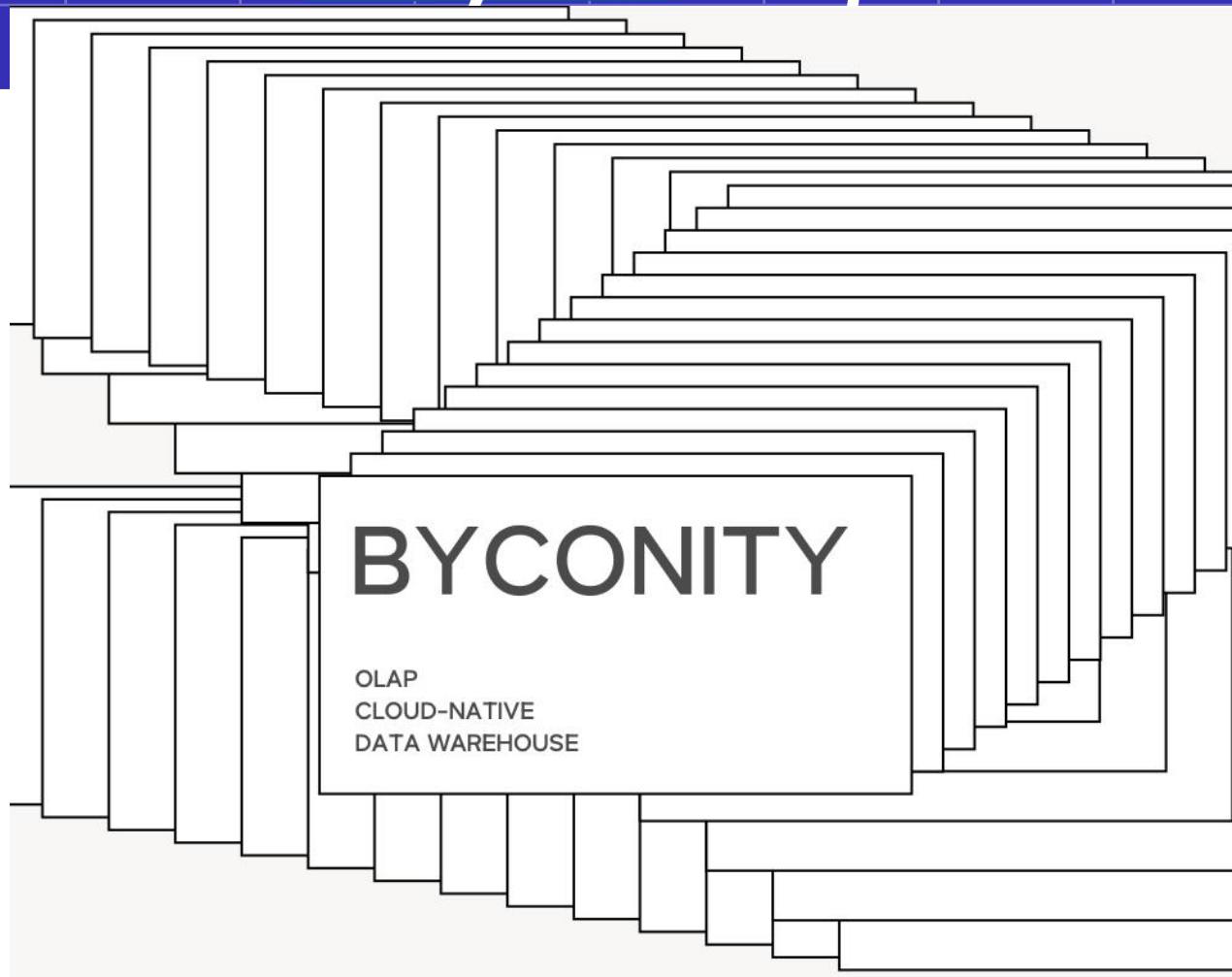
ByConity的架构与设计：从ClickHouse到云原生

王蕴博 字节跳动首席开源布道师

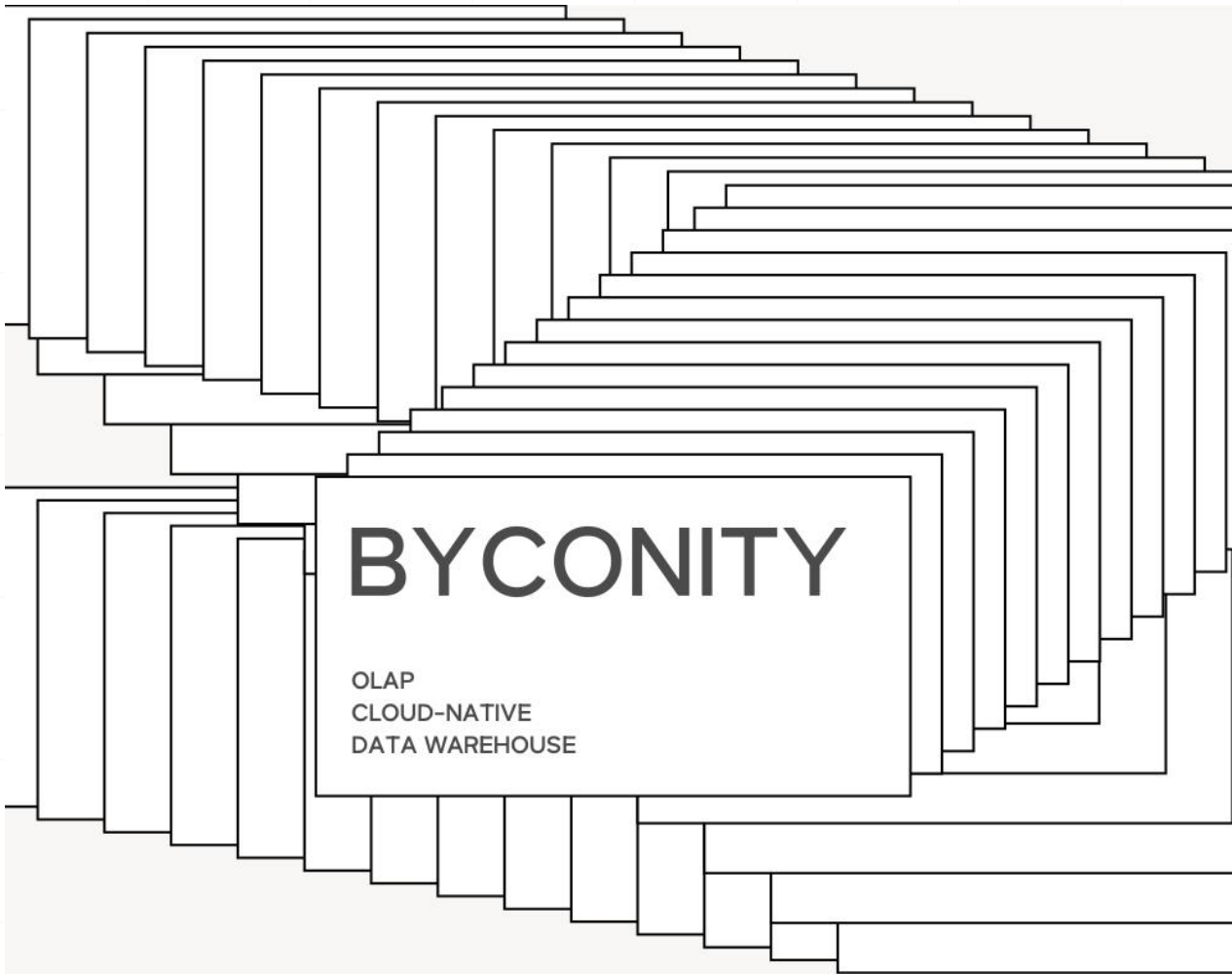


王蕴博

字节跳动首席开源布道师

字节跳动首席开源布道师、开源基础设施负责人。中国计算机学会(CCF)开源发展委员会副秘书长，CCF GitLink社区负责人，CCF GLCC发起人兼组委会主席。前腾讯开源联盟常委，前滴滴开源办公室负责人。长期专注于大数据、DevOps、AI等方向；对开源治理、项目孵化、开源合规等具有丰富的经验。

目录



背景

大规模使用ClickHouse集群至今

Machines
18000

Largest
Cluster
2400

Data Size
700PB

计算和存储紧耦合

Sharing Nothing架构，每个节点拥有部分数据，节点间不共享

扩缩容成本高

涉及数据迁移
资源浪费

多租户难以隔离

读、写相互影响

复杂查询性能低

多表Join等场景

从设计之初

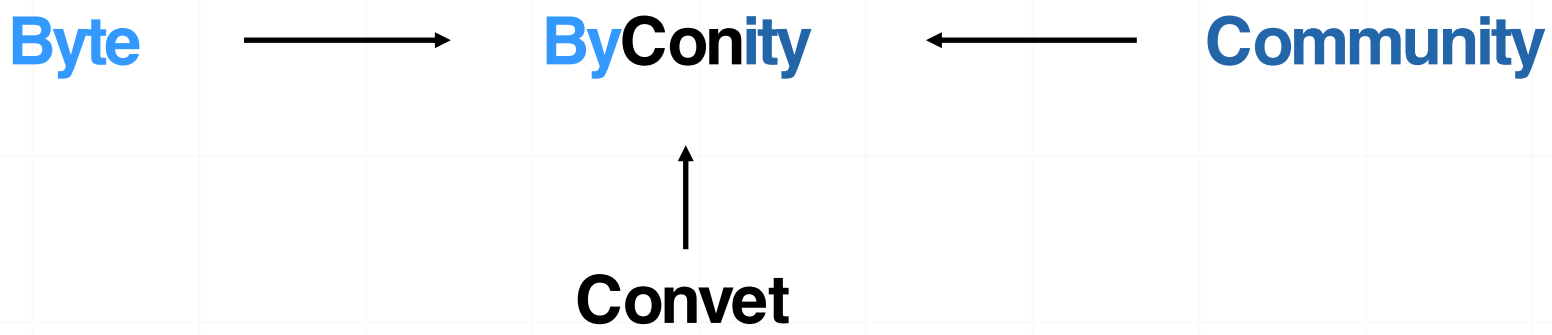
云原生

- 重用云基础设施，高可靠性和降低成本；
- 整个系统和架构设计从开始就基于云的需求；
- 存算分离避免了传统分布式系统的一些性能瓶颈和复杂性

开 源

- 开源让软件更早接触用户，了解用户真实需求；
- 吸引外部开发者参与，汇聚领域人才参与，传播影响力；
- 更加高效的迭代，软件更佳安全和健康

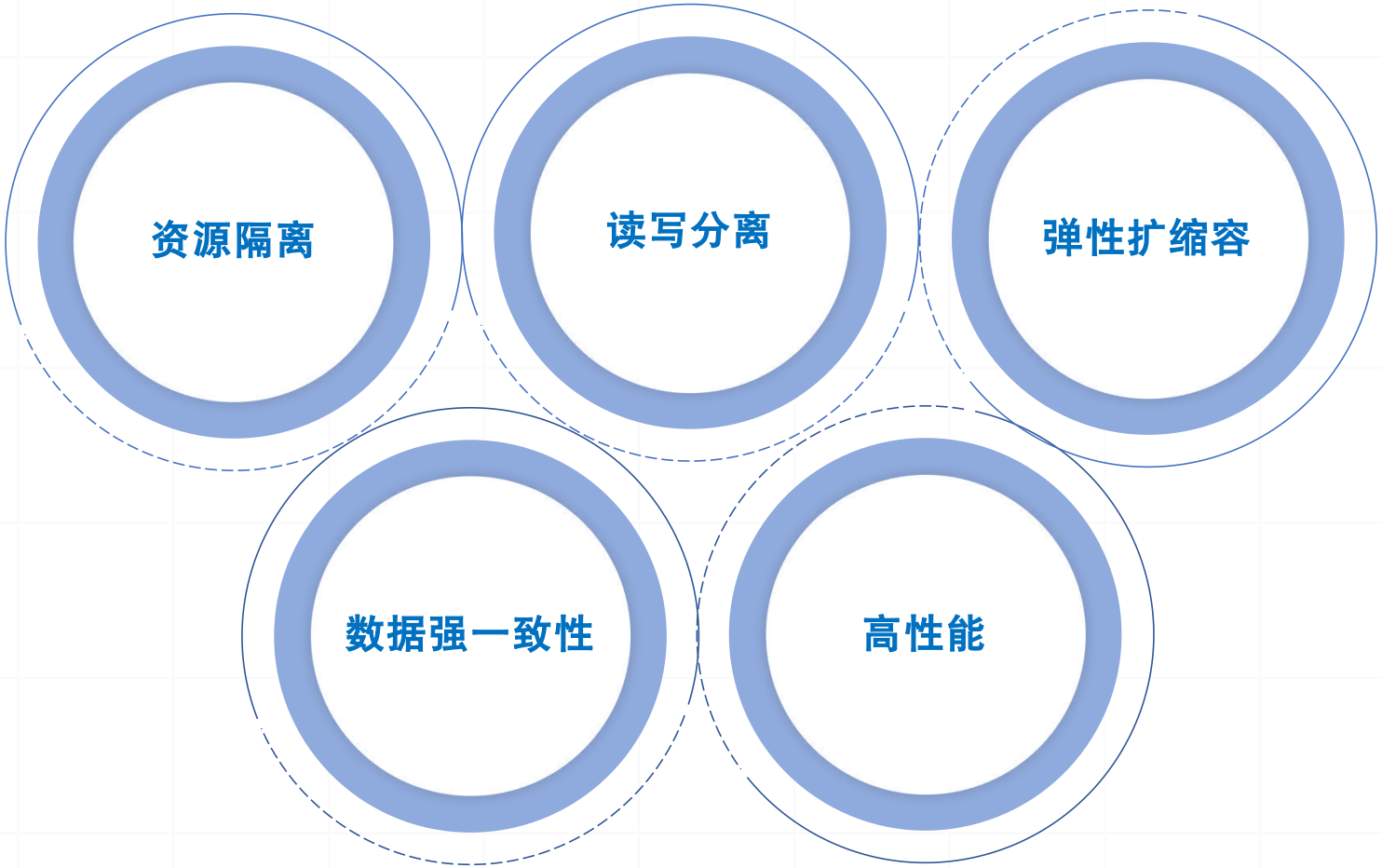
开源从“命名”开始



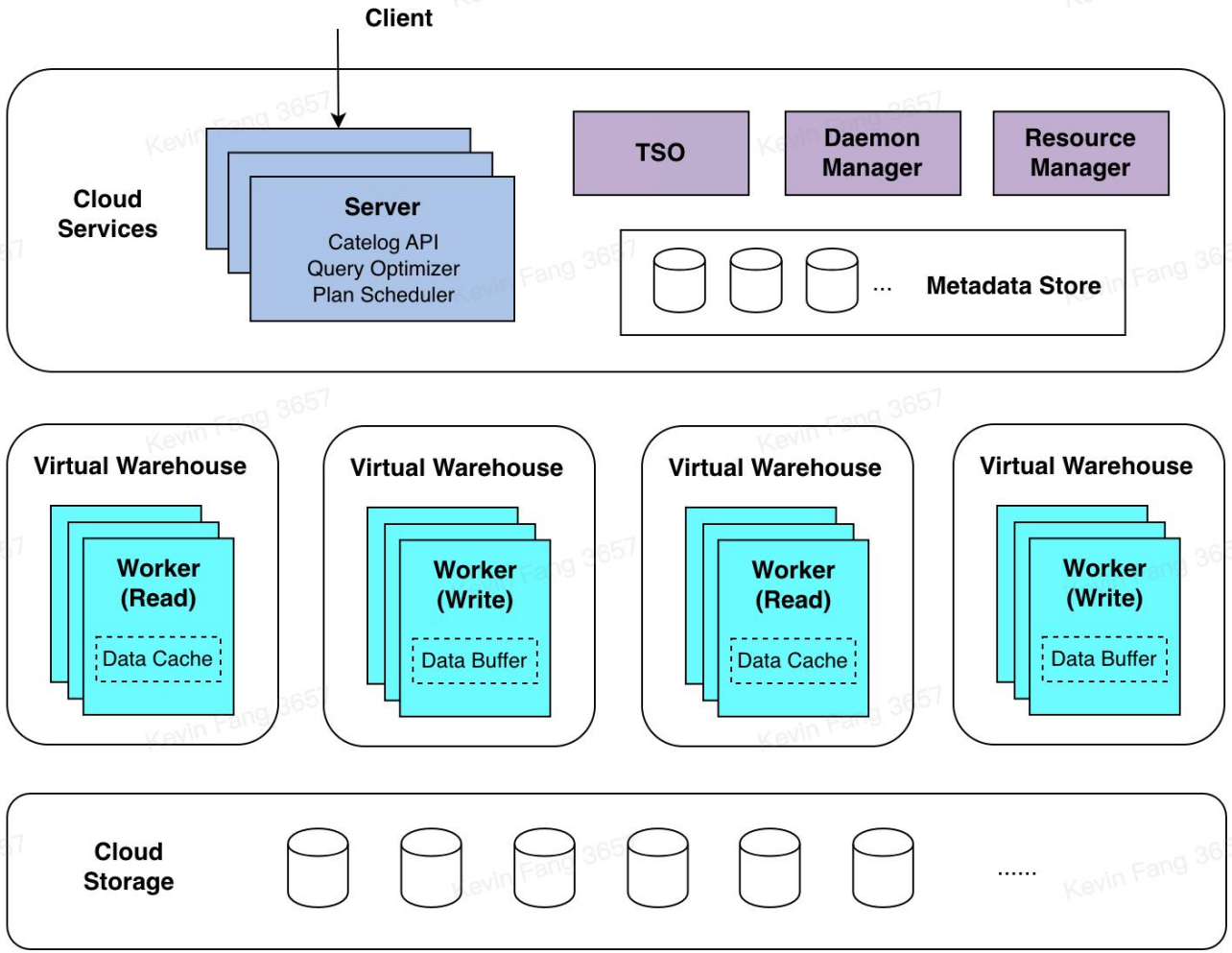
ByConity通过开源，融合希望打破常规技术的开发者，改变数据的使用方式

存算分离的优势

- ByConity是开源的基于存储计算分离架构的云原生数据仓库，具有以下特性：



基于云原生架构



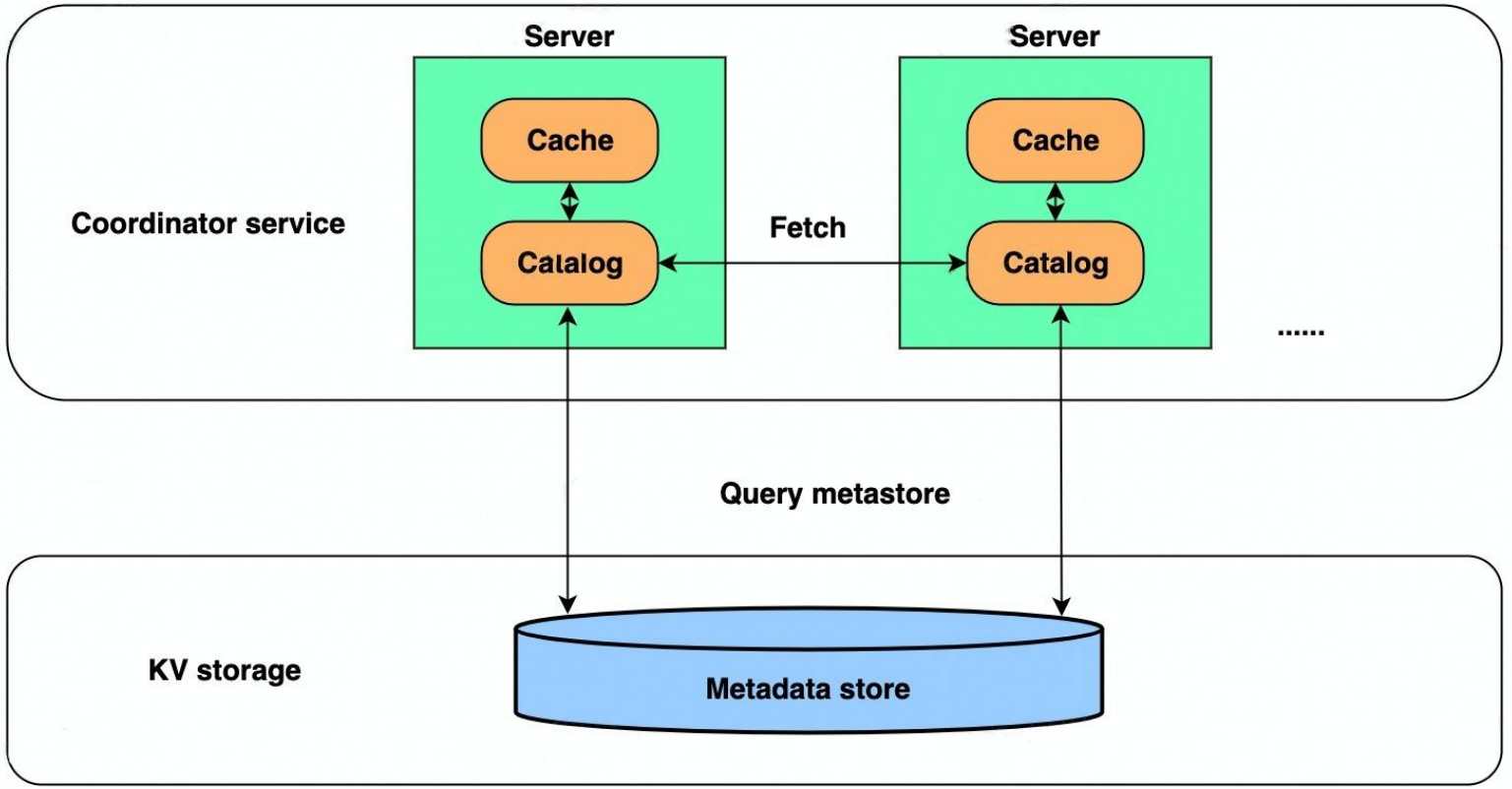
- 服务层（Cloud Service）
  - MetaDate：FoundationDB/ByteKV
  - Server：表元数据缓存、查询SQL解析、计划生成、调度和下发
  - Resource Manager：服务发现、负载心跳检测
  - TSO：全局唯一单调递增的时间戳
  - Daemon Manager：调度和管理任务
- 计算组（Virtual Warehouse，VW）
  - Worker：执行片段的执行，后台任务的执行、Local Disk Cache
  - 每个表可以设置默认的Read VW（查询）和Write VW（导入和Merge）
- 存储层（Cloud Storage）
  - 支持HDFS、S3

存算分离的思考

- 需要统一的元信息管理系统
- 分布式文件系统大多数存在元信息管理压力问题（nn）
- 分布式统一存储系统大多不支持rewrite，一些对象存储系统甚至不支持append
- 分布式对象存储系统大多move代价都比较高
- io latency通常情况对比本地文件系统下都存在增加的情况

统一的元数据管理

- 提供高可用和高性能的元数据读写服务
- 完备事务语义的支持
- 后端存储系统可插拔，方便扩展
- 高效的Part缓存管理
- 一致性hash



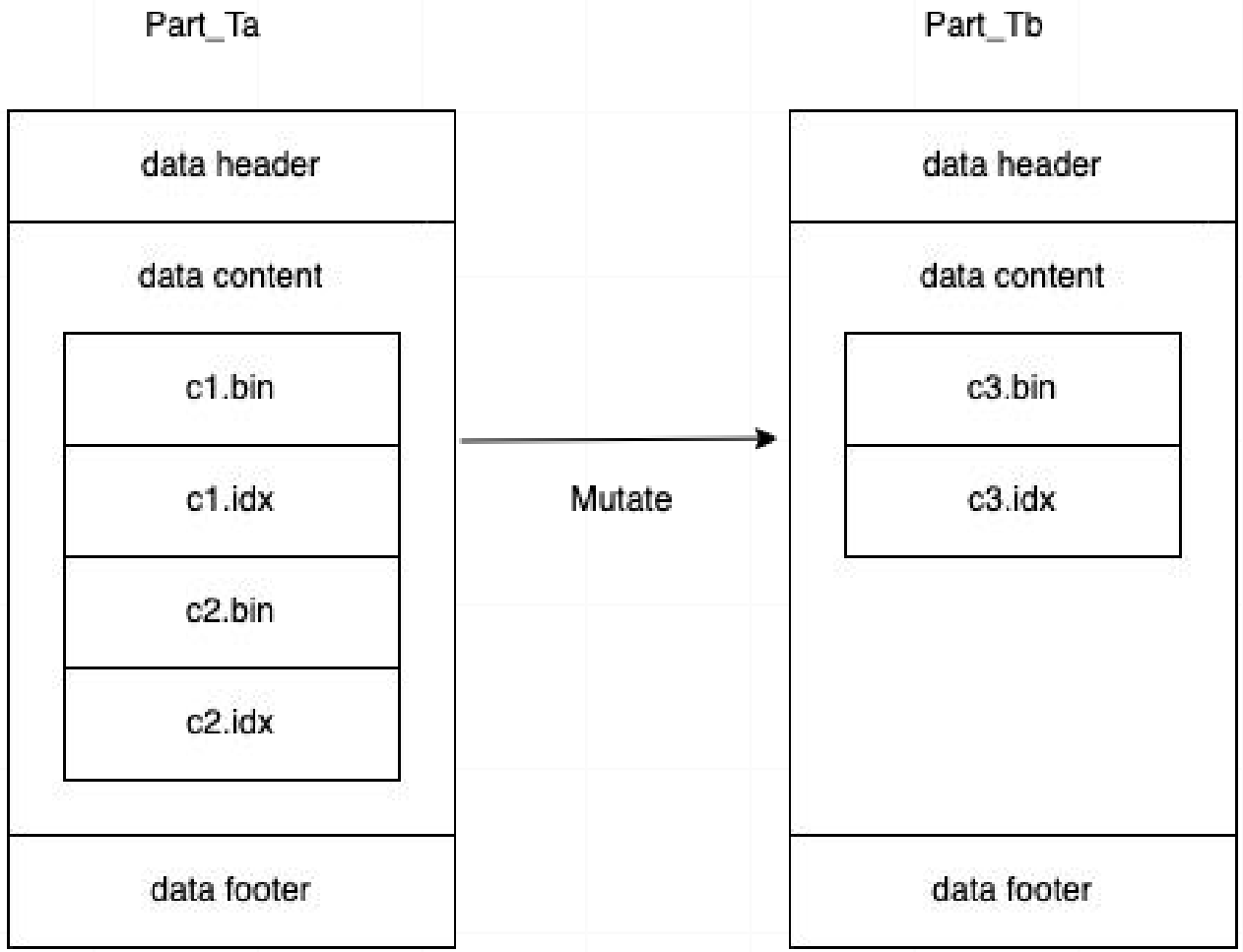
数据存储结构

- 合并小文件，每个part所有数据存储在一个文件中
- 保持按列存储特性

```
* Data information will be stored in cloud storage(e.g. s3 hdfs) as one data file,
* Which will not be updated once generated, only can be rewrited to new data file or be deleted.
*
* ------------data header---------
* magic_code(4 bytes)
* version(8 bytes)
* deleted(1 bytes)
* reserved size(256 - 12 bytes)
* -----------data content---------
* columns data files & idx files
* primary_index
* checksums
* metainfo
* -----------data footer----------
* primary_index offset(8 bytes)
* primary_index size(8 bytes)
* primary_index checksum(16 bytes)
* checksums offset(8 bytes)
* checksums size(8 bytes)
* checksums checksum(16 bytes)
* metainfo offset(8 bytes)
* metainfo size(8 bytes)
* metainfo checksum(16 bytes)
* unique_key_index offset(8 bytes)
* unique_key_index size(8 bytes)
* unique_key_index checksum(16 bytes)
* metainfo key (32 bytes)
* ------------------------------
*/
```

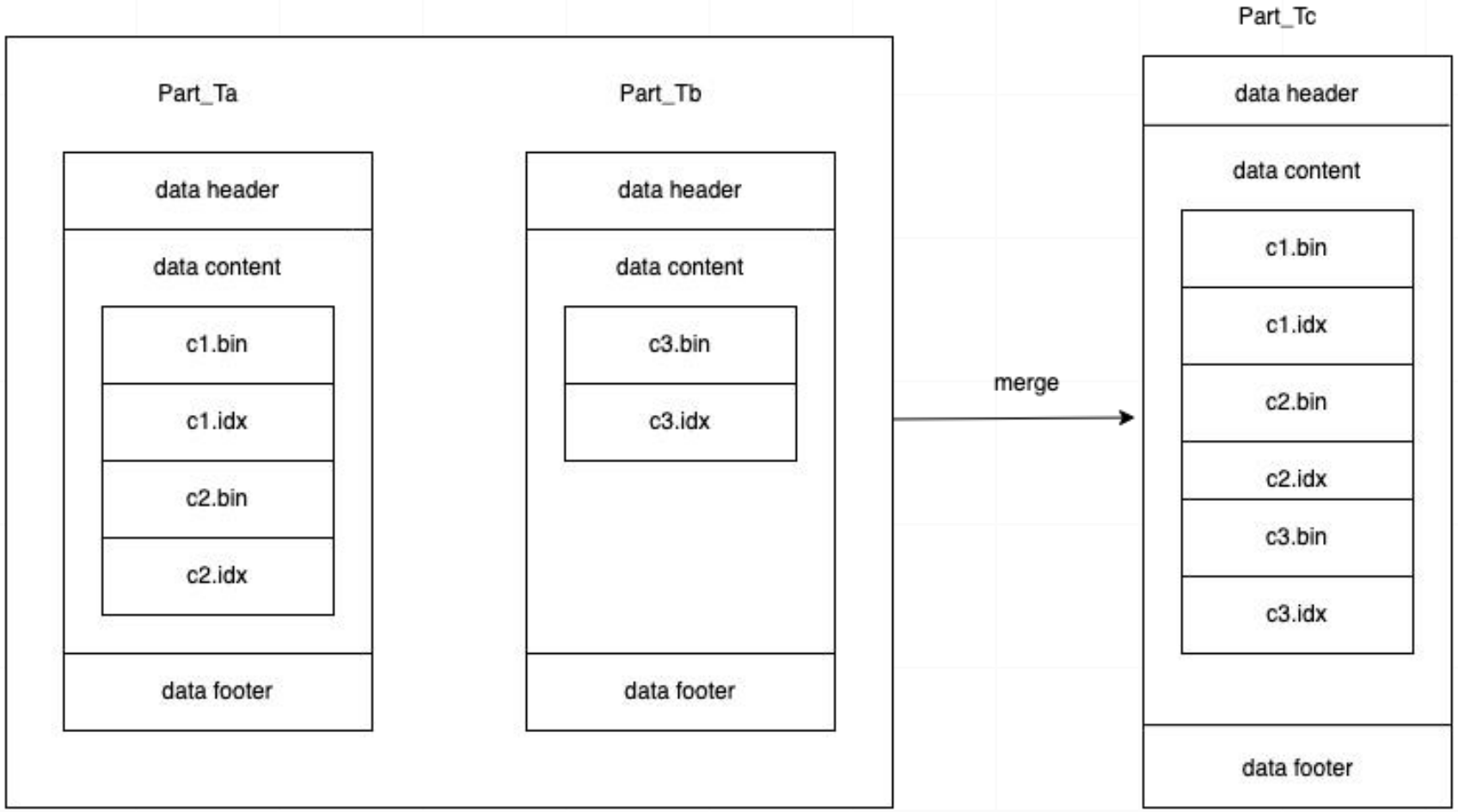
数据变更

- 文件生成后不再变动
- delta part + base part
- part chain（merge-on-write）
- 读放大



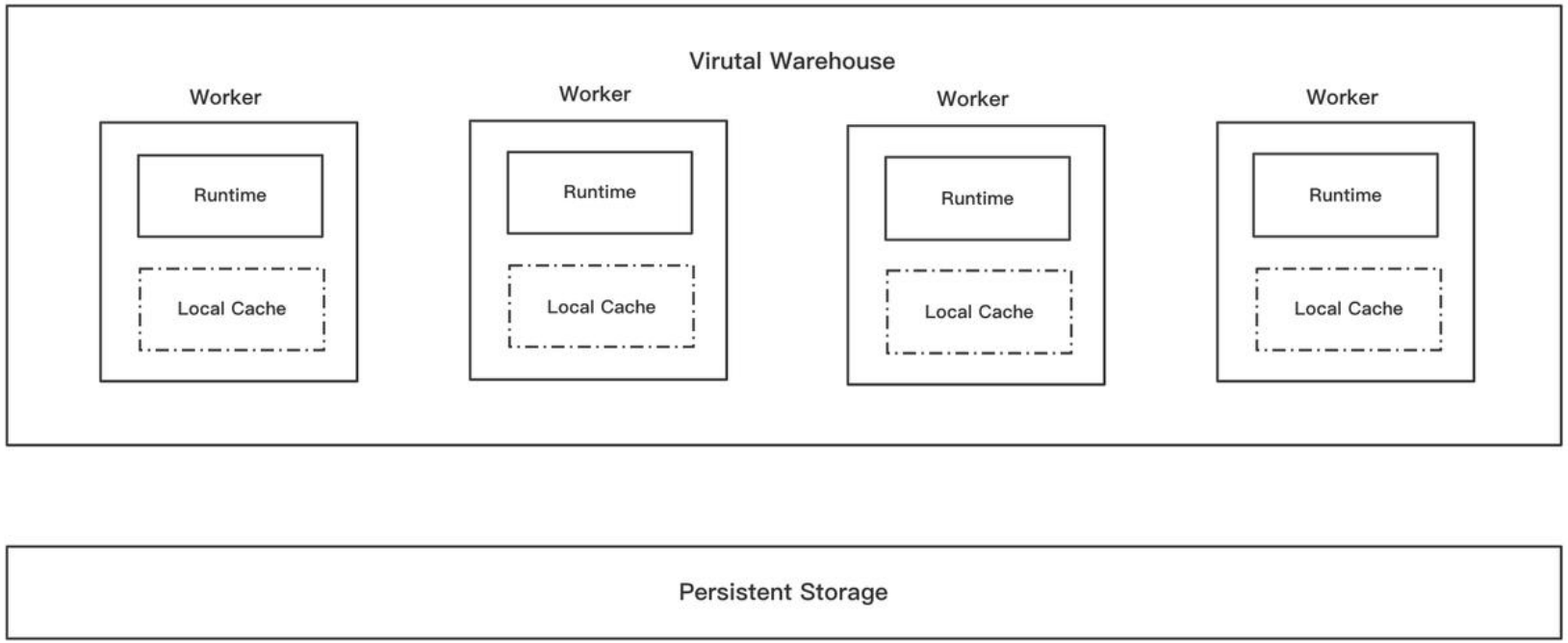
数据合并

- 异步merge
- Old parts通过GC清理



数据缓存

- 一致性hash分配parts
- 热数据worker节点自动缓存
- 改进bucket-lru算法
- 避免数据reshuffling



唯一键（UNIQUE KEY）

实际场景

- 数据源(如Kafka)包含重复数据，如何保障数仓表的数据质量?
- 业务数据流包含行更新，如何高效实时同步和分析?
- 如何提高RDBMS->数仓的同步时效性，并支持高效分析?

唯一键 + Upsert

- 面向读取操作进行优化
- 支持唯一键与排序键不同
- 支持基于版本字段的比较
- 支持行删除
- 支持表级别和分区级别

```
CREATE TABLE t
( `d` Date, `id` Int32, `name` String )
ENGINE = CnchMergeTree -- cloud native table engine
PARTITION BY d ORDER BY name
UNIQUE KEY id;

insert into t values (today(),1,'A'), (today(),2,'B');

select * from t order by id;
"2022-03-03",1,"A"
"2022-03-03",2,"B"

-- UPSERT semantics
insert into t values (today(),2,'B1'), (today(),3,'C');

select * from t order by id;
"2022-03-03",1,"A"
"2022-03-03",2,"B1"
"2022-03-03",3,"C"
```

查询优化

- 优化器：本质是对查询计划的等价转换，从中找到最优解或者较优解。ByConity实现了RBO和CBO
- RBO：基于规则的优化能力。使用一系列预定义的启发式规则来选择查询执行计划。
  - 基于visitor的全局改写，例如filter下推、列的裁剪、SQL指纹等
  - 基于pattern-match的局部改写，例如多个filter的merge、多个projection的merge
- CBO：基于代价的优化能力。通过收集和分析数据库中的统计信息来评估不同执行计划的成本，并选择成本最低的计划作为最佳计划。
  - 基于Cascades搜索框架，遍历等价计划，评估每种等价计划的代价，选出最优解
  - Join Reorder 超过10表启发式搜索
  - 分布式执行计划，属性传递，基于代价生成最优的分布式计划

查询调度

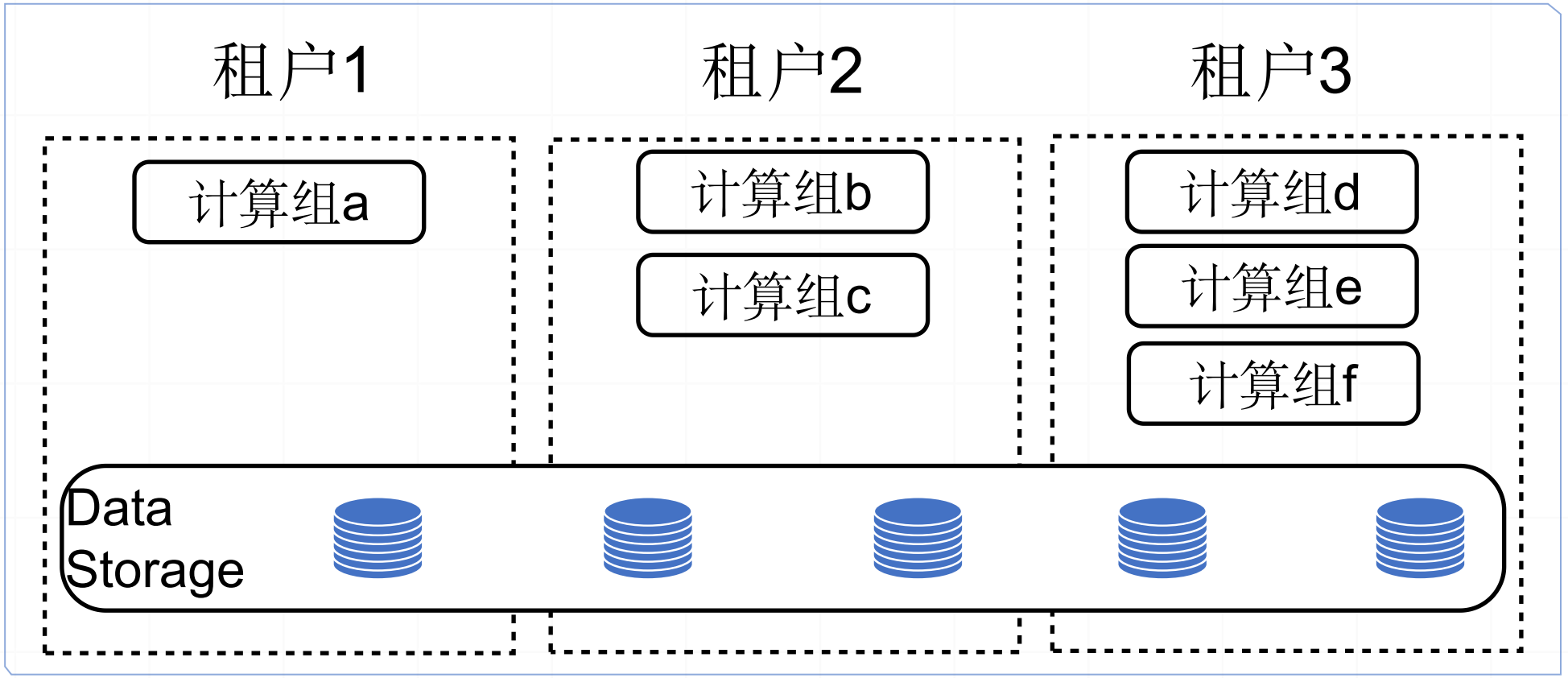
- 负责对生成的可执行计划plan segment tree进行调度

- Cache-aware调度 – 针对source，读取数据
  - 最大化cache命中率，提升读写性能
  - 拓扑发生变化时，最小化cache失效的影响

- Resource-aware调度和流量控制 – 针对计算节点，纯计算
  - 最大化资源利用率
  - 合理使用资源，避免负载过高

计算组

- 多租户隔离
- 读写分离
- 水平和垂直动态扩缩容
- 资源共享

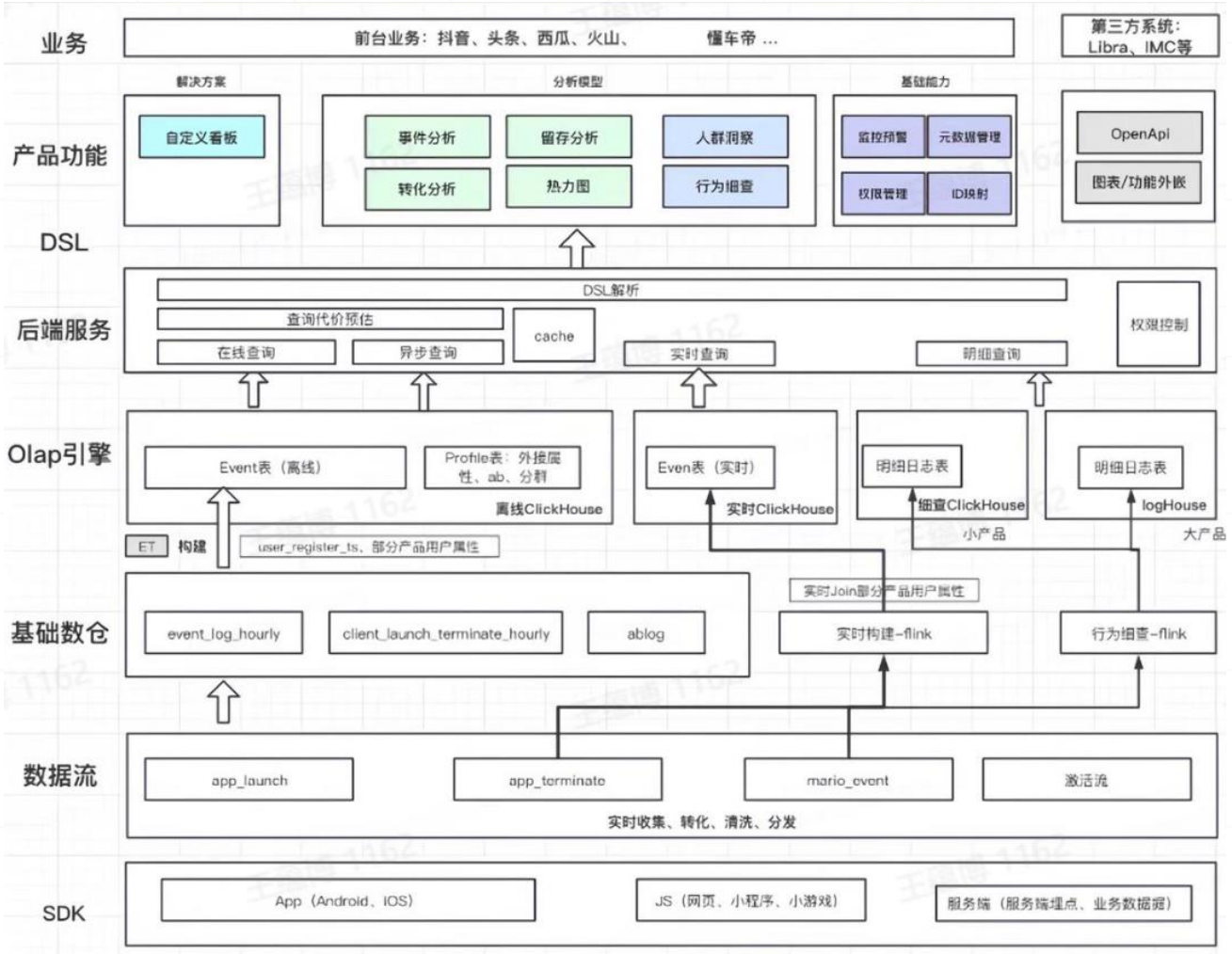


在分析系统场景实践（1/2）

320TB

2.3万亿行

2万个维度



在分析系统场景实践（2/2）

避免资源抢占

天然的资源隔离和租户隔离，不同用户查询相互不收到影响

节约资源成本

基于Kubernetes的弹性伸缩能力，实现无感扩缩容

运维成本降低

存算分离架构，计算节点为无状态节点，发生故障秒级替换

2024年整体规划

性能提升

- 优化动态构建filter能力
- 全局字典
- Zero-copy
- 非等值join算子优化
- 并行化重构
- Bucket表优化
- UncompressedCache优化
- 多表异步物化视图

数据湖分析

- 支持Hive表查询、写入
- 支持Hudi表查询、写入
- 支持Iceberg表查询
- 外表查询性能优化
- 外表物化视图
- Hive元数据Catalog同步

数仓能力

- 复杂大数据ELT稳定运行
- Runtime Filter性能优化
- Shuffle性能优化
- 算子落盘
- 长事物优化
- 失败重试
- MySQL兼容性
- SQL诊断优化

数据安全和备份恢复

- 透明加密
- 表级快照
- 全量备份恢复

易用性

- 一键部署
- Local mode

生态

- Dbeaver
- Kubeblocks
- SugarBI
- Quicksight

字节跳动开源

Web, Audio & Video

ByteX

xgplayer

IconPark

AlphaPlayer

ByteMD

Scene

arco.design

AabResGuard
POWERED BY DOUYIN ANDROID TEAM

BoostMultiDex

.....

AI

BytePS

ByteIR

MatxScript

LightSeq

.....

BigData

BitSail

ByConity

.....

Infrastructure

CloudWeGo

KubeWharf

Sonic

MonoIO

.....

Security

Elkeid

appshark

.....

Contributors 2000+ **Stars** 100000+

ByConity社群小助手

ByConity公众号

微信扫描左侧二维码

加入社区，共同成长

感谢观看

云原生社区
Cloud Native Community