DTCC 2010-2022

第十三届中国数据库技术大会

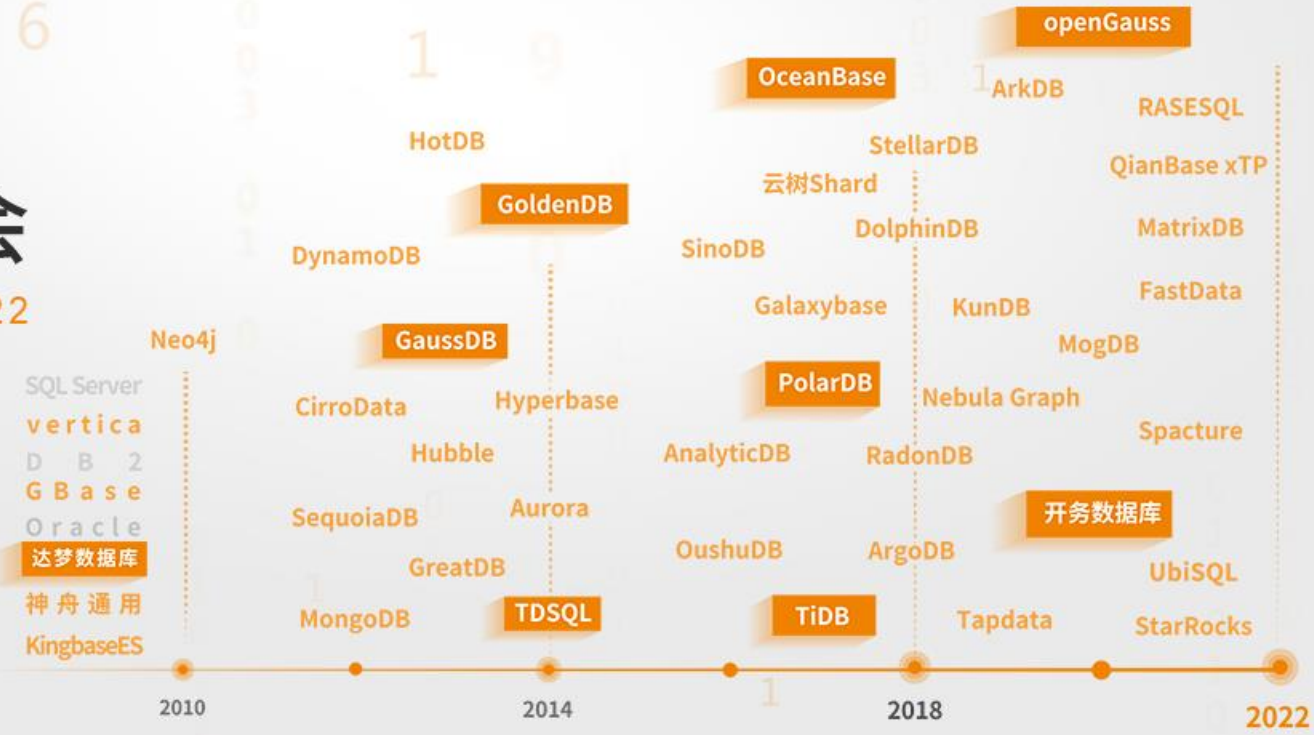
DATABASE TECHNOLOGY CONFERENCE CHINA 2022

数据智能 价值创新

IT168.com ChinaUnix.net ITPUB

线上直播 | 2022/12/14-16

数据来源：数据库产品上市商用时间



vivo KV存储探索与实践

汪翔·vivo互联网·专家工程师

关于个人

过往履历

网易，APM系统研发以及云游戏的探索
腾讯，数据库智能化运维平台研发以及数据库中间件研发

汪翔

vivo数据库技术专家

负责项目

- 负责 Redis 方向的研发，主导 Redis 双活项目从0到1的研发，及 Redis 内核的改进
- 负责 KV 方向的研发，主导 KV 存储项目从0到1的研发
- 负责 MySQL 方向的研发，包括MySQL Proxy 、MySQL HA以及SQL审核服务
- 负责 DTS 项目的研发，构建数据订阅、数据同步、数据迁移的一体化平台

兴趣方向

- 分布式数据库/分布式系统/存储引擎/编程语言/系统架构等技术方向

目录

背景和目标

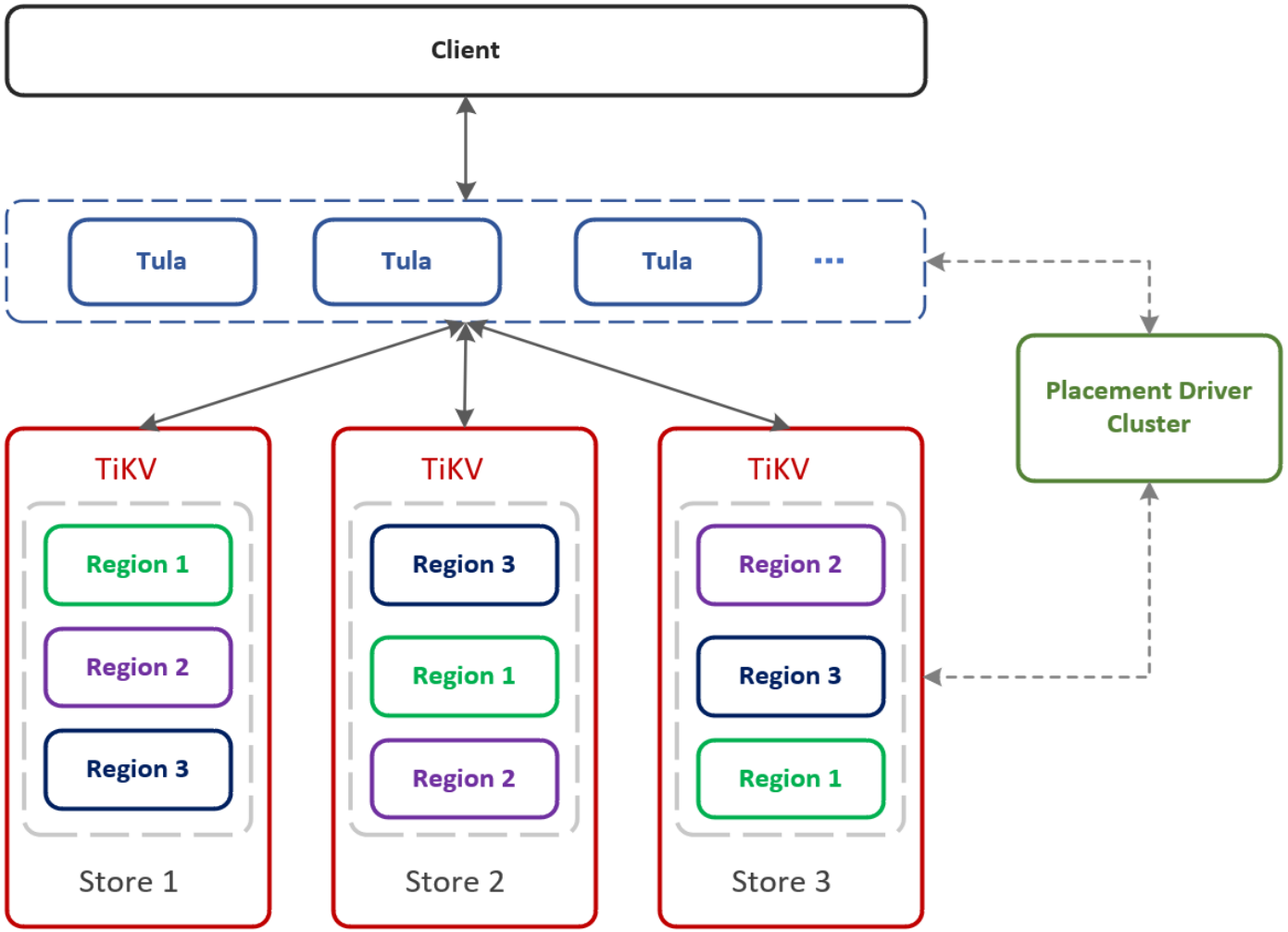
背景

当前公司内部没有统一的磁盘 KV 存储服务，很多业务都将 Redis 当作 KV 存储服务在使用，但是部分业务可能不需要 Redis 如此高的性能，却承担着巨大的成本（内存价格相对磁盘来说更加昂贵）。基于降低存储成本的需求，同时为了尽可能减少业务迁移的成本，我们基于 TiKV 研发了一套磁盘KV 存储服务。

目标

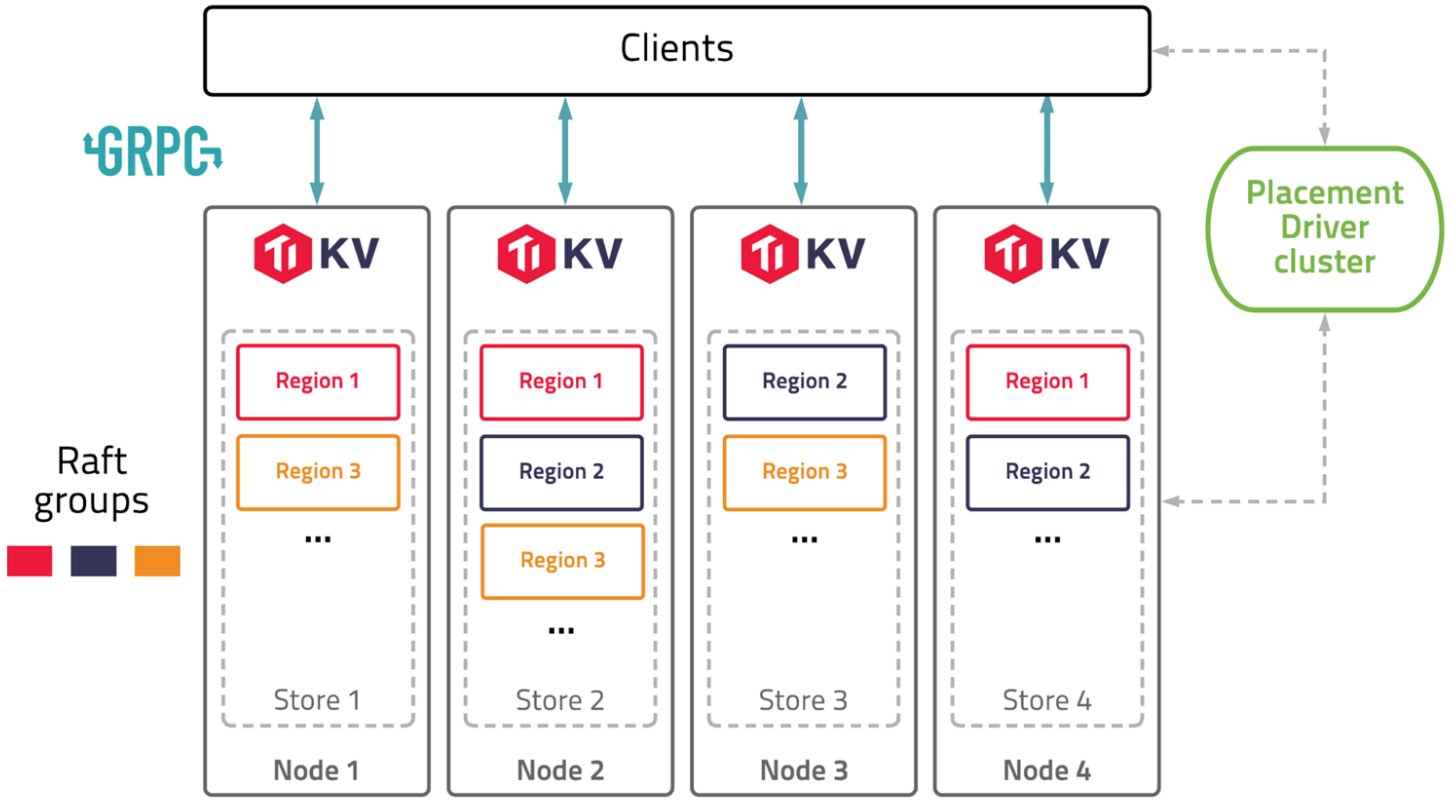
- 兼容 Redis 协议，便于业务进行数据库迁移
- 支持大容量存储，承载业务大规模数据
- 高性能，满足业务对性能的需求
- 高可用，能够容忍部分节点失效
- 易运维，降低整体运维成本

系统架构 - 整体架构



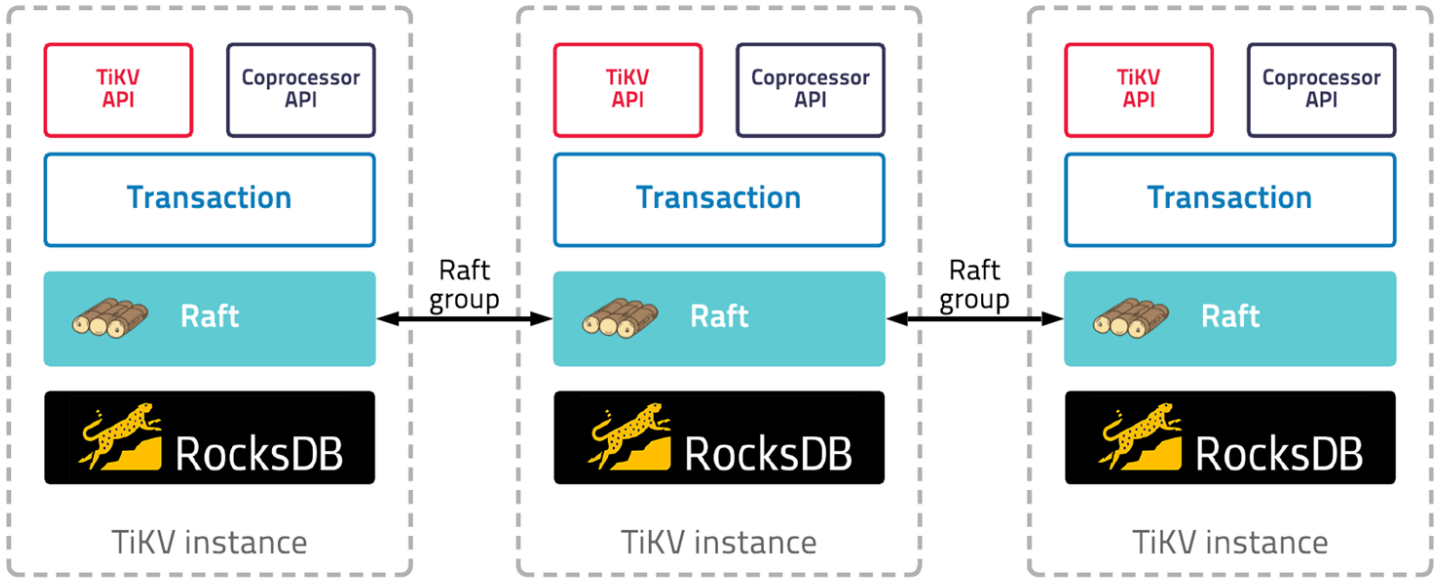
- 兼容 Redis 协议
- 存储计算分离
- 支持横向扩展
- 高可用性

系统架构 - TiKV 架构简介



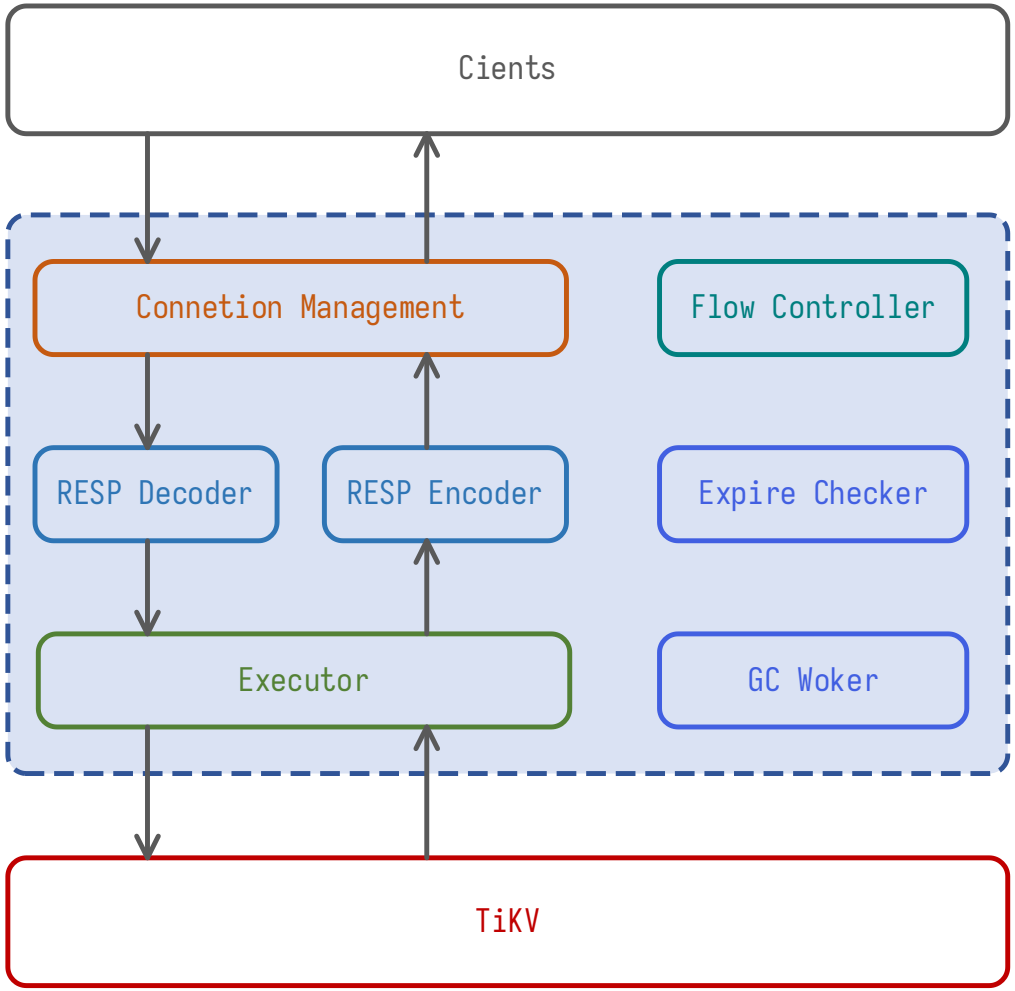
- **Placement Driver**: PD是集群的管理者，它会周期性检查TiKV状态，根据需要进行负载和数据的自动均衡
- **Store**: Store 表示一个存储点，每个Store中有一个RocksDB实例，负责将数据持久化到本地磁盘
- **Region**: Region是 Key-Value数据移动的基本单元，每个 region 的数据会使用**raft协议**复制到多个节点，共同组成一个 Raft Group
- **Node**: Node表示集群中的一个物理节点。每个Node上可以有一到多个Stores，每个Store上有多个Regions

系统架构 - TiKV 架构简介



- **Transaction Model**: TiKV使用类似于google percolator 的事务模型，支持snapshot isolation级别的事务隔离
- **Raft**: TiKV使用Raft协议来进行数据复制，支持线性一致性
- **RocksDB**: TiKV使用rocksdb作为底层存储引擎

系统架构 - Tula架构



- Connection Management模块管理客户端连接
- RESP Encoder 和 Decoder 负责对协议数据进行编解码
- Executor 将 Redis Command 转换为事务型的KV请求
- Expire Checker 负责检查Key是否过期
- GC Worker 负责异步删除过期数据

设计细节 – 数据编码

String

Meta Key:

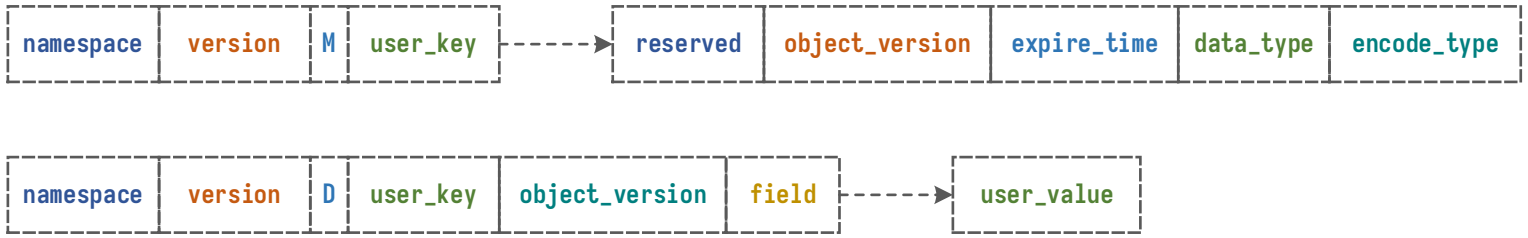
1. namespace，命名空间
2. version: 编码版本，为了前向兼容
3. 元数据Key标记
4. 用户key

Value:

1. reserved，保留字段
2. object_version，对象版本，用于快速删除
3. expire_time，过期时间
4. data_type，数据类型
5. encode_type，编码类型
6. user_value，用户value

设计细节 – 数据编码

Hash

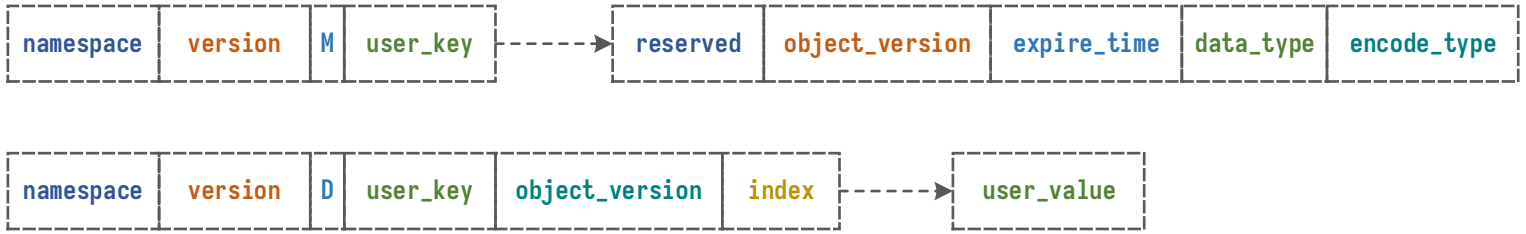


Data Key:

1. namespace，命名空间
2. version，编码版本
3. D，Data Key标记
4. user_key，用户Key
5. object_version，数据版本，用于异步快速删除
6. field，Hash中的field

设计细节 – 数据编码

List

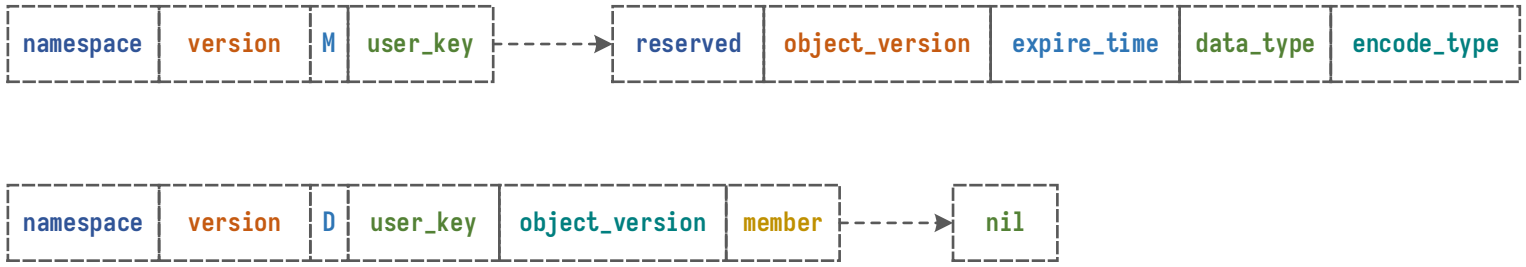


Data Key:

1. namespace，命名空间
2. version，编码版本
3. D，Data Key标记
4. user_key，用户Key
5. object_version，数据版本，用于异步快速删除
6. index，list中的元素的索引

设计细节 – 数据编码

Set

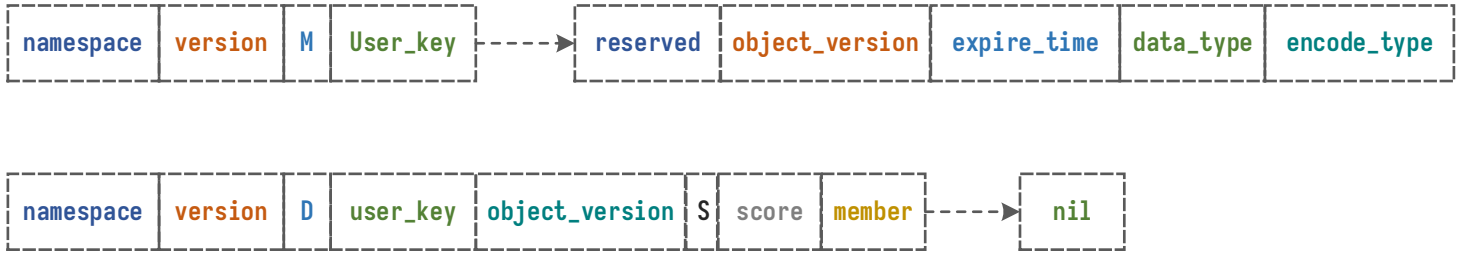


Data Key:

1. namespace，命名空间
2. version，编码版本
3. D，Data Key标记
4. user_key，用户Key
5. object_version，数据版本，用于异步快速删除
6. member，集合中的元素

设计细节 – 数据编码

ZSet



Data Key:

1. namespace，命名空间
2. version，编码版本
3. D，Data Key标记
4. user_key，用户Key
5. object_version，数据版本，用于异步快速删除
6. S，标识为Score Key
7. score，元素的score
8. member，元素的member

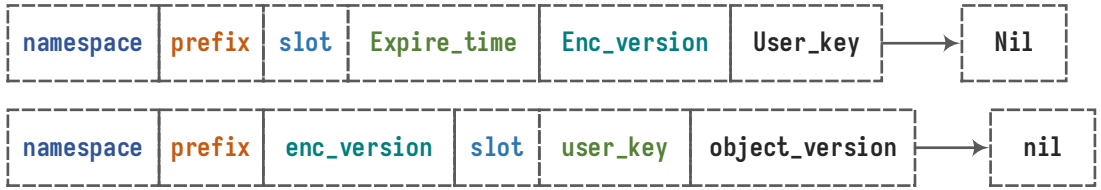
设计细节 – 过期数据回收

过期检测

- 被动检测，数据访问时判断Key是否过期
- 主动检测，定期检查带TTL的Key是否过期

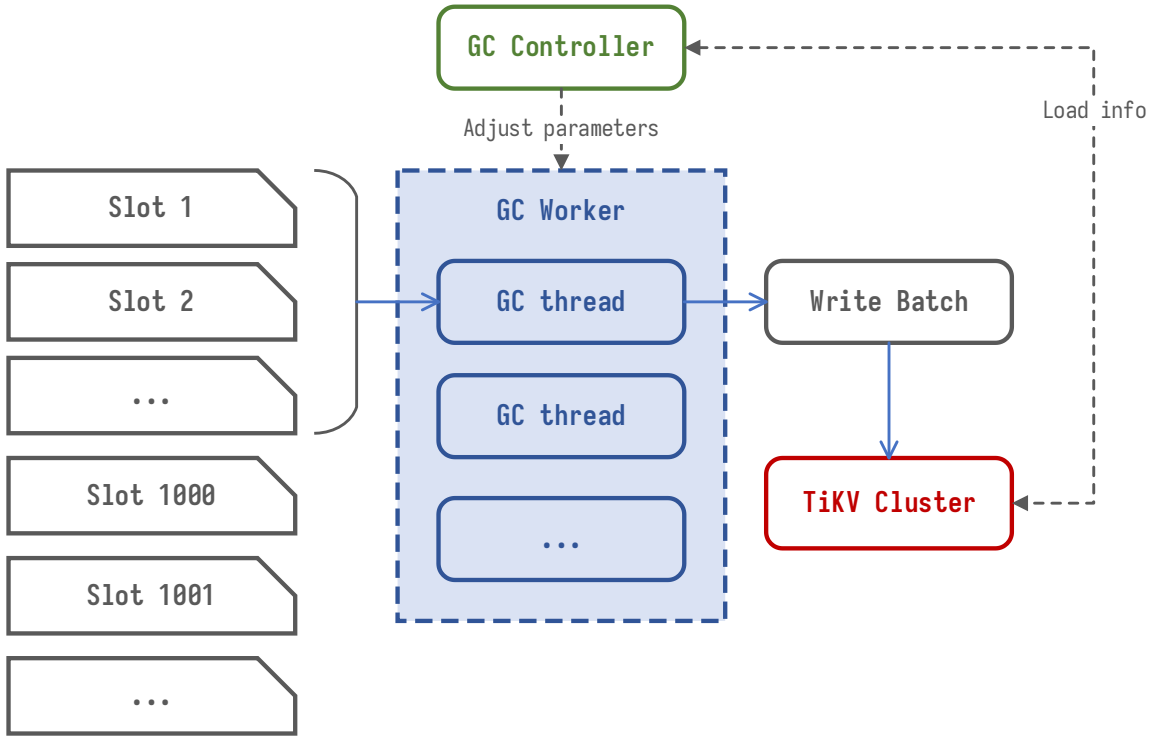
主动检测

- 带TTL的 Key 冗余一个 Expire Key，提高检测效率
- 对 Expire Key 空间进行划分，支持并发扫描
- 发现 Key 已过期，删除 Meta Key，并生成 GC Key



- GC Key 空间按照slot进行划分，支持并发删除
- <= object_version 的 Data Key 都需要被删除

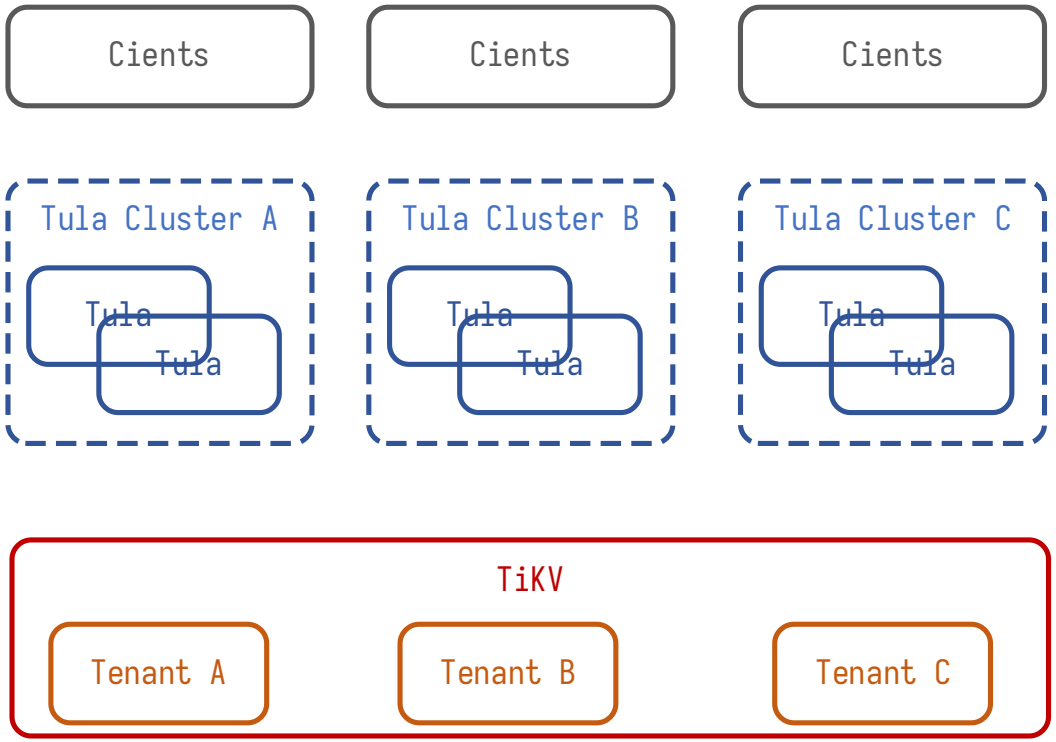
Adaptive GC



- 根据负载情况自动调整 GC 速度，提升整体吞吐
- 可调整参数：GC线程数、Write Batch大小、Sleep时间

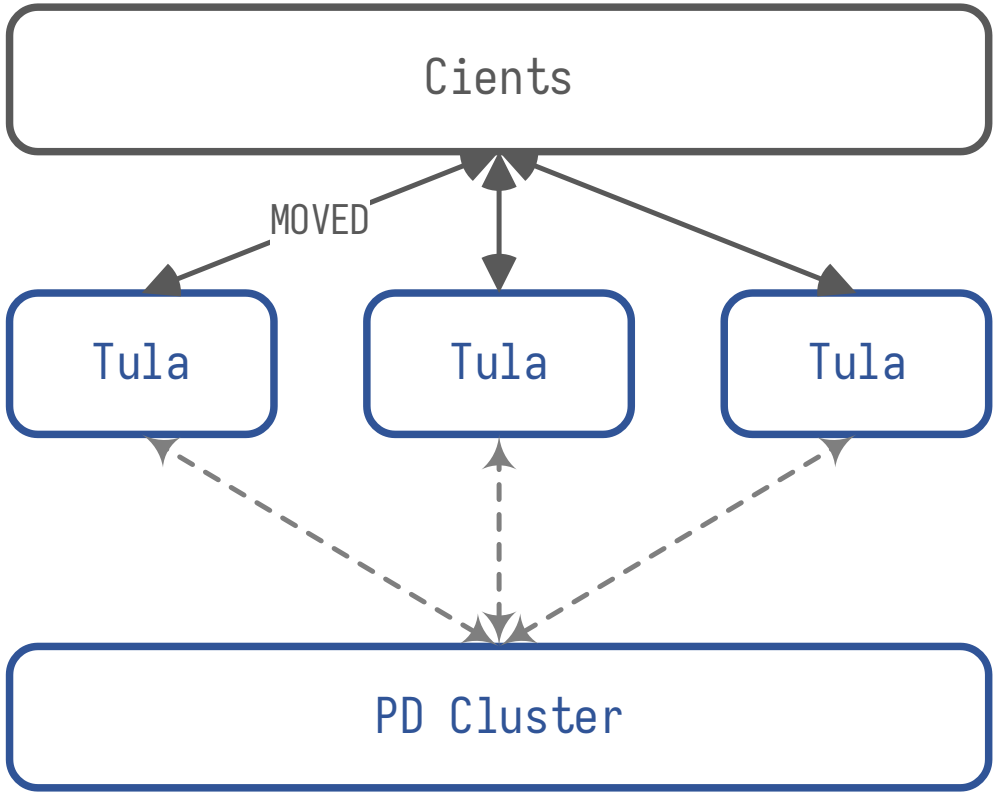
设计细节 - 多租户设计

- 解决小数据量集群资源占用问题
- 基于 namespace 进行数据隔离
- 底层共享一个TiKV存储集群



设计细节 – 高可用和水平扩展

- 基于 ETCD in PD 服务注册和服务发现
- 兼容Redis Cluster协议
- 每个 Tula 平分 slot 分配
- Tula 节点 Down 掉之后自动触发 slot 分配
- 扩缩容之后自动触发 slot 分配



性能指标 – 测试环境

TiKV（3节点）：

- CPU: CPU @ 2.30GHz 32 core
- 内存: 376G
- 硬盘: NVME SSD 4T

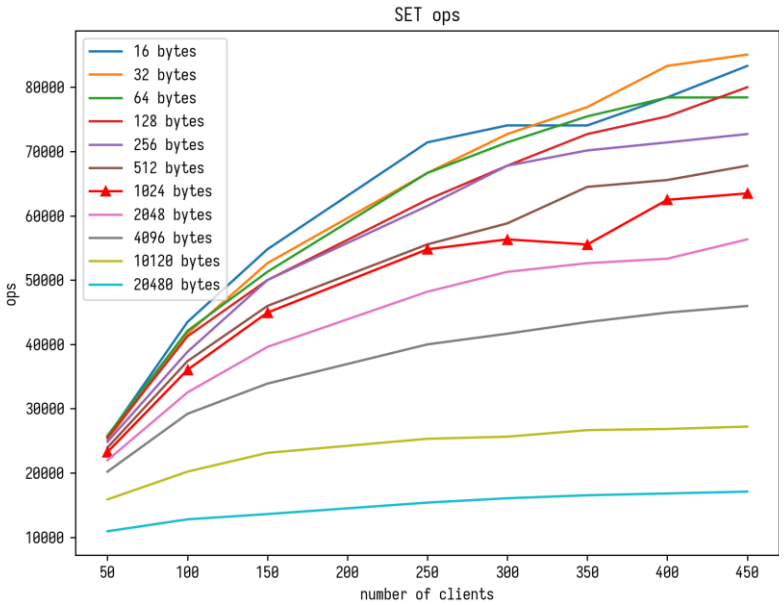
PD（3节点）：

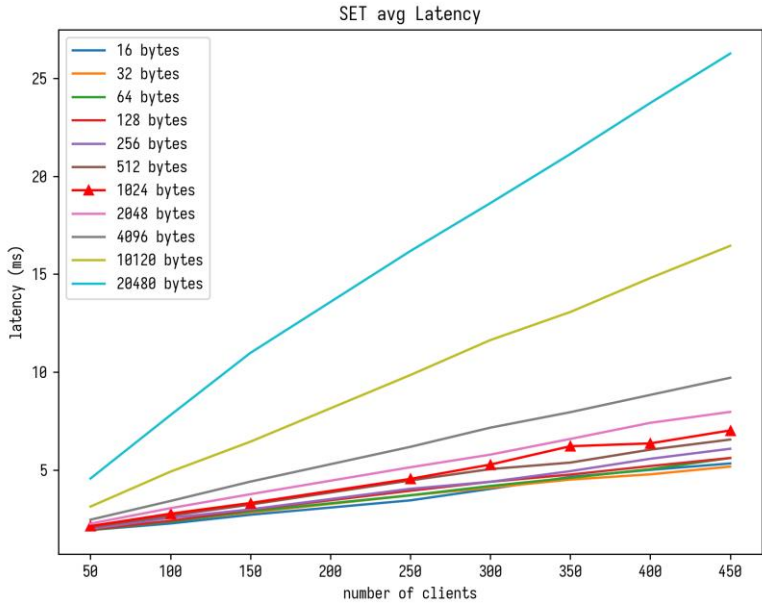
- CPU: CPU @ 2.30GHz 32 core
- 内存: 376G
- 硬盘: NVME SSD 4T

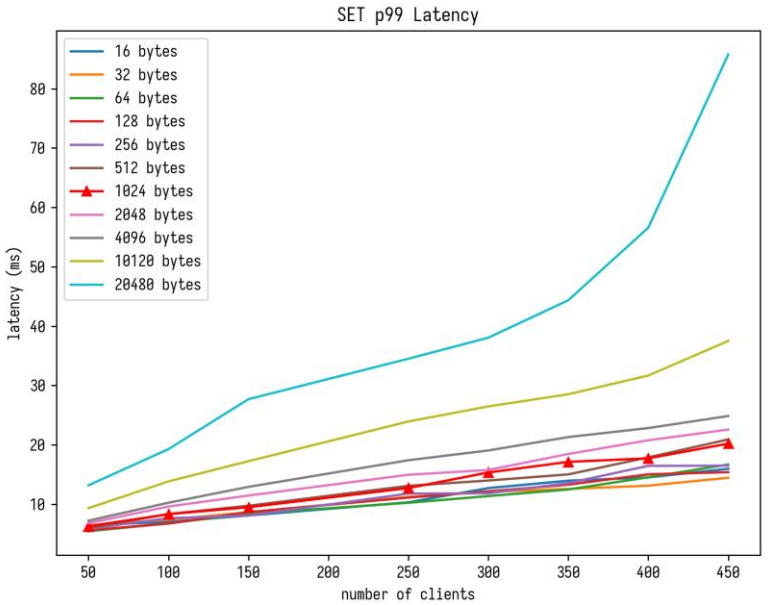
Tula（3节点）：

- CPU: CPU @ 2.10GHz 32 core
- 内存: 187G

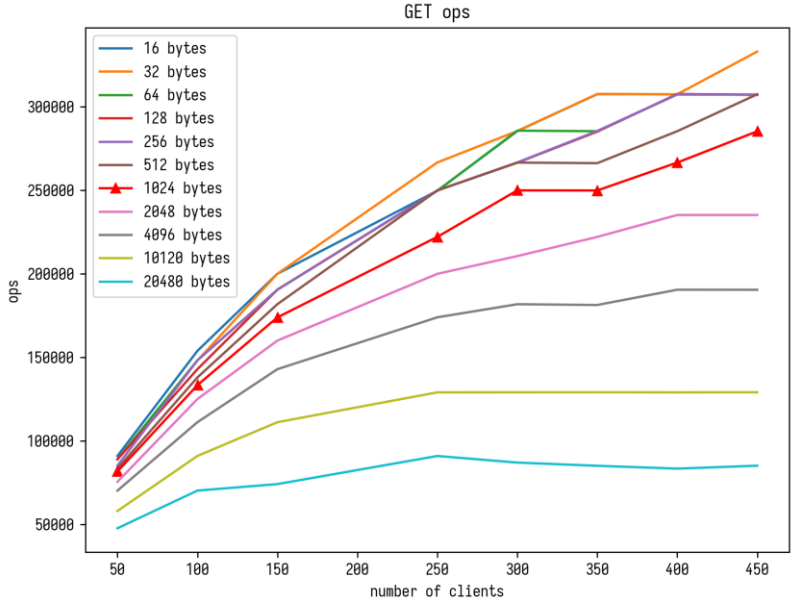
性能指标 - SET

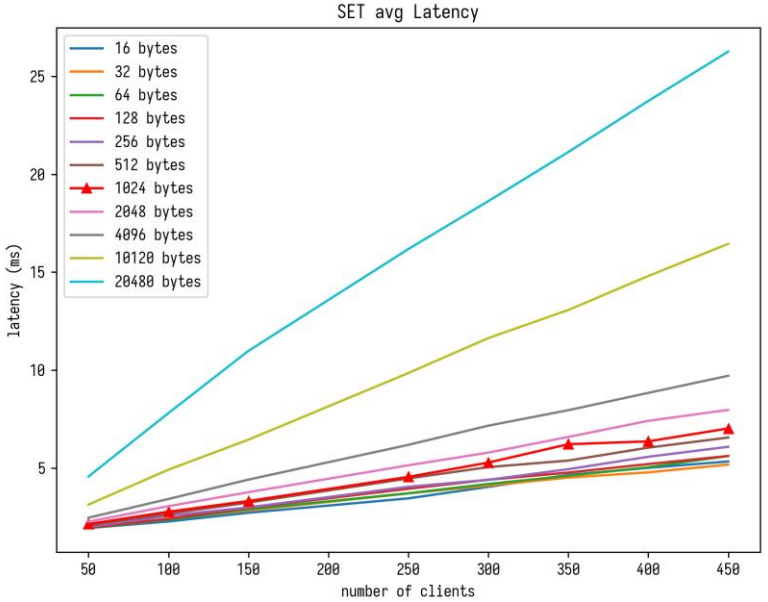

SET ops
16 bytes
32 bytes
64 bytes
128 bytes
256 bytes
512 bytes
1024 bytes
2048 bytes
4096 bytes
10120 bytes
20480 bytes
ops
number of clients

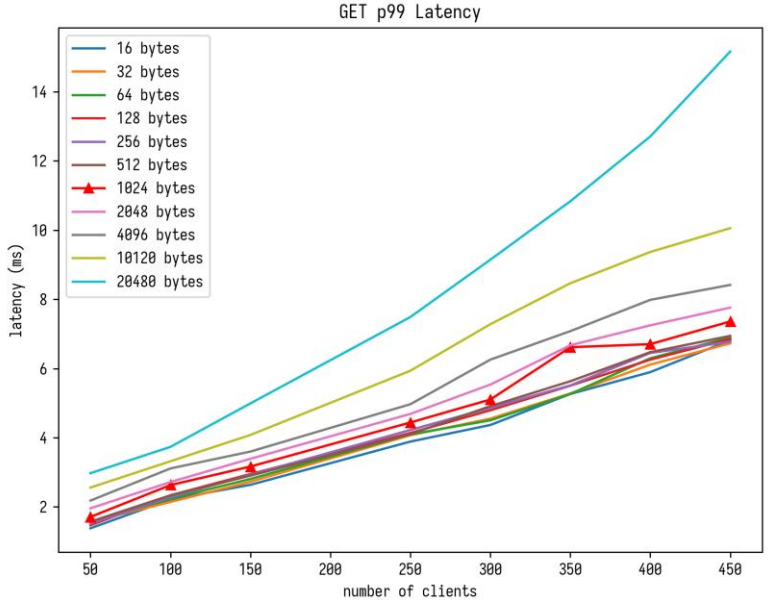

SET avg Latency
16 bytes
32 bytes
64 bytes
128 bytes
256 bytes
512 bytes
1024 bytes
2048 bytes
4096 bytes
10120 bytes
20480 bytes
latency (ms)
number of clients


SET p99 Latency
16 bytes
32 bytes
64 bytes
128 bytes
256 bytes
512 bytes
1024 bytes
2048 bytes
4096 bytes
10120 bytes
20480 bytes
latency (ms)
number of clients

性能指标 – GET

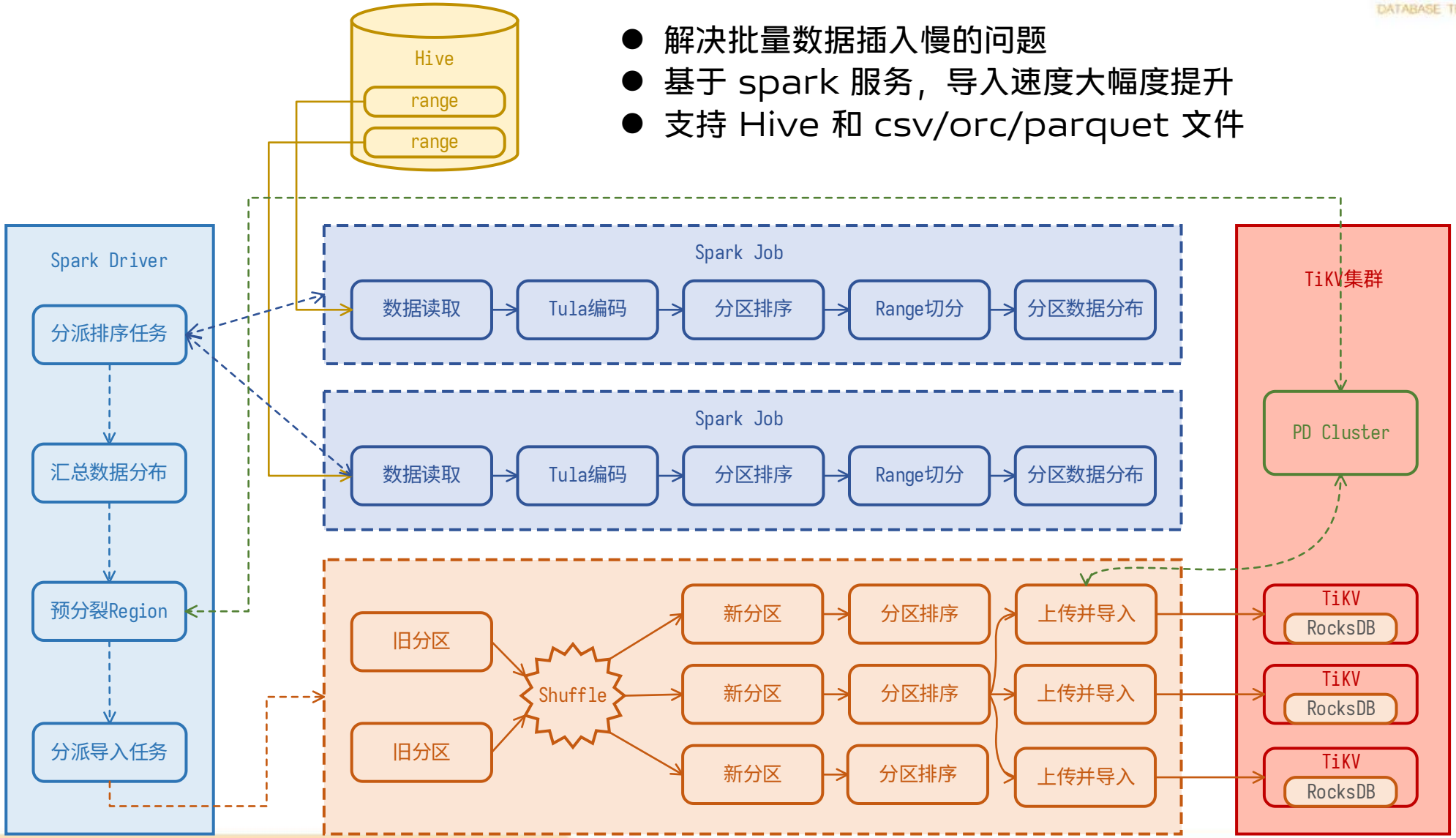

GET ops
16 bytes
32 bytes
64 bytes
128 bytes
256 bytes
512 bytes
1024 bytes
2048 bytes
4096 bytes
10120 bytes
20480 bytes
ops
300000
250000
200000
150000
100000
50000
50
100
150
200
250
300
350
400
450
number of clients


SET avg Latency
16 bytes
32 bytes
64 bytes
128 bytes
256 bytes
512 bytes
1024 bytes
2048 bytes
4096 bytes
10120 bytes
20480 bytes
latency (ms)
25
20
15
10
5
50
100
150
200
250
300
350
400
450
number of clients


GET p99 Latency
16 bytes
32 bytes
64 bytes
128 bytes
256 bytes
512 bytes
1024 bytes
2048 bytes
4096 bytes
10120 bytes
20480 bytes
latency (ms)
14
12
10
8
6
4
2
50
100
150
200
250
300
350
400
450
number of clients

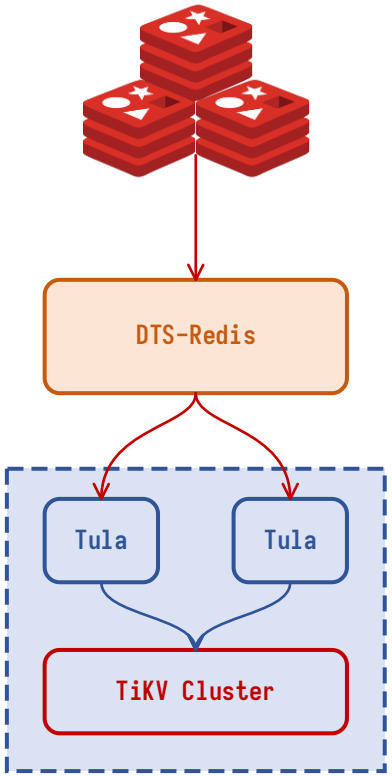
周边生态 – 离线数据批量导入

- 解决批量数据插入慢的问题
- 基于 spark 服务，导入速度大幅度提升
- 支持 Hive 和 csv/orc/parquet 文件

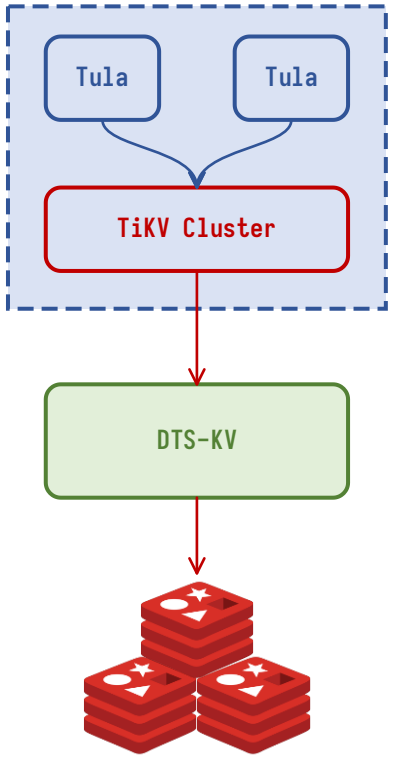


周边生态 – 数据迁移

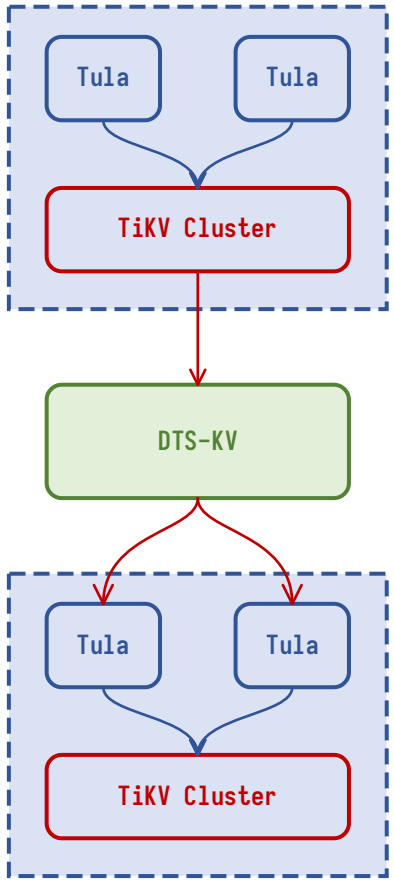
从Redis迁移到磁盘KV



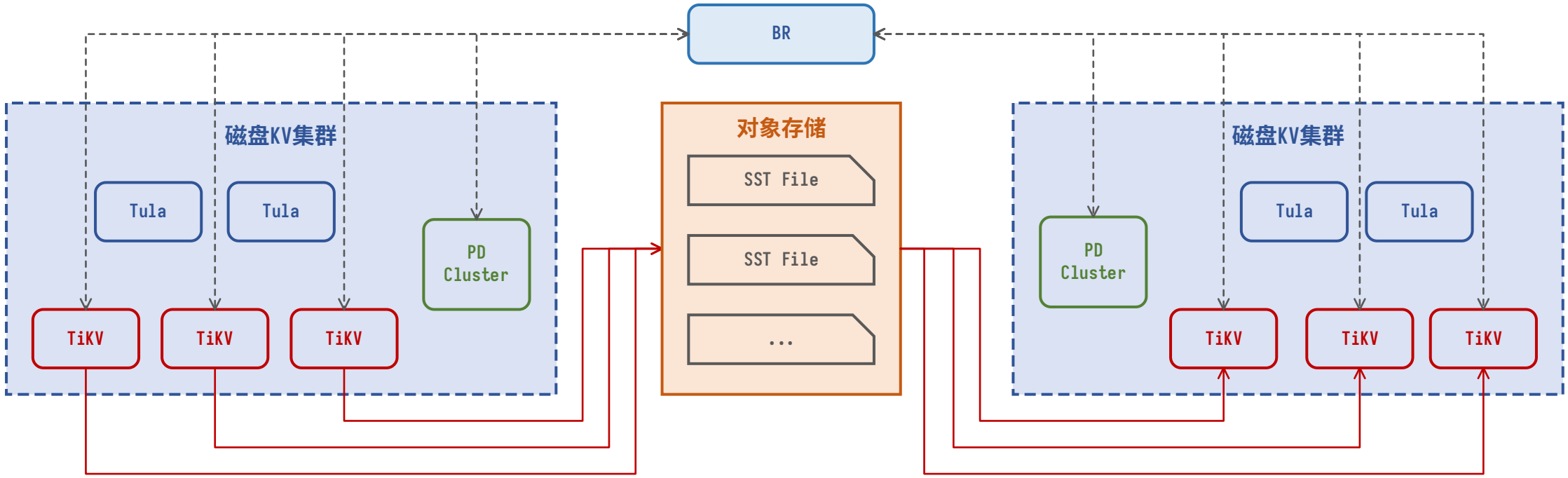
从磁盘KV迁回到Redis



从磁盘KV迁移到其他磁盘KV



周边生态 – 备份和恢复



- 基于TiKV BR 工具
- 数据编码部分修改为Tula编码

未来展望

- 自适应slot锁机制，解决部分场景事务冲突较多的问题
- 性能优化，下沉数据结构相关的指令到TiKV，提升整体性能
- 构建缓存和存储一体化系统
- 支持更多的协议，Table API等


openGauss
OceanBase
ArkDB
RASESQL
HotDB
StellarDB
QianBase xTP
云树Shard
GoldenDB
DolphinDB
MatrixDB
DynamoDB
SinoDB
FastData
Galaxybase
KunDB
SQL Server
vertica
D B 2
GBase
Oracle
达梦数据库
Spacture
开务数据库
UbiSQL
神舟通用
TDSQL
TiDB
Tapdata
StarRocks
KingbaseES
2010
2014
2018

THANKS