十年架构 成长之路

SACC 第十届中国系统架构师大会

SYSTEM ARCHITECT CONFERENCE CHINA 2018

2018年10月17-10月21日 北京海淀永泰福朋喜来登酒店

IT168.com ChinaUnix ITPUB

Tendis在腾讯游戏的演进和实践

汪清平
腾讯游戏DBA团队

议程

- 为何使用Redis
- Redis在腾讯游戏中的应用
- Tendis产生的背景
- Tendis的设计和实现
- Tendis的集群方案
- Tendis的使用规模

*Redis is a **DSL** (Domain Specific Language) that manipulates **abstract data types** and implemented as a TCP daemon.*

— redis manifesto

*Redis is an open source (BSD licensed), **in-memory data structure store**, used as a database, cache and message broker.*

— http://redis.io

为什么用Redis

Redis是个基于内存的数据结构服务器

- 统一了数据存储和抽象数据结构
  - 程序 = 算法 + 数据结构
  - 应用 = 业务逻辑 + 数据存储
- 提供适合互联网业务的丰富特性
- 适用于分布式环境
- 性能卓越

Redis@腾讯游戏

使用场景：手游、官网活动、平台业务

- string：账号体系的ID映射/用户信息等
- hash：角色信息/装备道具经验
- list：消息队列/邮件通知/评论列表
- set：资格/白名单/黑名单等
- sorted set：排名相关的场景，关系链，延时队列等
- ttl：活动礼包、访问限频等
- geo：某moba游戏的战绩在南山区的排名

Redis@腾讯游戏

使用规模：

- 实例总数：几千实例
- 内存总量：TB级内存数据
- 请求峰值：数百万QPS
- 日请求量：过千亿

Redis@腾讯游戏

使用Redis的痛点

- 访问密度低，成本高
  - 某游戏业务： 最大QPS 5620，数据量35620 MB
  - 某平台业务： 最大QPS 几十，数据量40338 MB
- 数据规模大了可运维性不佳
  - 主从关系不健壮，网络闪断时master压力大，重建slave困难
  - 内存易成为瓶颈，数据量增长快，导致经常要做扩容，运维压力大
  - 存在数据丢失风险

Tendis的产生

Tendis：做成什么样？

- 所有数据实时落地，保证数据安全性
- 以SSD为主存储介质，提高机器利用率
- 高度兼容官方Redis，存量能迁移，增量可接入
- 更稳定的主从关系重建
- 支持数据备份和定点回档
- 性能要足够好

Tendis的产生

Tendis：怎么做？

- 从零开始造轮子还是站在巨人的肩膀上
- 如何用RocksDB存储Redis的数据
- 如何设计主从同步协议实现master/slave热备
- 如何实现数据备份与定点回档的建设

Tendis的设计和实现

站在巨人的肩膀上，基于Redis和RocksDB开发

- Redis和RocksDB都是经过业界广泛验证的项目
- 可重用Redis很大一部分代码
- 可继承Redis已有的丰富特性
- 只需用RocksDB API实现Redis命令
- **投入小，周期短，见效快**

Tendis的设计和实现

沿用Redis单线程模型的影响

- 不允许有阻塞Redis主线程的操作
  - 删除一个包含很多元素的大key
  - 在一个包含上亿key的实例上执行keys *命令
  - …
- 如何避免阻塞Redis主线程
  - 技术实现上面避免
  - 业务使用时规避
  - 架构规范的指导

Tendis的设计和实现

Redis数据结构在RocksDB中的表达

- 一组key/value来表达Redis数据结构包含的一个成员，而非整个数据结构
  - 对于复合结构，减少了读放大，和写放大，性能更好
  - 无法完美支持所有操作
- 复合数据结构由两类key/value表示，一级key和二级key
  - 一级key记录数据结构的成员数量，TTL，版本号等元数据
  - 二级key记录数据结构中的具体元素
- 利用RocksDB中key/value的有序性表达元素之间的有序关系
  - list中如队列的相对顺序
  - zset中元素按照分数有序

Tendis的设计和实现

string和hash在RocksDB中的表达

string

hash

Tendis的设计和实现

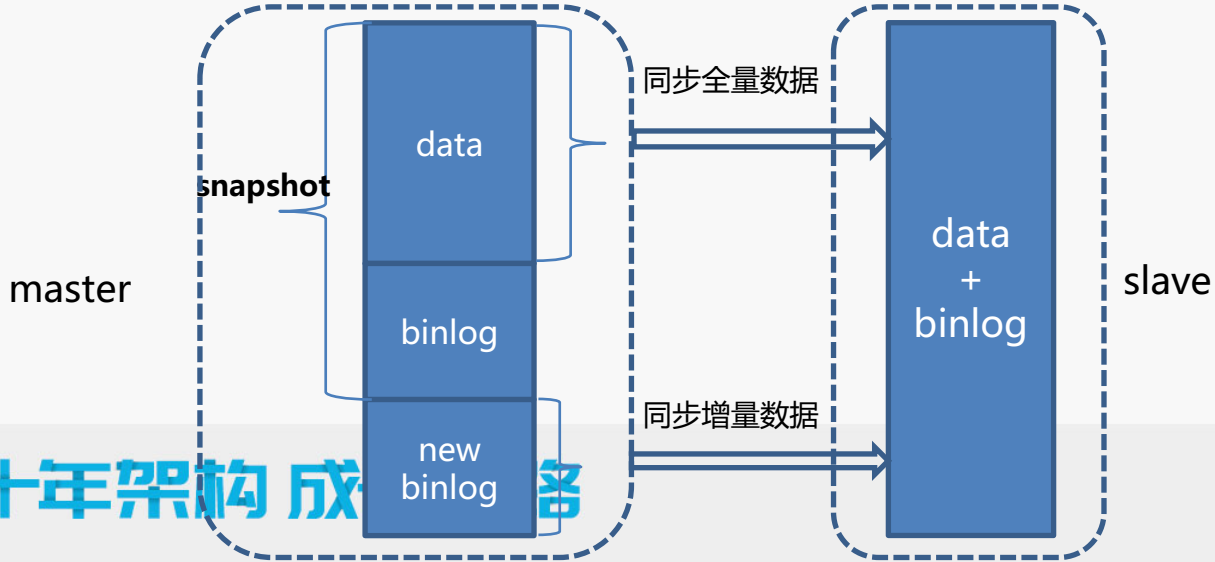
key/value模型表达Redis数据结构的影响

- string和hash没有任何限制，使用方式完全一样
- list退化成一个dequeue
  - 不支持lrem和linsert，新增一个lreplace操作做虚拟删除
- zset的score被限制是整型
  - 为了加速zrank操作，大的skiplist在内存中维护
- 放弃支持集合的交，并，差操作
  - 例如sunion，sinter，sdiff，zinter，zunion，zdiff
- 不同数据类型可以同名
  - set sacc 2018； hset sacc 2018 cool 这两个操作都会成功
- key内元素的聚集，一级key的聚集
  - key内元素的scan操作，lrange，hkeys，hscan…
  - 一级key的聚集

Tendis的设计和实现

主从同步方案

- 数据同步 = 全量同步 + 增量同步
- 全量同步使用RocksDB的snapshot/checkpoint来实现
- 增量同步使用binlog



Tendis的设计和实现

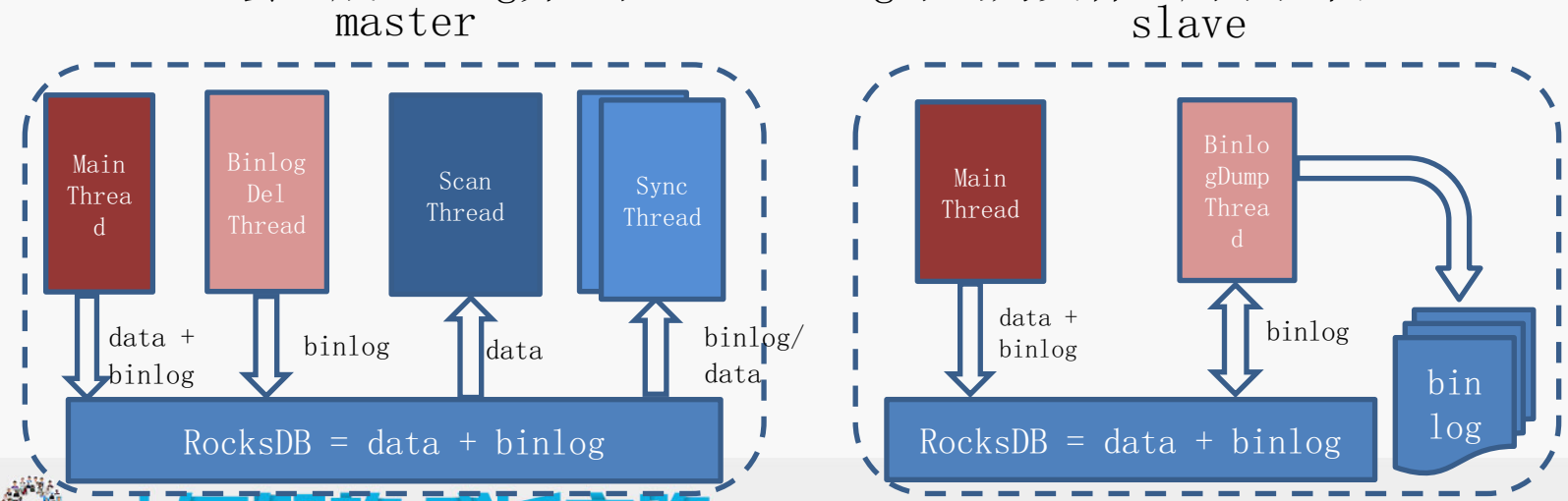
主从同步方案:基于binlog做增量同步

- Tendis在执行具有写语义的命令时往RocksDB中写入binlog
  - binlog以key/value对存储在RocksDB中
  - binlog的key中包含一个递增的序号，binlog按照该序号在RocksDB中有序
  - 写binlog和写数据位于同一个WriteBatch，为一个原子操作
- Tendis只为最基础操作的生成binlog
  - 每种数据结构的写命令归约成几种基础操作，如只有hset/hdel，没有hincrby
  - 每条binlog只记录对底层RocksDB中一个key/value对的修改

Tendis的设计和实现

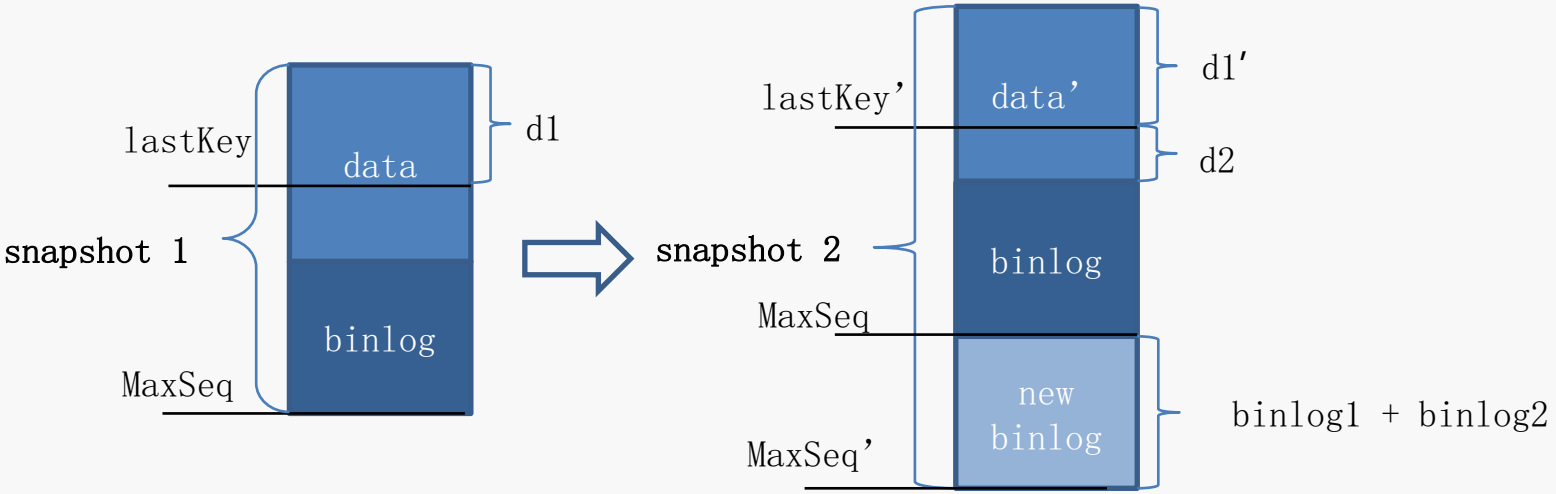
主从同步方案:基于binlog做增量同步

- master把新生成的binlog同步给slave
- master把已同步给slave的binlog删除
- slave上会生成binlog并且把这些binlog导出为文件，用于回档



Tendis的设计和实现

主从同步方案：基于snapshot逻辑全量同步



snapshot 2 = snapshot 1 + binlog[MaxSeq + 1 ... MaxSeq']

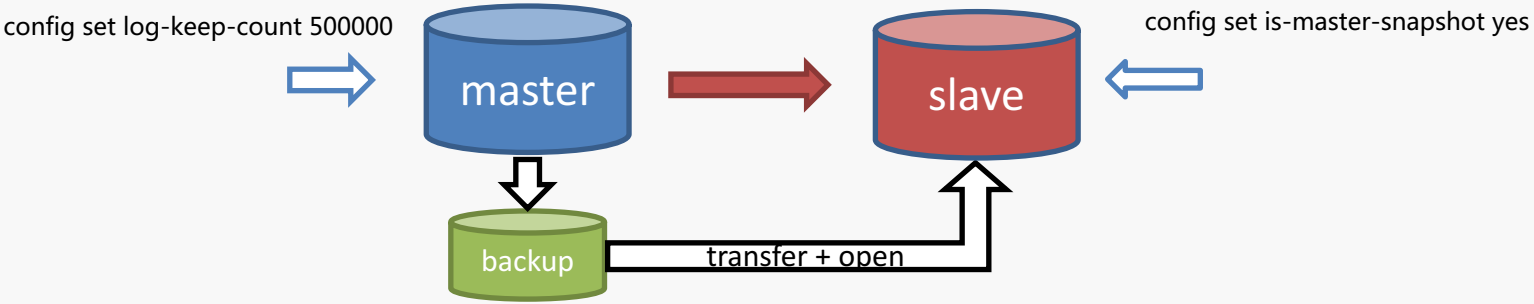
data` = d1' + d2 = d1 + binlog1 + d2

其中，binlog1应用于存放在lastKey以前的KV，binlog2应用于存放在lastKey以后的KV

Tendis的设计和实现

主从同步方案：基于checkpoint的全量同步

- 在线更新master配置，让其保存足够数量的binlog
- 在master上做checkpoint，生成RocksDB实例，将该实例传输到到目标机器
- 在目标实例上打开的RocksDB实例
- 将slave设置同步状态置为binlog跟进方式，从RocksDB中找到最后一条binlog，向master请求其下一条binlog



Tendis性能

Tendis单实例性能

- ./redis-benchmark -p 9988 -n 1000000 -c 24 -t set,get -r 100000000
  - SET: 40278.73 requests per second
  - GET: 44120.89 requests per second

- ./redis-benchmark -p 9988 -n 1000000 -c 24 -r 100000000 lpop list __rand_int__
  - LPOP: 24636.00 requests per second

Tendis性能

Tendis单机性能：单机部署多Tendis实例

搭建Redis集群，测试client、twemproxy及Tendis分别在3台不同机器上，48个client，每个并发20线程测试，twemproxy24实例，Tendis 12实例

- Set稳定性能: **31W QPS** (写入100G数据)

统计项	Client	Twemproxy	Tendis
流量（出/入）	370M/190M	550M/490M	120M/360M
包量（出/入）	33.5w/33.5w	65w/54w	20w/31w
CPU(total)	36%	50%	80%
IO util	0	0	60%

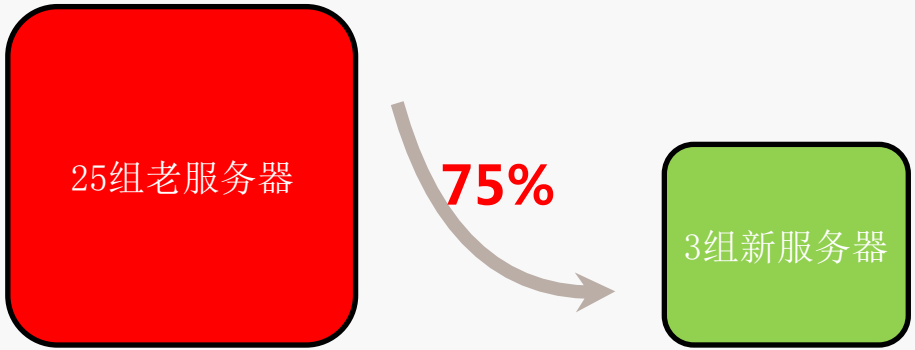
- Get稳定性能：**17W QPS**（100G 数据随机读取）

统计项	Client	Twemproxy	Tendis
流量（出/入）	161M/110M	266M/228M	67.2M/155M
包量（出/入）	18.8w/18.8w	36.8w/29.7w	10.8w/17.8w
CPU(total)	18%	26%	38%
IO util	0	0	91%

Tendis上线收益

Tendis上线收益举例

- 某业务单集群从原生Redis迁到Tendis，成本缩减75%!



Tendis集群方案

单实例部署的问题

- 单实例的性能和存储容量有限
- 管理大量实例，业务配置复杂
- 业务会去实现自己的数据分布策略，和集群部署策略

集群方案的选择

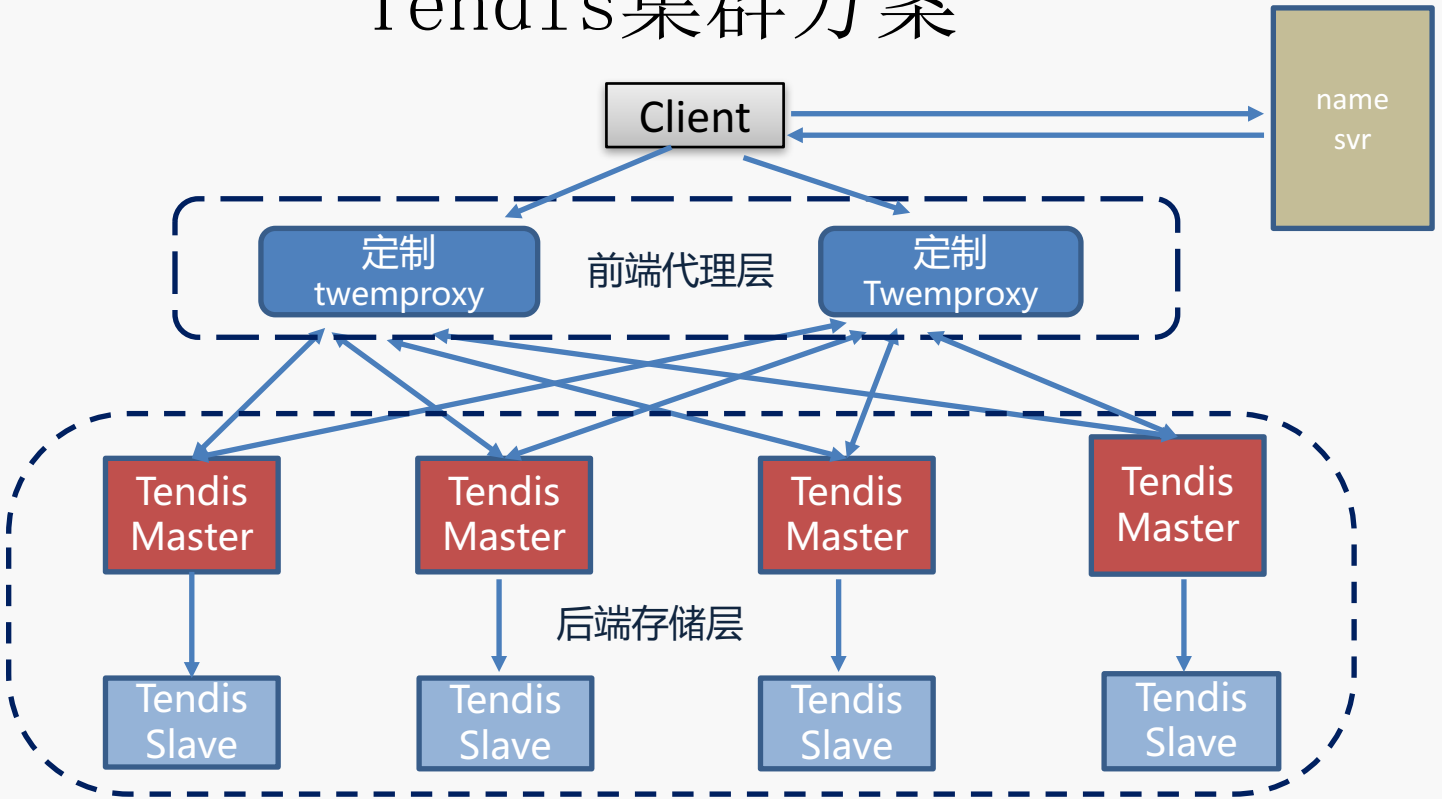
- 引入twemproxy
- 根据业务需求做定制

Tendis集群方案

定制twemproxy

- 新增数据分布方式hash bucket
  - key空间分为若干个bucket
  - 根据key的哈希值计算其所属bucket
- twemproxy一致性切换
  - 不重启进程加载新的配置
  - 不断开客户端连接
  - 可控制新配置生效时机
- Twemproxy功能增强和新命令支持
  - Twemproxy对任意合法格式的resp回包的支持
  - 对Redis中新增命令的支持，如geo
  - 请求延时统计
  - …

Tendis集群方案


Client
name
svr
定制
twemproxy
前端代理层
定制
Twemproxy
Tendis
Master
Tendis
Master
Tendis
Master
Tendis
Master
后端存储层
Tendis
Slave
Tendis
Slave
Tendis
Slave
Tendis
Slave

Tendis集群方案

Tendis集群架构之数据分布

- 整个key空间被划分为若干个bucket
- 每个key根据其hash值决定其所属bucket
- 所有后端存储节点共同负责所有的bucket
  - 每个后端节点负责一个连续的bucket区间
  - 任意两个后端节点负责的bucket区间不允许重叠
  - 所有后端节点负责的bucket区间的并集为全量bucket空间
  - [0, 100) [100, 200)

Tendis集群方案

Tendis集群架构之高可用

- proxy高可用
  - Twemproxy使用docker，至少部署2台，跨campus部署
- 后端高可用
  - 后端Tendis使用一主一备方式，跨campus部署
- 自动failover
  - GCS监控系统对proxy和后端master进行监控
  - 如果proxy不可达，从name svr中踢掉
  - 如果后端master不可达，将slave提升为master

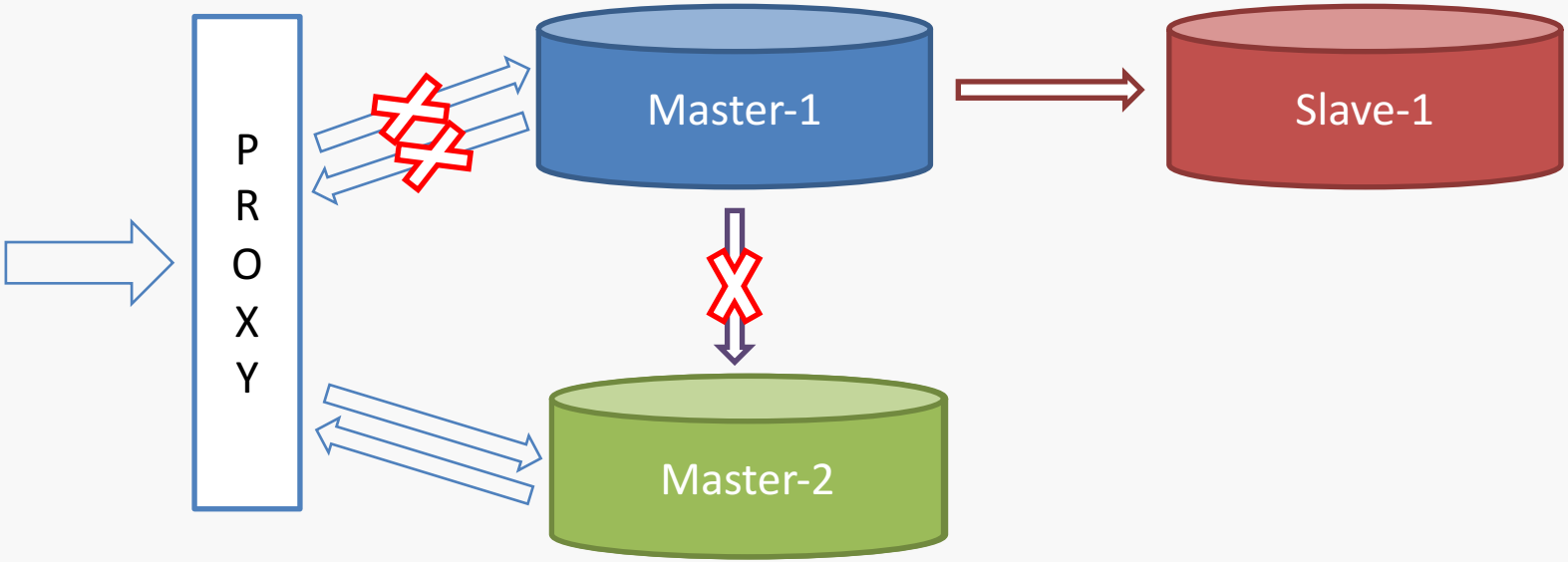
Tendis集群方案

Tendis集群的扩容

- Proxy扩容
  - 新增机器，更新namesvr配置
- 存储扩容
  - 数据搬迁
    - 实例搬迁
    - 实例拆分
  - 路由切换
    - 加载新路由
    - block客户端新请求
    - 使新路由生效

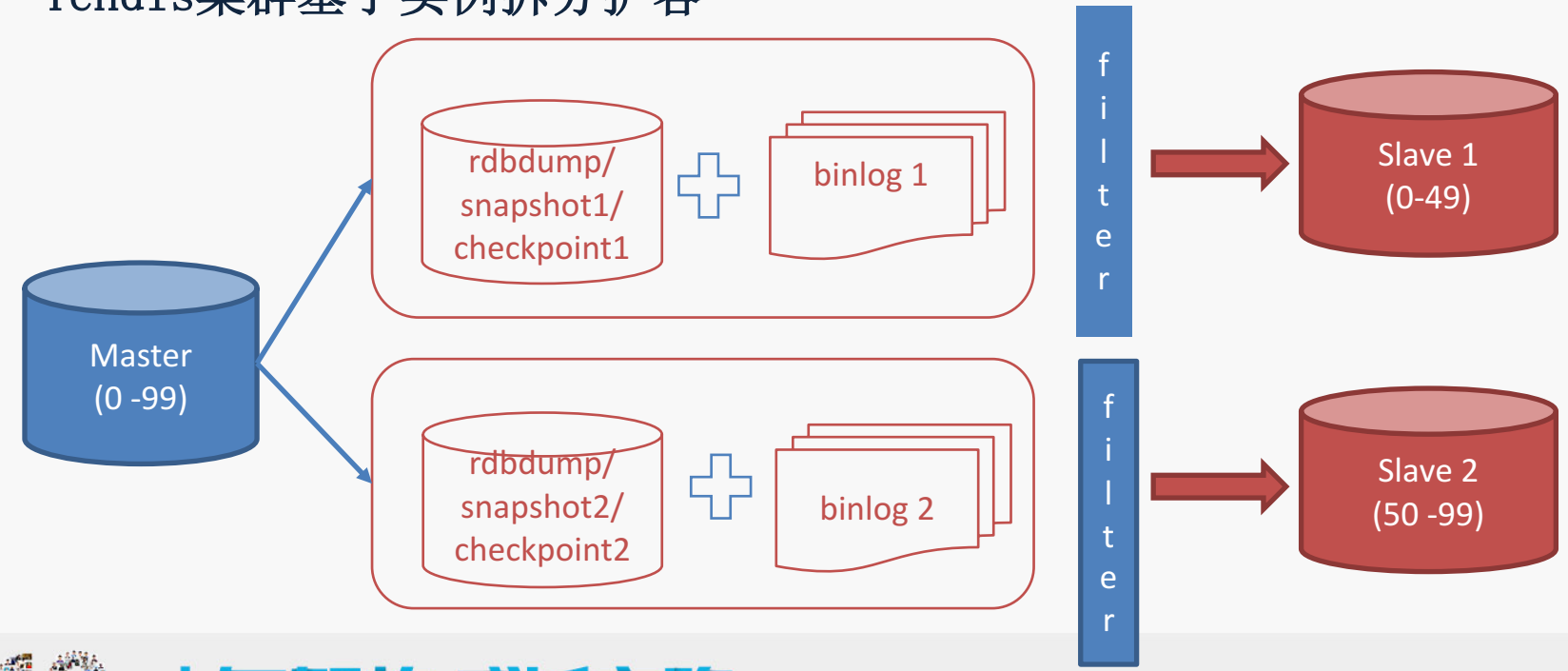
Tendis集群方案

Tendis集群基于实例搬迁扩容



Tendis集群方案

Tendis集群基于实例拆分扩容



Tendis的使用规模

Redis和Tendis在腾讯游戏中得到广泛的使用

- Tendis(Tendis SSD)使用情况
  - 数百集群
  - 上万实例
  - 数百TB数据量
  - 日访问量数千亿
- Redis(Tendis Cache)使用情况
  - 上千集群
  - 上万实例
  - 数十TB数据
  - 日访问量过千亿

广告时间

We are Hiring !

岗位职责:

负责腾讯游戏TenDB数据库(MySQL)的内核开发工作，包括innodb/replication等相关优化;
负责基于TenDB Cluster的分布式事务工作开发，制定适用于业务场景的分布式事务框架;
负责对线上运行的海量数据库实例进行疑难问题定位解决及优化工作;
负责MySQL相关组件和工具的开发工作，包括备份工具、回档工具、数据库中间件等;
负责MySQL数据库在腾讯游戏的最佳实践。

岗位要求:

热爱数据库和存储，以数据库和存储技术作为发展方向，并希望在该领域长期发展下去;
熟悉C/C++、网络，有分布式存储系统开发经验者优先;
熟练掌握Linux下shell、perl或python开发，熟悉go语言开发优先;
熟悉MySQL机制和源码，有MySQL源码开发经验者优先;
熟悉innodb、rocksdb者优先;
在开源社区活跃并有积极贡献者优先。

广告时间

We are Hiring !

附录

Redis数据结构在RocksDB中的表达

set

list

附录

Redis数据结构在RocksDB中的表达

zset

THANKS