
Architect
SACC
2022 中国系统架构师大会
SYSTEM ARCHITECT CONFERENCE CHINA 2022
激发架构性能 点亮业务活力
云上会议 网络直播 | 2022年10月27-29日
ITT168.com
ChinaUnix.net
ITPUB

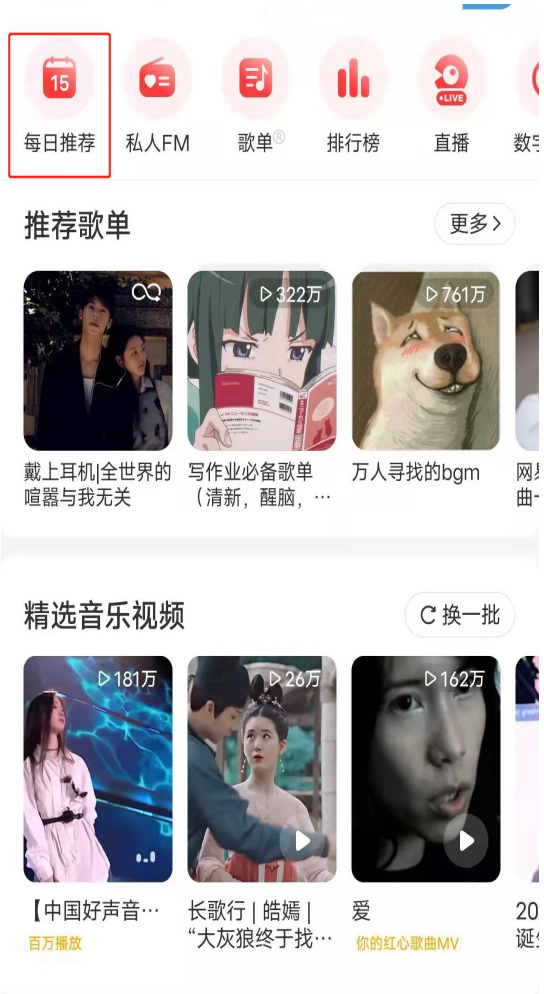
云音乐分布式KV存储实践和演进

网易云音乐 存储资深工程师 张磊

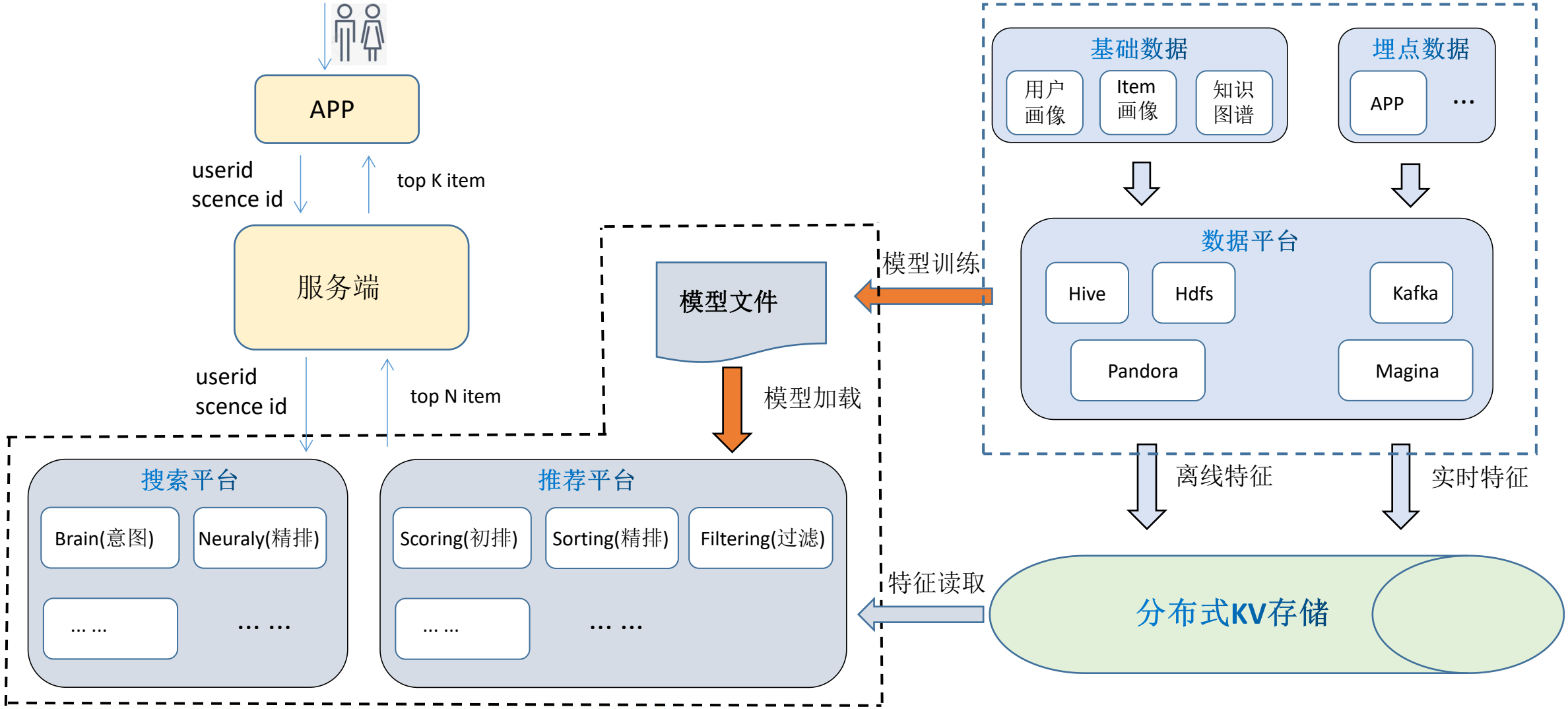
大纲

- 业务背景
- 分布式KV存储实践
- 存储架构演进和未来展望

业务背景 - 推搜场景



业务背景 - 推搜架构


APP
userid
scence id
top K item
服务端
userid
scence id
top N item
模型文件
模型训练
模型加载
基础数据
用户画像
Item画像
知识图谱
埋点数据
APP
...
数据平台
Hive
Hdfs
Kafka
Pandora
Magina
离线特征
实时特征
特征读取
分布式KV存储
搜索平台
Brain(意图)
Neuraly(精排)
... ...
... ...
推荐平台
Scoring(初排)
Sorting(精排)
Filtering(过滤)
... ...
... ...

业务背景 - KV存储需求

业务通用需求

1. 千万级的并发设计要求，0.x ms ~ x ms时延响应，持久化存储。

2. 支持方便的动态扩容、缩容，以应对大规模的特征数据存储。

3. 架构优秀，支持自动的容错恢复，维护简单方便。

业务定制化需求

1. 特征数据离线大规模集中写入，数据规模亿级或者TB级别挑战。

2. 特征数据Schema复杂，特征更新路径漫长而低效。

3. 随机读写、大value存储（x KB ~ xxx KB）挑战，如模型实时化样本快照存储、用户完整历史行为数据。

4. 海量用户特征数据的存储成本问题。

分布式KV存储 - 技术选型

Redis	Codis	Tair	HBase	MongoDB	…
高性能内存数据读写，但单线程易抖动，集群方案还有优化空间。	Redis的集群部署方案，但本质上为代理中间件、存储路径长、资源成本高。	具备多引擎支持能力，但代码老旧、社区开源后极少维护。	海量数据磁盘存储，适合写多读少场景，但读实时性不足。	适合读多写少场景，但写性能一般	

开源方案

优点：初期成本低，存储能力起步快。

缺点：存储能力天花板低，未来可能会制约业务发展。

自研方案

缺点：初期成本高，存储能力起步慢。

优点：存储能力天花板高，业务几乎不存在存储瓶颈。

存储挑战

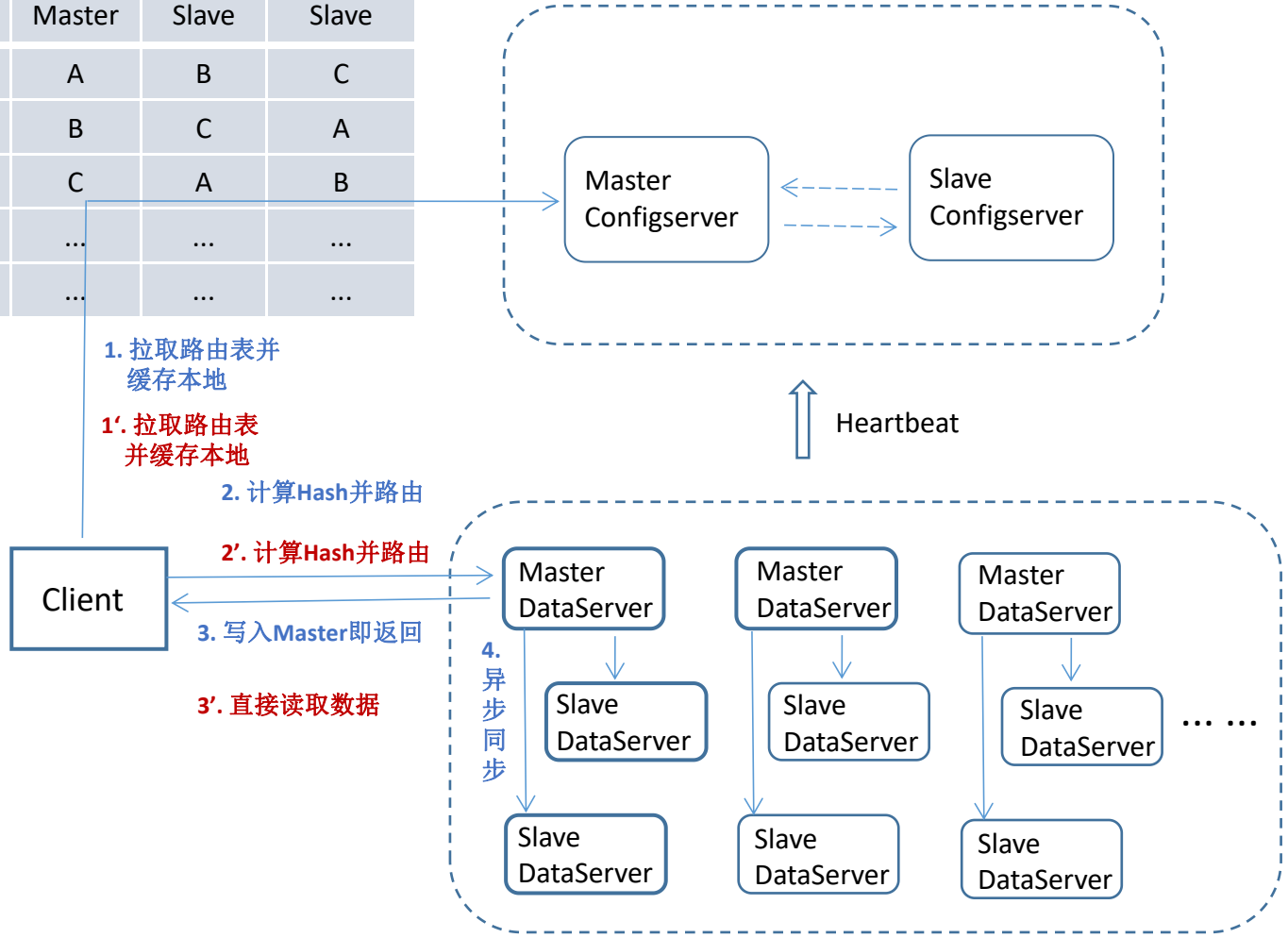
1. 数据规模大和请求并发高低延迟。
2. 存储场景复杂且多样。
3. 开源存储方案只能解决部分场景问题，且无法定制化。

基于Tair的分布式KV存储

1. 支持千万级的并发、低延迟和动态扩缩容能力。
2. 多存储引擎架构，支持业务场景定制化能力。

分布式KV存储 - 基于Tair的架构

Bucket	Master	Slave	Slave
0	A	B	C
1	B	C	A
2	C	A	B
…	…	…	…
1023	…	…	…



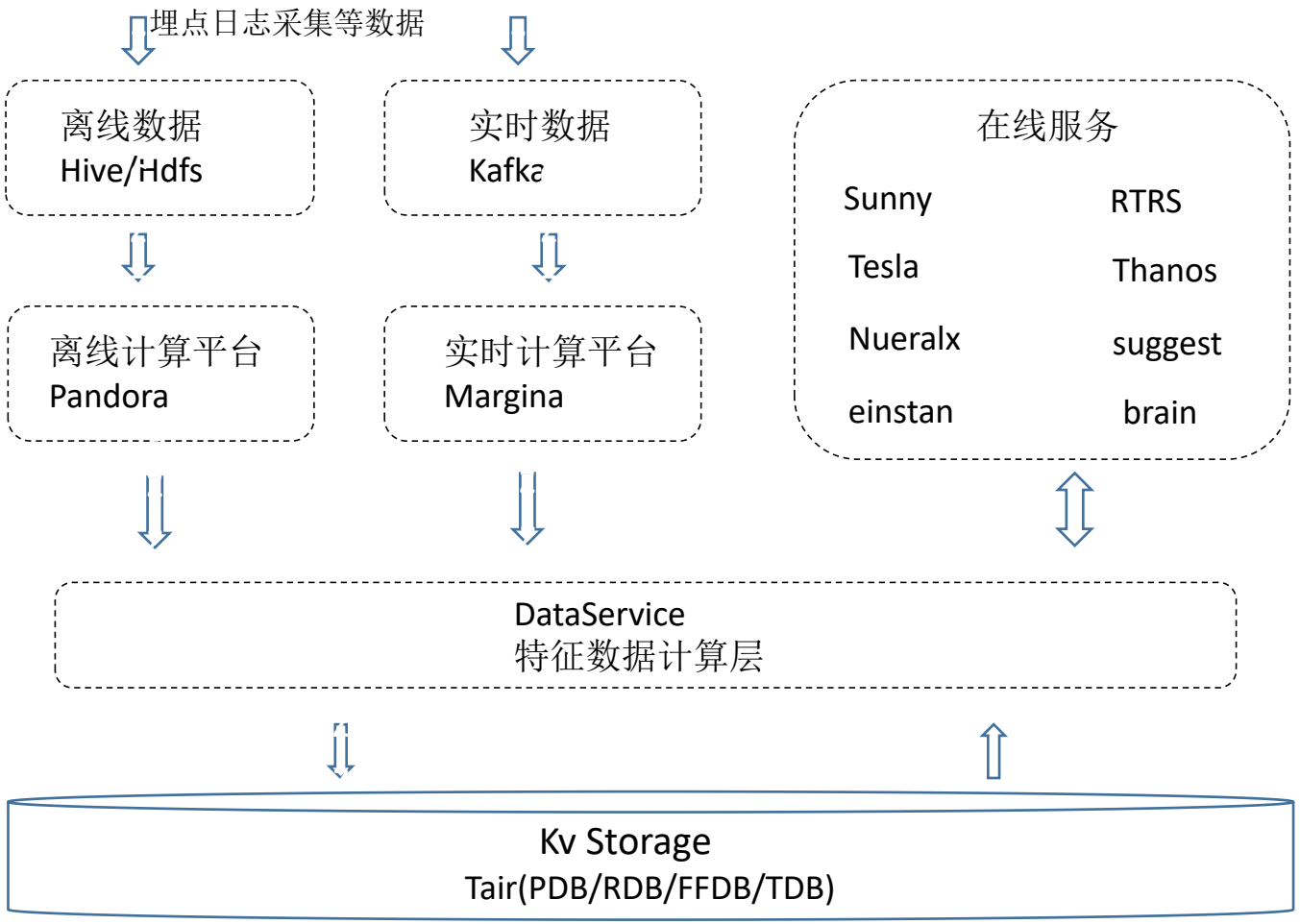
架构层

1. 内存数据持久化机制实现。
2. 网络框架重写和统一。
3. Configserver HA主备方案。
4. 细粒度监控能力的支持。
5. 磁盘存储能力。

特定场景

1. 存储层支持Protobuf Schema的CRUD。
2. 海量流式数据存储的随机高效读写。
3. 海量离线数据短时间集中写入。
4. 用户长序列特征存储(大value、增量更新)。
5. 时序数据存储和计算。

分布式KV存储 - 业务数据流



Tair-PDB

支持protobuf结构字段级CRUD能力

Tair-RDB

支持磁盘型海量数据存储

Tair-FFDB

支持海量流式数据高效随机读写

Tair-TDB

支持百万级指标实时计算的时序数据存储

分布式KV存储 - PDB存储方案

背景问题

1. Value部分更新操作流程复杂

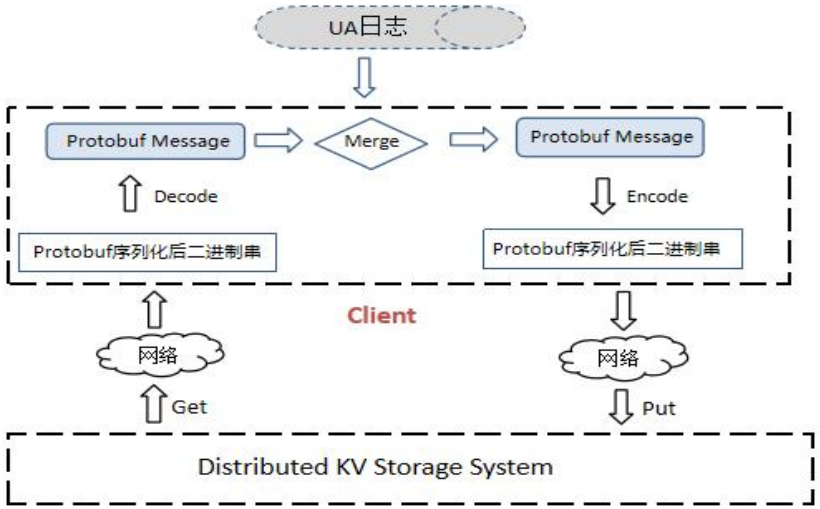
   Get+Decode+Merge+Encode+Set

2. 存储压力大

   大批量实时特征更新，存储 IO压力巨大。

3. 并发更新丢失

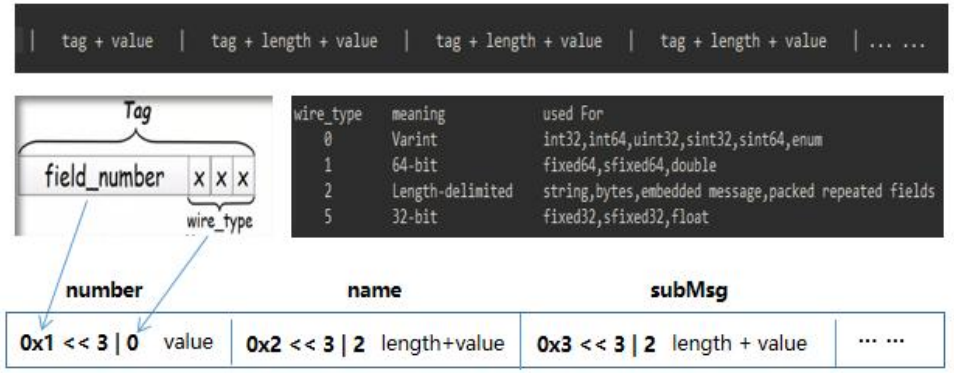
   并发更新相同key时会丢失更新。

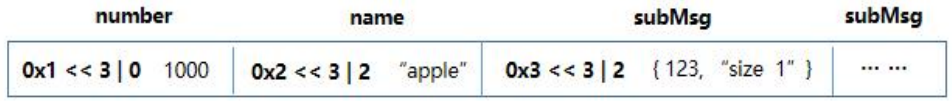


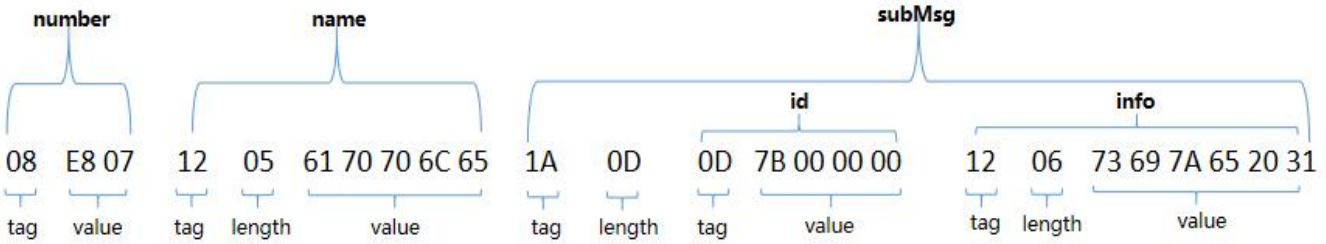

```
message ExampleMessage
{
    message EmbMsg
    {
        fixed32 id = 1;
        bytes info = 2;
    };

    int32 number = 1;
    string name = 2;
    repeated EmbMsg subMsg = 3;
}
```

```
number: 1000
name: "apple"
subMsg {
    id: 123
    info: "size 1"
}
subMsg {
    id: 456
    info: "size 2"
}
... ...
```







分布式KV存储 - PDB存储方案

业务效果

1. 简化业务操作

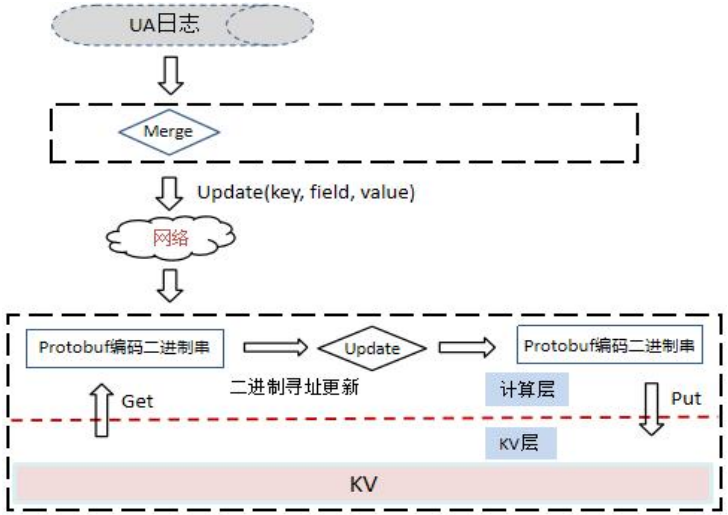
   存储层支持多种protobuf的CRUD操作

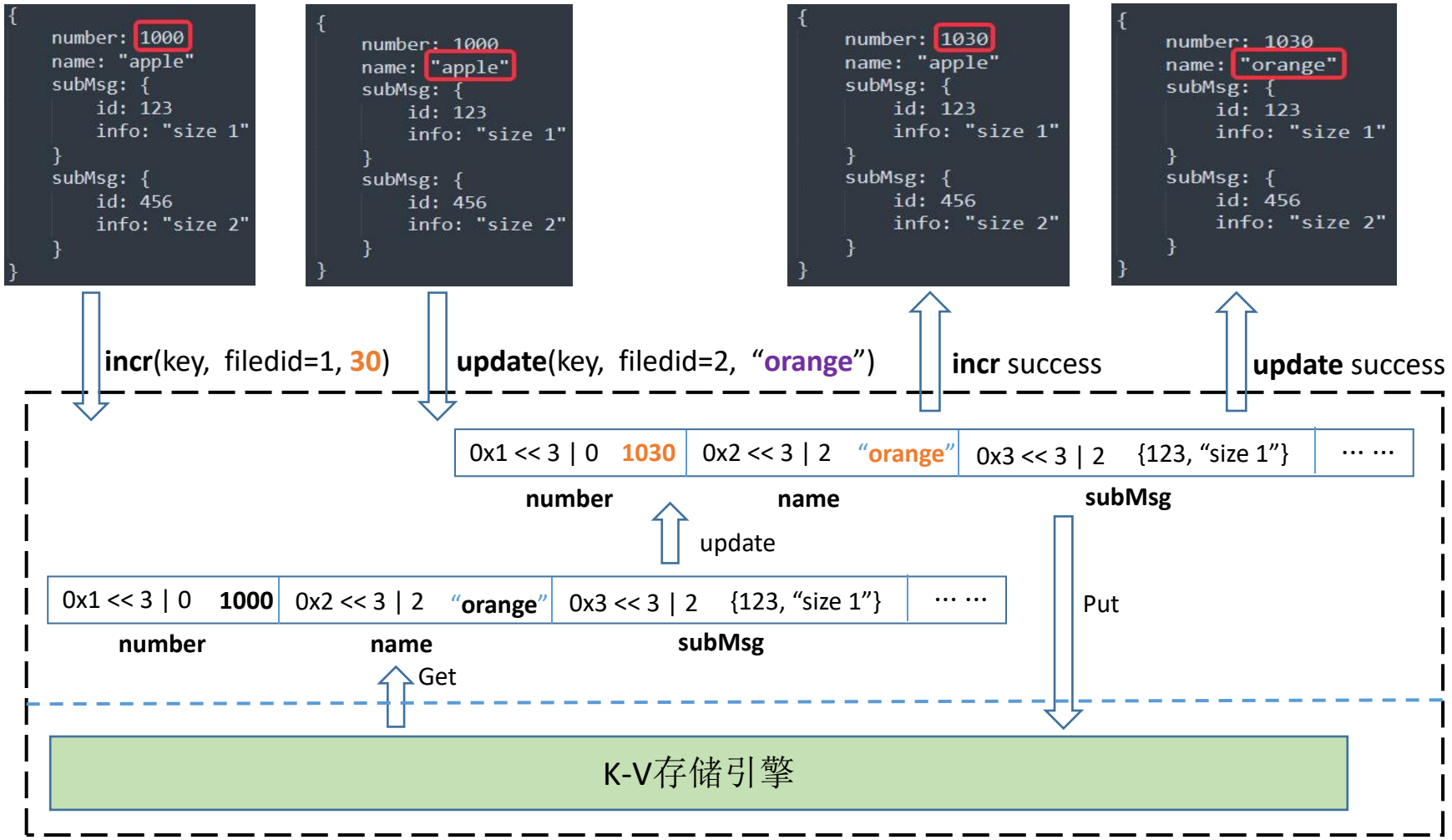
2. 存储效率提升

   两次存储IO减为一次存储IO开销

   计算资源降低60%，RT降低70%

3. 解决并发更新丢失（悲观锁、乐观锁）



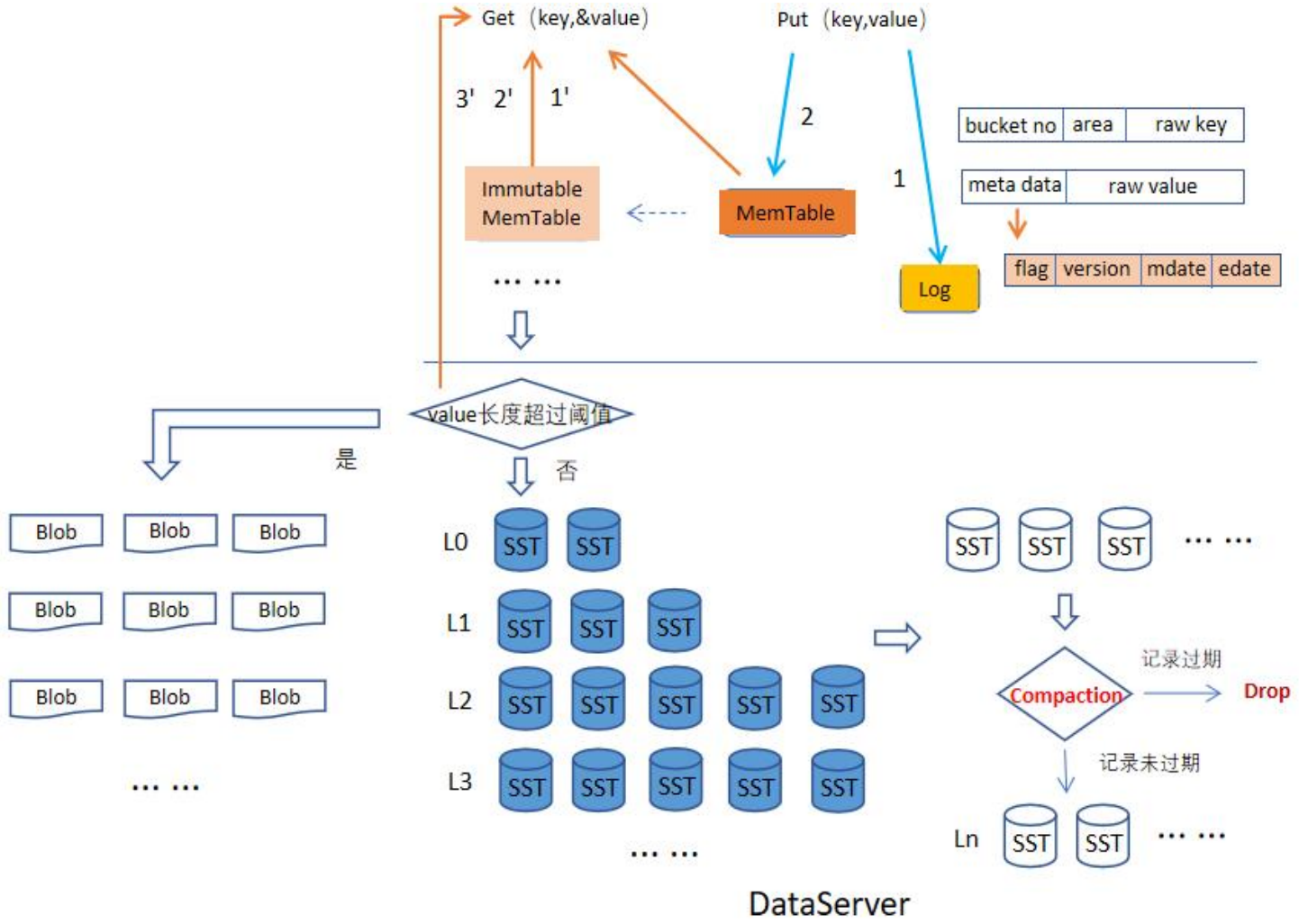


分布式KV存储 - RDB KV分离方案

Tair-RDB
基于Rocksdb的磁盘型大规模数据存储。

背景问题（大value）
1. 触发compaction概率更高，写放大更大。
2. compaction不可控，影响性能和数据写入。
2. 更低的cache命中率，影响性能。

KV分离存储
平均RT降低60%，空间增加30% ~ 40%。



分布式KV存储 - RDB Bulkload方案

背景问题

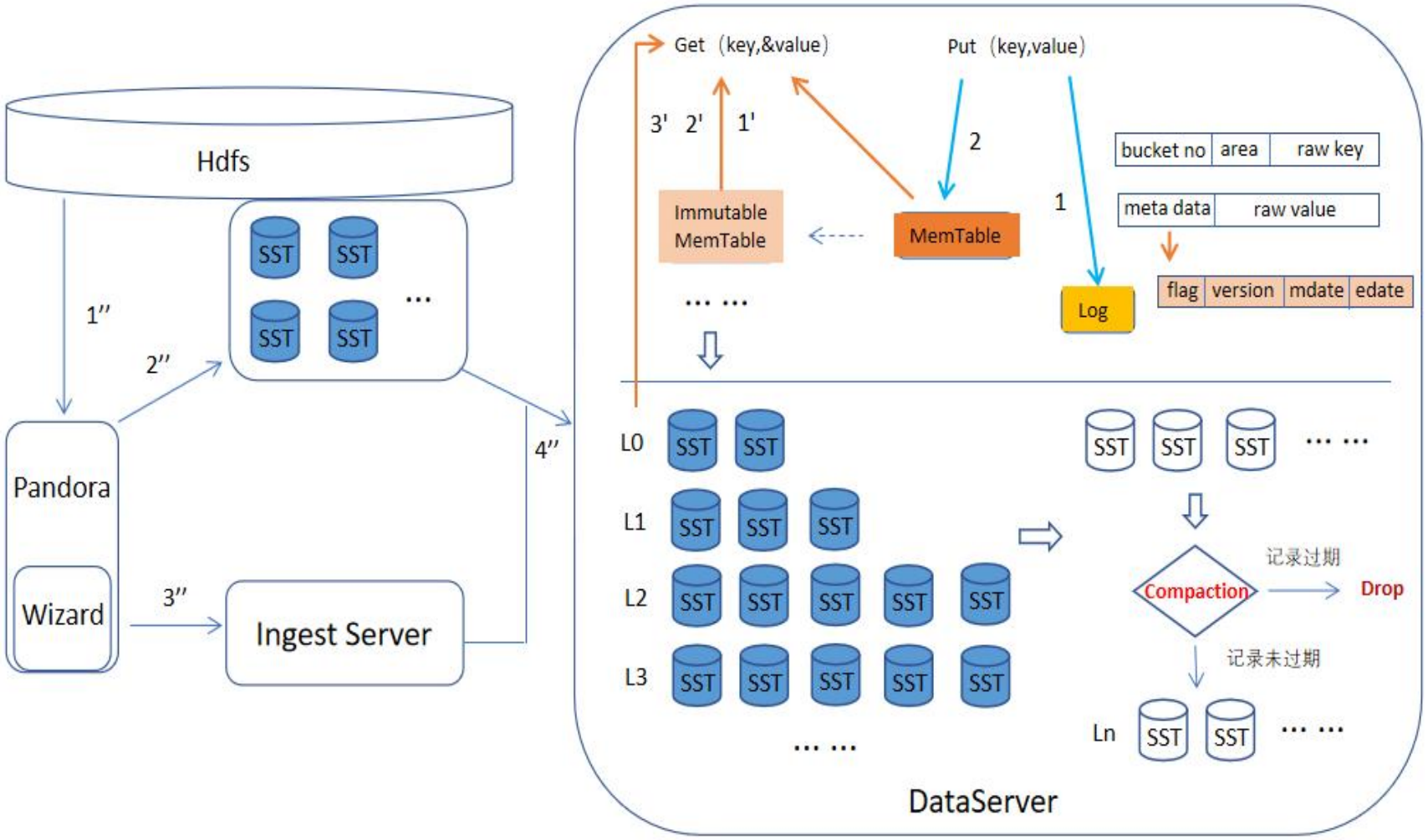
用户离线特征规模数亿条或TB级，如用户历史行为数据、标签数据，易出现短期集中写入问题。

存储挑战

1. 传统单条记录写入耗时长。
2. compaction压力大。
3. 存储极易抖动。

Bulkload方案

IO压力、RT相对逐条写入方式降低66%。



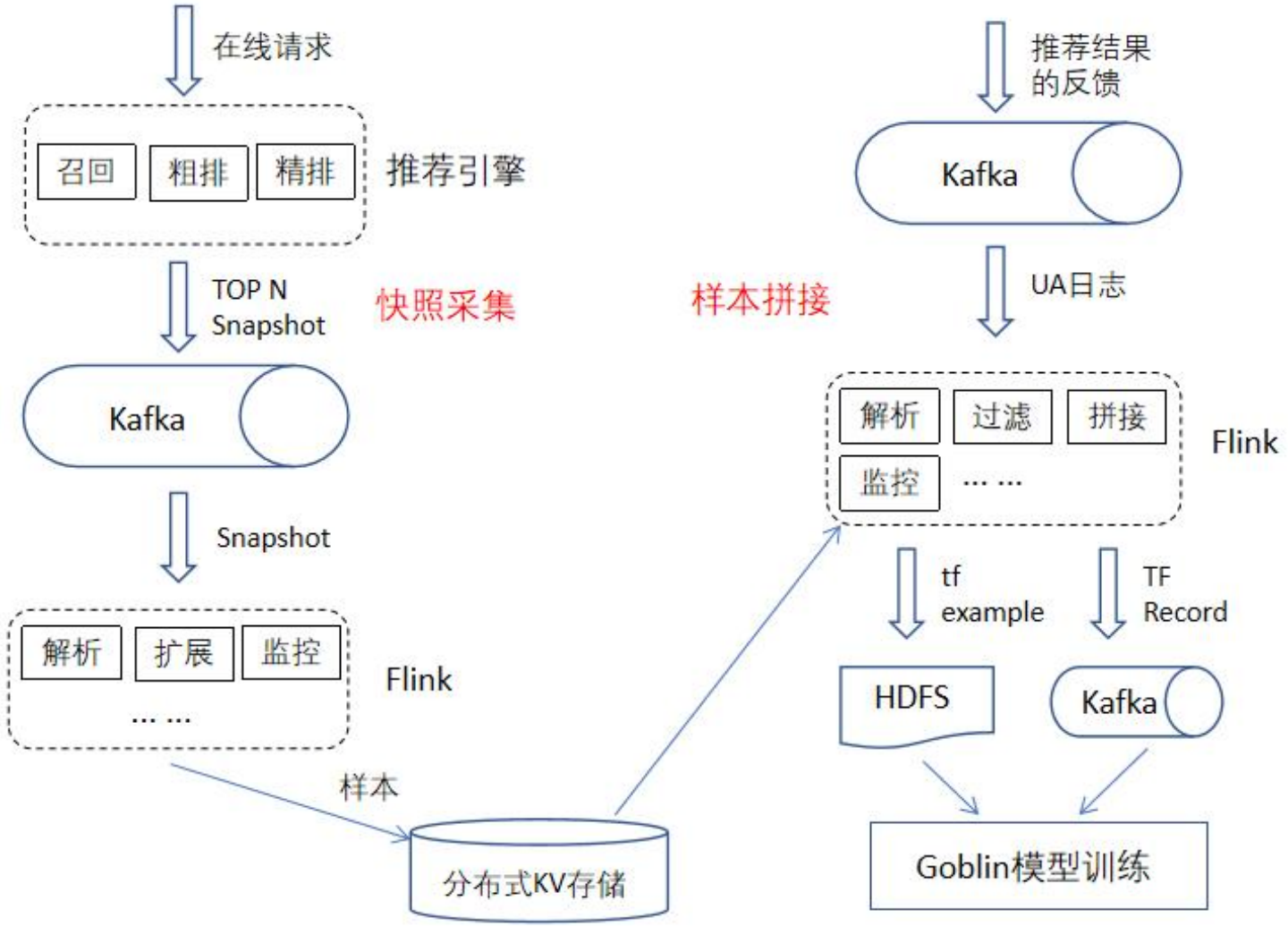
分布式KV存储 - FFDB存储方案

业务背景

模型实时化以更好的拟合数据的分布，帮助模型发现流行的数据pattern，推荐出实时的流行趋势。

存储挑战

1. key几乎不重复，属于完全的随机读写。
2. 没有热点key。
3. 实时样本写入的value非常大，几十KB ~ 上百KB。
4. 数据规模非常大，数TB ~ 数十TB。
5. 固定的数据过期窗口。



分布式KV存储 - FFDB存储方案

Memory内存结构

1. 提高写入并发。
2. 将随机的写入转换为后台追加写入。

倒排索引

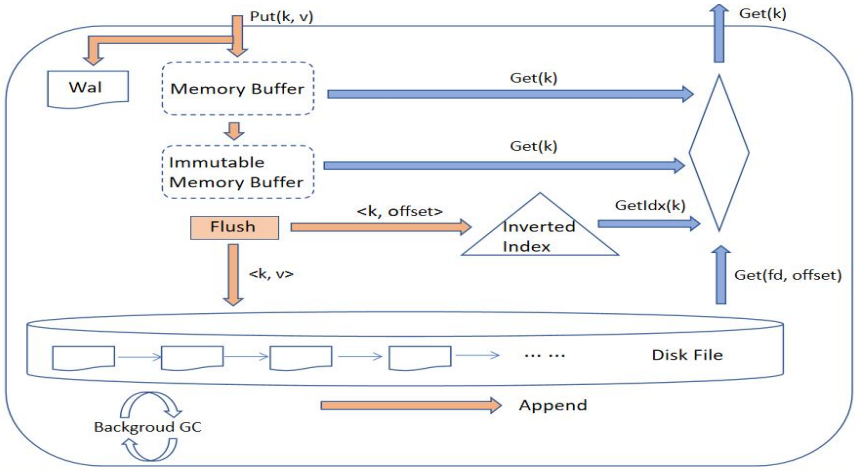
记录key-offset的索引，用于热点的随机数据读取。

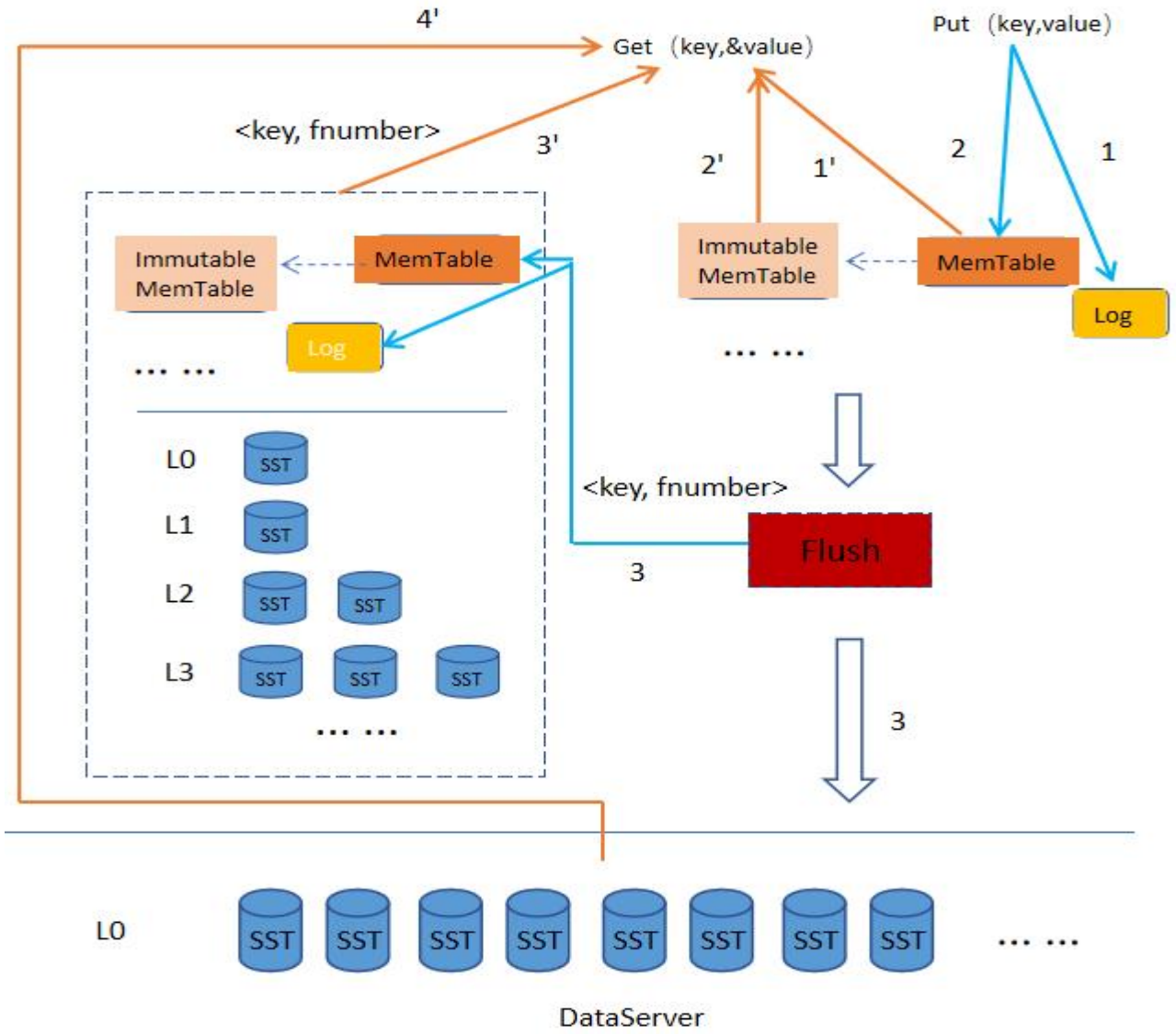
磁盘文件

按照创建的时间点先后以队列模式组织。

存储效果

资源开销、系统负载相比开源存储方案降低90%左右。





分布式KV存储 - TDB时序数据存储方案

OpenTSDB设计

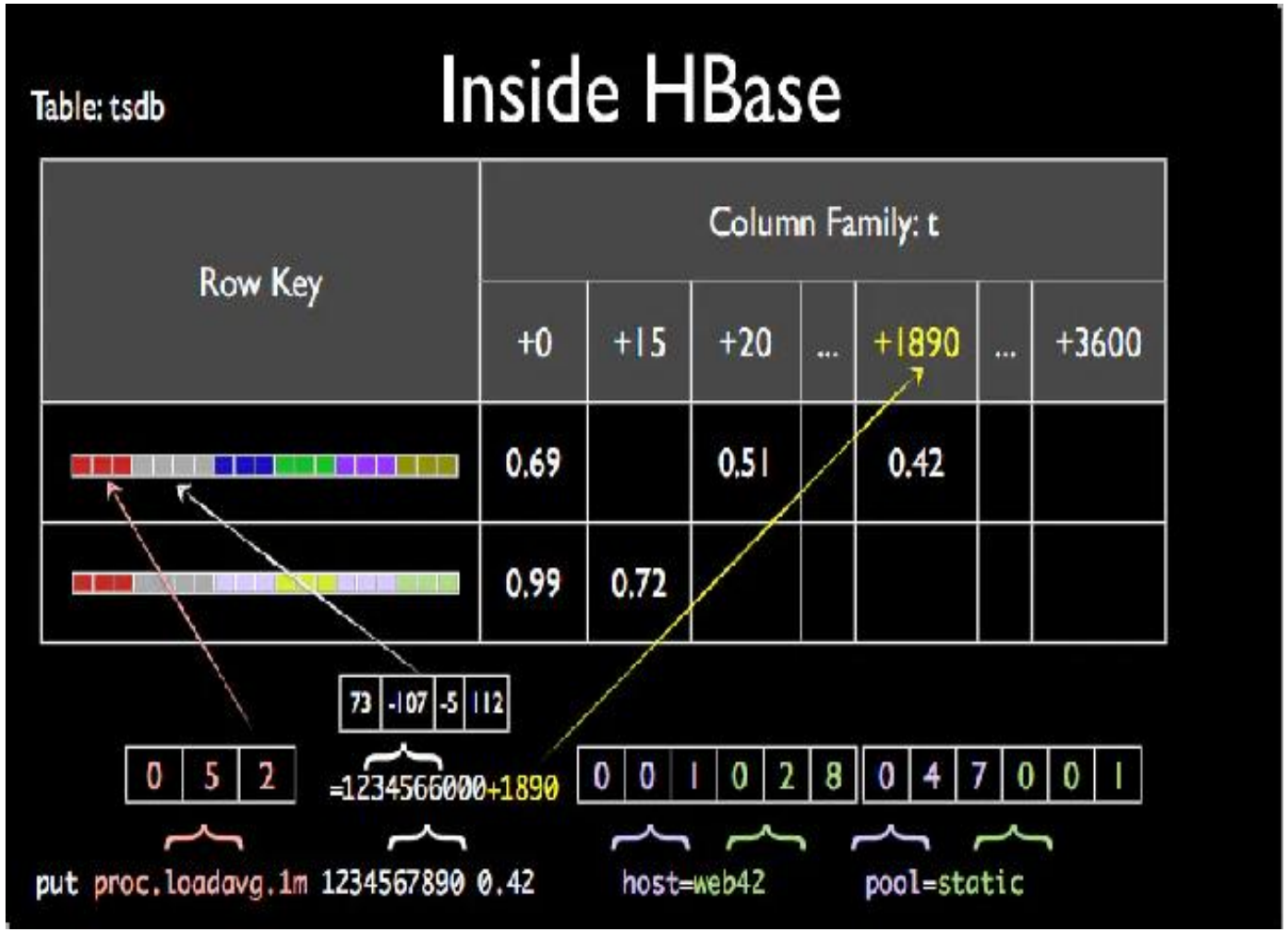
1. 依赖于HBase实现分布式存储能力。
2. rowkey设计：

   metric + base_time + tag1k + tag1v + tag2k +

   tag2v + ... + tagnk + tagnv
3. rowkey有序按time range存储，便于对metric下基于各标签的数据做聚合，以标签为条件做检索。
4. rowkey编码存储，CF存储/多列合并减少KV记录数并提升读写效率，预分桶方式热点优化。

OpenTSDB不足

1. 运维成本极高，不但需要维护TSD节点，还需要同时维护hbase和底层的HDFS。
2. 多维标签检索困难，因为没有针对时间线构建倒排索引，需要硬扫Hbase表。



分布式KV存储 - TDB时序数据存储方案

InfluxDB设计

1. Series Key设计：

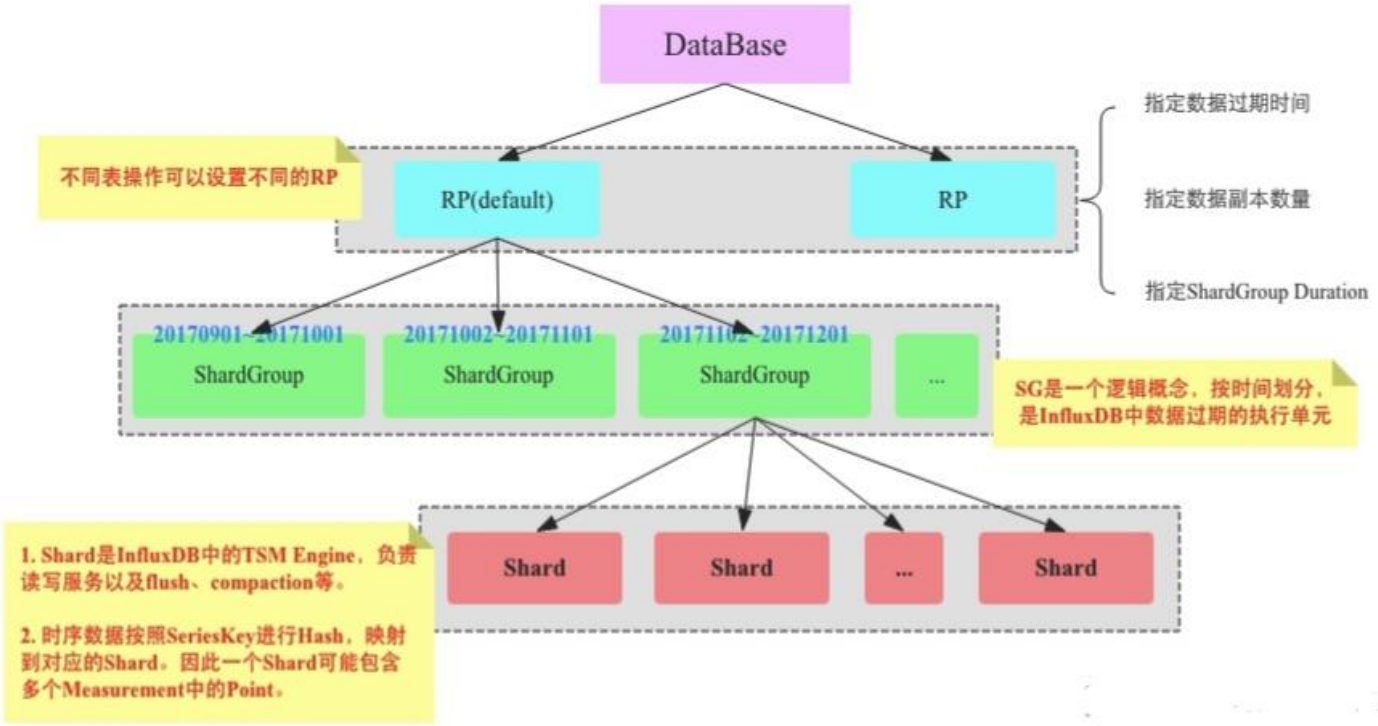
   <metric>[,<tag-key>=<tag-value>...] <field1-key>=<field1-value>[,<field2-key>=<field2-value>...]

2. 数据点两次路由设计：

   按时间戳进行Range路由 + 按Series Key进行Hash路由

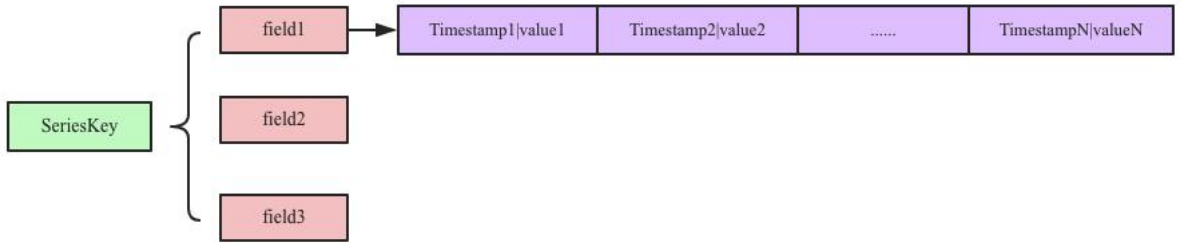
3. 数据存储设计：

   <SeriesKey, List<Timestamp|Value>>



InfluxDB不足

1. 开源版本只有单机模式，无集群模式。
2. 采用自研的存储引擎，维护和学习成本相对更高。



分布式KV存储 - TDB时序数据存储方案

时序数据库基本概念

Metric：度量，相当于关系型数据库中的table。

Timestamp：时间戳，代表数据点产生的时间。

Tag：标签，指定度量下的数据子类别。

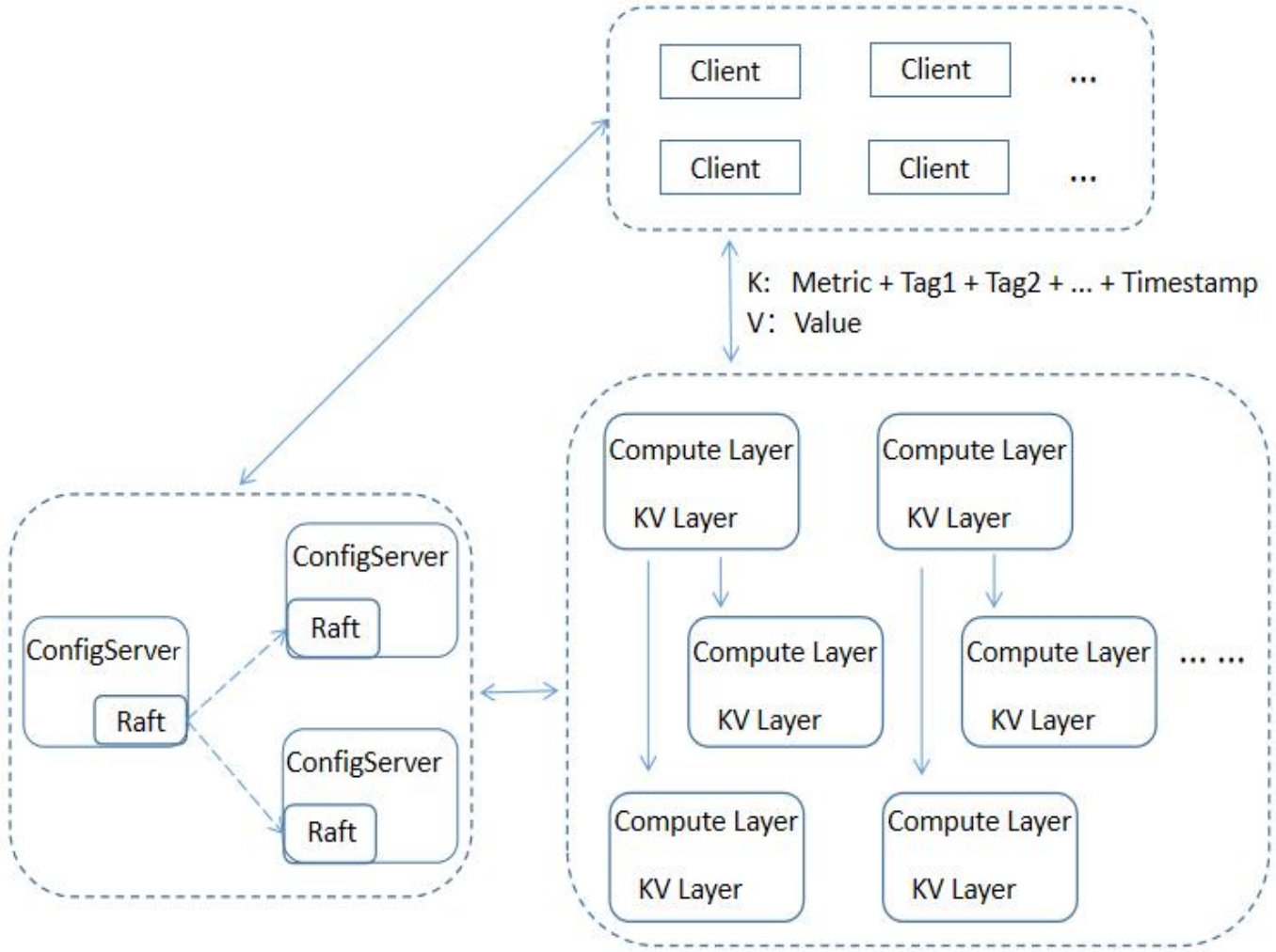
Data Point：数据点，度量 + 标签 + Timestamp。

挑战

1. 持续高并发追加写入，每天35亿数据点。
2. 百万级指标（数十亿级数据点）聚合查询，任意标签维度检索查询，毫秒级响应。
3. 半年或一年数据有效期，低成本持久化存储。

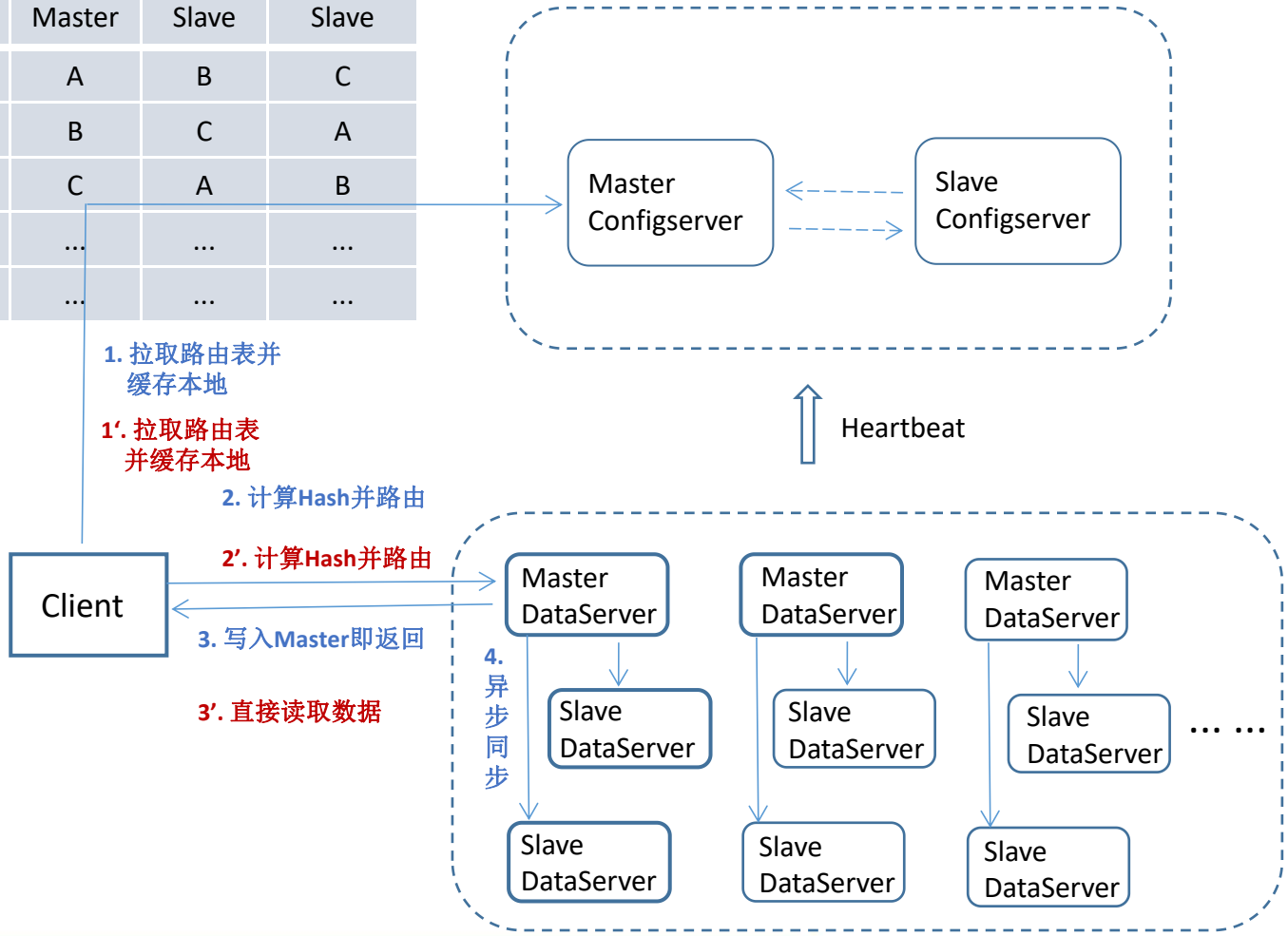
存储设计

1. 数据点存储设计：key - [ts-val, ts-val, ts-val, ...]
2. 针对时间线构建标签粒度倒排索引，优化多维标签检索性能。



存储架构演进

Bucket	Master	Slave	Slave
0	A	B	C
1	B	C	A
2	C	A	B
…	…	…	…
1023	…	…	…



基于Tair的分布式KV存储的不足

1. 开发和维护成本高

   整套架构实现已经非常老旧，且频繁的改造让代码结构变得较为混乱。

2. 设计上还存在固有缺陷

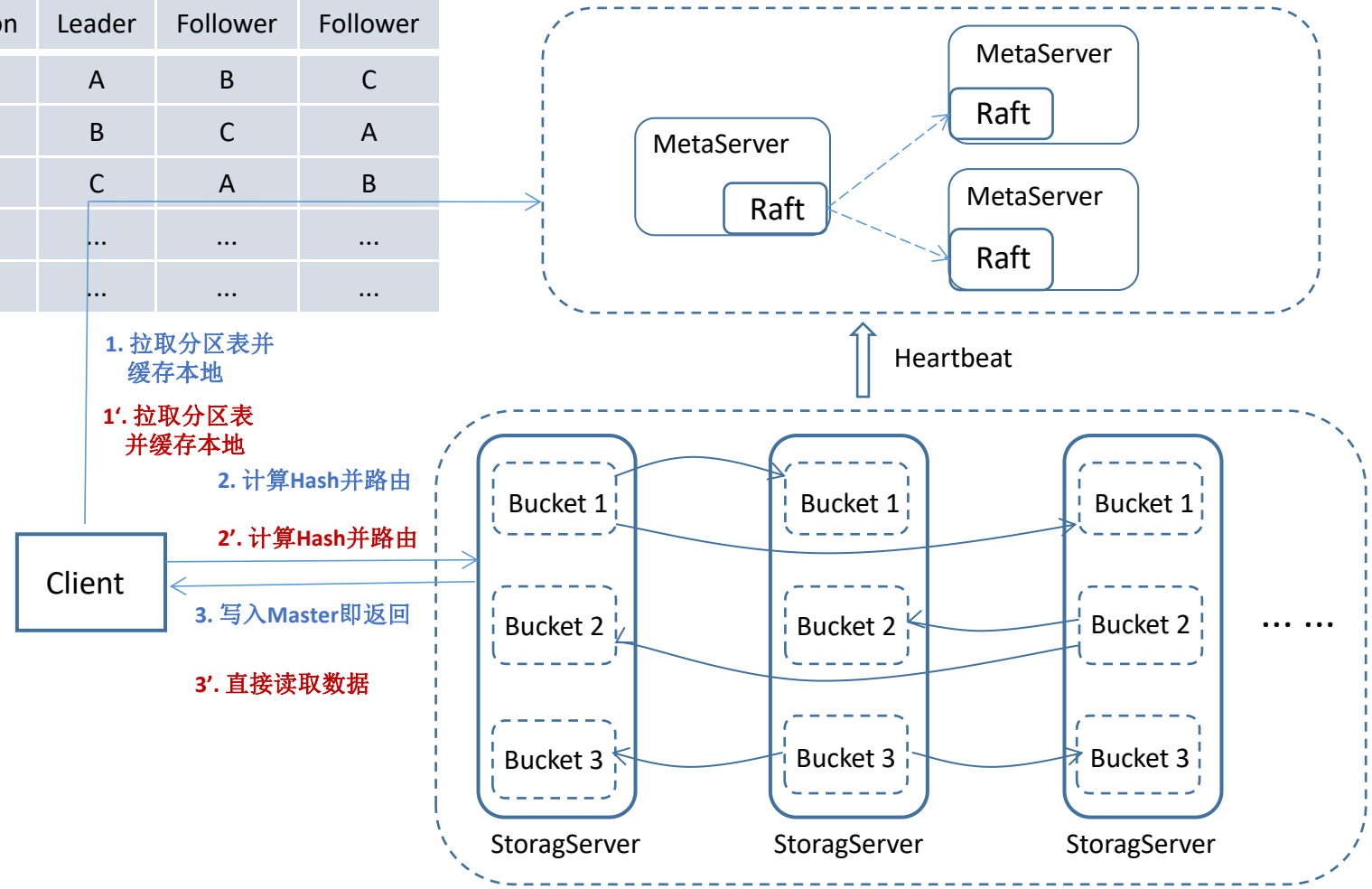
   Configserver的HA能力不足，原有实现有脑裂风险。

3. 数据强一致性的缺乏

   最终一致的系统应用场景较为有限。

存储架构演进

Partition	Leader	Follower	Follower
0	A	B	C
1	B	C	A
2	C	A	B
…	…	…	…
1023	…	…	…



演进的分布式KV存储

1. 主流的代码实现。

   c++ 11代码风格实现

2. 数据最终一致、强一致性能力支持

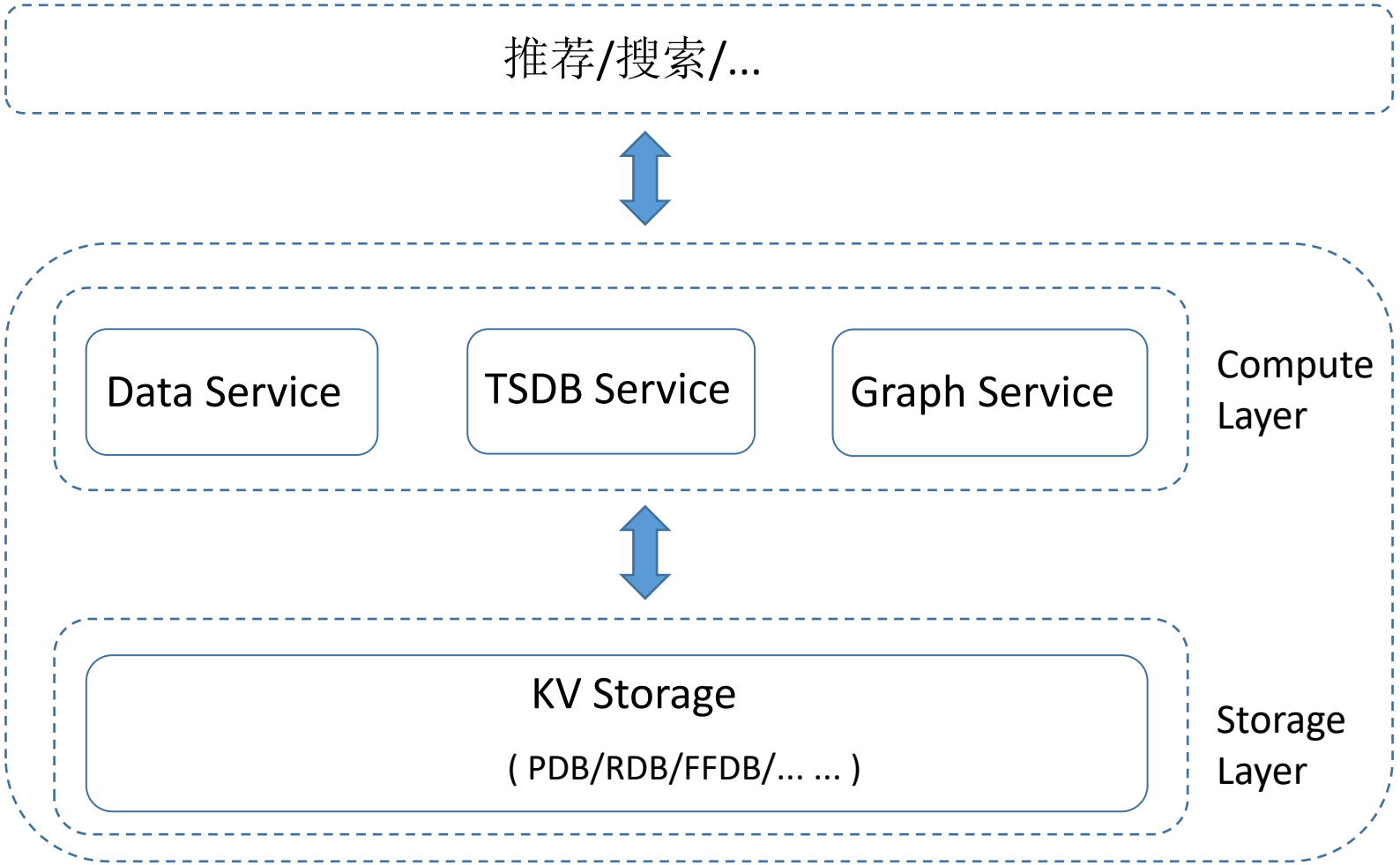
   应用于更广泛业务场景

3. 服务高可用和易用

   架构设计优秀

   完善的生态和工具。

未来展望



1. Data Service
   负责特征计算服务

2. TSDB Service
   时序数据库计算服务

3. Graph Service
   图数据库计算服务

4. KV Storage
   支持多引擎的存储服务

谢 谢!

THANKS