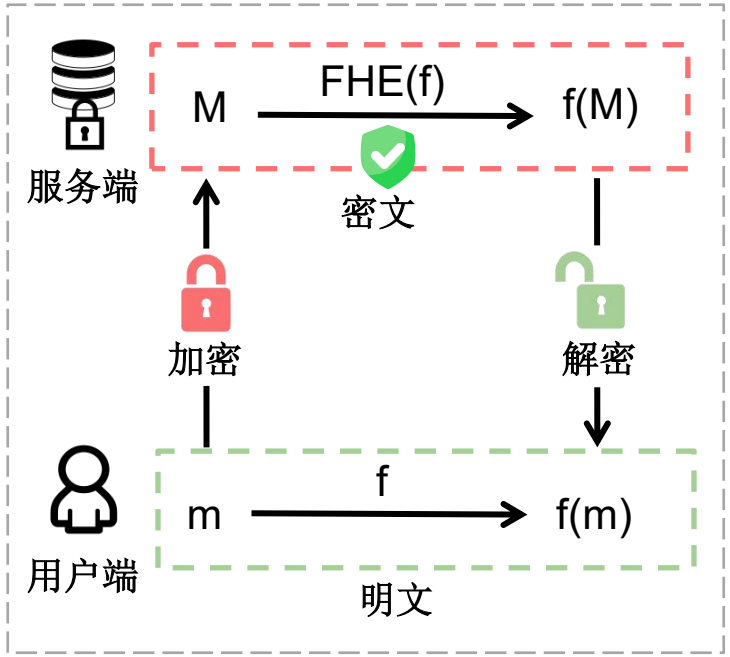



图4：全同态加密算法的运算流程

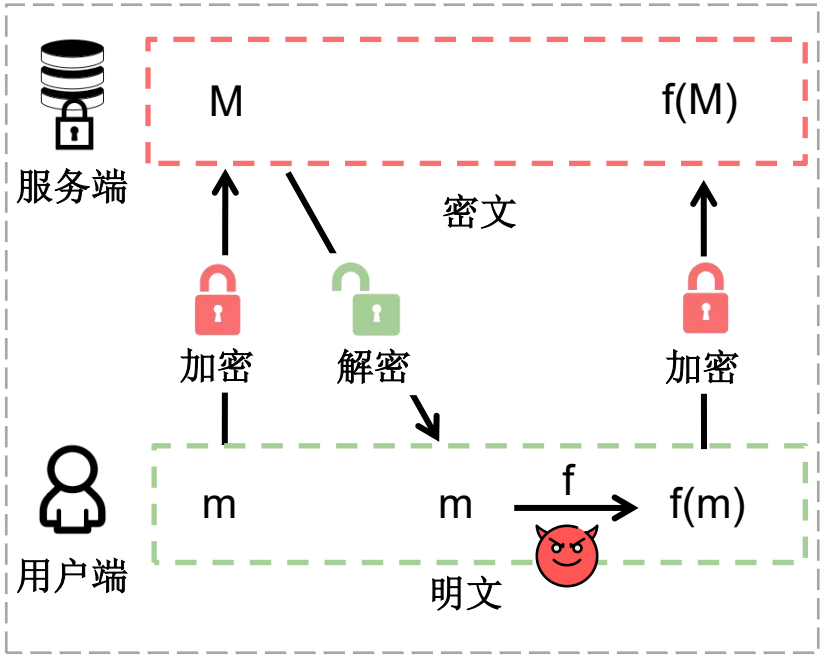



图5：传统加密算法的运算流程

# 发明背景 - 全同态加密

- 全同态加密发展迅猛，目前已迭代到第四代：
  - 第一代：基于格密码学，证明全同态加密的理论可行性，但性能不可用。代表作：[Gentry 2009]，[DGHV 2010]。
  - 第二代：基于LWE/RLWE，引入leveling schemes，支持整数算术运算，性能可用。代表作：[BFV 2012]，[BGV 2011]。
  - 第三代：基于第二代引入多种性能优化策略，例如避免relinearization和优化bootstrapping。代表作：[GSW 2013]，[FHEW 2014]。
  - 第四代：引入高效的密文取整操作，降低噪声增速，支持浮点数算术运算或布尔表达式运算。代表作：[CKKS 2017]，[TFHE 2016]。
- 本发明考虑纯软件的（不使用第三方可信硬件的）基于全同态加密的密态数据库场景，支持基于整数或浮点数算术运算的全同态加密算法。

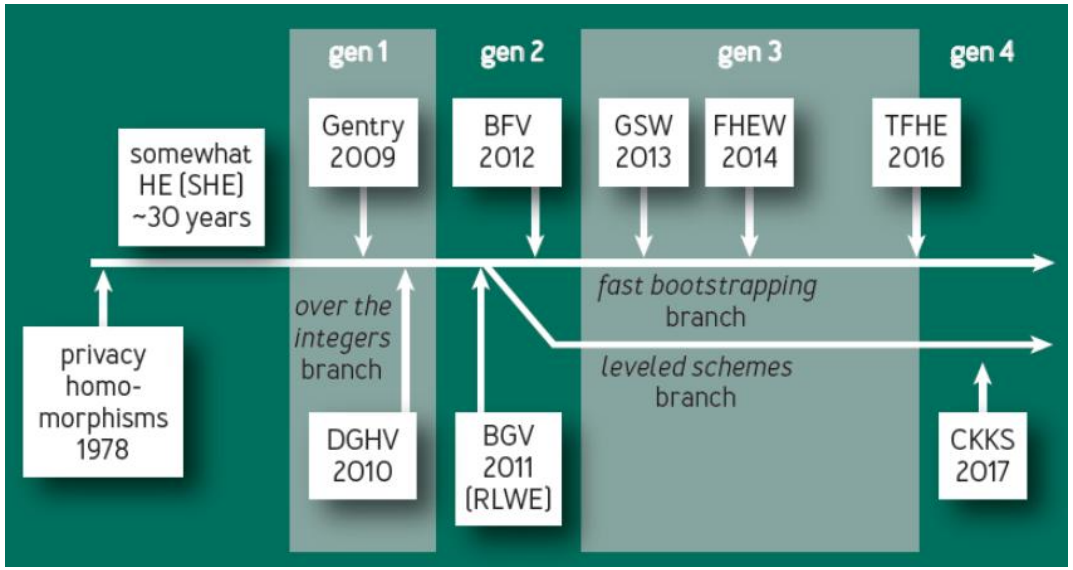


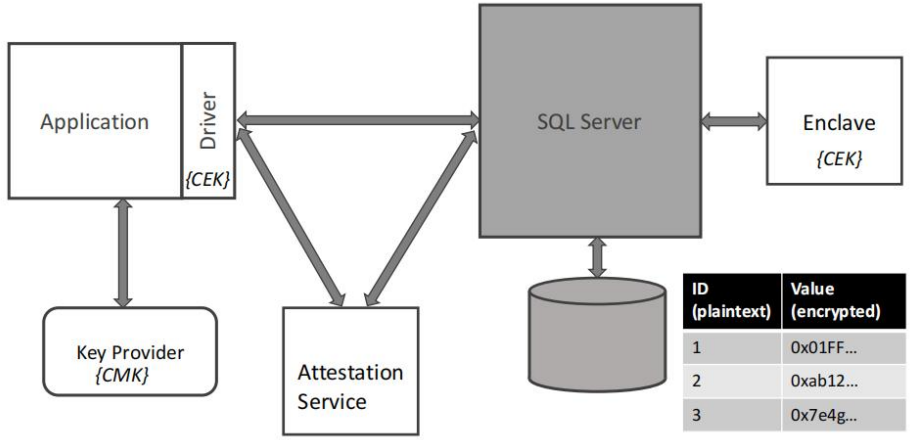

**图6：全同态加密算法发展路线**

# 现有工作 - 密态数据库

## 基于传统密码学的密态数据库

- 使用传统密码学算法加密关键数据
- 优点：
  - 增删查改性能优秀
- 缺点：
  - **不支持密文运算**，或需要在 TEE 中解密数据并对明文进行运算

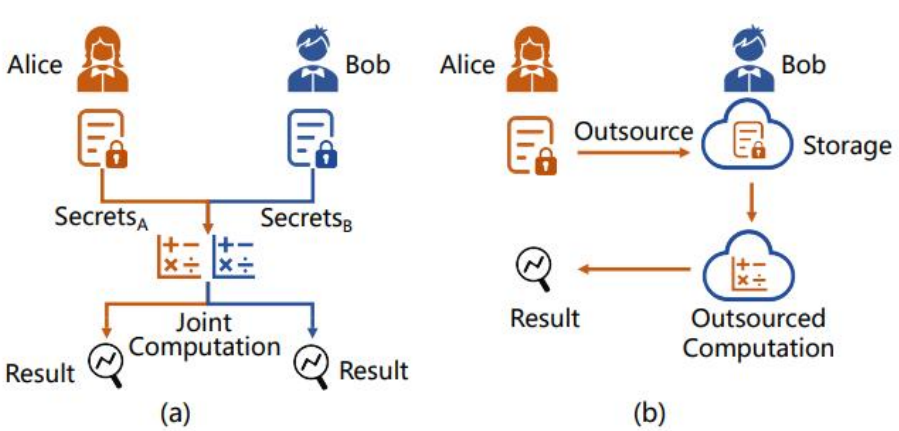
**图7：** Azure SQL AE 支持加密列级数据，并在 TEE 中提供明文比较和字符串匹配等复杂功能。




## 基于同态密码学的密态数据库

- 使用同态密码学算法加密全表数据
- 优点：
  - 支持密文查询和聚合运算
- 缺点：
  - **性能极差**，查询和修改为线性复杂度
  - **不支持多种异构数据库**

**图8：** HE3DB 使用 TFHE 完成查询的谓词比较操作，使用 CKKS 完成聚合运算。



# 发明目标 - 安全、高效、支持异构的全同态数据库中间件

- 本发明是首个使用GPU加速且支持异构数据库的全同态数据库中间件。
  - 安全性：使用全同态密文加密关键数据，可直接在全同态密文层级进行运算，保护关键数据不被泄露、篡改与删除。
  - 高效性：保持云端数据库操作原有复杂度，并使用GPU加速和关联事务融合技术加速全同态事务执行。
  - 异构支持：引入数据库驱动层对异构数据库算子进行统一抽象，为异构云端数据库（例如 SQL 和 NoSQL）提供支持。

| 系统 | 数据机密性 | 密文可算性 | 高效性 | 异构支持 |
|---|---|---|---|---|
| MySQL | ✗ | ✗ | ✓ | ✗ |
| Azure SQL AE [SIGMOD'20] | ✓ | ✗ | ✓ | ✗ |
| GaussDB [VLDB'21] | ✓ | ✗ | ✓ | ✗ |
| CryptDB [SOSP'11] | ✓ | ✓ | ✗ | ✗ |
| HE3DB [CCS'23] | ✓ | ✓ | ✗ | ✗ |
| 本发明GECO | ✓ | ✓ | ✓ | ✓ |

# 发明灵感 - 基于GPU加速和关联事务融合的全同态数据库工作流

## 现有工作痛点

**传统密态数据库性能低下且易泄露明文**

密文事务执行须经历密文解密、明文运算、再加密

**现有全同态数据库性能极差**

- 全同态密文比较需要进行全表扫描，无法查询海量数据
- 全同态算术运算开销高昂，难以应对高并发场景

**无法支持异构云端数据库**

绑定单一数据库架构(例如基于B+树的SQL数据库)，无法迁移到基于其他架构的数据库

## GECO 解决方案

**保护点1：多元加密策略**

使用全同态加密保护关键数据，使用明文/确定性加密存储参与谓词比较的数据，兼顾性能与安全

**保护点2：全同态数据库GPU二重加速协议**

使用GPU从单个同态算子和多个同态事务两个层面进行加速

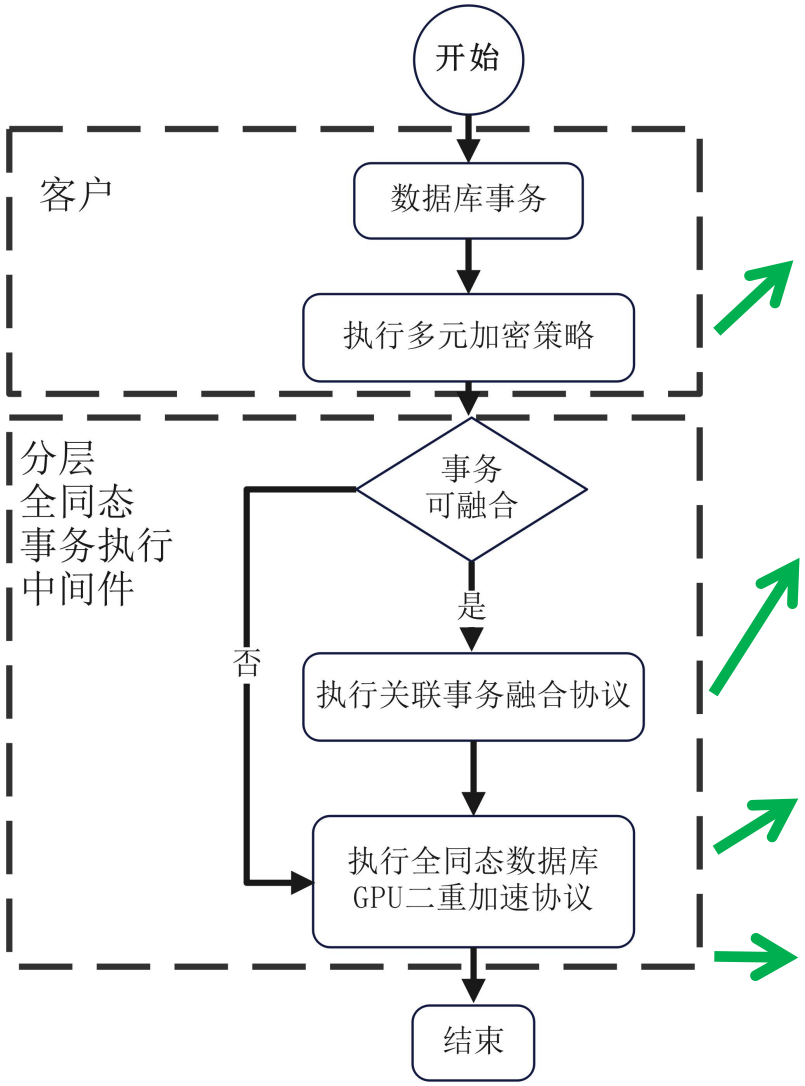
**保护点3：关联事务融合协议**

将多个同态事务融合为单一等价事务，减少密文运算次数

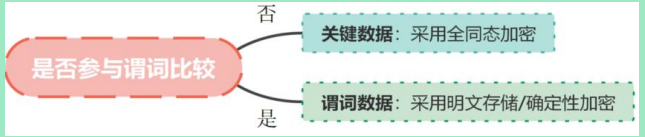
**分层全同态事务执行中间件**

使用中间件分层架构，为客户提供统一API，并通过提供特定于数据库的驱动适配异构云端数据库事务。

# 本发明工作流程与实现目标回顾

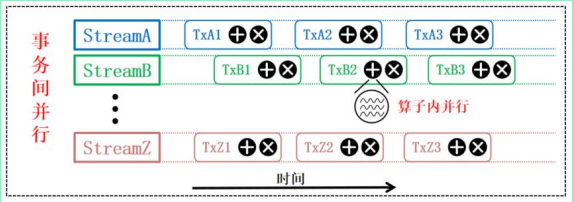



**保护点1：多元加密策略。**实现目标：(数据机密性&密文可算性)使用全同态加密保护关键数据，确保关键数据全生命周期明文不可见；(数据完整性)确保明文与密文数据不被篡改；(高效性)使用明文/确定性加密存储参与谓词比较的数据，避免全同态密文谓词比较。




**保护点3：关联事务融合协议。**实现目标：(高效性)减少事务执行与全同态算子运算次数，降低事务并发冲突导致的资源浪费。

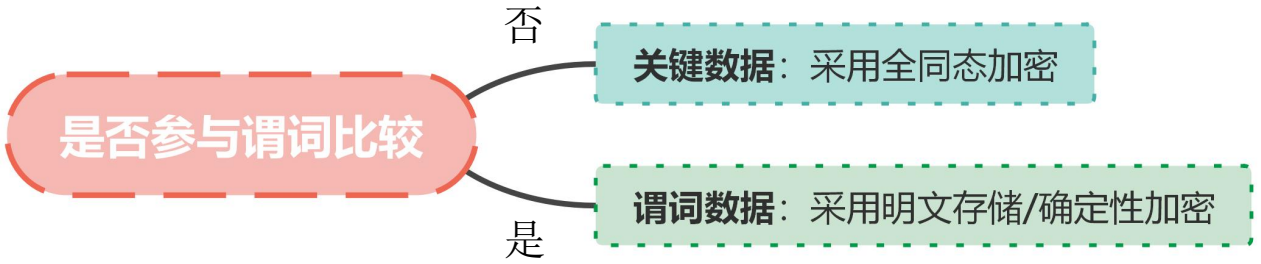
**保护点2：全同态数据库GPU二重加速协议。**实现目标：(高效性)利用GPU并行执行全同态事务与全同态算子，从而提高吞吐量和降低时延。




**架构设计：分层全同态事务执行中间件。**实现目标：(异构支持)通过提供特定于数据库的驱动适配异构云端数据库事务。

# 保护点1 – 多元加密策略

- **多元加密策略：**根据数据在事务中是否参与谓词比较，将数据分为以下两类并采取相应的(加密)存储策略：




- 为什么采用多元加密策略？
  - 避免开销极大的全同态密文比较：全同态密文比较基于迭代算法，时间复杂度为O(MN)，其中M为被比较密文差值的精度，N为比较的密文总数。为了保证数据库的可伸缩性，必须避免全同态密文参与谓词比较。
  - 最大化保护数据机密性：采用全同态加密保护不参加谓词比较的关键数据，从而保证关键数据在全生命周期的明文不被泄露（即机密性不受损）。

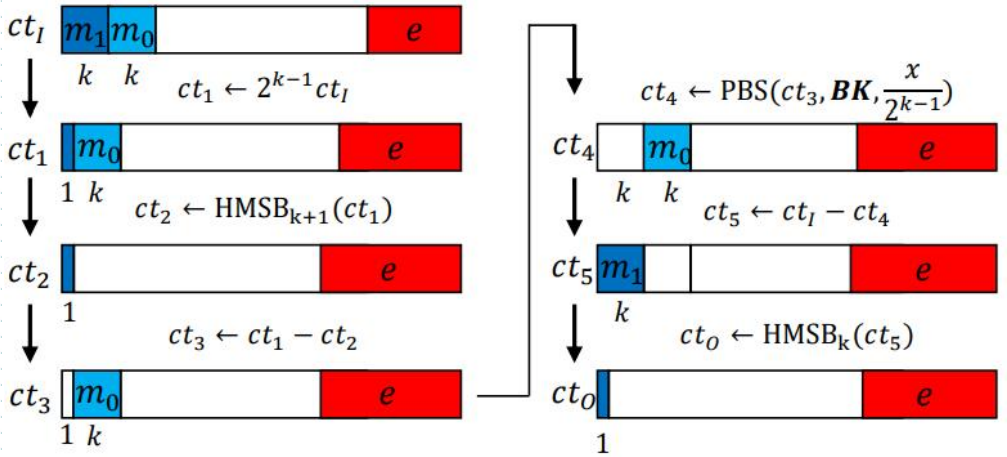



图9：全同态密文比较迭代算法

# 保护点2 - 全同态数据库GPU二重加速协议

- 使用GPU在两个层级上加速全同态运算：
  1. **单个全同态算子内部并行**：利用全同态算子内部运算(例如key-switching)的高并行度，将全同态算子封装为多GPU线程并行执行内部运算的GPU核函数。
  2. **多个全同态事务之间并行**：利用多个GPU计算流(stream)可并行执行的特点，将多个全同态事务调度到不同的GPU计算流中，实现多个事务的并行执行。
- GPU二重加速的优势：显著提升GPU资源利用率
  1. 算子内部运算并行化缩短单个算子时间开销→**降低事务处理时延**
  2. 事务处理并行化提高GPU流水线利用率→**提升事务处理吞吐量**

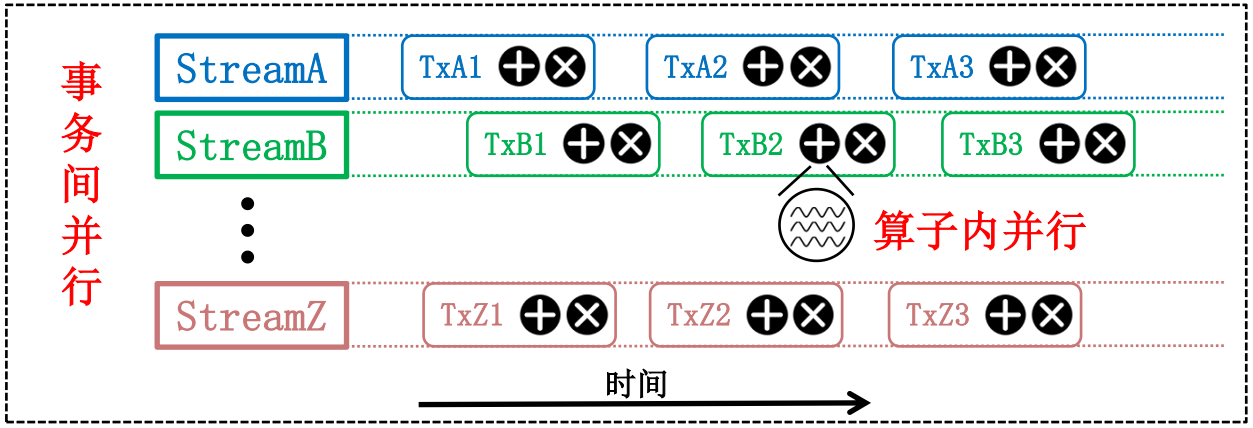



图10：全同态数据库GPU复合加速协议

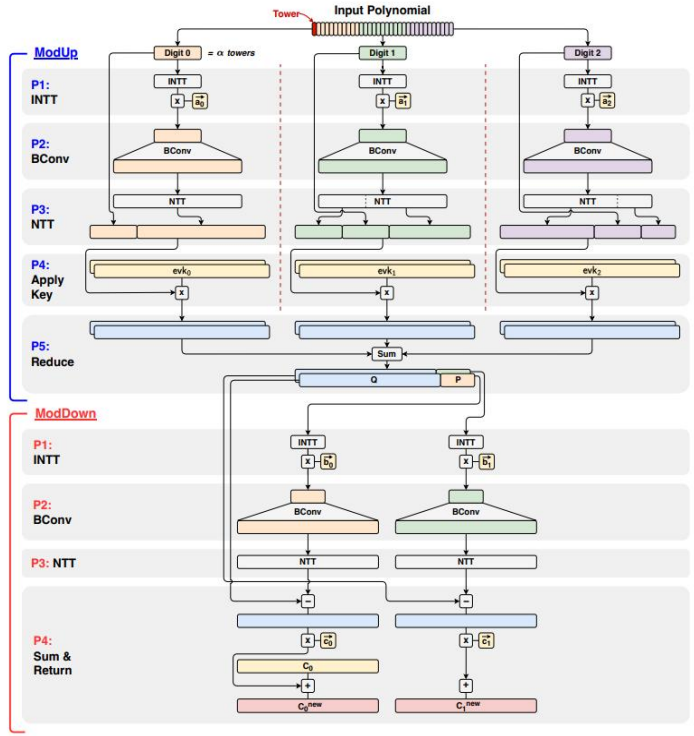



图11：key-switching并行执行示意图

# 保护点3 – 关联事务融合协议

- **可融合的关联事务**：关联事务是指业务逻辑相同但传入参数不同的数据库事务。如果两个或多个关联事务在满足给定条件下，能够通过融合原有传入参数生成新事务，那么这类关联事务是可融合的。
- **重要观察**：在各类数据库应用中，可融合的关联事务广泛存在且高频执行，例如TPC-C的Payment事务和SmallBank的DepositChecking事务。
- **关联事务融合协议**包含两部分
  1. **事务融合元信息声明**：在事务静态编码阶段，开发者提供事务的融合元信息，例如(1)事务可融合性；(2)两个关联事务可融合的判定谓词；(3)关联事务融合指令，等等。
  2. **关联事务在线融合**：在事务动态执行阶段，根据融合元信息，该发明对不同事务执行分流处理：
     - ◆对于不可融合事务，直接执行。
     - ◆对于可融合事务，先将一定时间窗口内（例如100ms）接收的可融合事务进行缓存，然后将多个关联事务融合为单一等价事务，最终执行等价事务。
- 关联事务融合具有两大优势：
  - ■ 显著降低密文运算次数
  - ■ 减少并发事务冲突，避免运算资源浪费

# 保护点3 - 关联事务融合协议

- 示例：可融合的 TPC-C Payment 事务

```
def payment(
  h_amount: hadd,
  w_id: equal,
  d_id: equal,
  c_id: equal,
  h_date: nil):

  # 0. 开始事务
  tx = DB.startTx()
  # 1. 获取仓库、区域、顾客信息
  ...
  # 2. 更新顾客余额和顾客今年累计支付金额
  tx.execute(update_customer_sql, (
      hsub(customer.balance, h_amount),
      hadd(customer.ytd_payment, h_amount),
      c_id))
  # 3. 更新仓库金额和区域金额，插入支付历史
  ...
  # 4. 提交事务
  tx.commit()
```

Payment关联事务融合指令是对h_amount执行同态加法操作

Payment关联事务可融合的判定谓词是w_id, d_id, c_id各自满足等价关系

融合所得的新Payment事务可取任意原事务的h_date作为新事务的h_date入参

融合元信息

图12：带融合元信息的TPC-C Payment事务部分实现代码

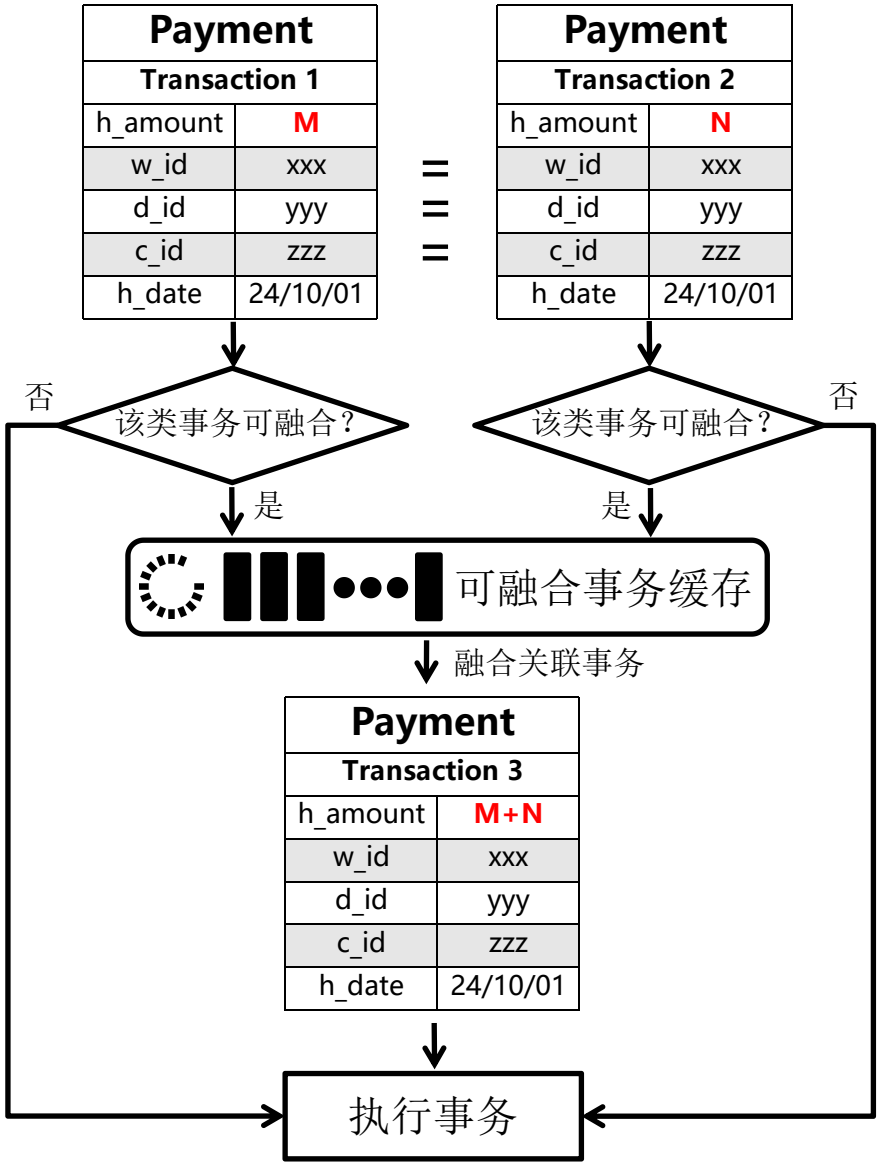



图13：关联事务在线融合

# 技术挑战

- 在数据库引擎层面实现本专利技术保护点面临两大技术挑战：
  1. **性能衰减**：本专利的计算开销(例如全同态运算)挤占数据库主机宝贵的运行资源，不利于释放本专利的最佳性能。
  2. **不支持异构数据库**：需要针对各种数据库分别实现该专利技术方案，无法做到一套专利实现在多种数据库上运行。
- 解决方案：适配异构数据库引擎的**分层全同态事务执行中间件**
  1. **事务调用接口层**：面向客户提供事务统一实现和调用接口，使客户可以无感知地在异构云端数据库上实现与执行事务。
  2. **事务调度执行层**：运行全同态数据库GPU二重加速协议与关联事务融合协议。
  3. **数据库驱动层**：通过数据库驱动连接中间件与异构数据库，每个驱动只需提供针对于特定数据库的操作原语(例如查询与更新)。

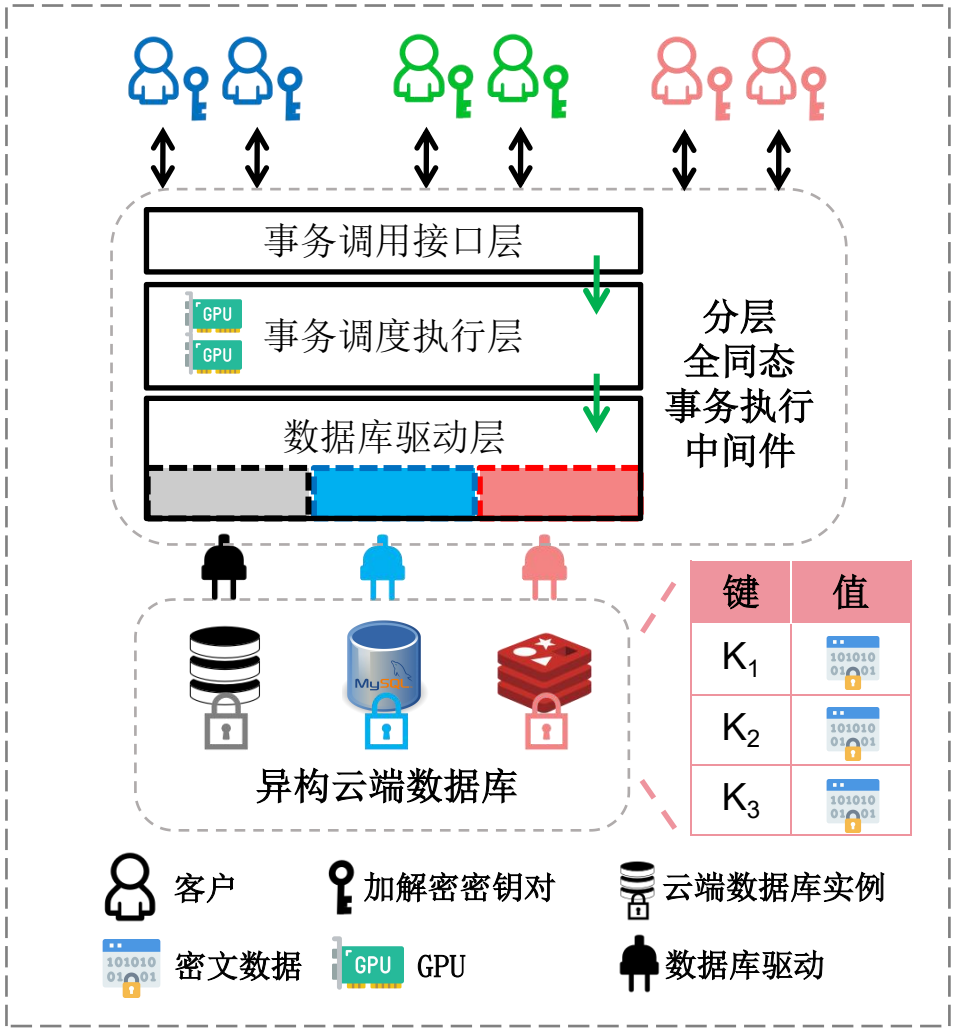



**图14：** 可插拔全同态事务执行中间件使用场景

# 本发明技术方案实施案例（对传统异构非密态数据库进行加密与加速）

- 实施例：使用本专利加速传统异构非密态云端数据库(例如MySQL和Redis)运行关键数据加密的银行转账业务
  - 银行账户数据库可以简化为三元组(账户ID，账户余额，最近操作时间)，其中账户余额是关键数据，使用全同态加密进行保护；其余数据例如账户ID与最近操作时间是非关键数据，使用明文存储或确定性加密。
  - 客户执行本专利编译完成的客户端程序，生成或获取用户密钥对，根据银行账户数据库的多元加密策略对银行转账事务传入参数进行加密，随后将全同态银行转账事务执行请求发送至中间件程序。
  - 中间件程序在云端部署，负责对银行转账关联事务的融合与使用GPU加速事务执行。
  - 中间件程序通过数据库驱动与异构云端数据库(例如MySQL和Redis)进行数据查询、插入、更新与删除等操作。

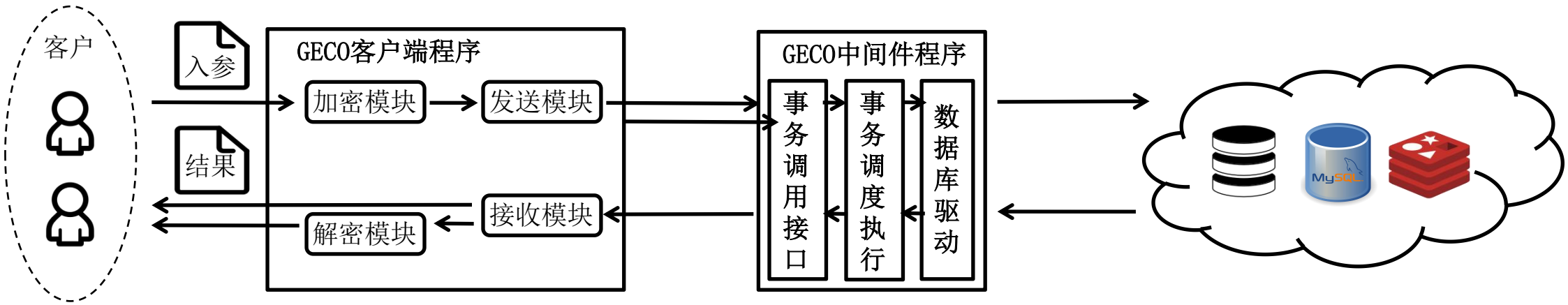



**图15：**本发明技术方案实施案例

# 本发明防侵权检测方法

- **防侵权核心方法**
  - （保护点1）观察第三方系统的数据存储格式
  - （保护点1,2）提交事务执行请求，观察第三方系统的硬件运行指标
  - （保护点3）分别提交多个可融合事务和不可融合事务，观察第三方系统吞吐量与时延

- 本发明具有以下外显特征：
  - 外显特征1：第三方系统端到端关键数据被全同态加密，其余数据采用明文存储或确定性加密 → 保护点1
  - 外显特征2：第三方系统执行事务的吞吐量和时延不与数据库的记录数量成显著负相关 → 保护点1
  - 外显特征3：第三方系统使用GPU加速，并且流多处理器(SM)使用数量与正在处理事务数量成正相关 → 保护点2
  - 外显特征4：第三方系统执行可融合事务和不可融合事务的吞吐量和时延存在明显差异 → 保护点3

- **若第三方系统声称满足本发明所有的发明目标，且同时满足上述所有外显特征，则该第三方系统极有可能侵权**

# 本发明防侵权检测方法

- **外显特征1：第三方系统端到端关键数据被全同态加密，其余数据采用明文存储或确定性加密** → **保护点1**
  - 对客户端发送到中间件的事务请求进行网络抓包，查看事务请求数据是否全为密文或密文与明文共存
  - 向中间件发送密文数据运算事务执行请求，执行过程中使用perf等性能监控工具观察本地机器是否不参与事务执行
  - 向中间件发送密文数据运算事务执行请求，获取运算前后的密文进行解密，判断两个明文是否符合事务执行逻辑
- **外显特征2：第三方系统事务执行时延不与数据库的记录数量成显著负相关** → **保护点1**
  - 使用10^N条记录对数据库进行初始化，其中N的取值为(1,2,3,4,5,6)。在不同N取值的条件下执行非关键数据参与谓词判断的查询事务并记录事务执行时间，判断事务执行时间是否不与数据库的记录数量成显著负相关
- **外显特征3：第三方系统使用GPU加速，并且流多处理器(SM)使用数量与正在处理的事务数量成正相关** → **保护点2**
  - 向中间件同时发送多条密文数据运算事务执行请求，执行过程中使用nvidia-smi等性能监控工具观察中间件在使用GPU执行不同数量事务时，是否使用了数量与事务数量正相关的流多处理器(SM)
- **外显特征4：第三方系统执行可融合事务和不可融合事务的吞吐量和时延存在明显差异** → **保护点3**
  - 向中间件同时发送M(M>16)条**可融合**事务执行请求，记录平均时延L1并计算吞吐量T1
  - 向中间件同时发送M条**不可融合**事务执行请求，同样记录平均时延L2并计算吞吐量T2
  - 如果L1显著小于L2并且T1显著小于T2，则说明发生关联事务融合，存在侵权

# 有益效果

- **本发明原型系统：**在同一个中间件实现的基础上，采用 MySQL 和 Redis 作为后端数据库分别实现 GECO SQL 和 GECO NoSQL 两个原型系统，测试本发明分别在 SQL 和 NoSQL 数据库下的有益效果。
- **对比系统：**本发明对比了**MySQL EE，MySQL FHE 和 Redis FHE**
  - MySQL EE (MySQL Enterprise Encryption)是一个广泛商用的SQL密态数据库，支持传统加密算法（例如 RSA）。
  - MySQL FHE 基于 MySQL 8 引入全同态加密算法，但不具备GPU关联事务加速和关联事务融合能力。
  - Redis FHE 基于 Redis 7.4 引入全同态加密算法，但不具备GPU关联事务加速和关联事务融合能力。
- **数据集：**本发明使用TPC-C数据集对比了所有SQL系统（GECO SQL, MySQL EE 和 MySQL FHE），使用Smallbank数据集对比了所有NoSQL系统（GECO NoSQL 和 Redis FHE）。
- **实验问题**
  - 高效性：对比现有系统，本发明是否取得更低平均时延与更高吞吐量？
  - 可伸缩性：本发明能否高效扩展到更大数据集？
- **天然满足的目标**
  - 数据机密性与完整性：在传输、存储和计算过程中，关键数据使用全同态加密，保障明文不泄露与不被篡改。
  - 密文可算性：全同态加密保障密文可直接执行算术运算。
  - 异构支持：采用分层全同态事务执行中间件架构，通过提供数据库驱动即可适配异构云端数据库。

# 有益效果 – 本发明实现更高吞吐量和更低平均时延

- 图16使用 TPC-C 数据集对比了三个SQL密态数据库（本发明 GECO SQL，以及两个对比系统 MySQL EE 和 MySQL FHE）的端到端性能。Table Size 表示 Stock 表的记录数量，tput 表示事务执行吞吐量，latency 表示事务执行平均时延。
- 图17使用 Smallbank 数据集对比了两个 NoSQL 密态数据库（本发明 GECO NoSQL 和对比系统 Redis FHE）的端到端性能。Table Size 表示 Account 表的记录数量。
- 可以看出：本发明带来端到端约3-30倍吞吐量提升和最高60.9%平均时延降低。
  - （保护点1）多元加密策略可带来约1.5-3倍吞吐量提升
  - （保护点2）全同态数据库GPU二重加速协议可带来约5倍的吞吐量提升
  - （保护点3）关联事务融合协议可带来约2倍吞吐量提升

| | | MySQL EE | | | MySQL FHE | | | GECO SQL | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Table Size | | $10^5$ | $10^6$ | $10^7$ | $10^5$ | $10^6$ | $10^7$ | $10^5$ | $10^6$ | $10^7$ | Max Improvement |
| New Order | tput (tps) | 28 | 28 | 26 | 77 | 71 | 67 | 752 | 737 | 704 | 28.5x |
| | latency (ms) | 2105 | 2130 | 2159 | 923 | 960 | 1028 | 824 | 855 | 901 | 60.9% |
| Payment | tput (tps) | 110 | 99 | 90 | 164 | 155 | 147 | 3175 | 2988 | 2683 | 30.2x |
| | latency (ms) | 394 | 398 | 411 | 245 | 248 | 261 | 219 | 240 | 271 | 44.5% |
| Order Status | tput (tps) | 961 | 940 | 909 | 945 | 912 | 884 | 3958 | 3792 | 3531 | 4.2x |
| | latency (ms) | 71 | 73 | 77 | 91 | 95 | 102 | 87 | 88 | 94 | 7.9% |
| Delivery | tput (tps) | 30 | 29 | 27 | 47 | 45 | 41 | 216 | 195 | 171 | 7.2x |
| | latency (ms) | 1811 | 1887 | 2031 | 1672 | 1840 | 1932 | 1332 | 1361 | 1423 | 30.1% |
| Stock Level | tput (tps) | 744 | 720 | 698 | 712 | 694 | 604 | 2410 | 2271 | 2098 | 3.4x |
| | latency (ms) | 165 | 179 | 190 | 201 | 213 | 224 | 175 | 180 | 188 | 16.1% |

图16：基于TPC-C数据集的SQL密态数据库端到端性能

| | | Redis FHE | | | GECO NoSQL | | | |
|---|---|---|---|---|---|---|---|---|
| Table Size | | $10^5$ | $10^6$ | $10^7$ | $10^5$ | $10^6$ | $10^7$ | Max Improvement |
| Balance | tput (tps) | 718 | 712 | 701 | 3352 | 3340 | 3309 | 4.7x |
| | latency (ms) | 221 | 230 | 238 | 140 | 144 | 145 | 39.1% |
| Deposit Checking | tput (tps) | 682 | 670 | 669 | 4212 | 4179 | 4158 | 6.2x |
| | latency (ms) | 241 | 253 | 271 | 155 | 161 | 167 | 38.4% |
| Transaction Saving | tput (tps) | 701 | 689 | 674 | 4288 | 4247 | 4192 | 6.2x |
| | latency (ms) | 238 | 249 | 258 | 149 | 152 | 158 | 39% |
| Amalgamate | tput (tps) | 655 | 647 | 638 | 2875 | 2794 | 2743 | 4.4x |
| | latency (ms) | 296 | 305 | 317 | 243 | 249 | 257 | 19.9% |
| Write Check | tput (tps) | 532 | 529 | 514 | 3183 | 3119 | 3087 | 6x |
| | latency (ms) | 312 | 329 | 330 | 265 | 271 | 279 | 17.7% |

图17：基于Smallbank数据集的NoSQL密态数据库端到端性能

# 有益效果 – 本发明可高效伸缩

- 图18和图19分别展示了 GECO 在 TPC-C 和 SmallBank 数据集不同数据量下的端到端吞吐量变化。
- 可以看出：
  - 本发明**支持高效横向伸缩**：GECO 可在**数据量**显著增加的场景下，维持事务执行吞吐量基本稳定。
  - 本发明**支持高效纵向伸缩**：GECO 可在**数据库表结构**显著复杂的场景下，维持事务执行吞吐量基本稳定。

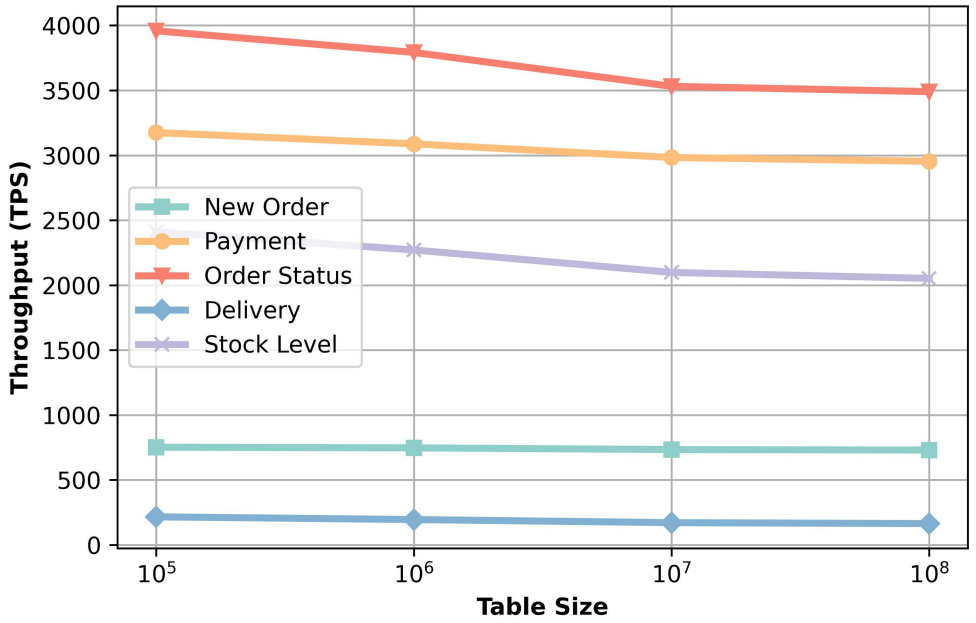



图18：GECO SQL 的 TPC-C 端到端吞吐量

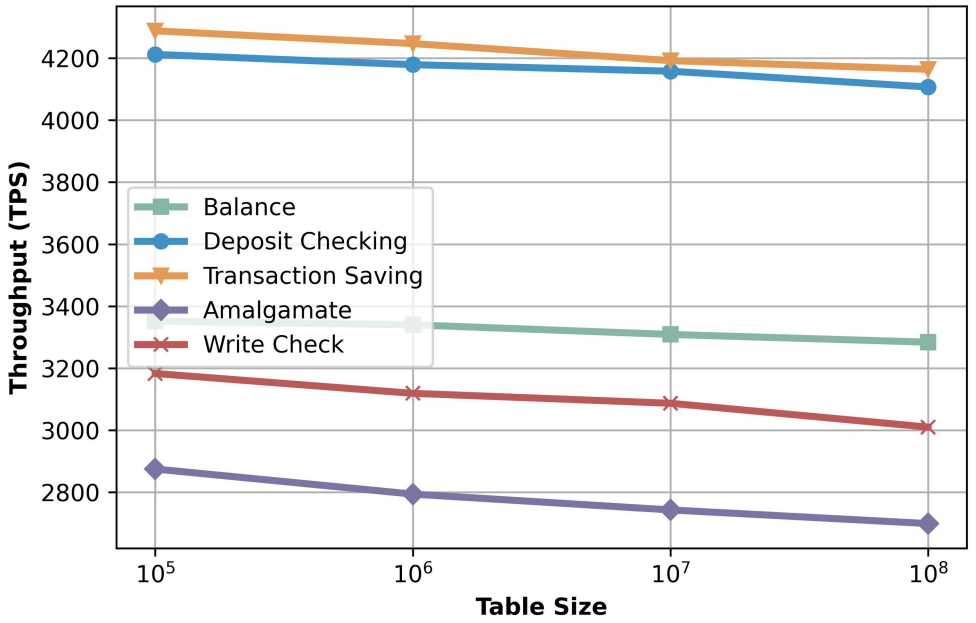



图19：GECO NoSQL 的 SmallBank 端到端吞吐量

# 结论：申请者认为本发明有重大专利价值

- 云端密态数据库目前广泛应用于金融、政务、电商等多个领域，蕴藏极高的**商业价值**和**落地潜力**。

- 现有系统**无法同时满足**事务执行**高效性**、**数据机密性**和**异构数据库支持**这三个核心目标，极大阻碍了来自不同行业的客户（例如政府或企业）采纳云端密态数据库。

- 本发明利用全同态加密数据库的**密文可算性**的特点，基于对全同态运算特征和事务数据依赖性的观察，通过使用**GPU**和**关联事务融合**加速数据库全同态事务执行，并引入**分层事务中间件**设计，实现了全同态数据库的**高效执行和异构支持**。

- 本发明有益于华为云、GaussDB、数据存储产品线等部门，**作为现有数据库的有力补充**，守护核心机密数据（例如技术方案实施案例）。

# 检索情况

| 国家/地区 | 检索网站 | 检索公式<br>(关键词及组合方式) | 文献数量(指依据检索关键字直接获得的文献数，不需要列出具体的文献信息) |
|---|---|---|---|
| CN/US/EP | http://www.incopat.com （推荐使用。账号获取：直接通过主页“IP登录”进入）<br>使用教程及密码：<br>http://3ms.huawei.com/hi/group/2033427/wiki_5014231.html | 云上机密完整的密态数据库<br>布隆过滤器高效索引<br>对数级复杂度多功能隐私查询系统<br>Bloom filter-based logarithmic index | |
| 其它网站 | https://www.google.com（适用于使用**专利号**直接检索）<br>标准网站等其它网站，请自行增加。 | 云上机密完整的密态数据库<br>布隆过滤器高效索引<br>对数级复杂度多功能隐私查询系统<br>Bloom filter-based logarithmic index | 1150<br>491<br>8560<br>725 |

# Thank You

www.huawei.com

www.huawei.com



HUAWEI

# 一种基于GPU加速和关联事务融合的全同态数据库中间件

- **现有方案一：传统密态数据库使用传统加密算法（例如RSA）**

☒ 缺点1：**性能低下**。密文事务执行牵涉**先解密、后加密**的繁琐流程

☒ 缺点2：**明文易泄**。密文事务执行期间会导致数据**明文暴露**。

- **现有方案二：现有全同态数据库使用全同态加密（例如CKKS）**

☒ 缺点1：**密文运算性能极差**。全同态算术运算开销高昂，并且涉及全同态密文比较的数据查询需要进行**复杂度高达O(MN)的全表扫描**

☒ 缺点2：**不支持异构数据库**。系统设计**无法迁移到不同数据库架构**

- **有益效果：分别使用TPC-C和Smallbank进行端到端性能测试**

☑ GECO带来端到端约**3-30倍吞吐量提升**和**最高60.9%平均时延降低**

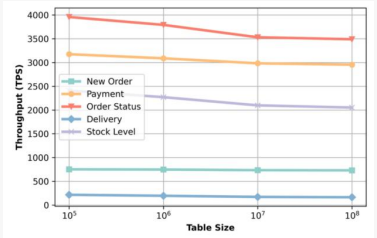
☑ 多元加密策略可带来**约1.5-3倍**吞吐量提升（贡献1）

☑ 全同态数据库GPU二重加速协议可带来**约5倍**的吞吐量提升（贡献2）

☑ 关联事务融合协议可带来**约2倍**吞吐量提升（贡献3）

☑ GECO具备**横向(数据量)**和**纵向(数据库表结构)**的**高可伸缩性**（贡献1）

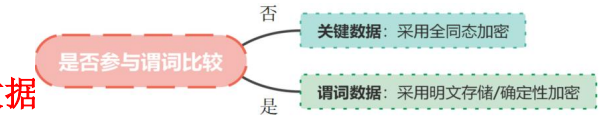
| Table Size | | MySQL EE $10^5$ | MySQL EE $10^6$ | MySQL EE $10^7$ | MySQL FHE $10^5$ | MySQL FHE $10^6$ | MySQL FHE $10^7$ | GECO SQL $10^5$ | GECO SQL $10^6$ | GECO SQL $10^7$ | Max Improvement |
|---|---|---|---|---|---|---|---|---|---|---|---|
| New Order | tput (tps) | 28 | 28 | 26 | 77 | 71 | 67 | 752 | 737 | 704 | 28.5x |
| | latency (ms) | 2105 | 2130 | 2159 | 923 | 960 | 1028 | 824 | 855 | 901 | 60.9% |
| Payment | tput (tps) | 110 | 99 | 90 | 164 | 155 | 147 | 3175 | 2988 | 2683 | 30.2x |
| | latency (ms) | 394 | 398 | 411 | 245 | 248 | 261 | 219 | 240 | 271 | 44.5% |
| Order Status | tput (tps) | 961 | 940 | 909 | 945 | 912 | 884 | 3958 | 3792 | 3531 | 4.2x |
| | latency (ms) | 71 | 73 | 77 | 91 | 95 | 102 | 87 | 88 | 94 | 7.9% |
| Delivery | tput (tps) | 30 | 29 | 27 | 47 | 45 | 41 | 216 | 195 | 171 | 7.2x |
| | latency (ms) | 1811 | 1887 | 2031 | 1672 | 1840 | 1932 | 1332 | 1361 | 1423 | 30.1% |
| Stock Level | tput (tps) | 744 | 720 | 698 | 712 | 694 | 604 | 2410 | 2271 | 2098 | 3.4x |
| | latency (ms) | 165 | 179 | 190 | 201 | 213 | 224 | 175 | 180 | 188 | 16.1% |




## 本发明解决方案

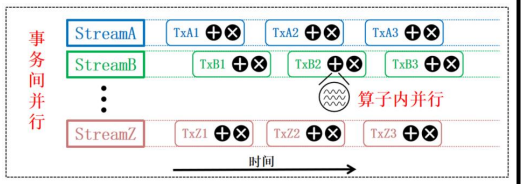
- **贡献1：多元加密策略**

**实现目标**：通过使用**全同态加密关键数据**与**使用明文/确定性加密谓词数据**，同时实现**高效数据查询**与**机密数据保护**



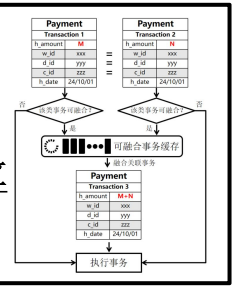

- **贡献2：全同态数据库GPU二重加速协议**

**实现目标**：使用GPU在**单个全同态算子内部并行**和**多个全同态事务之间并行**这两个层级进行加速，**显著降低事务处理时延和吞吐量**




- **贡献3：关联事务融合协议**

**实现目标**：通过在事务静态编码阶段**声明事务融合元信息**和在事务动态执行阶段对**关联事务实施在线融合**，**显著降低密文运算次数与并发事务冲突**，提高计算资源有效利用率




- **架构设计：分层全同态事务执行中间件**

**实现目标**：通过将GECO设计为三层架构（即统一的**事务调用接口层**和**事务调度执行层**，与针对于特定数据库的**数据库驱动层**实现了**对异构数据库引擎的适配**