数字转型 架构演进

SACC

2019 中国系统架构师大会

SYSTEM ARCHITECT CONFERENCE CHINA 2019

IT168.com ChinaUnix.net ITPUB

2019年10月31-11月2日 | 北京海淀永泰福朋喜来登酒店

爱奇艺Key-Value数据库HiKV应用实践

郭磊涛
guoleitao@qiyi.com

HiKV 是什么？

- 爱奇艺基于Scylla开发的Key-Value数据库
  - Scylla : C++重写的Cassandra，高吞吐，低延时
  - HiKV : High Performance Key-Value Database
- 特点
  - 大容量：单集群百TB数据量
  - 高负载：单节点10w+ QPS
  - 低延时：读写延迟 P99<10ms
  - 部署：多数据中心、在线扩容
  - 最终一致性
  - 查询缓存、物理备份、认证和授权

开源系统那么多，为什么还要开发新的KVDB？

提纲

- **为什么要开发 HiKV ?**
- HiKV 设计思路与关键技术
- HiKV 在爱奇艺的应用

Redis in iQIYI

2013前

2.6.13

- 服务侧Lua脚本
- 支持只读Slave
- Master+Slave部署

2014年

2.8.13/17

- 动态设置最大连接数、PUBSUB命令、keyspaces变化通知
- PSYNC代替SYNC

2015年

3.0.5

- 可以在从库读到过期key的问题
- Sentinel+DNS 高可用

2016~2019年

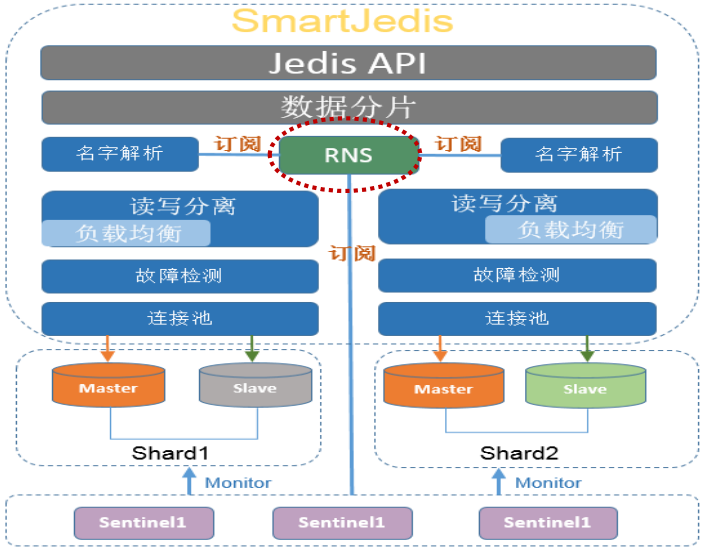
3.2.7/9

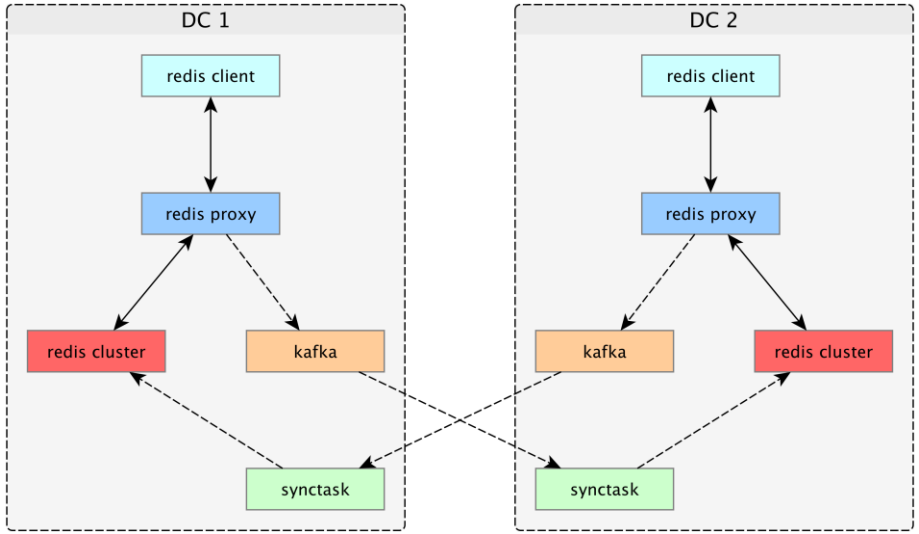
- 解决从库读到过期key的问题
- 客户端分片SmartJedis
- 上线服务端Cluster集群
- Redis Cluster Proxy

2020年

5.0

- PSYNC2 部分同步
- 异步删除
- Modules扩展
- 主动碎片整理





适用场景：

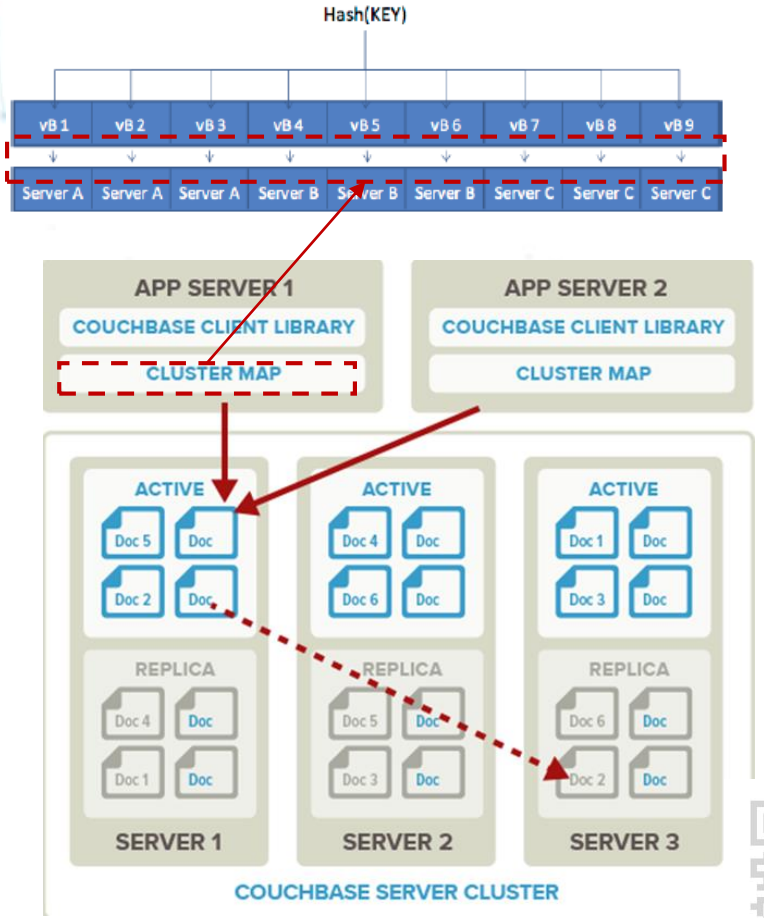
- 各个集群之间处理的数据没有重叠
- 同一集群上不对同一个key做并发修改

Redis扩展性差，无法存储百TB级数据量

Couchbase in iQIYI

分布式高性能NoSQL数据库

- 2种bucket（等价于Database）
  - Memcached：KV, 不持久化，无副本
  - Couchbase：JSON，持久化，有副本，Rebalance
  - 1 Bucket 包含 1024 vbucket
- 支持N1QL（类SQL）
- 容量：无上限，扩容方便，目前线上TB级
- 性能：与key/value size相关
- XDCR：支持跨数据中心集群间同步



Couchbase in iQIYI

2012年 → 2013年 → 2014年 → 2015~2018年 → 2019年 → 2020

CB1.8

- 对**erlang client**进行改进，防止底层C库中的error callback使整个进程退出
- 上线BI、广告、聊天室等项目

CB2.x

- 开发了**访问代理**，防止服务端压力过大导致崩溃
- 上线推荐、播放记录、搜索等

CB3.x

- 解决**NTP**时钟不一致导致集群崩溃等问题
- 解决**XDCR**开启同步的过程中**Rebalance**会导致集群崩溃等问题
- 上线大量业务

CB4.1.0

- 单集群控制在**2个XDCR**，配置过多的XDCR会导致主集群稳定性下降
- 调整XDCR同步顺序时，需要做再强制全量同步，否则可能会有部分数据由于CAS冲突导致同步失败
- 支持节点failover自愈功能

CB5.1.1

- XDCR增加细粒度**延迟**监控，提前预知XDCR同步异常
- 上线**JAVA动态客户端**，实现集群级别故障透明切换
- 研发**CB->HiKV**同步工具，实现快速数据同步迁移
- 100%运维自动化，监控全覆盖（延迟监控，拨测监控），50%告警实现自愈

CB6.0

- 调研阶段
- **Ephemeral** buckets
- **多数据结构**
- 权限管理
- 二级索引

- 未来可期...

Couchbase易扩展，“数据量<可用内存”时性能极高。但是，存储成本过高！

大容量开源NoSQL系统

Couchbase

SCYLLA.

选择Scylla、mongoDB、Couchbase进行性能对比

大容量开源NoSQL - 测试场景

服务器

服务端：3台物理机
CPU: 2 x Intel Xeon Gold 5118 @ 2.30GHz, 24 Cores 48 Threads in total
Memory: **192GB**
SSD: 8 x INTEL SSDSC2KB96 (960GB)

客户端：3台物理机
CPU: 2 x Intel Xeon Gold 6148 @ 2.40GHz, 40 Cores 80 Threads in total
Memory: 512GB

负 载

只读（RO）
加载数据集后，10w Read + 0 Write

随机读写（RW）
加载数据集后，5w Read + 5w Write

数据集

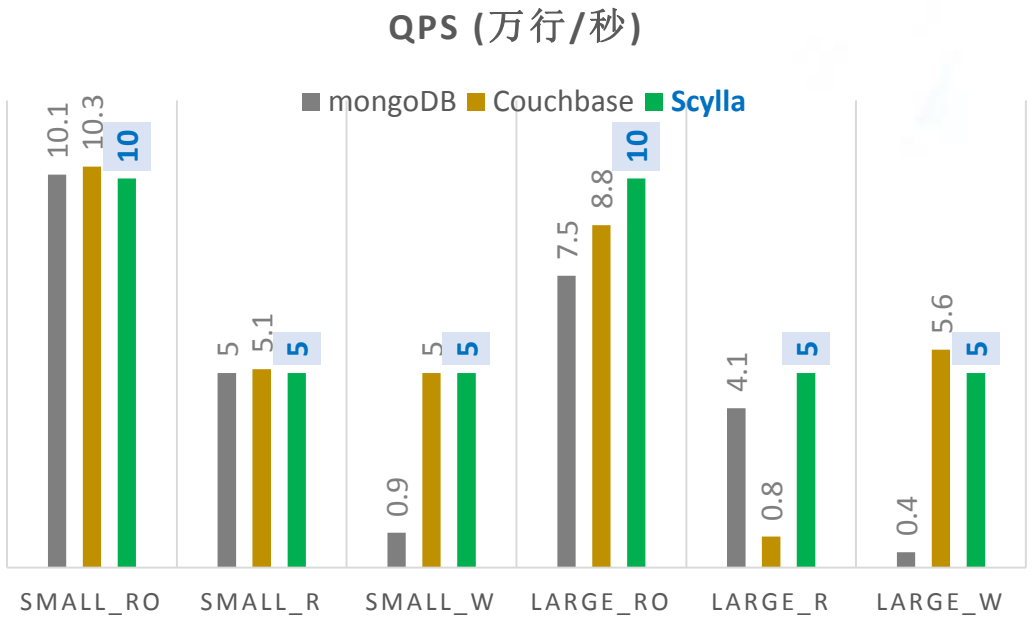
小数据集（Small）
1亿条 X 1KB长度 X 3副本
数据量<内存

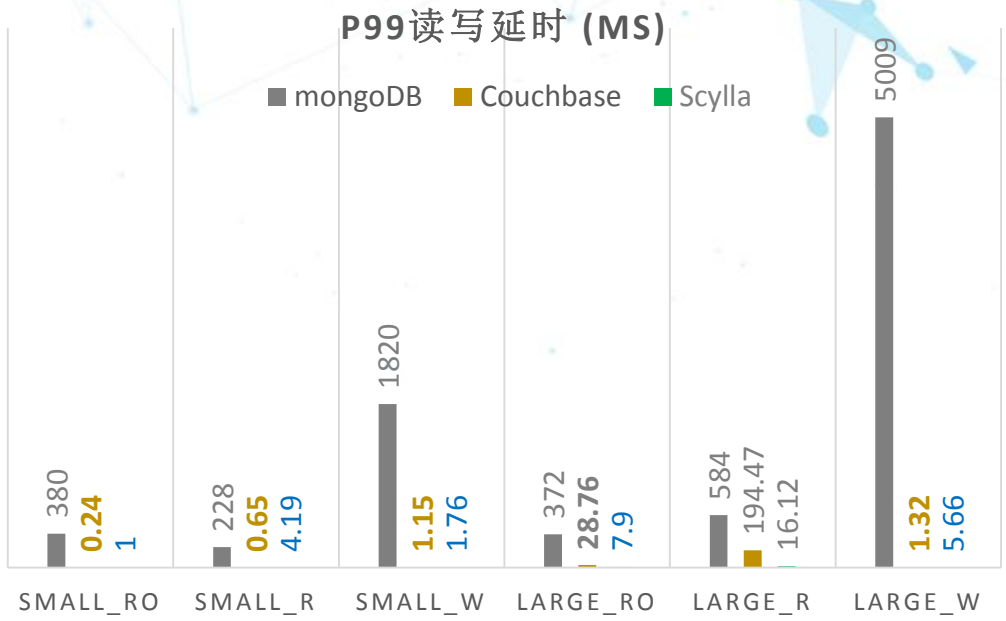
大数据集（Large）
5亿条 X 1KB长度 X 3副本
数据量>内存

版 本

MongoDB 4.2.0
Couchbase 6.0.0
Scylla 2.0.4/3.0.10

大容量开源NoSQL - 测试结果





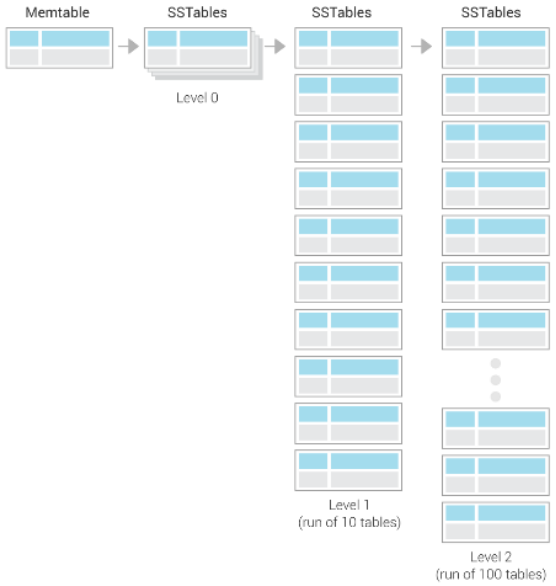
- 仅有 **Scylla** 满足所有场景QPS要求
- **mongoDB** 在cache dirty比例达到20%后，写请求开始排队（qw），**写** QPS急剧下降
- **Couchbase** 在大数据量读写场景下，数据异步持久化造成**读**QPS急剧下降

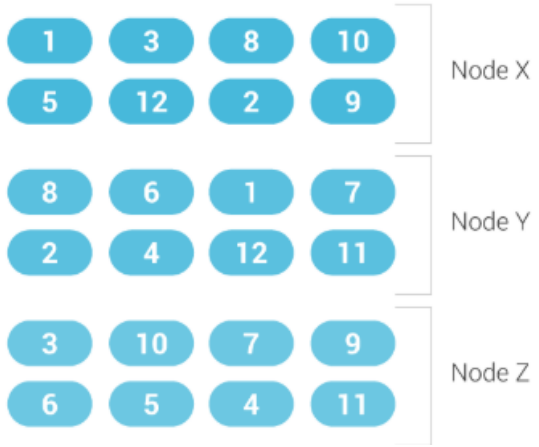
- 均不满足读写延时需求（P99<10ms）
- **Couchbase** 在小数据量，数据均缓存在内存，延时最优。大数据量读写时，读延时较大
- **Scylla**大数据量读写时，读延时最优。压测2小时后，开始出现

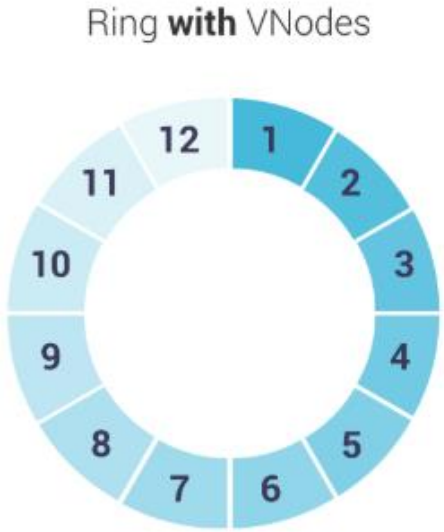
Scylla最接近我们的需求，但高负载下长尾延时明显。是什么原因？

Scylla 工作原理

```
CREATE TABLE loads ( machine, cpu, mtime, load ,
PRIMARY KEY ((machine, cpu), mtime) )
WITH CLUSTERING ORDER BY (mtime DESC);
```





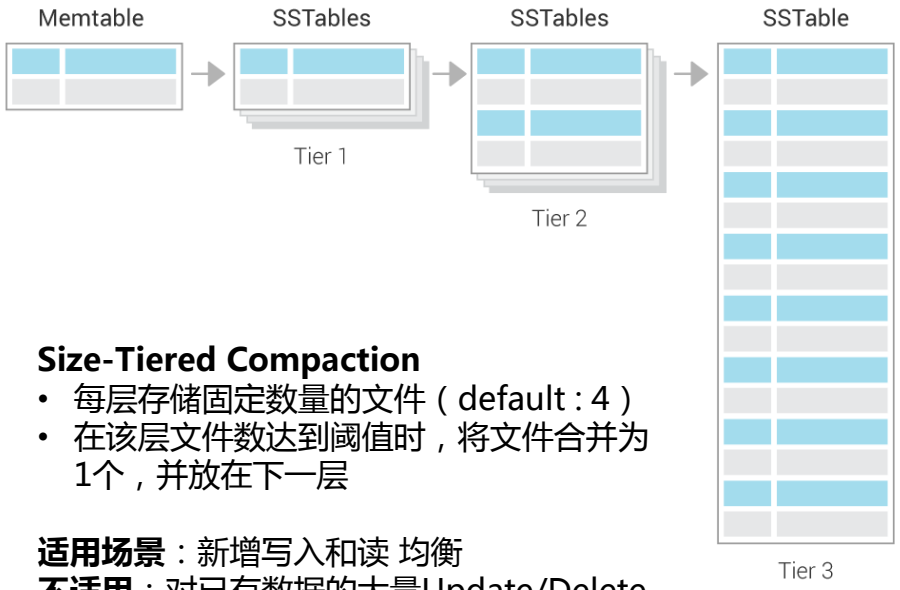


Coordinator → **Murmurhash3** (PartitionKey) → **Token** (64bit) → **Token Range** → VNo → Physical Node → Mem Table → Commit Log

Physical Node → SSTable → Compaction

基于LSM (Log Structured Merge) Tree的存储方式，会造成“写/读/空间的放大”

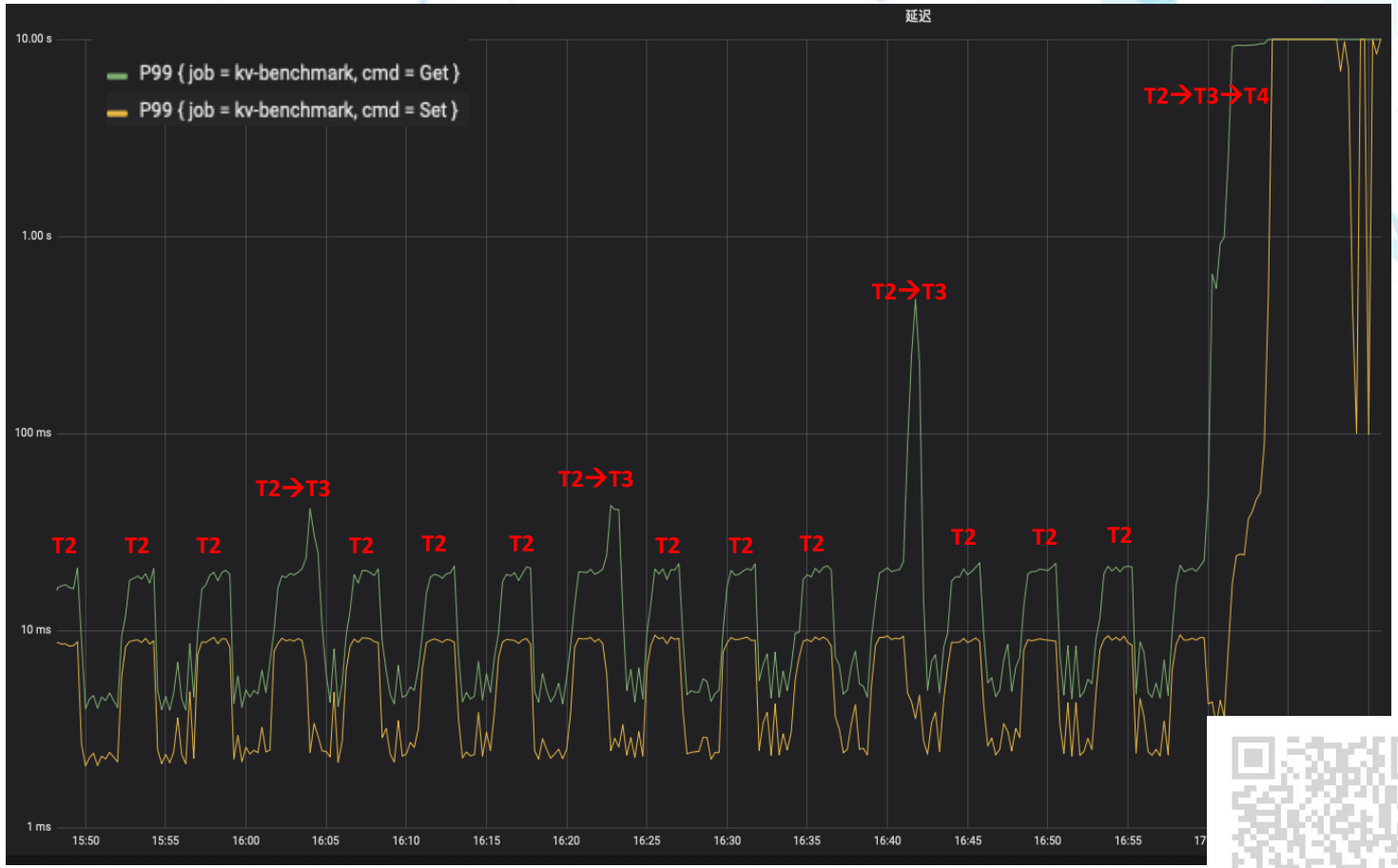
Scylla 延时抖动的原因



Size-Tiered Compaction

- 每层存储固定数量的文件（default : 4）
- 在该层文件数达到阈值时，将文件合并为1个，并放在下一层

适用场景：新增写入和读 均衡
不适用：对已有数据的大量Update/Delete



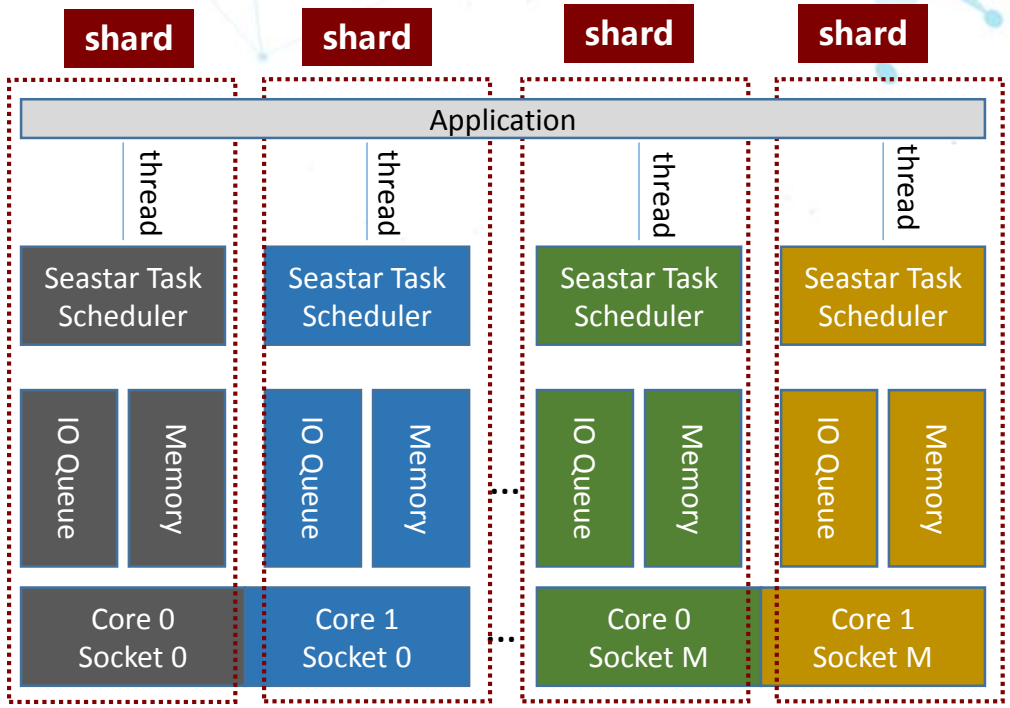
是否采用其他存储引擎，以减小“写/读/空间放大”带来的性能抖动和空间浪费？

提纲

- 为什么要开发 HiKV？
- **HiKV 设计思路与关键技术**
- HiKV 在爱奇艺的应用

HiKV 设计思路

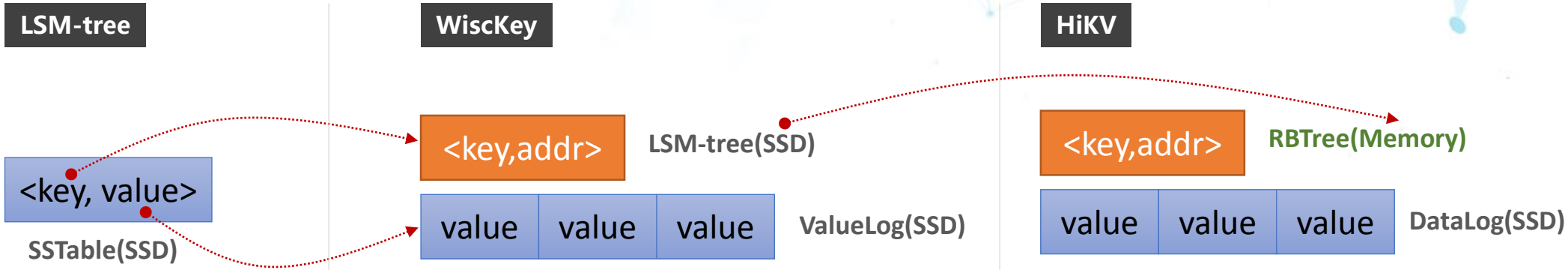
- 继承Scylla的优势，发挥硬件的性能
  - Seastar Shared-nothing架构
  - 用户态 I/O 队列
  - CPU和I/O调度
  - DPDK
  - 查询缓存
  - 无单点+多副本+多数据中心+多活



- 参考WiscKey的设计，减小写/读放大，降低长尾延时

HiKV = Scylla + WiscKey存储引擎

HiKV 数据存储方式



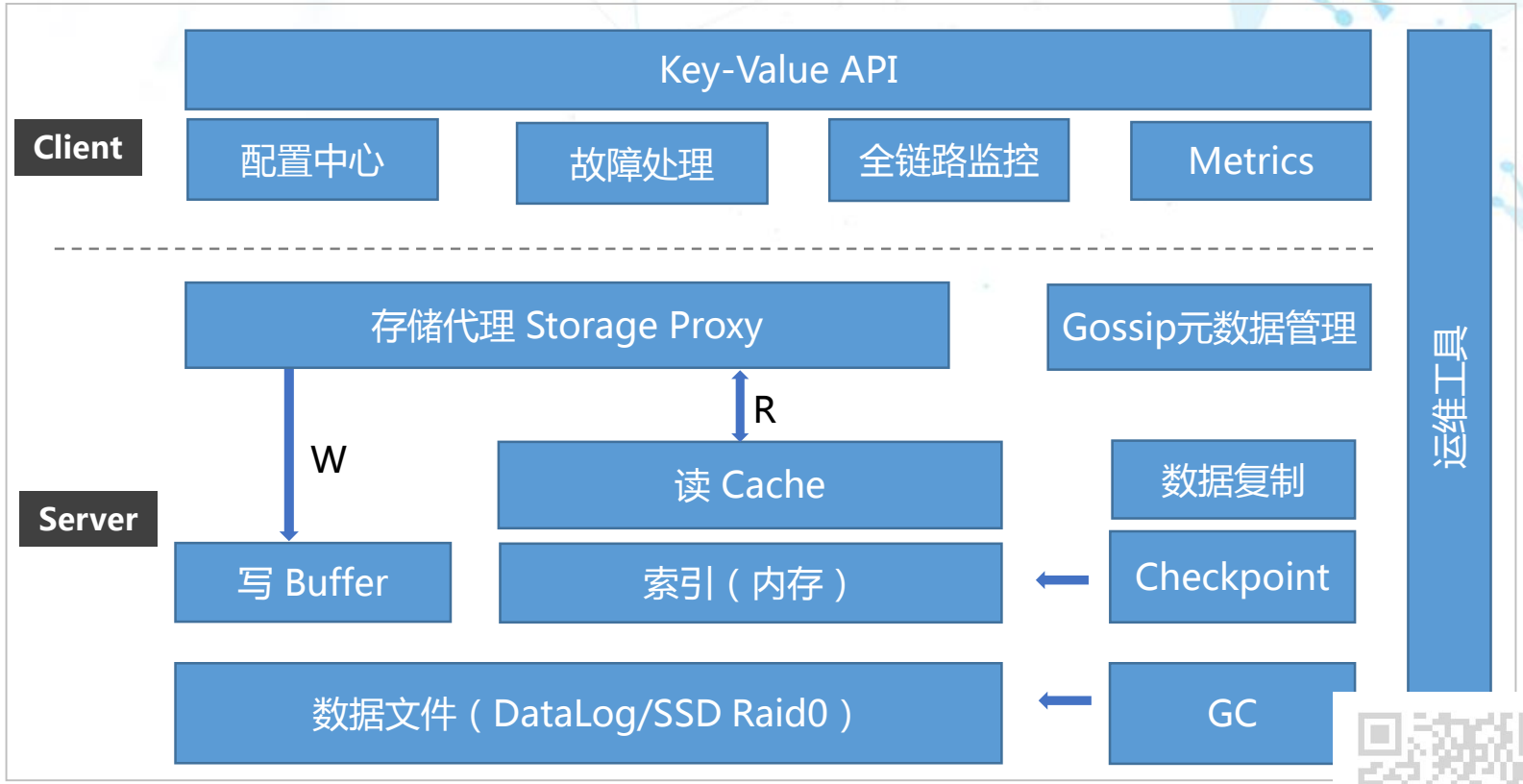
- 优化写：批量顺序写盘
- 优化读：Compaction（排序+GC）
- 缺点
  - Key与Value混存，检索效率低
  - 读/写放大
  - 浪费存储

- 减小“读写放大”：索引（Key）与数据（Value）分开管理
- 优化写：Value批量顺序写盘
- 优化读：Key排序+缓存，支持Range
- 空间优化：ValueLog 轻量级GC
- 缺点
  - 索引（Key）读放大

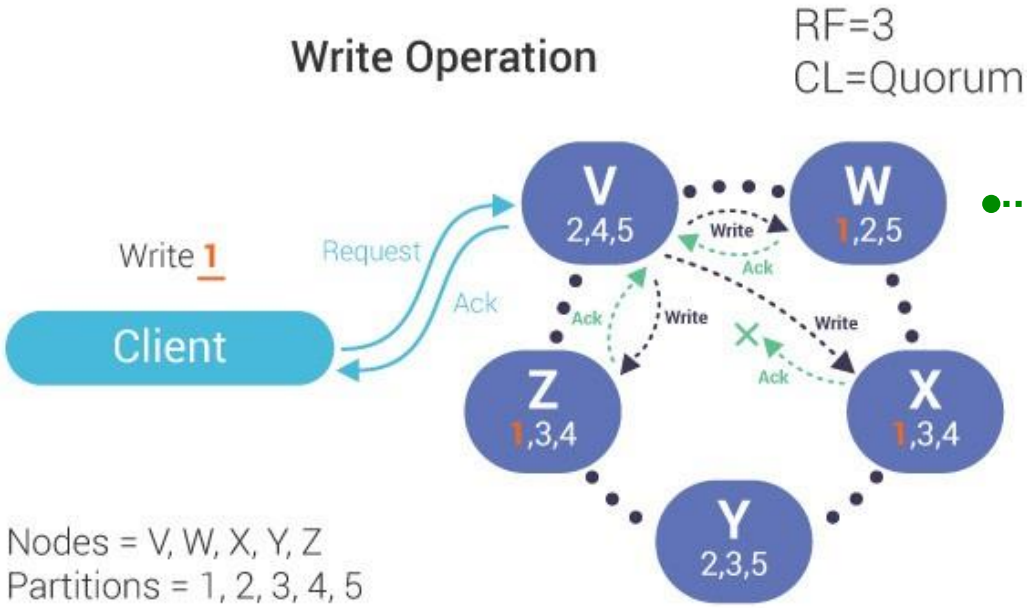
- 消除索引（Key）的读放大
  - LSM-tree → RBTree/Hashtable
  - 读：1次盘IO
- 提高索引检索速度：SSD → Memory
- 索引持久化：索引日志 Checkpoint
- 空间优化：DataLog轻量级GC+手动全量清理
- 缺点
  - 不支持range查询

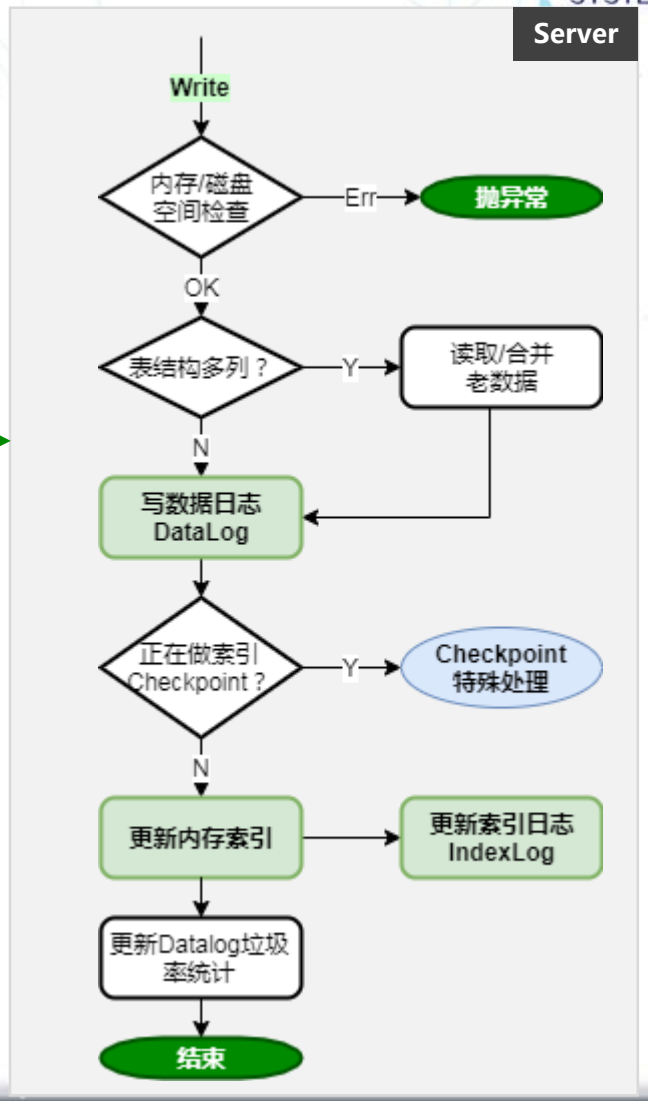
HiKV 整体架构

- 数据模型
  - <Key, Value>
  - Value 为单/多列，支持类型、TTL
- 支持的操作
  - Get/MGet
  - Set/MSet
  - Scan （ token range ）
- 分布式复制
  - Coordinator负责读写
  - 任何一个节点都可以是Coordinator
  - Coordinator通过Gossip共享元数据
- 可配的一致性级别
- 高可用
  - 多副本（ Hinted Handoff/Read Repair ）
  - 多数据中心

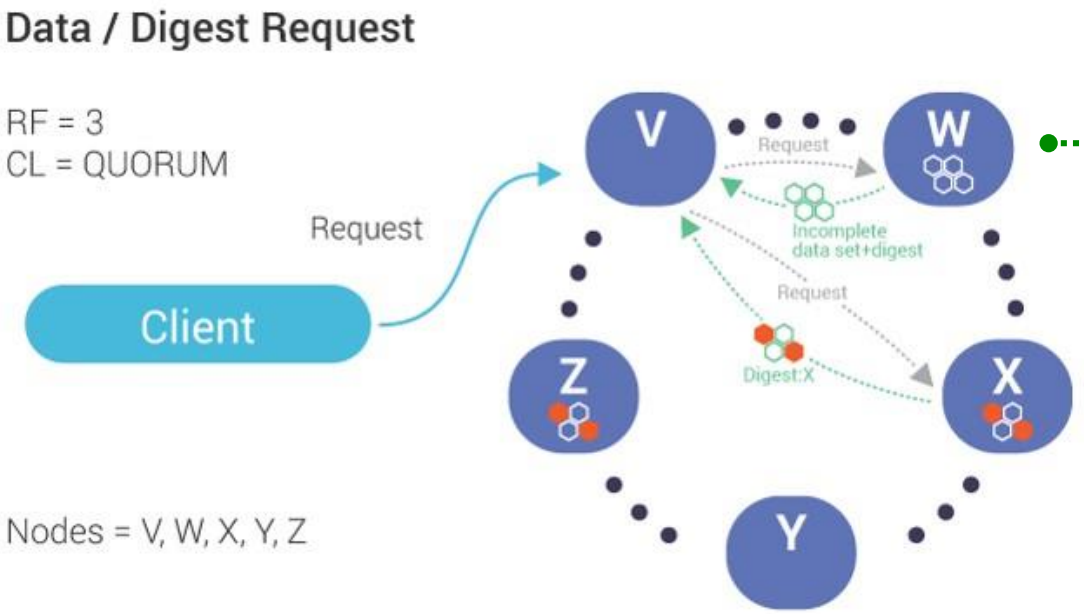


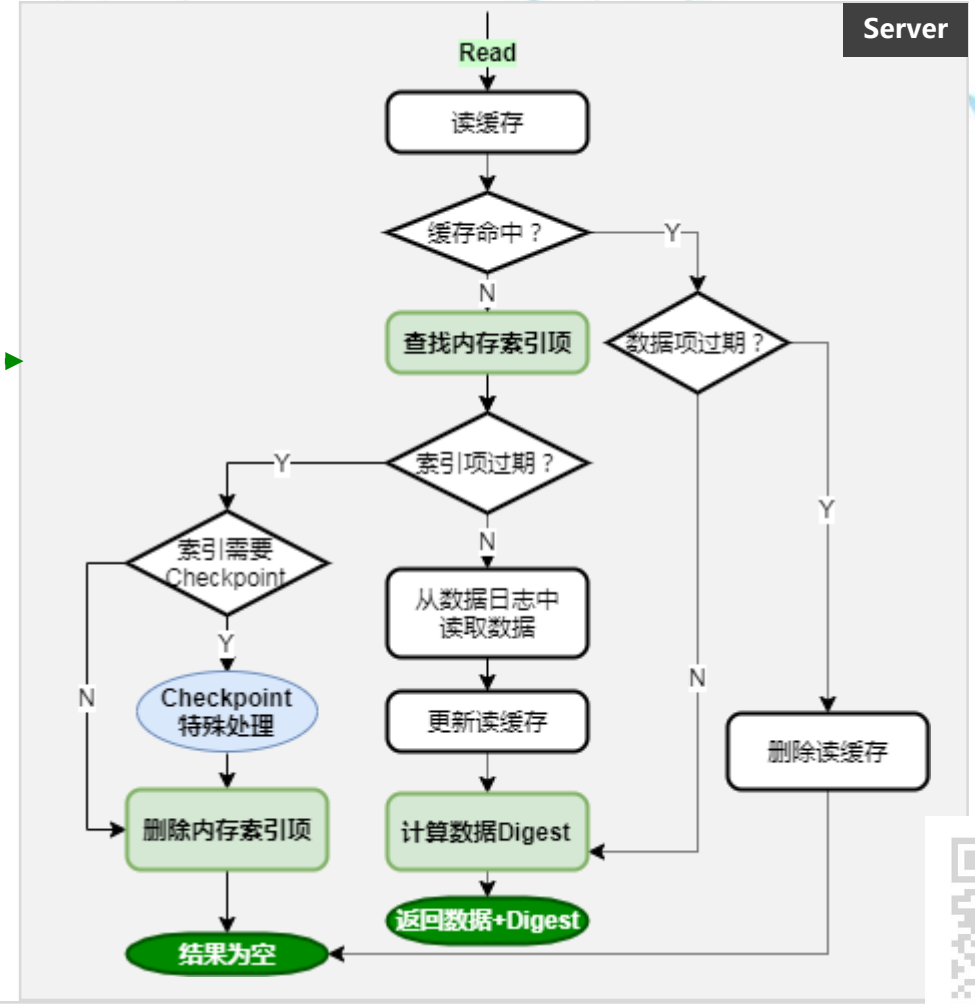
HiKV 写数据流程





HiKV 读数据流程



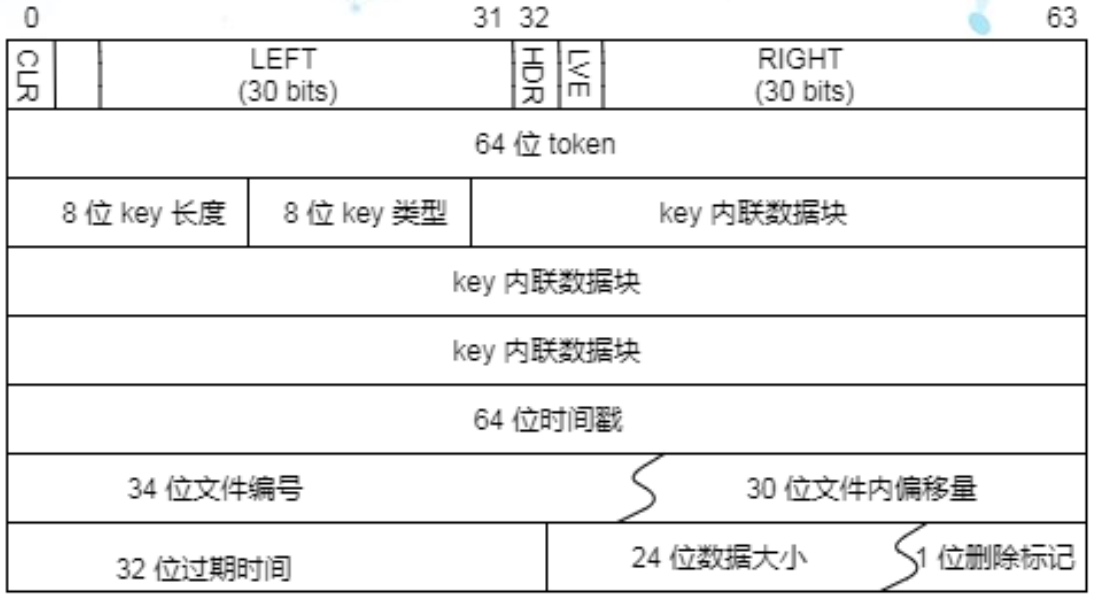


索引数据全部进memory，单机数据存储量是否受限？

HiKV 索引内存结构

- **每条数据 ⇔ 1条索引信息**
  - SET → 索引指向最新的Value
  - Delete/TTL → 删除索引
- 索引在内存中组成 **RBTree**
  - 支持 Token Range遍历
- 索引记录长度
  - 112 Byte (v1.0) → **64 Byte (v2.0)**
    - 数据文件编号 16B→8B
    - TTL 8B→4B
    - 数据记录大小 8B→4B
    - Key 16B→24B (**RipeMD160**摘要)
    - Token 20B→8B
    - 颜色CLR 8B→1bit
    - 父指针 8B→0 (保存在LRU缓存)
    - 2*孩子节点 2*8B→2*4B (**地址→下标**)
    - 4B对齐→0

HiKV 2.0 索引内存结构，每条索引长度 64 Byte



- 物理机内存：512GB
- 索引数据内存占比：90%
- 单条数据记录长度：1KB

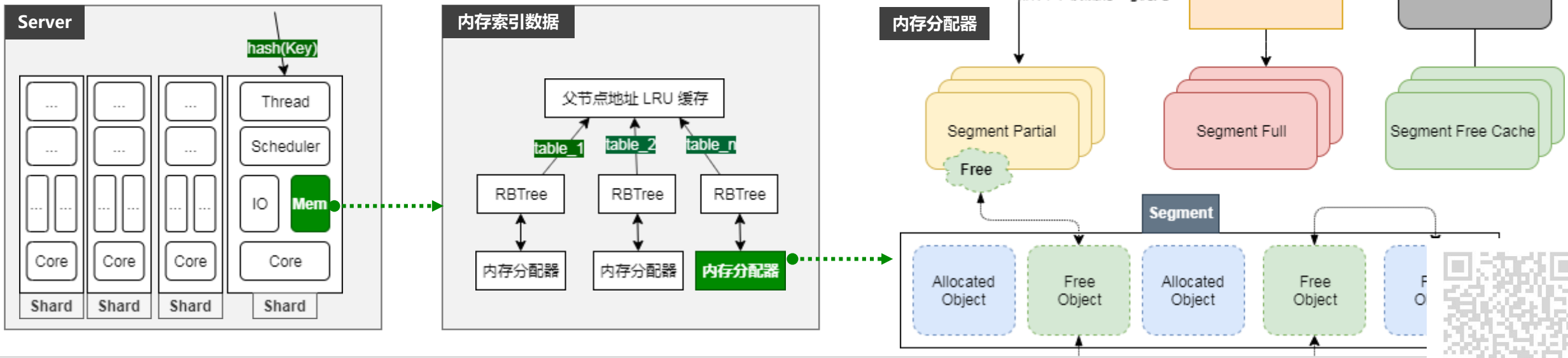
单台物理机存储容量

- HiKV 1.0： **4.1** TB
- HiKV 2.0： **7.2** TB

内存如何管理？ 使用Seastar的内存分配器LSA (Log-Structured Memory Allocator) ？

HiKV 内存管理

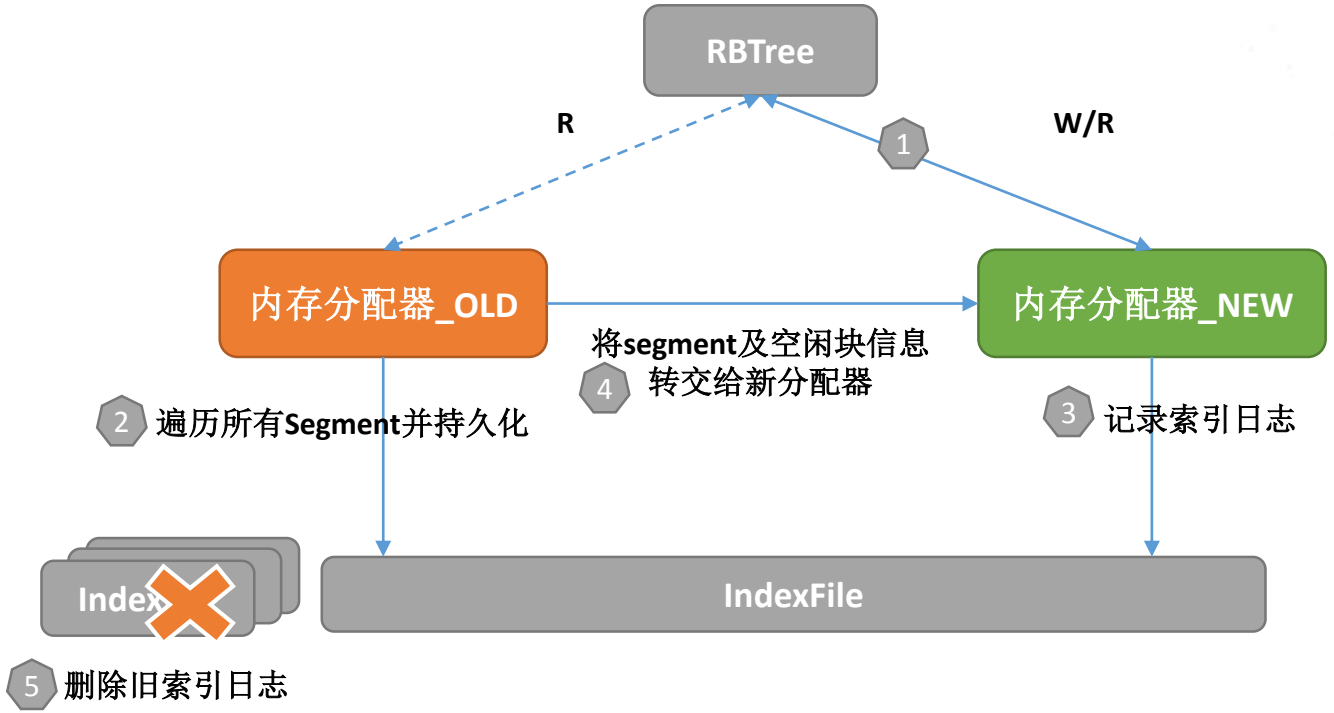
- Seastar LSA
  - 延迟释放 + 自动碎片整理 ➔ 索引记录长度需要96Byte，单机存储量下降33%
- HiKV 定长内存分配器
  - 延时释放 + 自动碎片整理 + 其他功能紧耦合(checkpoint)



索引数据如何进行 持久化 及 快速加载？

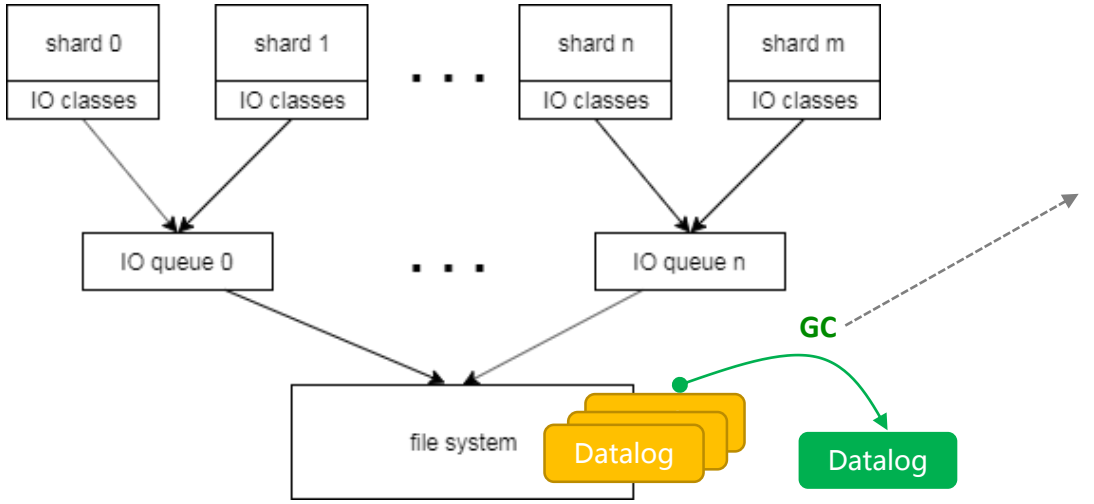
HiKV 索引数据Checkpoint

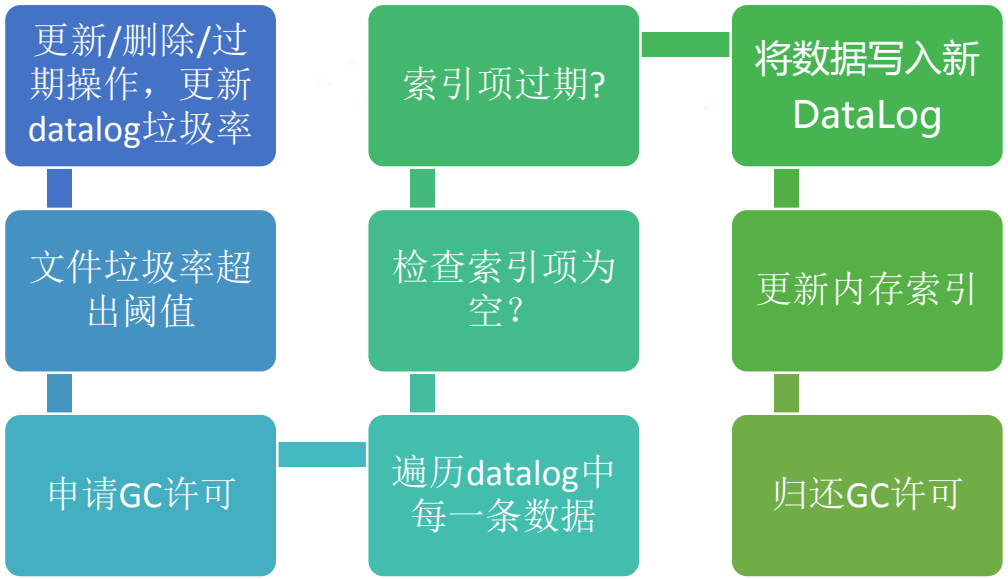
将某时刻内存索引数据持久化，并删除之前的IndexLog，加快节点启动速度



HiKV 数据文件GC

- Datalog中的垃圾数据
  - 删除的数据
  - 被覆盖的旧数据
  - 过期的数据
- Datalog GC
  - 将垃圾率超出阈值的Datalog重写，释放存储空间
  - 控制GC Thread并发度和优先级，避免影响正常操作





HiKV 设计思路总结

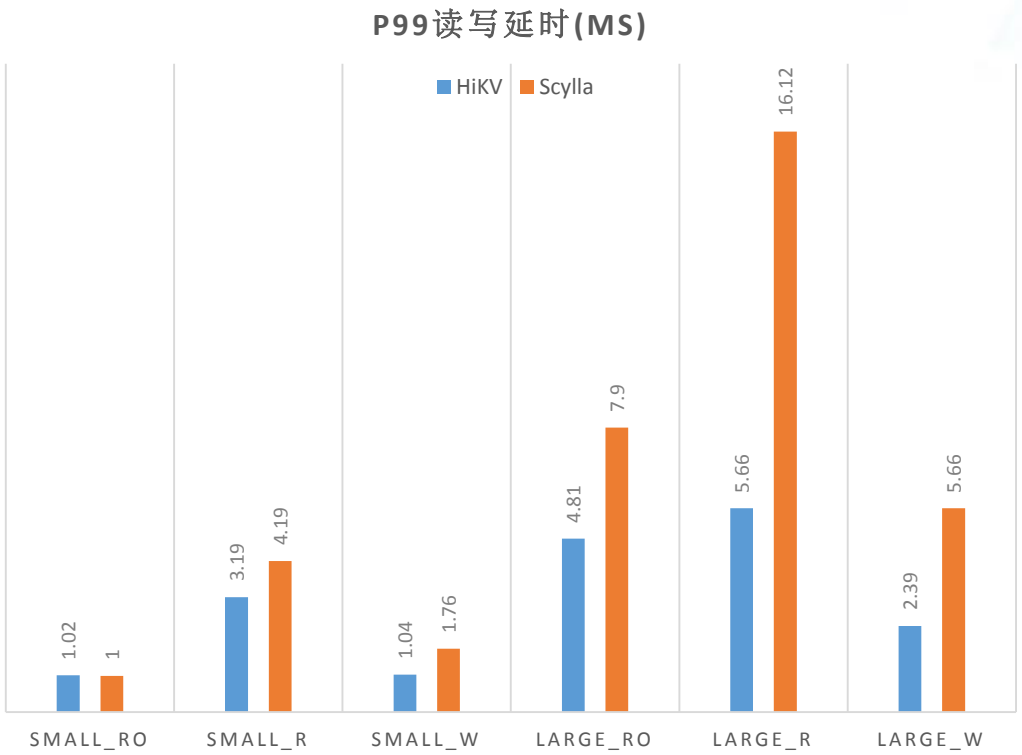
HiKV = Scylla + WiscKey存储引擎

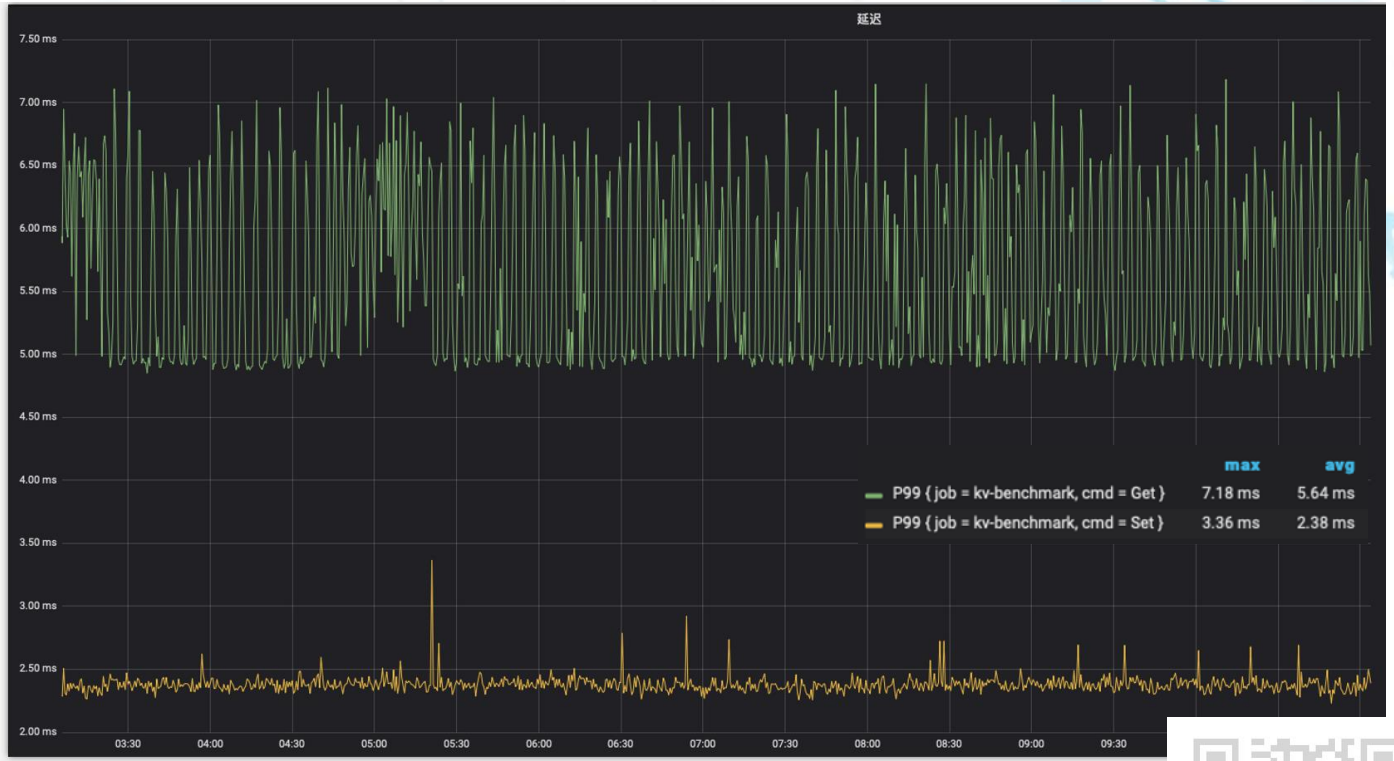
- 索引
  - 索引以RBTree全部保存在内存，1次盘IO读到数据，消除读放大
  - 优化索引结构，并定制定长内存分配器，减小索引内存占用，提高单机数据存储量
- 数据
  - 数据以Log方式追加写入到Datalog文件，减小写放大
  - 轻量级的GC（并发控制+IO带宽控制）清除垃圾数据，释放存储空间
- 框架
  - Scylla分布式架构
  - Seastar thread-per-core 框架

提纲

- 为什么要开发 HiKV？
- HiKV 设计思路与关键技术
- **HiKV 在爱奇艺的应用**

性能比较 HiKV vs. Scylla

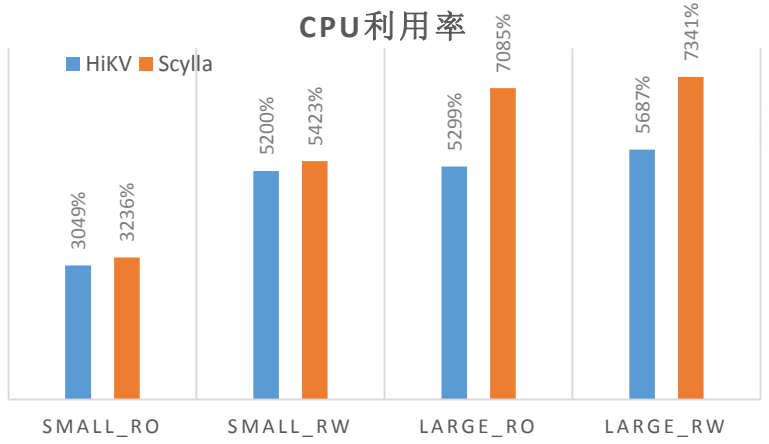


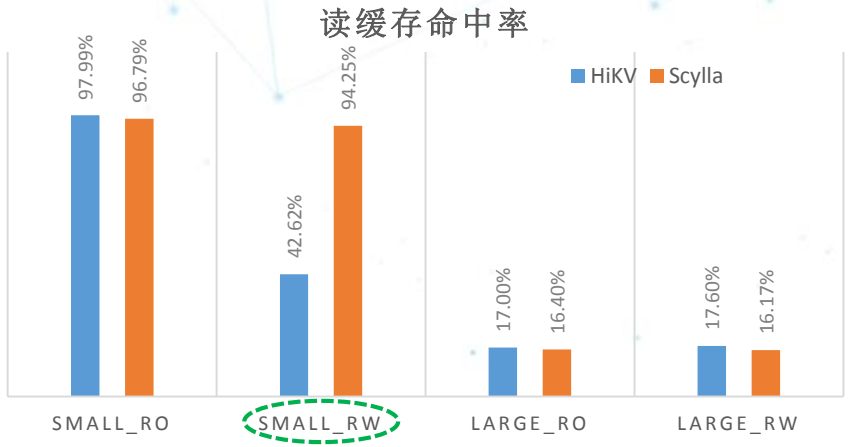


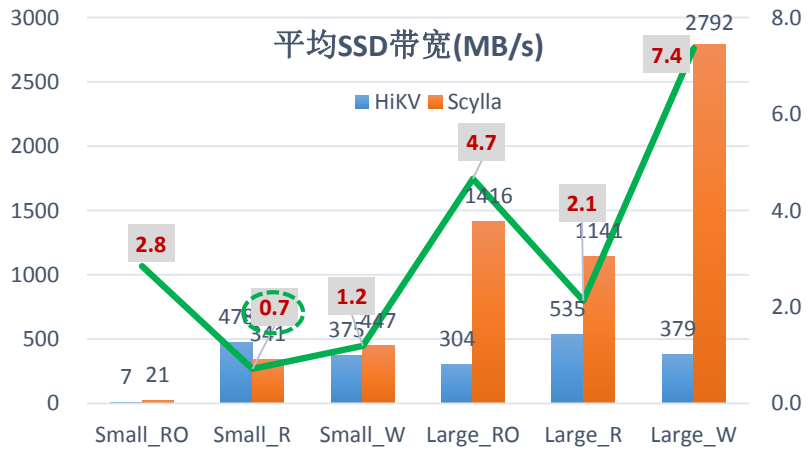
随机读写各5w QPS 下 HiKV P99延时

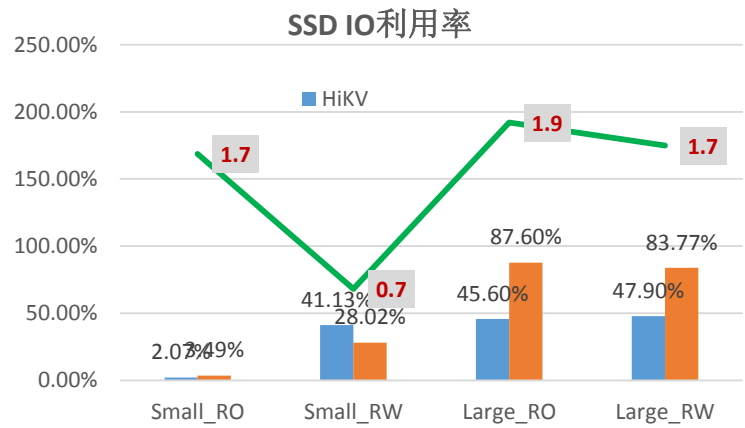
小数据量下，延时相当；大数据量下，HiKV 读写延时稳定，且明显优于Scylla

性能比较 HiKV vs. Scylla









HiKV 默认缓存策略对小数据量不友好，但CPU利用率较Scylla低，且极大降低了读写IO放大

HiKV在爱奇艺的应用实践

2017Q4

立项，封装Scylla

团队1.5人

2018Q1

开发WiscKey引擎

团队2人

2018Q2

发布v1.0，支持扩容、多列、视图等

在内部IM工具灰度上线

2018Q3

v1.2，支持缓存、GC、定长内存分配器等

推荐系统用户兴趣与行为/内存偏好数据

2018Q4

v1.5，内存索引结构优化至72Byte

推荐历史、播放记录、码流数据等

2019Q2

v2.0，内存索引结构优化至64Byte，RipeMD160 摘要、索引checkpoint、写保护等

应用 推荐、BI、搜索、广告、视频播放、安全等

数据 视频特征、用户特征、视频元数据、推荐历史、设备指纹、KV持久化等

成本 约等于 Couchbase 1/6，Redis 1/9

Small Size KV: 性能接近内存KV

6节点/9TB数据
KV平均长度 <1KB
读写比 50:1，QPS峰值20w
缓存命中率：73%
延时 P99<1ms

Large Size KV: 长尾延时不超过10ms

12节点/50TB数据
KV平均长度 5KB
写QPS峰值35w，读QPS峰值10w
缓存命中率：51%
延时 P99<9ms

未来计划

- 用Optane扩容内存
- 优化内存碎片整理
- 启用DPDK
- 数据压缩

招聘 ➔ 邮件我，直接聊

guoleitao@qiyi.com

Thanks