DTCC

第十四届中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA

数智赋能 共筑未来

IT168.com ChinaUnix ITPUB

北京国际会议中心 | 2023/8/16-18

字节跳动自研万亿级图数据库ByteGraph架构演进

演讲人：字节跳动-图数据库研发-周冰玉

目录

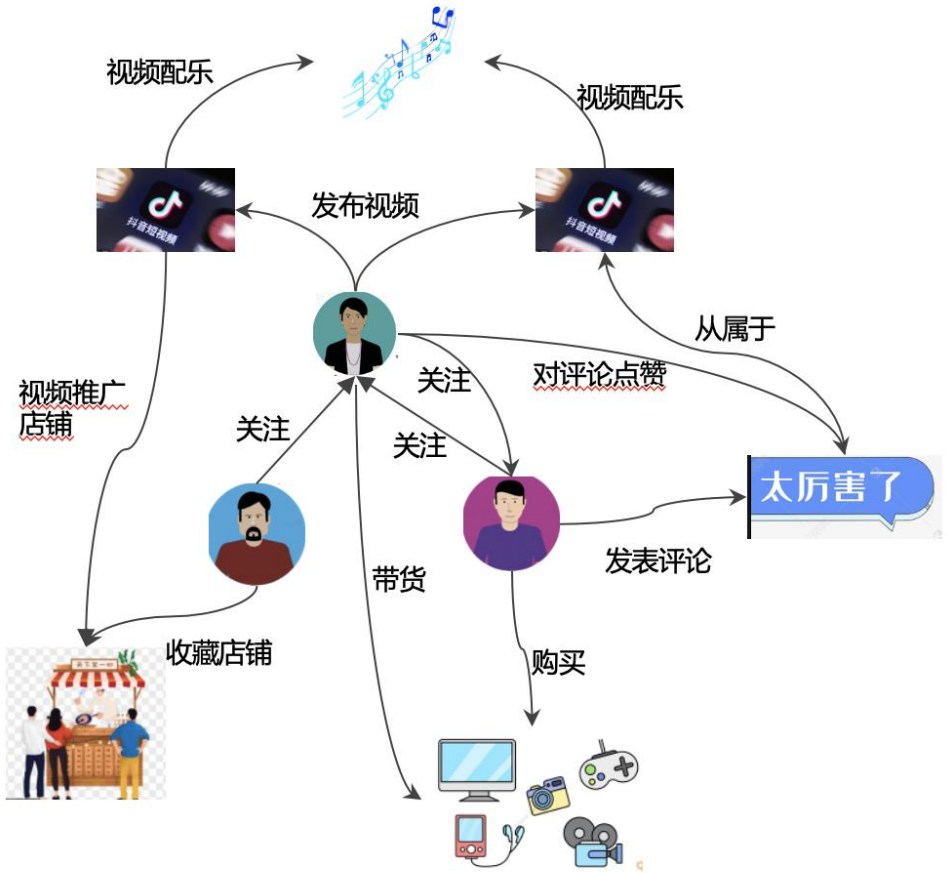
简介

ByteGraph 2.0

ByteGraph 3.0

1.1 ByteGraph 可以做什么

- 字节有哪些业务数据呢？
  - 用户信息、用户关系
  - 内容（视频、文章、广告等）
  - 用户和内容联系（点赞、评论、转发、点击）
- 使用图表达业务场景的优势
  - 建模直观简洁
  - 挖掘数据关联
- ByteGraph特点
  - 高吞吐
  - 低延迟
  - 最终一致性
  - 兼容Gremlin
- ByteGraph学术论文 VLDB-2022



1.2 ByteGraph 查询介绍

- Gremlin语法
  - Gremlin是一种图灵完备的图遍历语言
    （相较其他查询语言，功能全面，上手容易，使用广泛）
  - 主流云厂商图数据库都提供了Gremlin支持，ByteGraph目前支持一个子集
- 数据模型
  - 有向属性图
  - 点和边上都可以携带多属性，支持动态加减属性列
- 查询举例
  - 用户A所有一度好友中满足粉丝数量大于100的子集

    g.V(vertex(A.id, A.type)).out('好友').where(in('粉丝关注').count().is(gt(100))).toList()
  - 求中国出生且配偶是日本人的女明星

    g.V().has('出生地','中国').has('性别', '女').and(out('配偶').has('出生地', '日本'), has('职业', '明星'))

1.3 ByteGraph 应用场景

业务场景分类	分类	图模型	查询举例：
抖音用户关系的服务端在线存储	社交网络关系	点：用户 边：用户之间关系	关注/粉丝列表，关注关系判断
抖音推荐： 推人、推视频	社交推荐	点：用户、视频 边：用户关系(多种）、用户发文	好友的好友等多度查询
知识图谱： 搜索百科、教育、电商	知识图谱	点: 各种实体（课程、知识点，商品） 边：实体之间逻辑关系（报名课程、掌握知识点、收藏商品）	实体推荐 某个人在某个商铺昨天的订单数等
lib库、项目、线上服务之间的网状关系	IT系统	点: lib库、repo、线上服务 边：点之间依赖关系	给定某个库的确定版本，求所有依赖这个版本的库

集群数量：1000+ （计算资源 100W+ 核；存储资源 100+ PB）

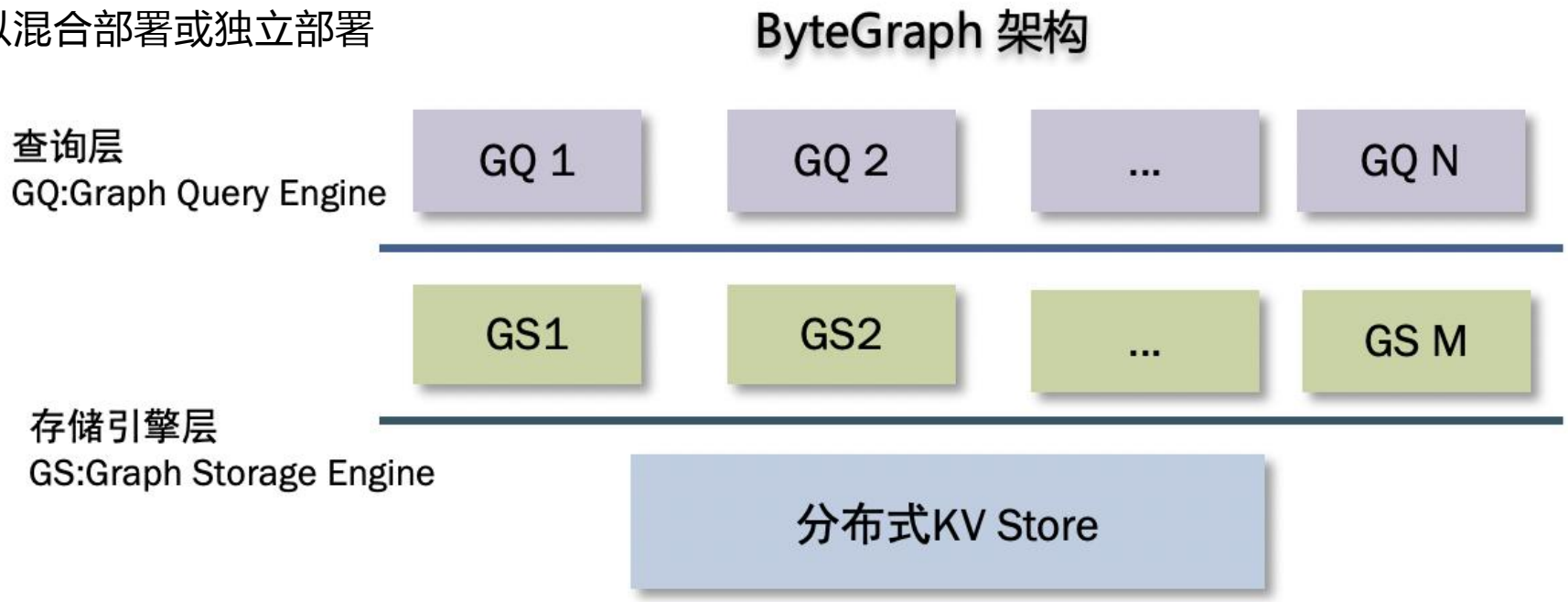
简介

ByteGraph 2.0

ByteGraph 3.0

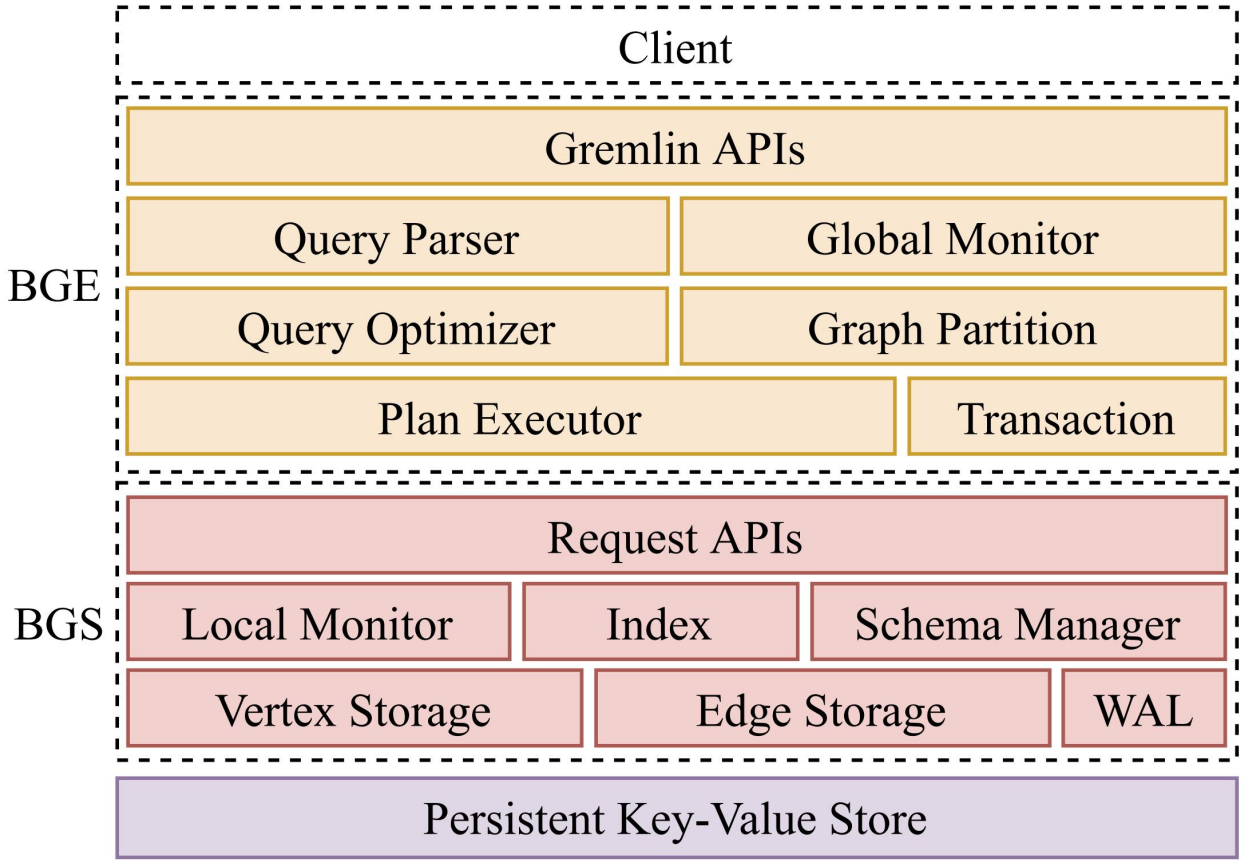
2.1 ByteGraph 2.0 架构 - 整体架构

- 整体分为三层
  - 查询层、内存存储层、分布式KV
- 每层由多个进程实例组成集群
- 查询层和内存存储层可以混合部署或独立部署



2.2 ByteGraph 2.0 架构 - 系统组件

- 查询引擎
  - 服务Proxy层，用户会话管理
  - Gremlin语言解析器
  - 分布式的数据库执行器
  - GS层数据分布路由模块
  - Go语言实现
- 存储引擎
  - 子图（Partition）存储和缓存
  - Partition数据内存组织和磁盘组织
  - 实现Redo Log，支持事务
  - C++实现，追求极致性能
- 分布式KV
  - 数据持久化
  - 可插拔

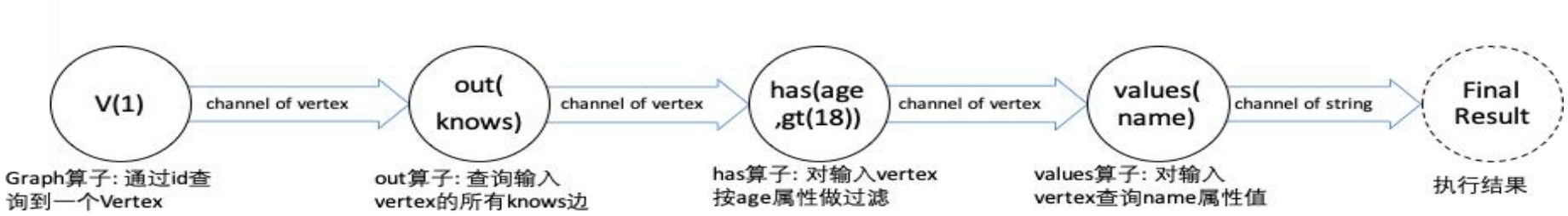


2.3 ByteGraph 2.0 架构 - 执行引擎

查询层(GQ)和Mysql的sql层一样，主要做查询的解析和执行，大致分三个步骤：

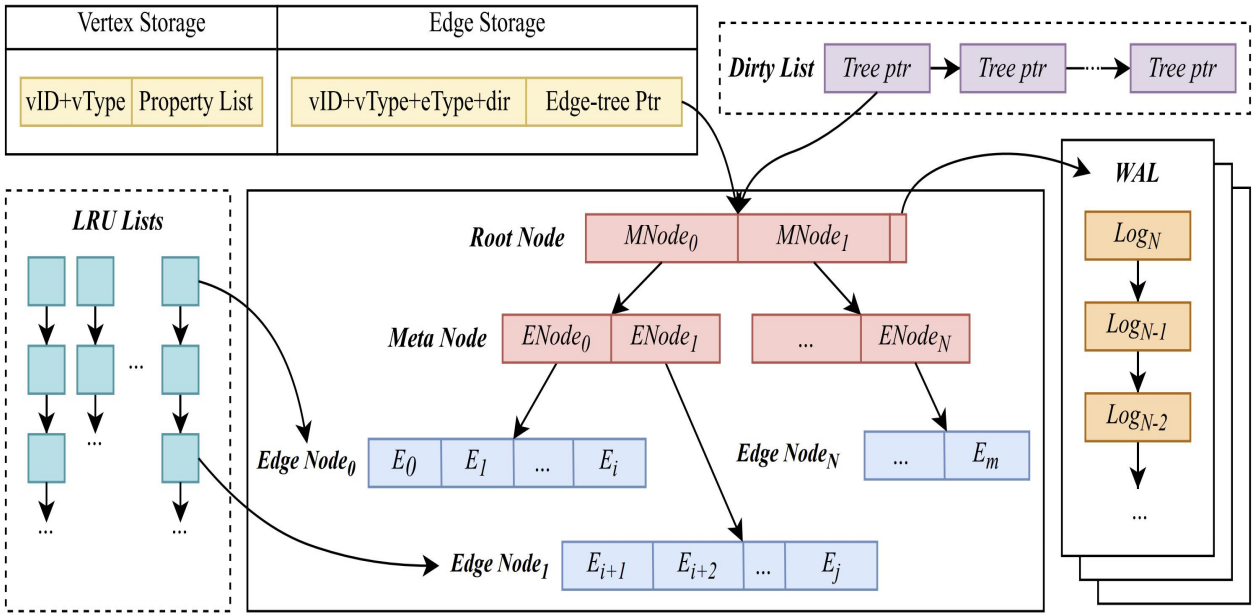
- 解析查询请求：
  - 直接实现的递归下降解析器，将Gremlin语句解析成查询语法树
- 生成查询计划：
  - 把步骤1中查询语法树按照一定查询优化策略（RBO & CBO)转换成执行计划
  - 支持查询计划缓存，减少解析和优化开销
- 执行查询计划：
  - 理解存储层数据分区，与存储层(GS）交互，下推执行算子，合并执行结果，完成查询
- 流水线执行示例

g.V().has('id', 1).has('type', person).out('knows').has('age', gt(18)).values('name')



2.4 ByteGraph 2.0 架构 - 存储结构

- 全局哈希表：
  - Key是起点ID+边Type，当确定点ID时实现快速查找
  - 哈希表中每个元素是Btree，实现快速Scan
- 单机全局多个Btree
  - 森林关系
- 全局LRU表：
  - Partition和Page按需加载，根据一定策略(LRU) Swap到磁盘(KV)
- 全局Dirty链表：
  - Page被修改后，Redo Log同步写磁盘，Page再插入到Dirty链表中， 异步Flush回磁盘



2.5 ByteGraph 2.0 问题

- 高成本的分布式KV
  - 冗余副本，3AZ 5副本 / 3副本
  - 基于LSM Tree KV构建图存储引擎不够极致
  - 多层Cache冗余
  - 内存、CPU预留（Block Cache、Compaction）
  - 磁盘预留写放大高
- 性能
  - 架构分层过多
  - 图上多跳性能难以做到极致
  - 最基础的GetOneHop算子读取性能不够高
  - 执行器基于Channel做通信，涉及大量数据复制
- 数据一致性
  - 主从同步延迟不可控

简介

ByteGraph 2.0

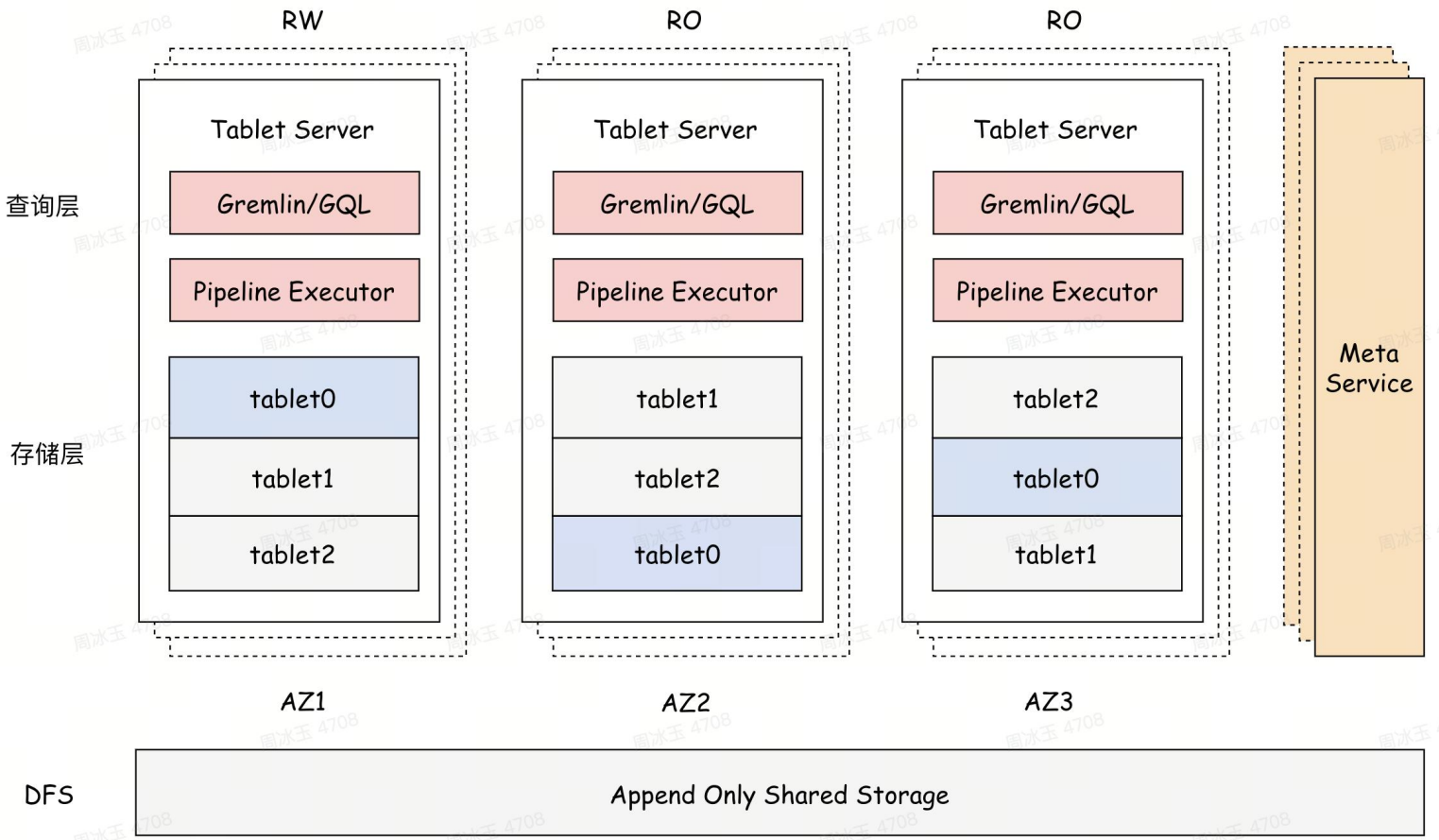
ByteGraph 3.0

3.1 ByteGraph 3.0 设计实现

- 成本
  - 基于EC技术降低副本数量，KV->DFS
  - 3AZ/3副本减少到3AZ/2副本
  - 高密度存储机型进一步降低总体成本
  - 自研基于DFS的BwTree存储引擎（合并2.0存储层Btree引擎和分布式KV引擎，减少写放大）
- 性能
  - 合并进程：减少穿透层数，减少多跳查询RPC开销
  - 减少分片数量：主推单分片一主多从架构，非必要不分片（利用大内存机器来满足性能），提高1PC事务比例
  - Btree Page内部列式存储
  - 新一代Pipeline执行引擎，减少通信拷贝开销，算子并行化
- 数据一致性
  - Tablet级别WAL主从同步

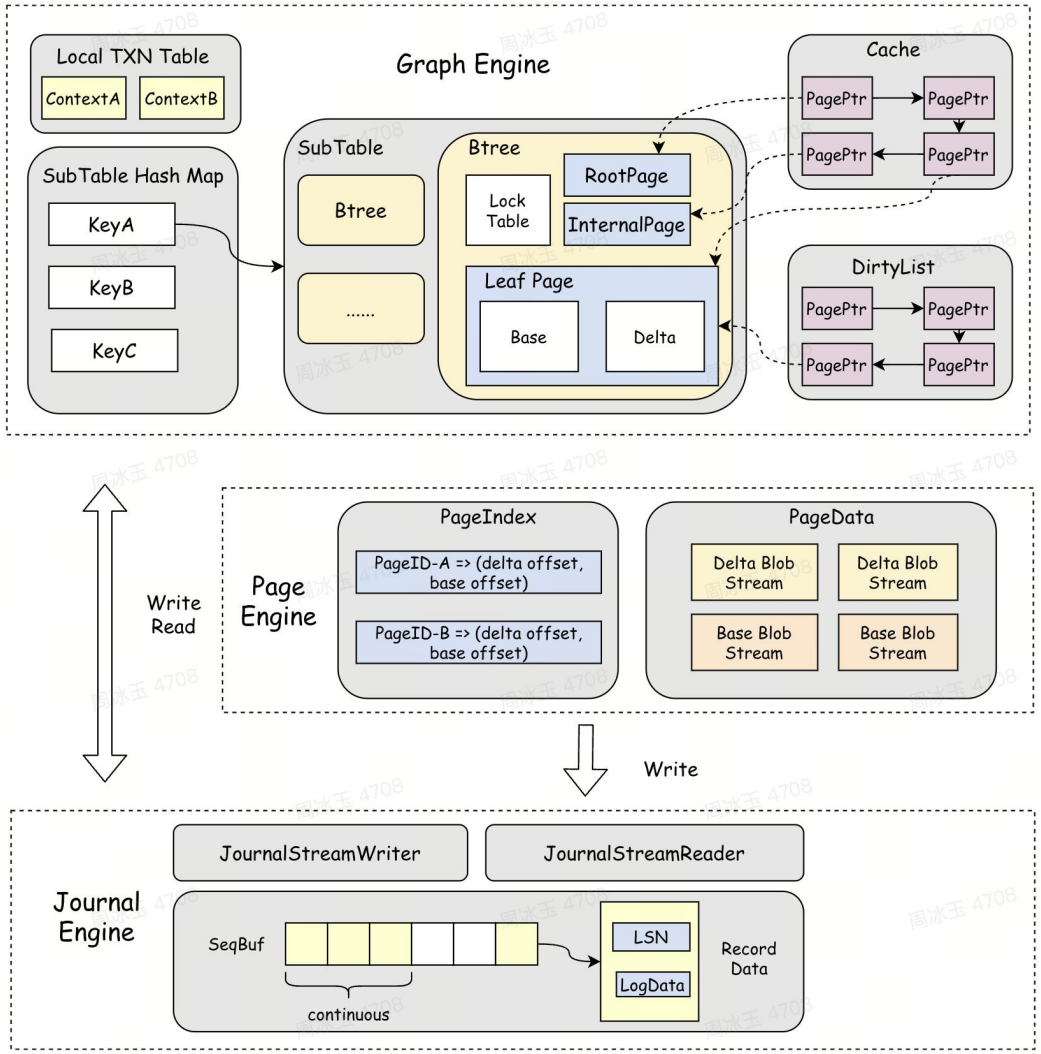
3.2 ByteGraph 3.0 架构设计

- 基于共享DFS构建存储
  - 图数据分区
    （数据按Hash分片，Tablet存储子图数据）
  - 引入只读节点
    （统一日志流，方便实现RO）
- 统一进程
  - 减少RPC开销
  - 全链路异步化
  - Rust & C++
- Pipeline引擎
  - 算子级别并行执行
- 元数据管理
  - Tablet路由管理
  - Schema管理



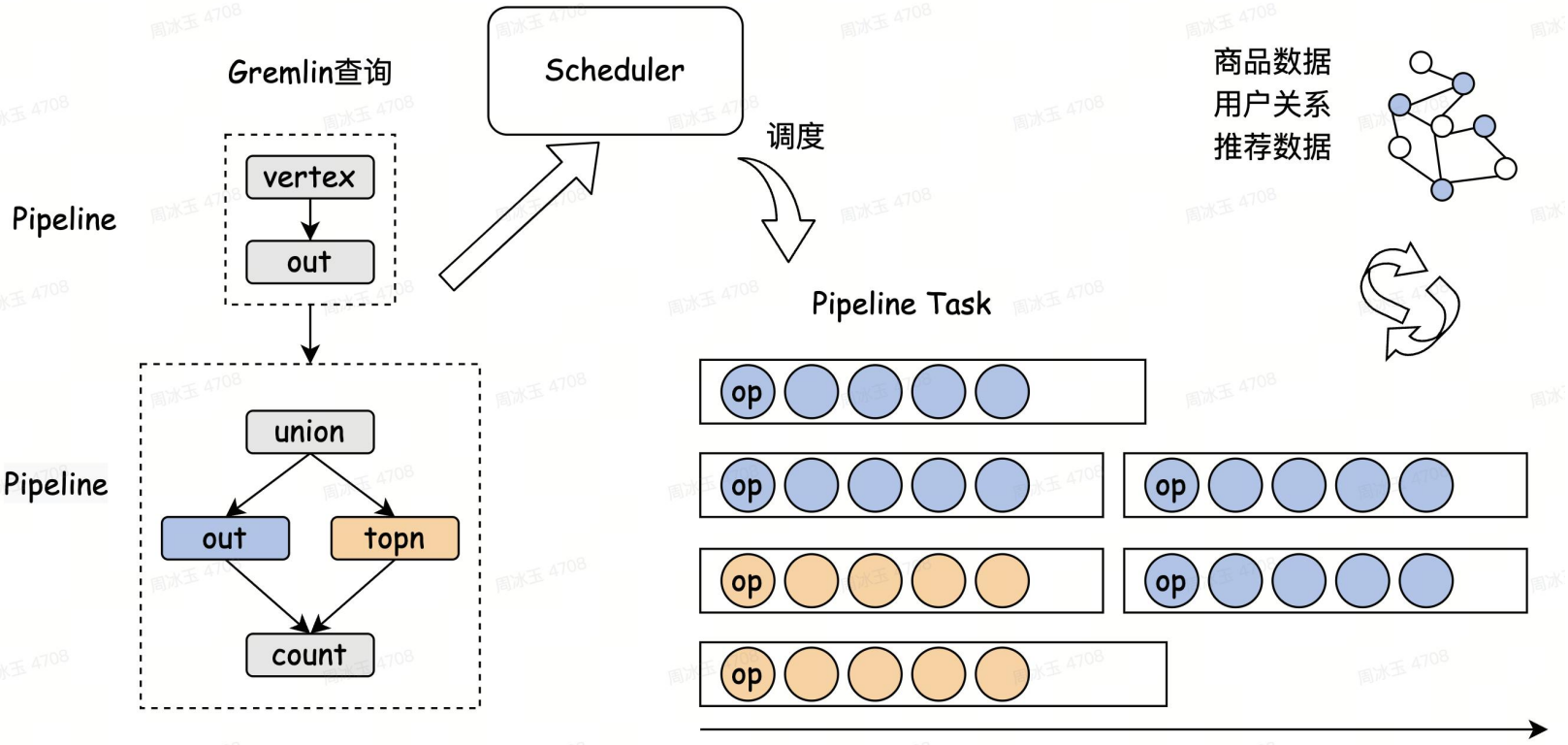
3.3 ByteGraph 3.0 存储引擎

- Graph Engine
  - SubTable HashMap
    维护起点与SubTable映射
  - SubTable存储了单个起点所有出边
  - Cache/Dirty List
- Page Engine
  - Page Index
    存储PageID到物理Page位置映射
  - Page Data存储Page内容
- Journal Engine
  - 负载日志读写



3.4 ByteGraph 3.0 查询引擎

- Pipeline
  - 部分Gremlin Step链
  - 并行执行Pipeline Task
- Pipeline Builder
  - 拆分Gremlin Step
  - 构建Pipeline
- 全局Scheduler
  - 调度执行Pipline Task



4 总结展望

- 业务收益
  - 存储成本降低 30% ~ 50%。
  - 在单分片场景下，多跳召回场景上可提供数倍于原有系统的性能。
- 未来工作
  - 统一存储底座，打通图数据库，GNN，图计算系统，提供一站式图服务。

THANKS