大规模分布式缓存系统
——Squirrel技术内幕

王辉@基础架构中心 2018

- 2015年加入美团点评存储平台中心
- 负责过 avatar-cache、medis、squirrel

主题

- 背景&现状
- Redis Cluster原理简介
- Squirrel 技术内幕
- 最佳实践

一 背景&现状

- 从关系型存储到全内存KV存储
- 继承自 Avatar-Cache 缓存框架.
- 基于Redis-Cluster 的 Key-Value 存储框架

一 背景&现状

- 集群

850+

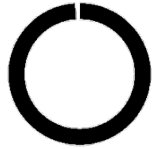
- 节点

1W+

- 平均响应延迟

0.7ms

- 总体可用性



>99.999%

二 Redis Cluster

- Remote Dictionary Server

  一个支持丰富数据结构的全内存Key-Value存储系统

- 数据结构

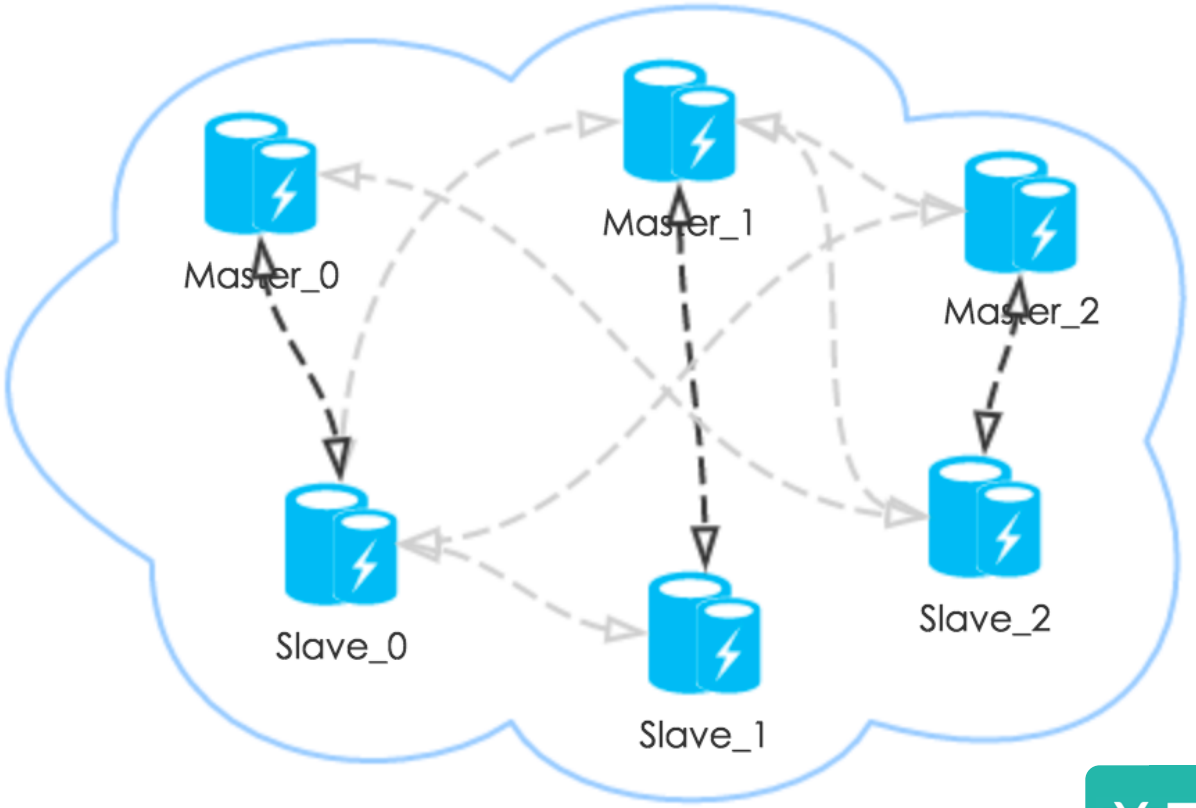
  String、Hash、 List、 Set、 SortedSet 等

- 设计目标

  高性能、高可用、可扩展

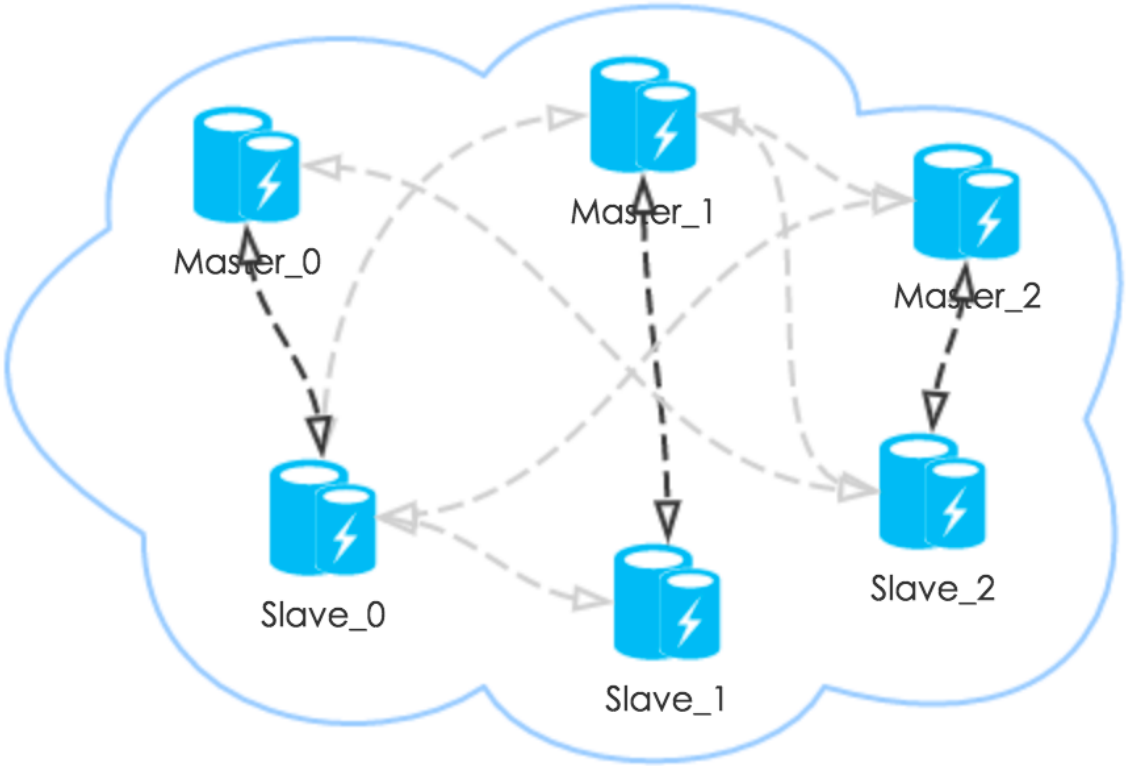
2.1 Redis Cluster

- 分布式
  - 无中心节点
  - 节点间Gossip协议通信
- 多副本
  - 分片主从全量备份



2.2 Redis Cluster – 高可用特性

- 主从模式——数据多副本
- 部分节点失效不影响整个集群
- 集群自动检测fail节点
- 主节点故障后自动的主从切换
- 可扩展上千节点



2.3 Redis Cluster -- 数据路由

- 预分片（一致性哈希）

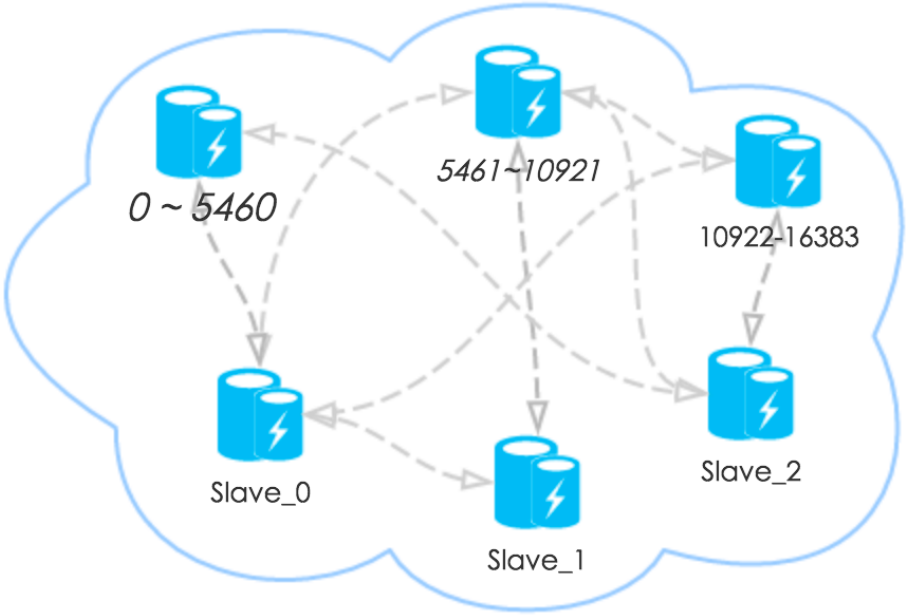
  16384个哈希槽

  每个分片负责一部分哈希槽

  HASH_SLOT = CRC16(key) mod 16384

- 扩展性

  通过增加分片提高集群的写能力以及容量



2.4 Redis Cluster – 主从数据备份

- 主从复制

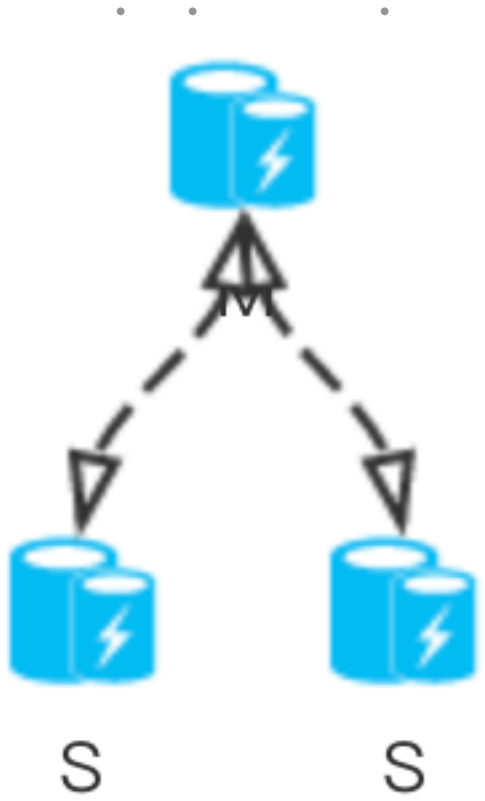
  命令传播方式

  异步复制——最终一致性

  从节点只读

- 扩展性

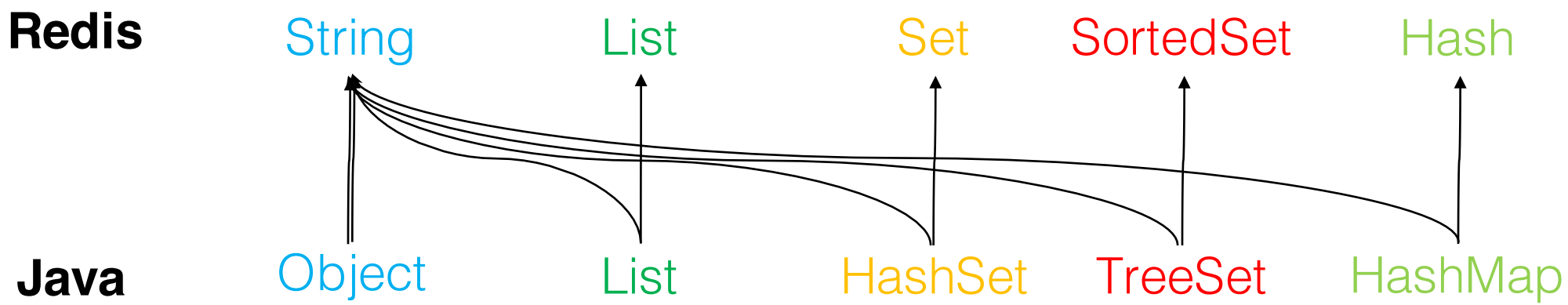
  通过增加Slave节点来提高查询能力



2.5 Redis Cluster – 持久化

- RDB
  - 1 保存和还原Redis 服务器所有的键值对数据
  - 2 save 900 30
- AOF
  - 1 appendfsync（always、everysec、no）
  - 2 aof-rewrite

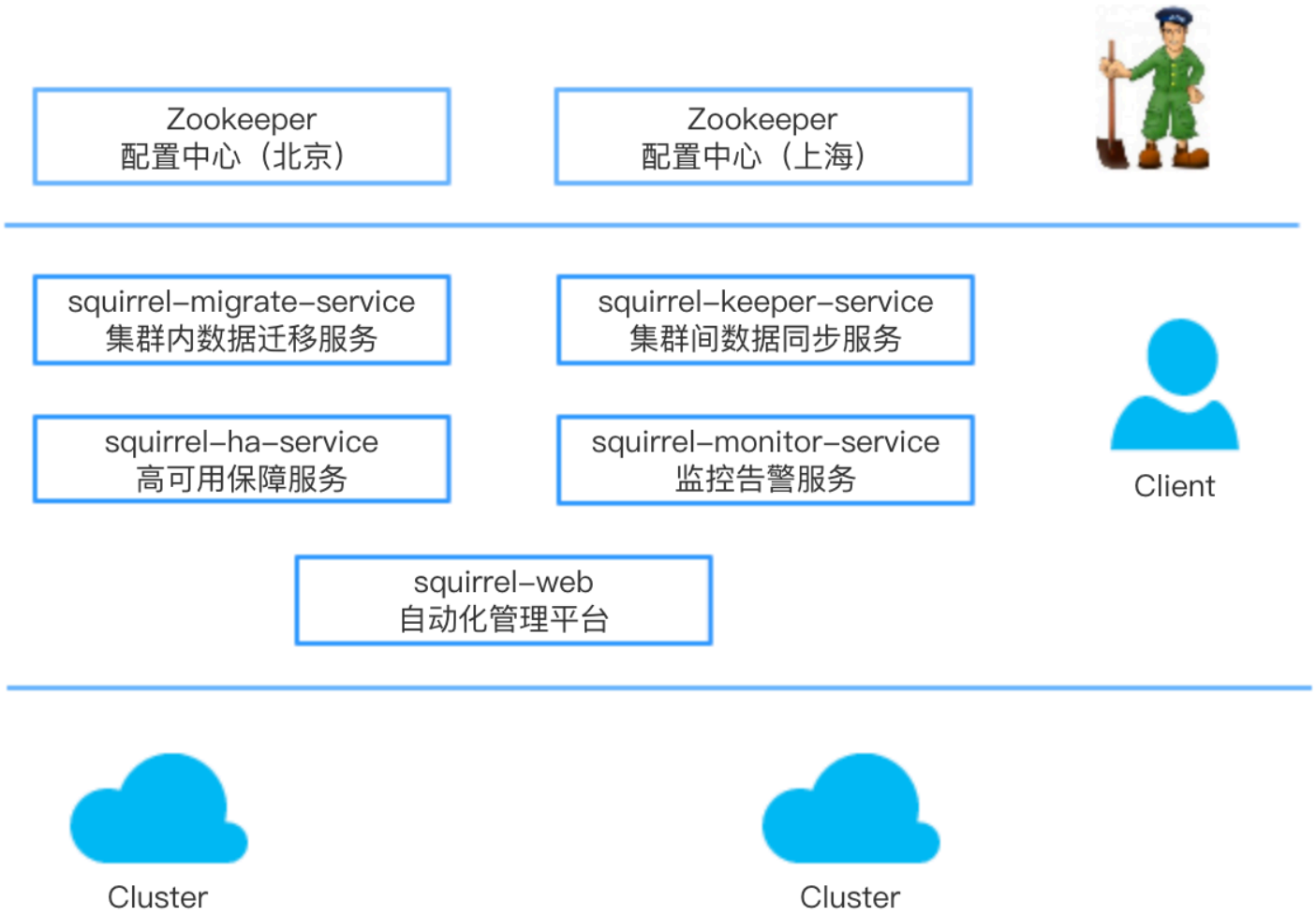
2.6 Redis Cluster – 数据结构



三 Squirrel技术内幕

- 整体架构
- 技术细节
- 最佳实践

3.1 整体架构


Zookeeper
配置中心（北京）
Zookeeper
配置中心（上海）
squirrel-migrate-service
集群内数据迁移服务
squirrel-keeper-service
集群间数据同步服务
squirrel-ha-service
高可用保障服务
squirrel-monitor-service
监控告警服务
squirrel-web
自动化管理平台
Client
Cluster
Cluster

3.2 技术细节

- StoreKey
- 可用性保障
- 数据无损扩容

3.2 StoreKey

- 概念： Key = new StoreKey(Category, params)

  Category：抽象的，类似于数据库中的table

  IndexTemplate：模板参数，例如 c{0}d{1} 。类似于primary key

  Key ： category.c+*param0*+d+*param1*+_version

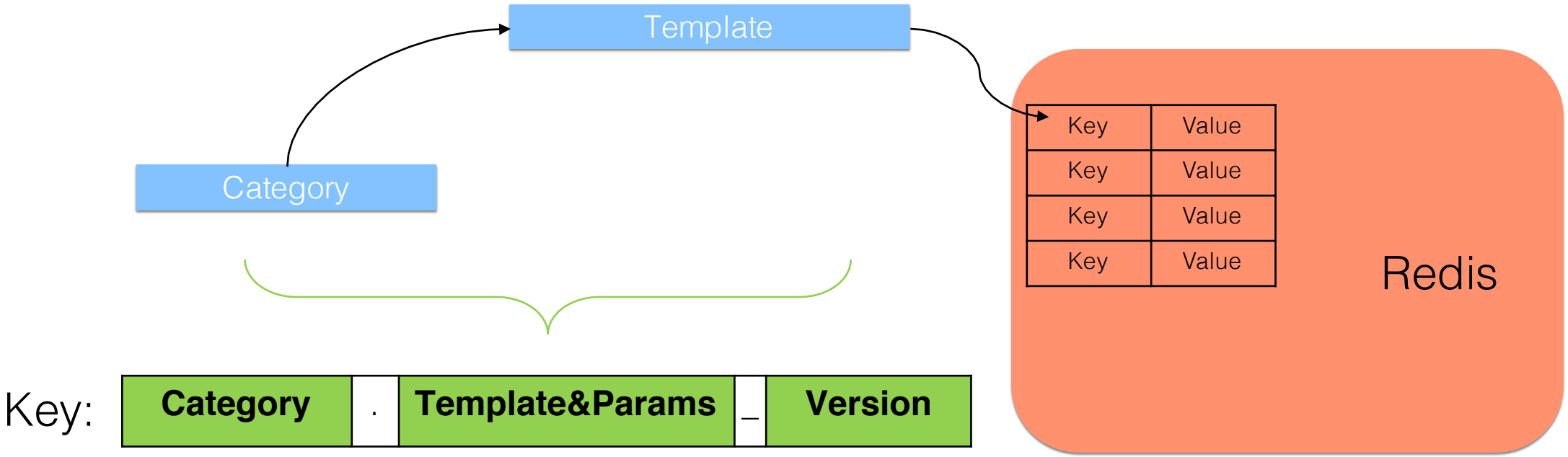
- 意义

  规范化：程序封装，key的格式统一

  优化管理：所有业务访问、运维操作、权限分配都基于Category

  缓存清理：version+1

3.2 StoreKey


Template
Category
Key
Value
Key
Value
Key
Value
Key
Value
Redis
Key:
Category
.
Template&Params
_
Version

3.2 可用性保障

- Squirrel维护近千个集群，共上万个节点，保证集群的可用性是重中之重。
- 从哪些方面去保障高可用？

  1 快速发现、定位问题 → 监控系统

  2 快速恢复 → 自动化故障处理

  3 避免问题、预知问题 → 运营治理、故障演练

3.2.1 监控体系

- 端到端监控：Cat

  快速定位出是服务端还是应用端的问题

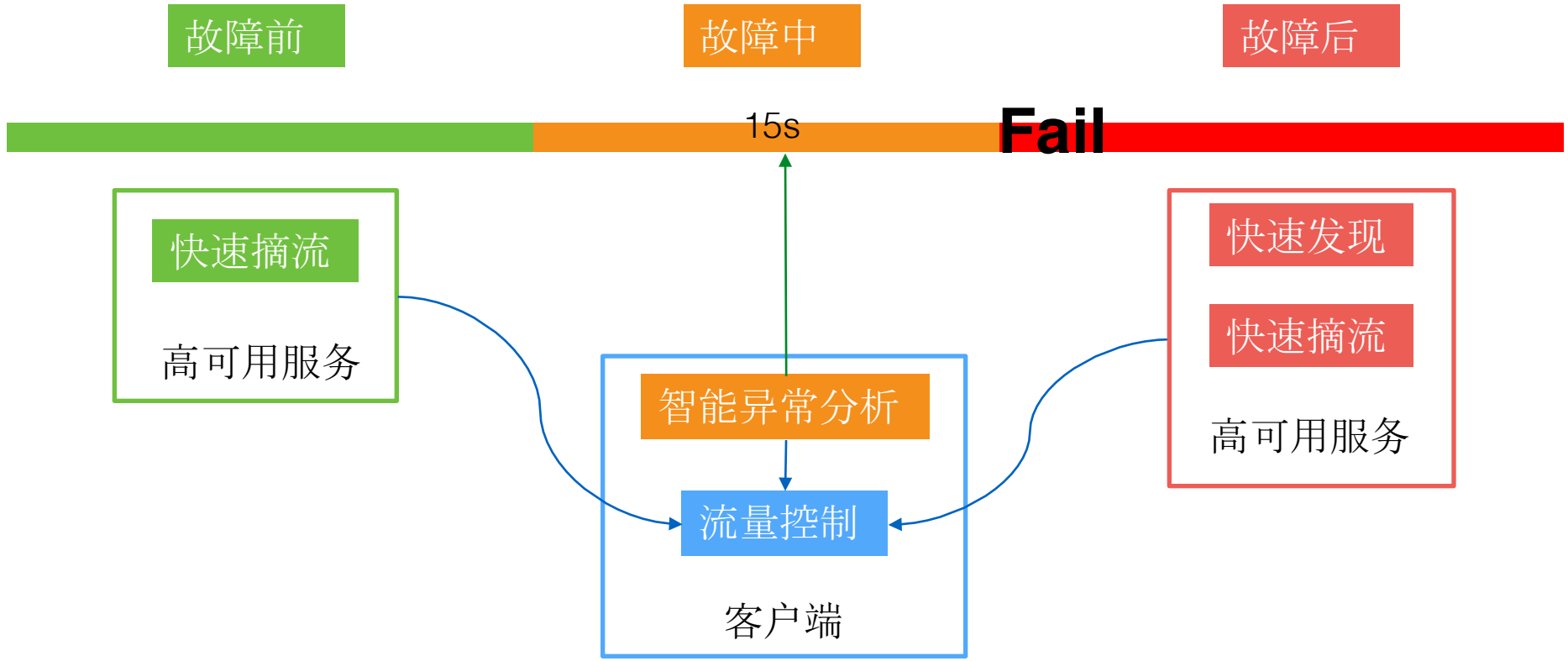
  Squirrel的Cat监控介绍

- 服务端监控：Falcon + Squirrel-monitor-service

  Falcon：物理机以及docker维度的系统监控

  Squirrel-monitor-service ：redis 实例状态的监控

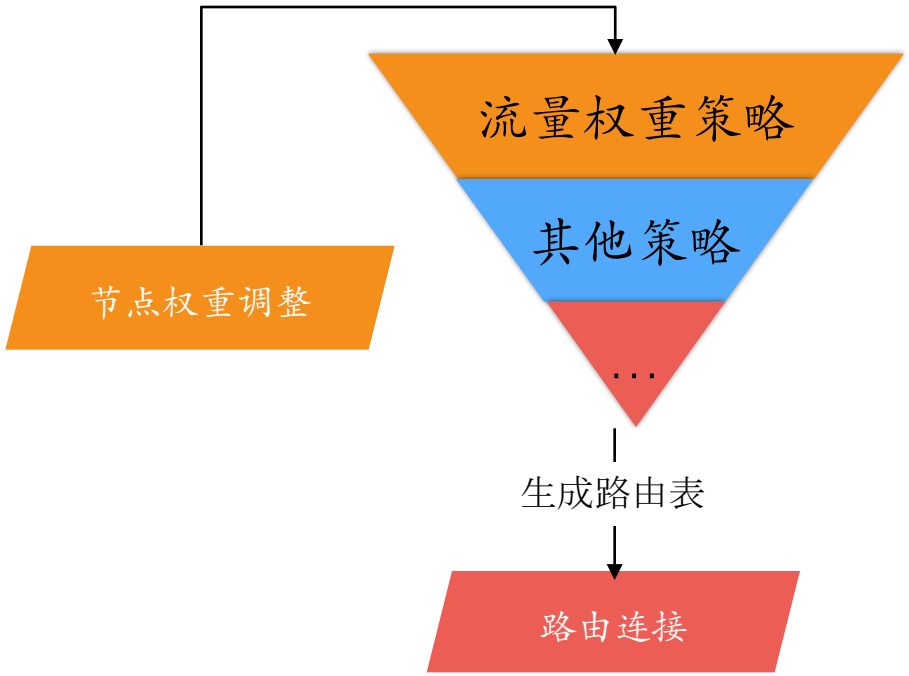
3.2.2 自动化故障处理



- ✓ 不同的故障场景可以在不同的阶段尽早的发现和处理
- ✓ 流量控制 -» 不访问故障节点

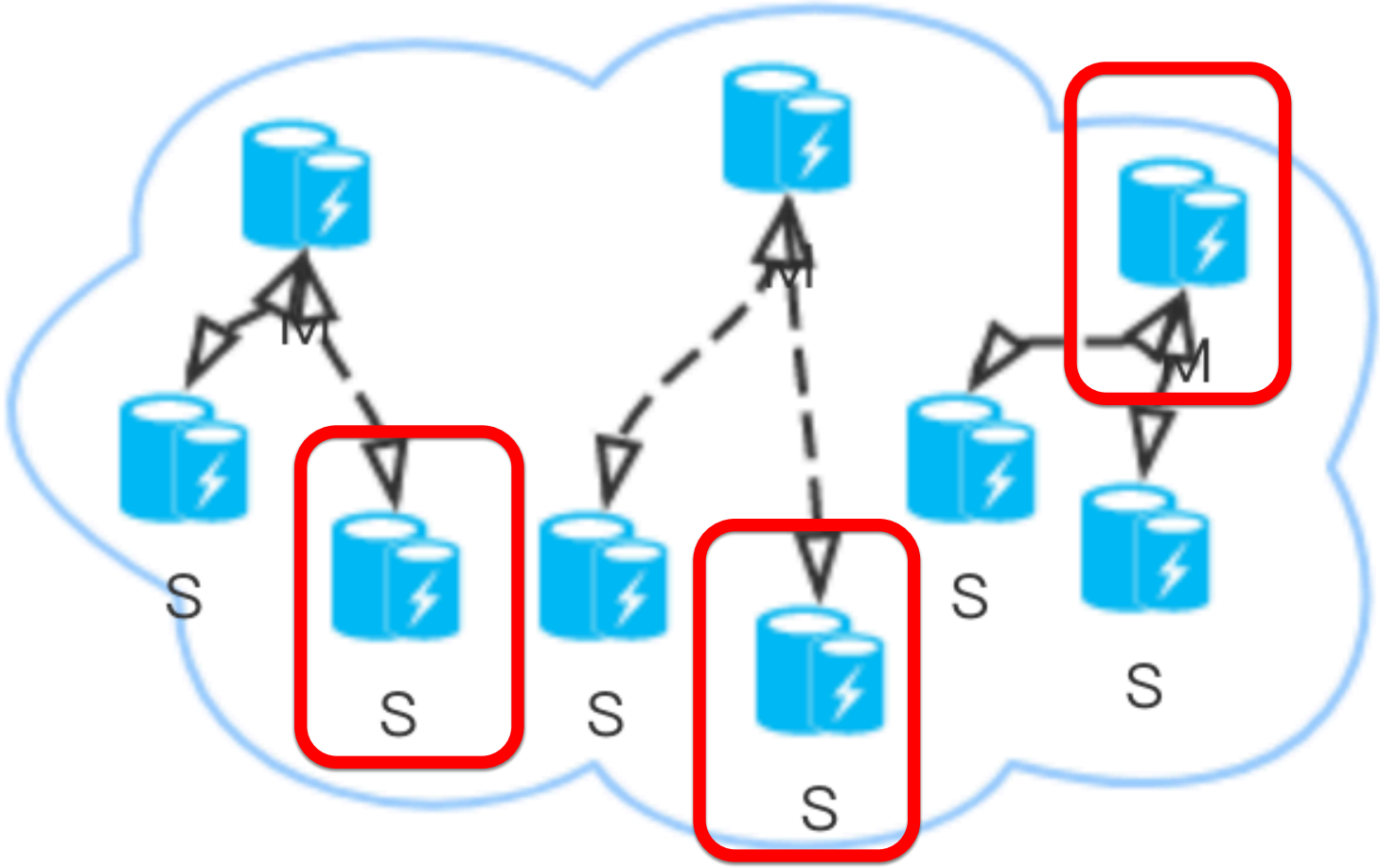
3.2.2 客户端流量控制

- **流量权重策略**
  - 通过调整集群中节点的权重，配合流量权重策略来控制最终的客户端本地路由
- **运行时动态调整**
  - 客户端可随时调整节点的路由权重，高可用服务则通过修改ZK配置来通知客户端做调整
- **异步节点恢复检测**
  - 客户端会异步检测因智能异常分析被摘除流量的节点，待其恢复正常后会逐步的恢复其节点流量

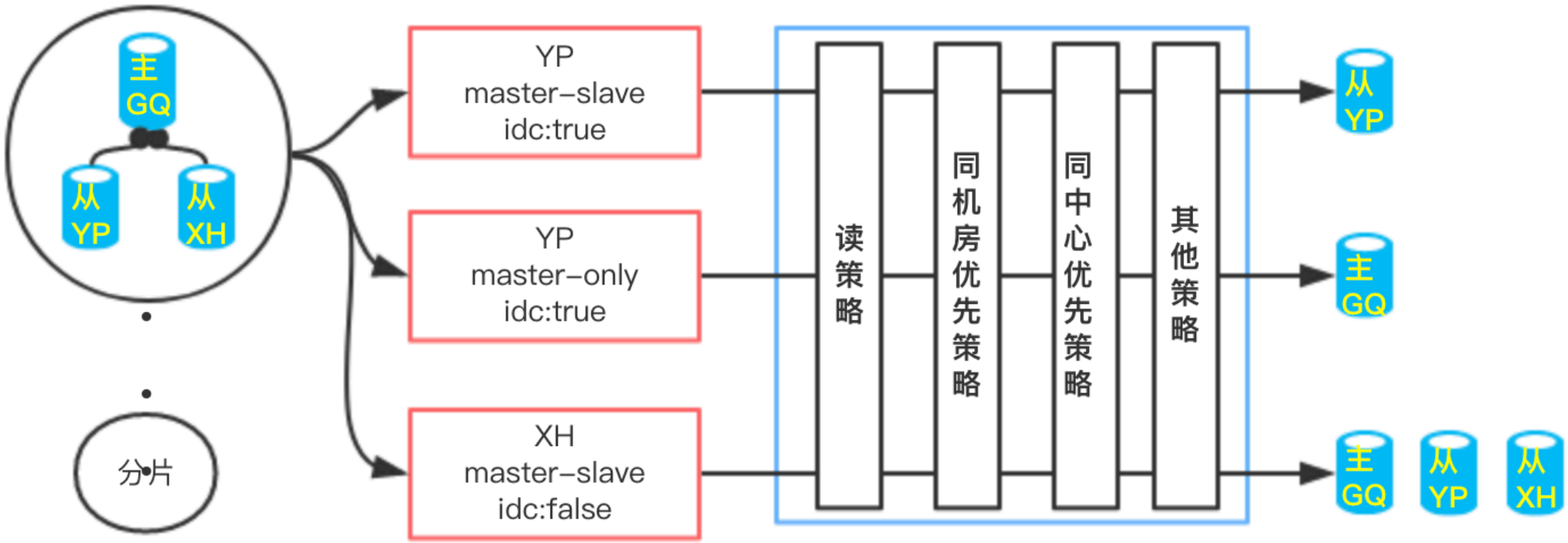


3.2.2 路由

- 分片中的每个节点数据一样
- 客户端根据策略筛选合适的节点
- 客户端读操作从本地的路由表中选择



3.2.2 路由策略


主
GQ
从
YP
从
XH
分片
YP
master–slave
idc:true
YP
master–only
idc:true
XH
master–slave
idc:false
读策略
同机房优先策略
同中心优先策略
其他策略
从
YP
主
GQ
主
GQ
从
YP
从
XH

3.2.2 路由策略

- 路由策略（只针对读操作）

1. master-only:

   所有的读操作都在主节点上操作，适合对于一致性要求非常高的业务

2. master-slave:

   主从负载均衡，主从节点均负责一部分的读操作

3. Slave-only:

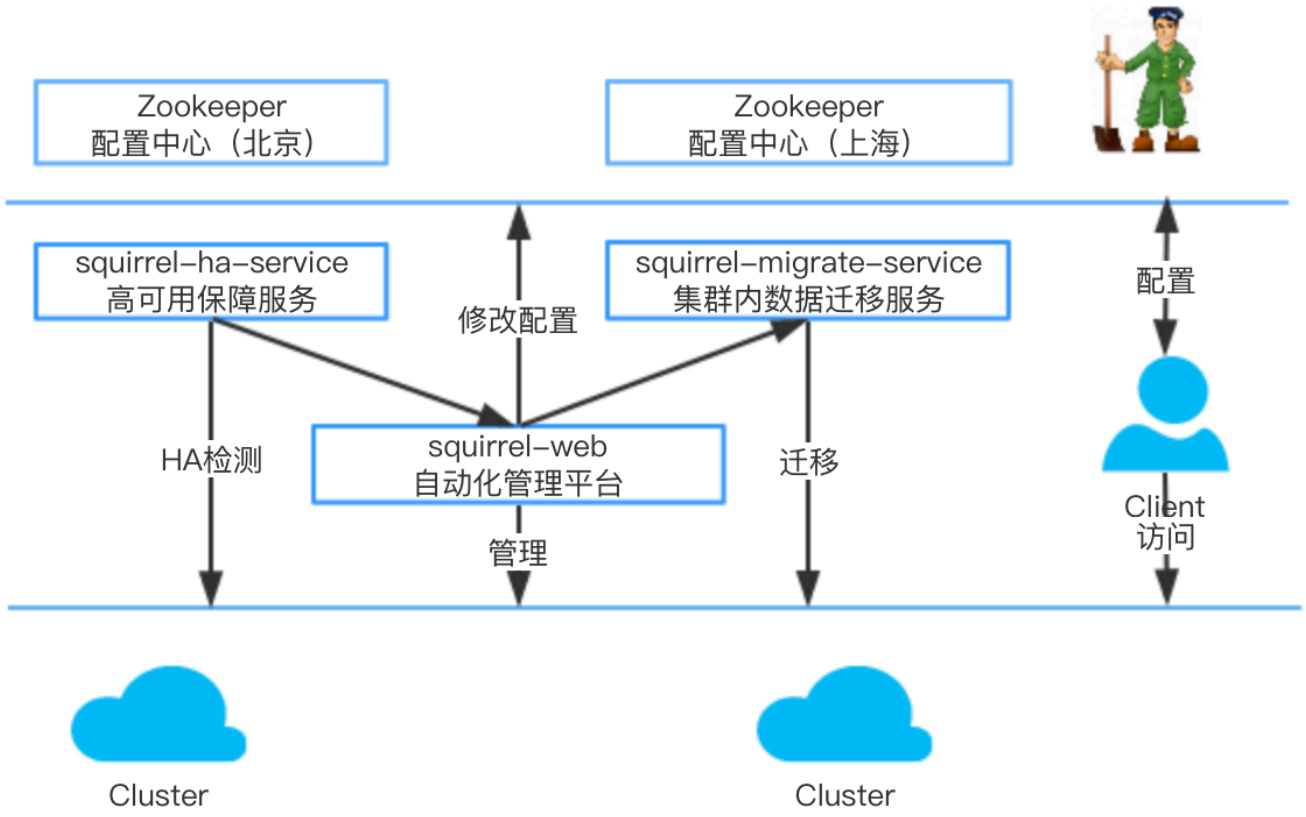
   只读从节点，适合用于读写分离场景，主节点只负责写操作

3.2.2 路由策略

- 随着业务不断扩大，各个机房均有部署应用机器，跨机房的调用会增加响应延迟（平均 1ms左右）

- 就近路由策略
- **idcSensitive（true/false）：同机房优先**
- **regionSensitive (true/false)：同中心优先**

3.2.2 Squirrel自动化故障处理体系



3.2.2 故障恢复时长

- 多种故障场景下的恢复时长

运维操作

0

异常关闭

1~15s

异常抖动

1~60s

加载数据

1s

宕机

15~30s

- 基本保证业务报错在 ＜1分钟

3.2.3 运营治理、故障演练

- 解决问题最好的方法是避免问题发生

1. 避免已知

   资源隔离：docker部署

   调度系统：统一管理资源的分配，自动化资源编排

   满足各种节点分布限制

   运营治理：大Key治理，资源利用率，内存倾斜等

2. 发现未知

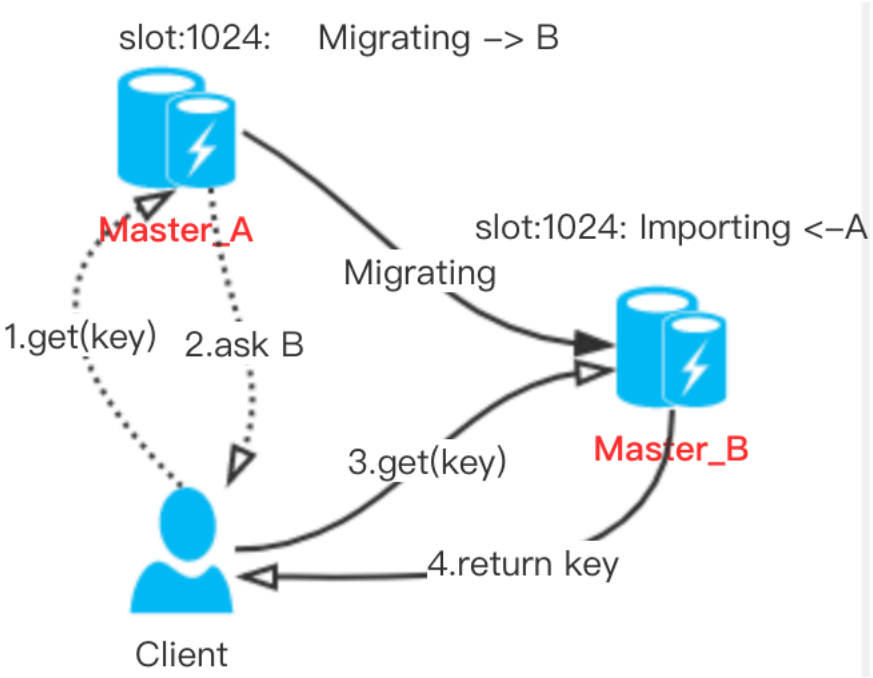
   故障演练：通过对线上真实环境、真实业务的故障演练来发现之前没有发现的问题。防患于未然。

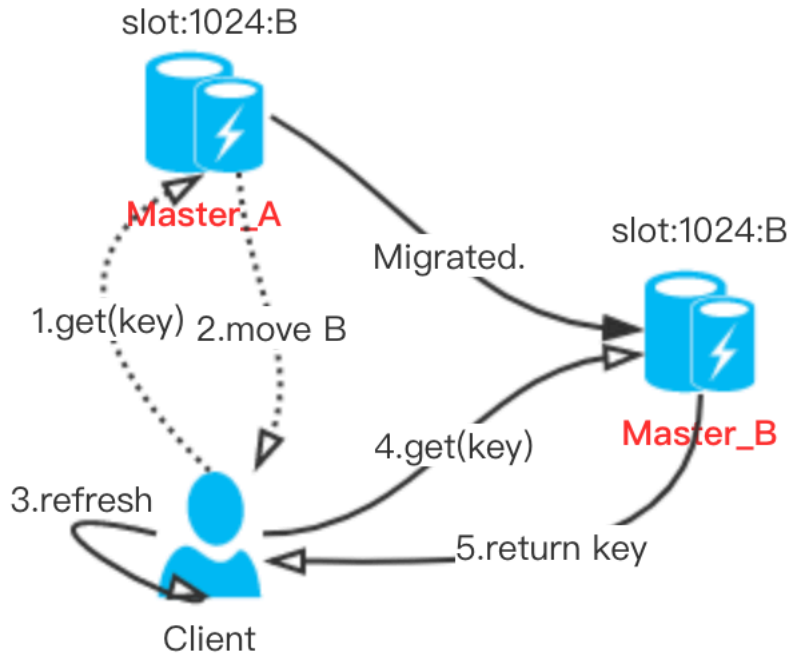
3.3 可扩展

- 服务端
- 可线性扩展，支持动态增删节点，最大支持近千节点规模
- 客户端
- 集群的扩容缩容对客户端无感知
- 管理端
- 扩容缩容，数据无损迁移

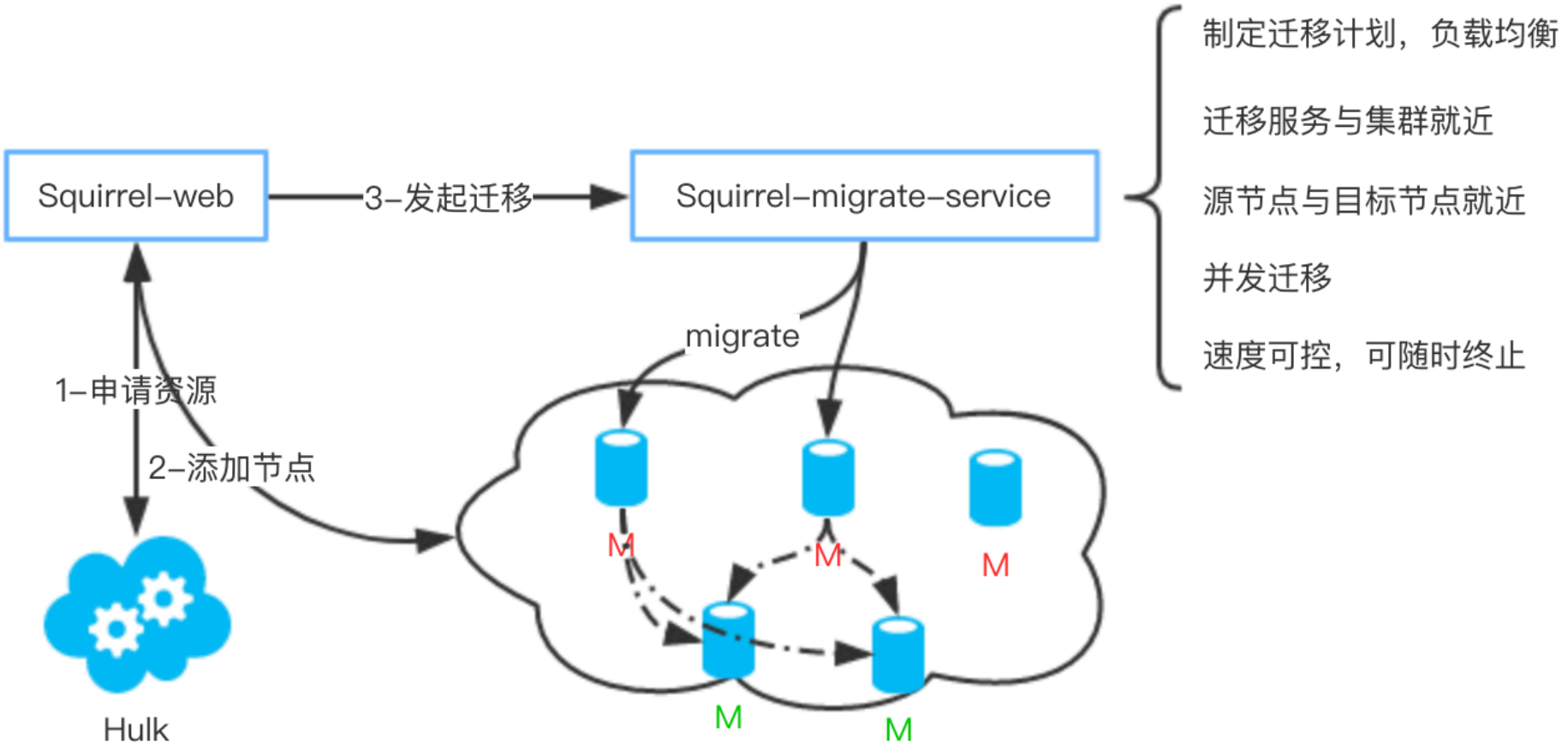
3.3.1 客户端无损路由

- 迁移中的读流程





3.3.2 数据迁移服务


Squirrel-web
3-发起迁移
Squirrel-migrate-service
1-申请资源
2-添加节点
migrate
Hulk
M
M
M
M
M
制定迁移计划，负载均衡
迁移服务与集群就近
源节点与目标节点就近
并发迁移
速度可控，可随时终止

四 最佳实践

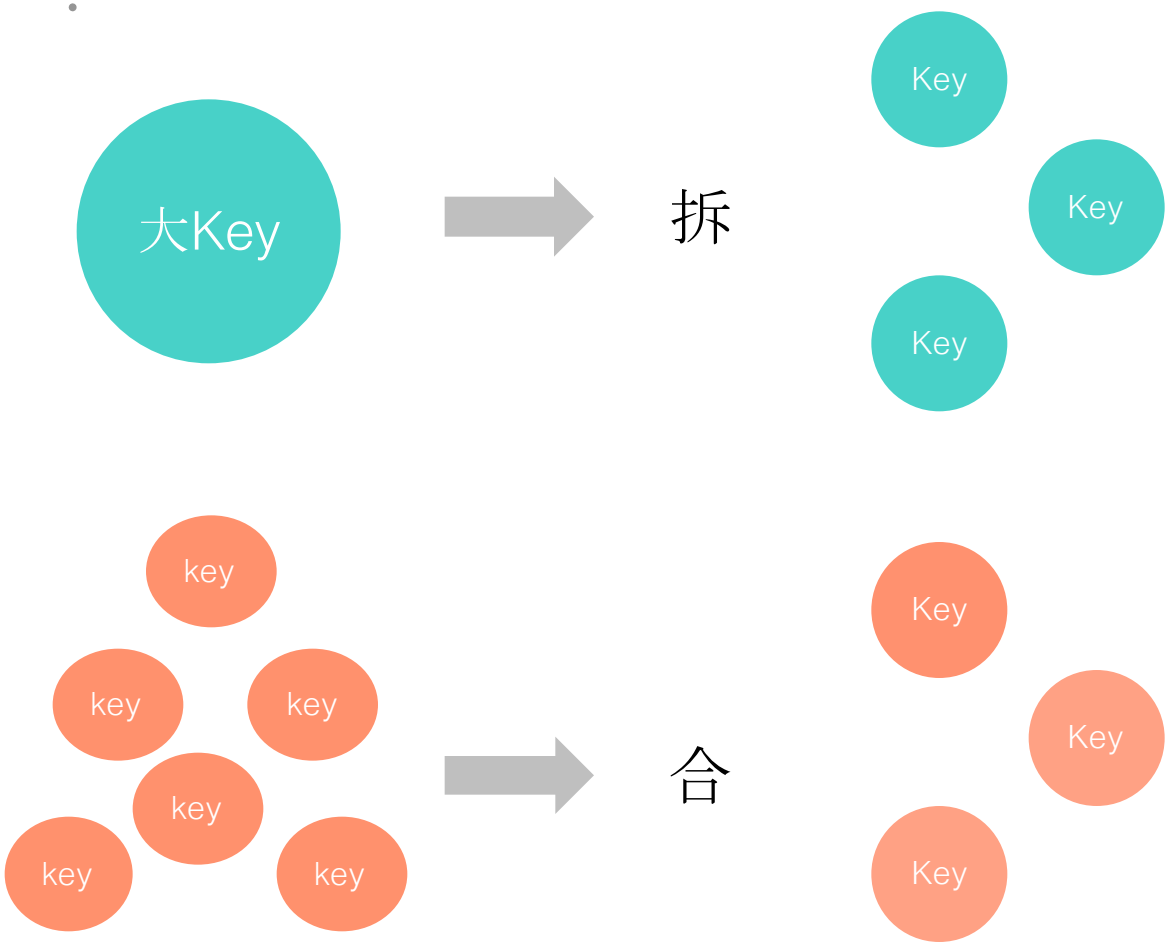
- 大Key拆分，多Key合并
- 多次操作可以使用批量方式（Multi，Pipeline），减少网络开销
- 多使用redis原生的数据结构
- 主动更新缓存，不要同时过期大量key，防止缓存穿透、雪崩

4.1 大Key&多Key

扩容难
易阻塞
性能有损

内存浪费
1亿 ≈ 10G

4.1 大Key&多Key

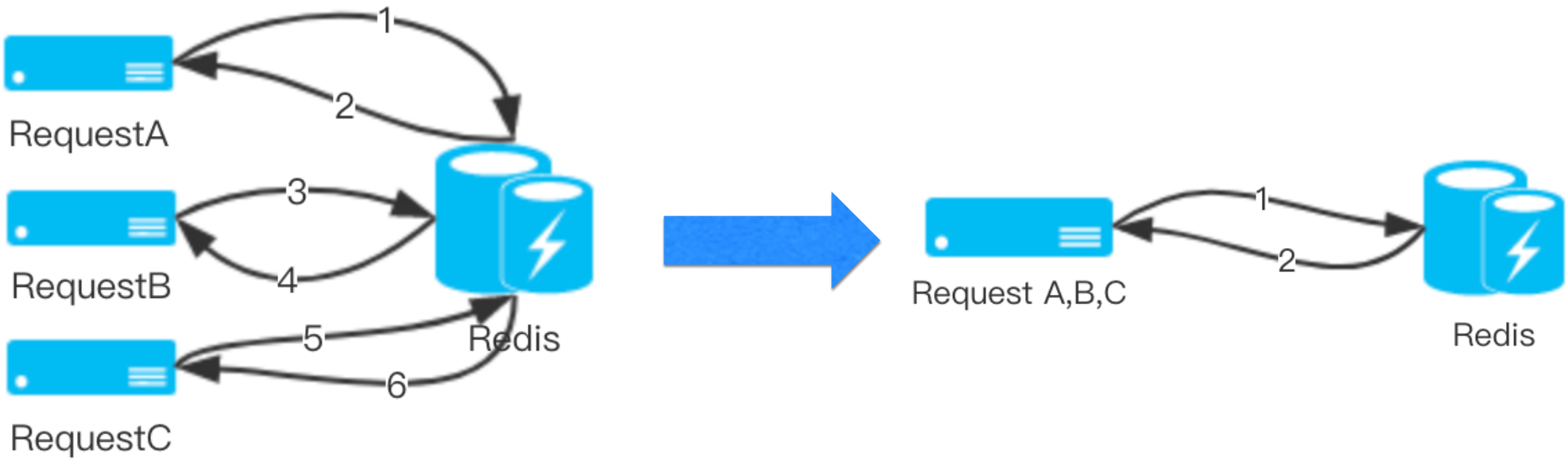


- 对象 -> Hash
- 预分桶

- 多String -> Hash

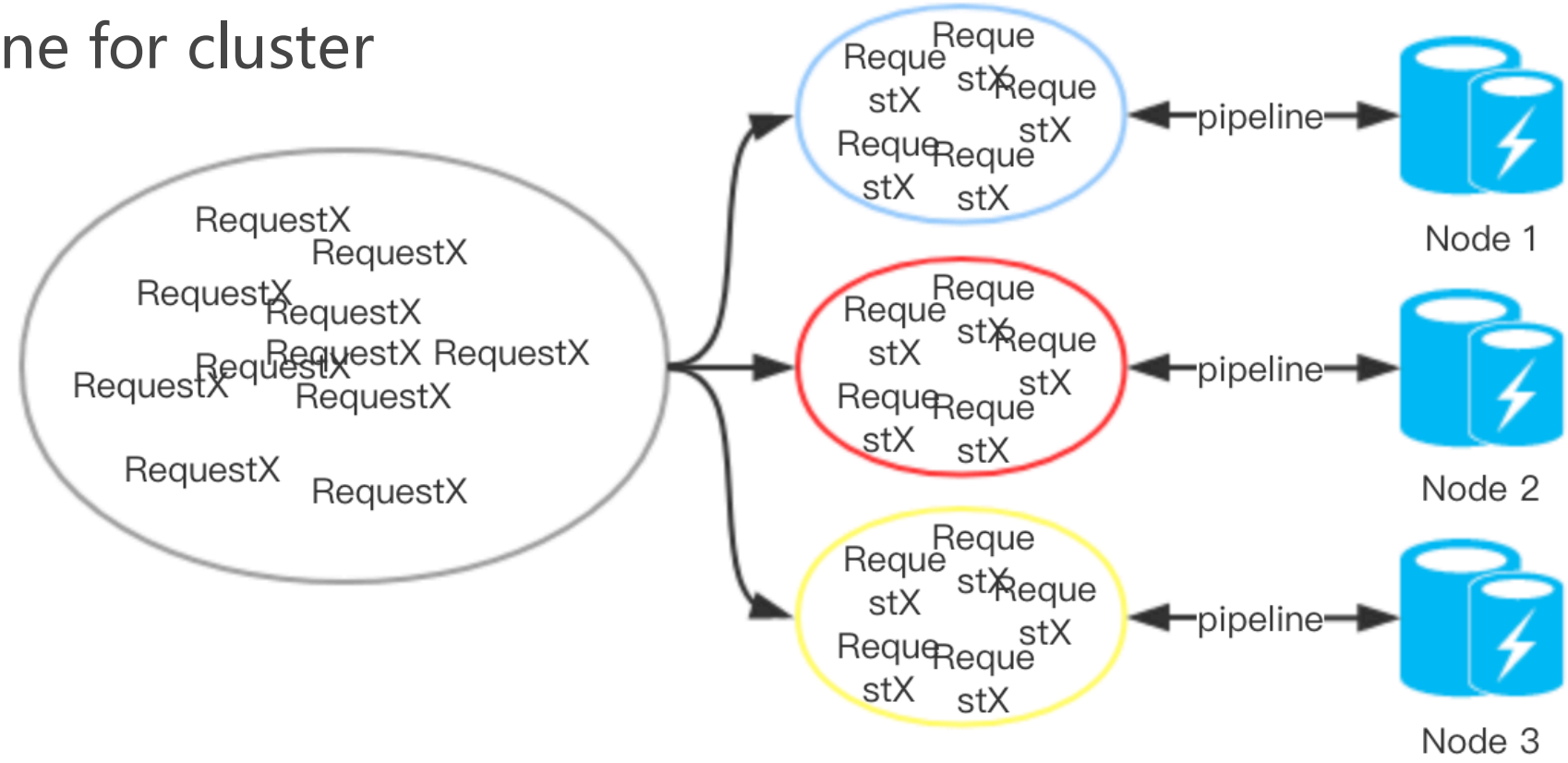
4.2 pipeline

- Pipeline



4.2 cluster pipeline

- Pipeline for cluster



4.2 wiki

大Key多Key ： https://wiki.sankuai.com/pages/viewpage.action?pageId=852403792

Redis数据结构：http://wiki.sankuai.com/pages/viewpage.action?pageId=730773314

Squirrel高可用服务设计：https://wiki.sankuai.com/pages/viewpage.action?pageId=826183083

路由策略：https://wiki.sankuai.com/pages/viewpage.action?pageId=768799970

热点Key策略：https://wiki.sankuai.com/pages/viewpage.action?pageId=1153178498

RD开发规范建议：https://123.sankuai.com/km/page/14770127

谢 谢

Q & A