DTCC

2016中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2016

数 据 定 义 未 来

SequeMedia 盛拓传媒　ITT168.com　ChinaUnix.net　ITPUB

游戏云存储-TRedis高性能缓存及持久化

腾讯游戏/康中良/DBA
Email: zhongliangkang@qq.com

提要

- Redis@腾讯游戏
- TRedis简介
- TRedis特点及设计
- TRedis集群实现
- TRedis在腾讯游戏应用实践

Redis is an open source, BSD licensed, advanced **key-value cache** and **store**. It is often referred to as a **data structure server** since keys can contain **strings, hashes, lists, sets, sorted sets, bitmaps** and **hyperloglogs**.

— http://redis.io

Redis特点

❖ 高性能	get/set 10w+
❖ 单线程	单线程：收包、发包、解析
❖ K-V存储	所有数据按“key”访问
❖ 纯内存+持久化（V2.8）	数据全内存，高性能，RDB/AOF落地
❖ 支持多种数据结构	strings,hashes,lists,sets,sorted sets
❖ 协议简单	各种语言的api支持
❖ 生产者/消费者消息队列	高性能，异步处理请求队列
❖ 过期时间	到期数据自动删除

Redis@腾讯游戏

- **string**： openid映射/用户信息等
- **hash**： 角色信息/装备道具等
- **list**： 消息队列等
- **set**： 属性记录/资格等
- **sorted set**： 全局排行/消息评论等
- **TTL**： 活动礼包、Cache过期等

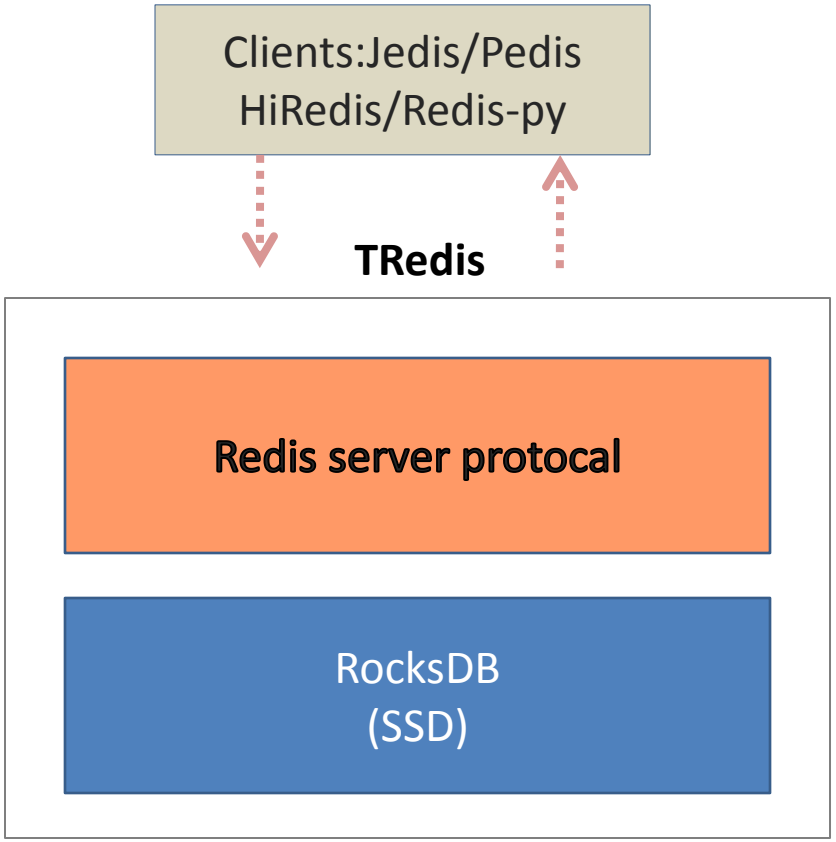
Redis@腾讯游戏

场景：手游(代理)、官网活动、平台业务

- 实例总数： **3500+**
- 内存总量： **5T+**
- 请求峰值： **300w+**
- 日请求量： **1500亿+**

TRedis介绍

TRedis是腾讯DBA基于官方2.8.17定制，采用Rocksdb作为底层存储引擎，完全兼容Redis协议，支持分片管理和迁移及增量同步的高性能Redis版本。



TRedis特点

- “RocksDB” 作为存储引擎，充分利用Rocksdb特性
- “高性能” 单实例set/get QPS 4W+ *(SSD)*
- “协议兼容” 支持大部分Redis指令,应用透明
- “省内存” string/list/hash存储零内存*(no set/zset?)*
- “数据安全” 数据落地到磁盘才返回，安全

TRedis特点

- **“大量存储”** 单实例可以提供亿级数据存储访问
- **“增量复制”** slave同步断重连后，可继续同步
- **“分片管理”** 集群功能支持，在线迁移及管理
- **“数据过期”** 数据过期自动清理
- **“启动快”** 没有Load RDB/AOF的时间
- **“物理备份”** 基于文件copy备份Redis数据，快

Rocksdb

- RocksDB是Facebook基于Leveldb改进，能充分利用多核CPU并提供系列的扩展特性
- LevelDB是google开源的高性能KV数据库，基于文件系统，支持海量存储，配合SSD能提供高性能的随机读写
- RocksDB的Key有序存储，将Redis的结构化数据以KV的形式落地到RocksDB中

http://rocksdb.org

TRedis性能

基于SSD，单实例

- ./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

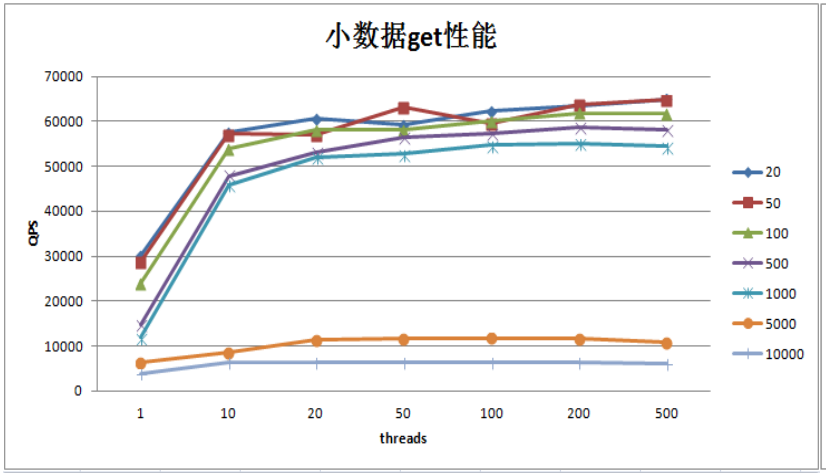
TRedis性能

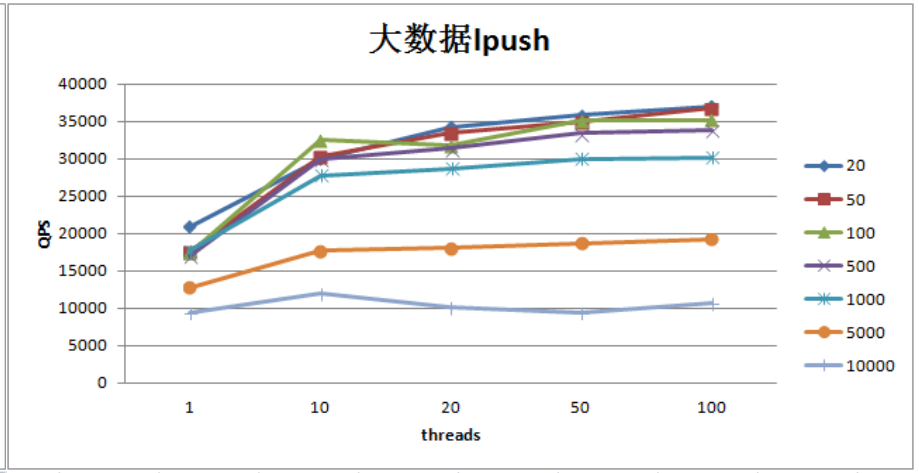
- ./redis-benchmark -p 9988 -n 1000000 -c 24 -r 100000000 rpush TRedis_list __rand_int__
- **RPUSH: 33705.21** requests per second

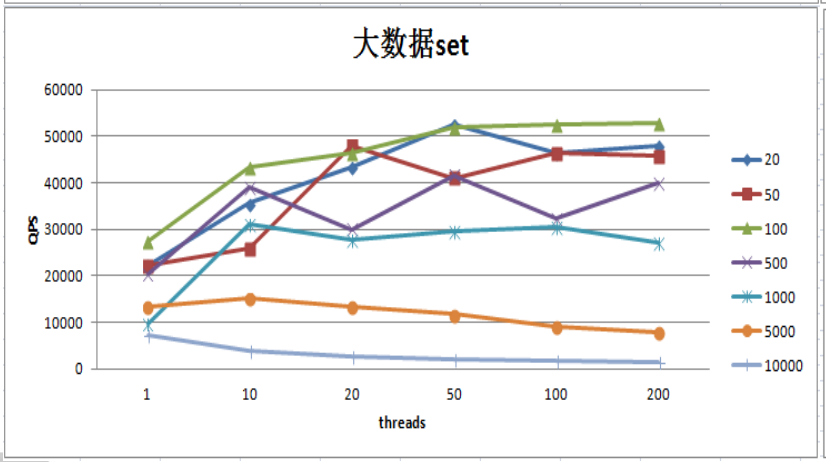
- ./redis-benchmark -p 9988 -n 1000000 -c 24 -r 100000000 lpop TRedis_list
- **LPOP: 24636.00** requests per second

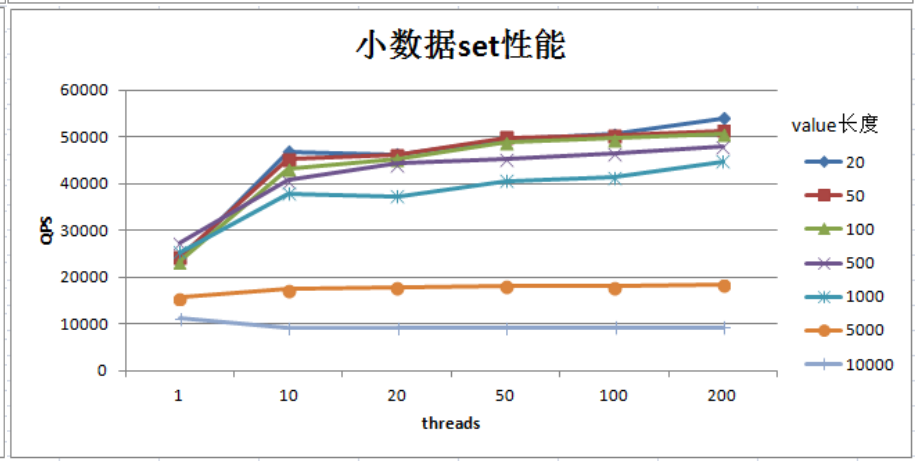
TRedis性能

基于FIO测试









TRedis性能-单机

搭建Redis集群，测试client、twemproxy及TRedis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，TRedis 12实例。

Set稳定性能：**31W QPS** (写入100G数据)

统计项	Client	Twemproxy	**TRedis**
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	**TRedis**
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%

TRedis设计-存储实现

- 每个key在RocksDB中均以k-v存储

  [DBID][Type]key -> [EncodingType][TTL]value
- 除string外的数据结构，用额外的数据来存储
- 所有key在Rocksdb中按字典序存储
- 支持数据过期TTL
- 不同指令，性能不同

TRedis设计-存储实现

- 各种结构存储格式-KEY

Redis结构	RocksDB结构	RocksDB子元素
String	[ID]a[key]	无　　*ID为DB index id, 下同
List	[ID]L[key]	[ID]l[keylen][key][seq]
Hash	[ID]H[key]	[ID]h[keylen][key][hashkey]
set	[ID]S[key]	[ID]s[keylen][key][setkey]
Zset	[ID]Z[key]	[ID]z[keylen][key][zsetkey] [ID]c[keylen][key]['>'|'<'][score][zsetkey] *大于表示正，小于表示负，score编码

String实现

- String key: [DBID]a[key]
- String value: [EncodingType][TTL][value]

- 例：
  - set dtcc 2016 EX 9999

DBID	Type	Key		EcdType	TTL(ms)	value
4byte	1byte	N-bytes	→	1byte	8bytes	String/int
0	a	dtcc		1	9999000	2016

List实现

- List key: [DBID]L[key]
- List item key:[DBID]l[keylen][key][seq]

- 设计：
  - list包含顺序，因此在落地时以序列(seq)来标识
  - 每个list会记录一个头、尾序列(seq)，范围为64位整形
  - 每push/pop一个key,都会对该list的头尾做调整，同时key的seq也会固定。所有list的元素的seq必须连续。
  - List的操作保证item写入之后，内部seq就固定不变,且有序

List	TAIL					HEAD
seq	-999	-998	……	998	999	1000

Hash实现

- Hash key:[DBID]H[key]
- Hash item key:[DBID]h[keylen][key][hashkey]

- 设计：
  - 根据hash的特点，按照上面的key名的设计即可落地
  - 单独存储hash的key数量
  - Set结构与hash类似，可以把set看作没有value的hash

Zset实现

- Zset key:[DBID]Z[key]
- Zset item key:
  - [DBID]z[keylen][key][zsetkey]
  - [DBID]c[keylen][key]['>'|'<'][score][zsetkey]

    >表示正，<表示负，score bigendian编码
- 设计：需要存储3类信息：
  - Zset的元素的k-score对应关系，按zsetkey字典排序
  - 按score排序的对应关系，score-v映射
  - score只支持整数型
  - Zset的元素数量

TRedis使用约束限制

- 所有数据实时落地磁盘，string/list/hash只存储在磁盘上，set/zset会驻内存
- TRedis允许string/list/hash/set(zset) 使用相同的key名
- 指令行为改变：
  - Dbsize 近似计算，最近一次扫描的结果
  - Flushdb 只清理内存中的数据，数据还在磁盘上，只是从内存中删掉，磁盘清理用新命令取代
  - Randomkey 从前1024个key中随机返回一个key
  - Keys keys将返回前1024 个key
  - Type/ttl 由于TRedis支持不同数据类型使用相同的key，因此type/ttl返回其中一个类型的结果(第一个）
  - Expire/del/persist 对所有该key名字的数据类型执行相同的操作。
- 单个key大小不能超过内存设置大小 (与官方相同，但我们内存设置更小)
- Sorted set的权值不支持浮点数，浮点数在落地时会转换为整数
- List长度限制为： 9223372036854774783 ,总长度(正常场景够用了)
- 大key的delete会block住其他操作,慎用（与磁盘删除效率为10w/s k-v）

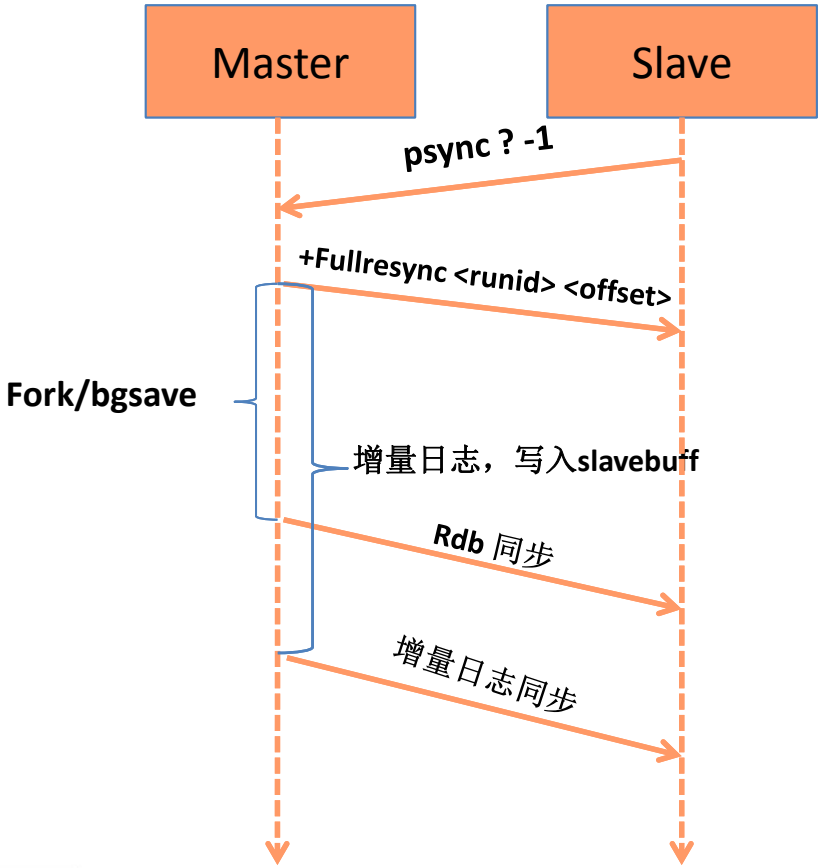
TRedis指令支持情况

- 除了以下指令，其他都支持

与redis差异指令	指令说明
KEYS	只返回前1024个keys
MIGRATE, MOVE	不支持
OBJECT	只支持set/zset
RANDOMKEY	取前1024个key随机
RENAME, RENAMENX	暂不支持
SCAN	只取内存数据
SORT	不支持list
TYPE， TTL, PERSIST, DEL, EXPIRE	行为改变
HSCAN	不支持
LINSERT, LREM	不支持
DBSIZE	后台异步计算dbsize，非实时
DEBUG OBJECT	只支持set/zset
DEBUG SEGFAULT	只支持set/zset
FLUSHALL, FLUSHDB	只清空redis内存，被新指令替代
PFADD, PFCOUNT, PFMERGE	不支持
SAVE	不支持, save阻塞时间过长，禁用

TRedis设计-增量同步

官方Redis复制流程



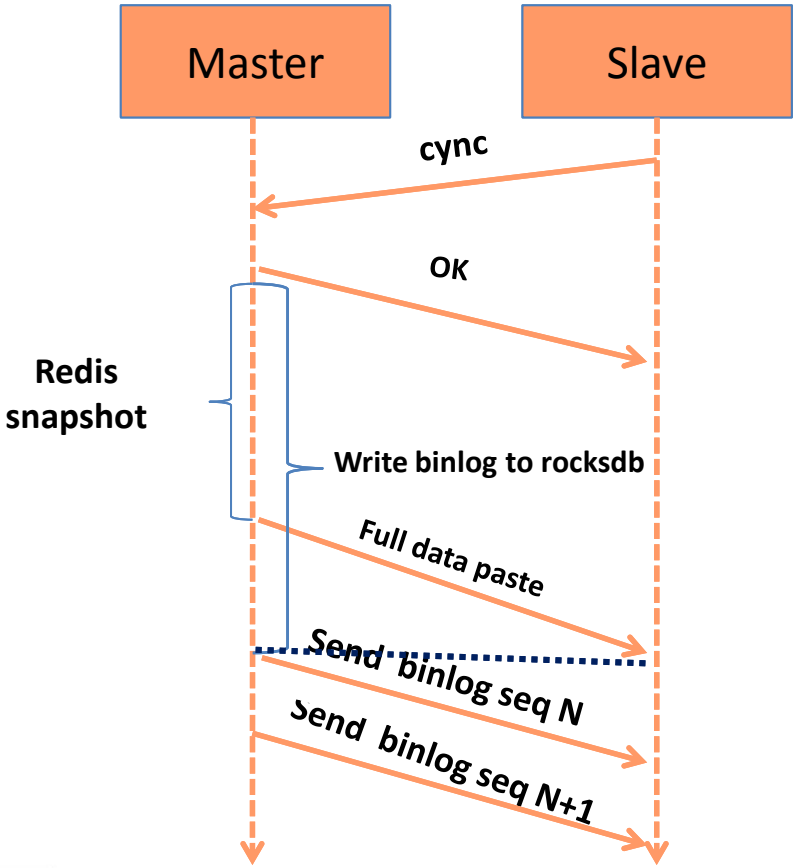
存在的问题：

1.主从同步断开后，slave基本都需重新同步

2.单机多实例的情况下，如果网络闪断，全部slave断开，全部同时重新同步，master压力巨大（IO/内存/CPU）

3.全部重做时，master负载很大，重做时间可能很长，甚至可能重复失败，导致master崩溃

TRedis设计-增量同步

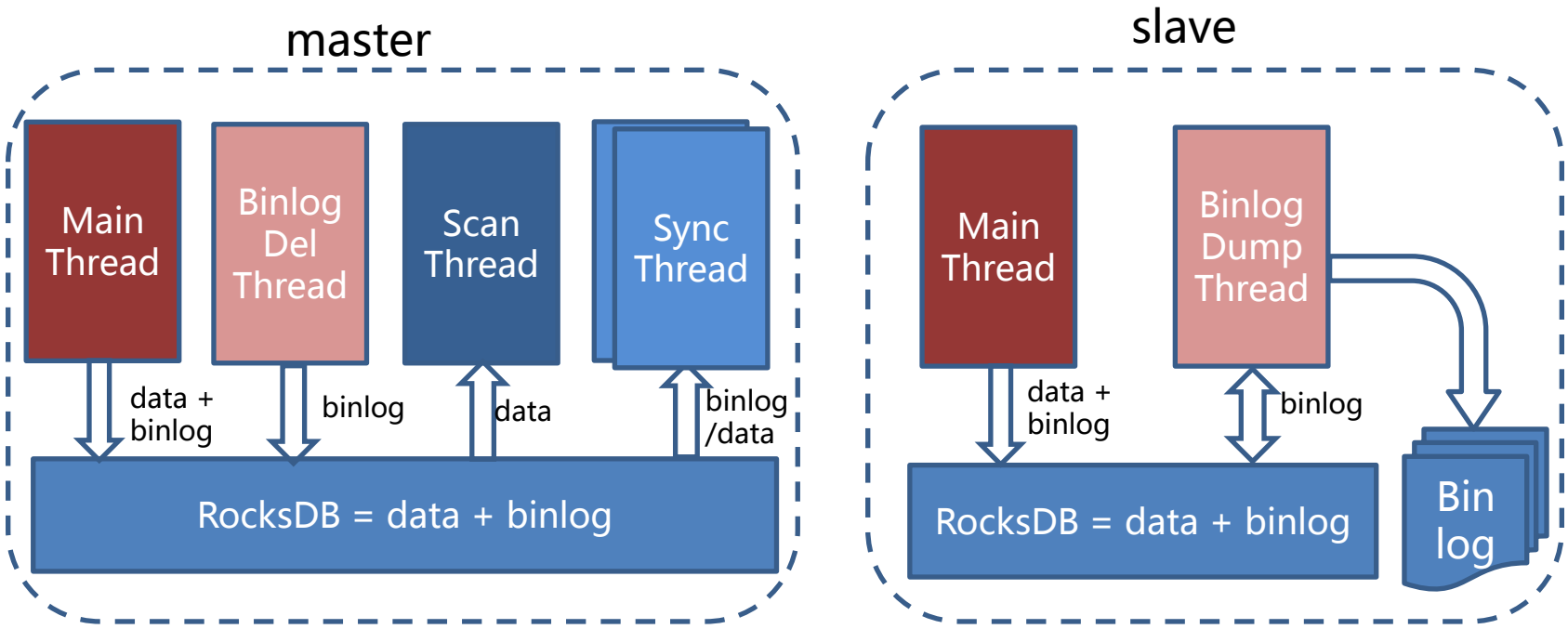


新同步方式：

1. 所有Redis操作记录操作binlog，与数据一起写入Rocksdb，每个binlog操作记录一个递增的seq序号
2. 首次同步，slave发送cync命令到master，master发送数据快照到slave，并将同步快照最新的binlog位置发送给slave
3. Master将快照之后产生的binlog按序号发送给slave
4. Slave断开之后，master会保留slave最后同步的seq之后的binlog
5. Slave断开重连时，会将最后一次同步的seq位置+1，发送给master进行增量同步
6. Master校验slave信息，并根据slave请求的位置，继续同步

TRedis设计-线程管理

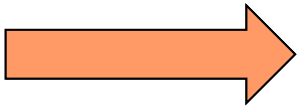
TRedis主线程仍是单线程，后台多线程



TRedis集群介绍

Redis/TRedis实际运营中遇到的困难:

- 单实例性能有限，容量不易扩展
- 大量实例管理，业务配置复杂
- 业务需要自己实现Redis分片，扩缩容困难



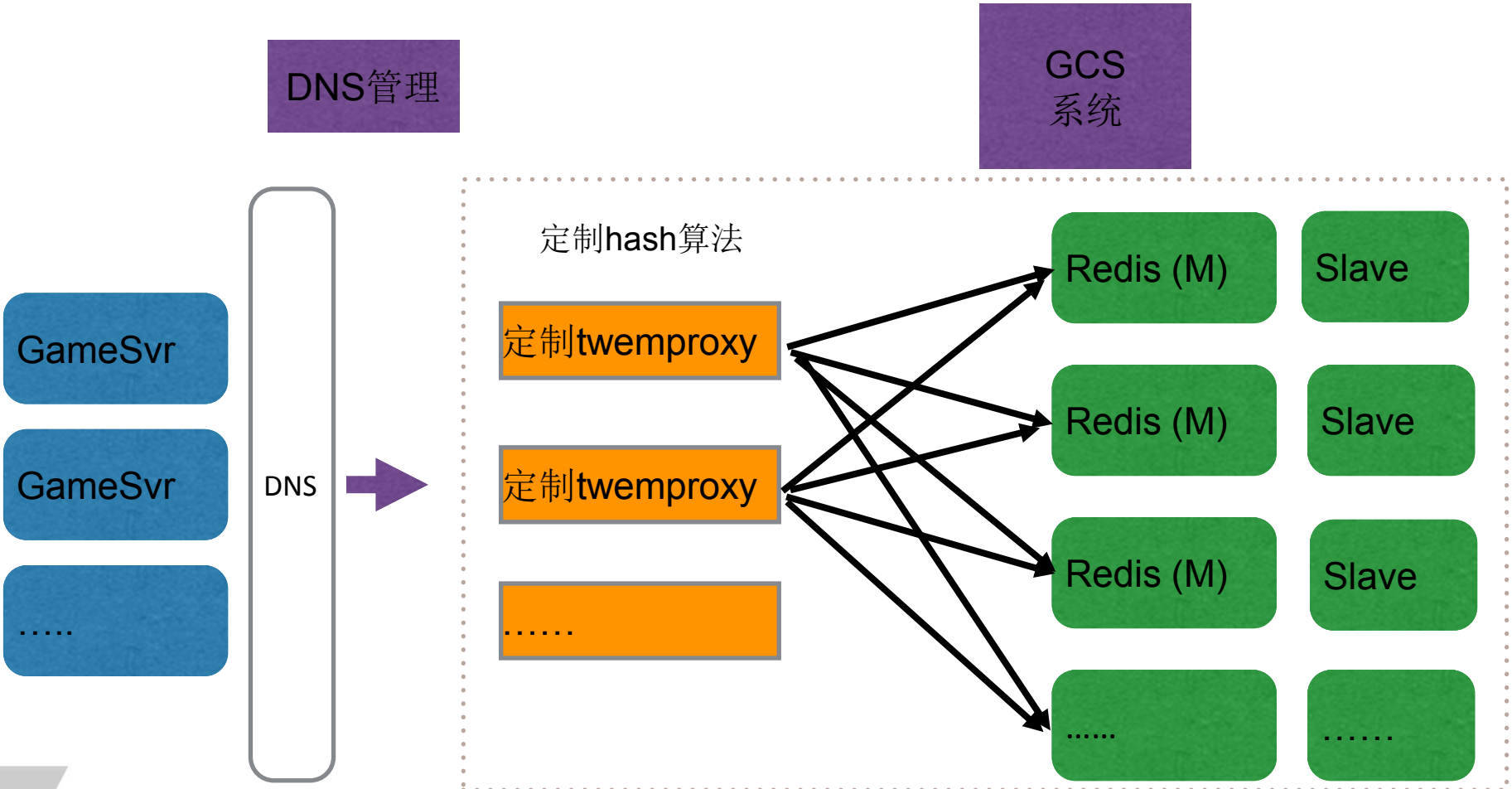
TRedis集群

集群设计

- Tredis集群目标:
  - 容量易于伸缩
  - 管理方便
  - 扩缩容业务无感知

- 方案实现:
  - 引入Twemproxy，定制改造
  - Redis分片管理支持，细粒度管理数据
  - 故障切换支持
  - 增删节点支持

集群设计


DNS管理
GCS
系统
GameSvr
GameSvr
.....
DNS
定制hash算法
定制twemproxy
定制twemproxy
.....
Redis (M)
Redis (M)
Redis (M)
.....
Slave
Slave
Slave
.....

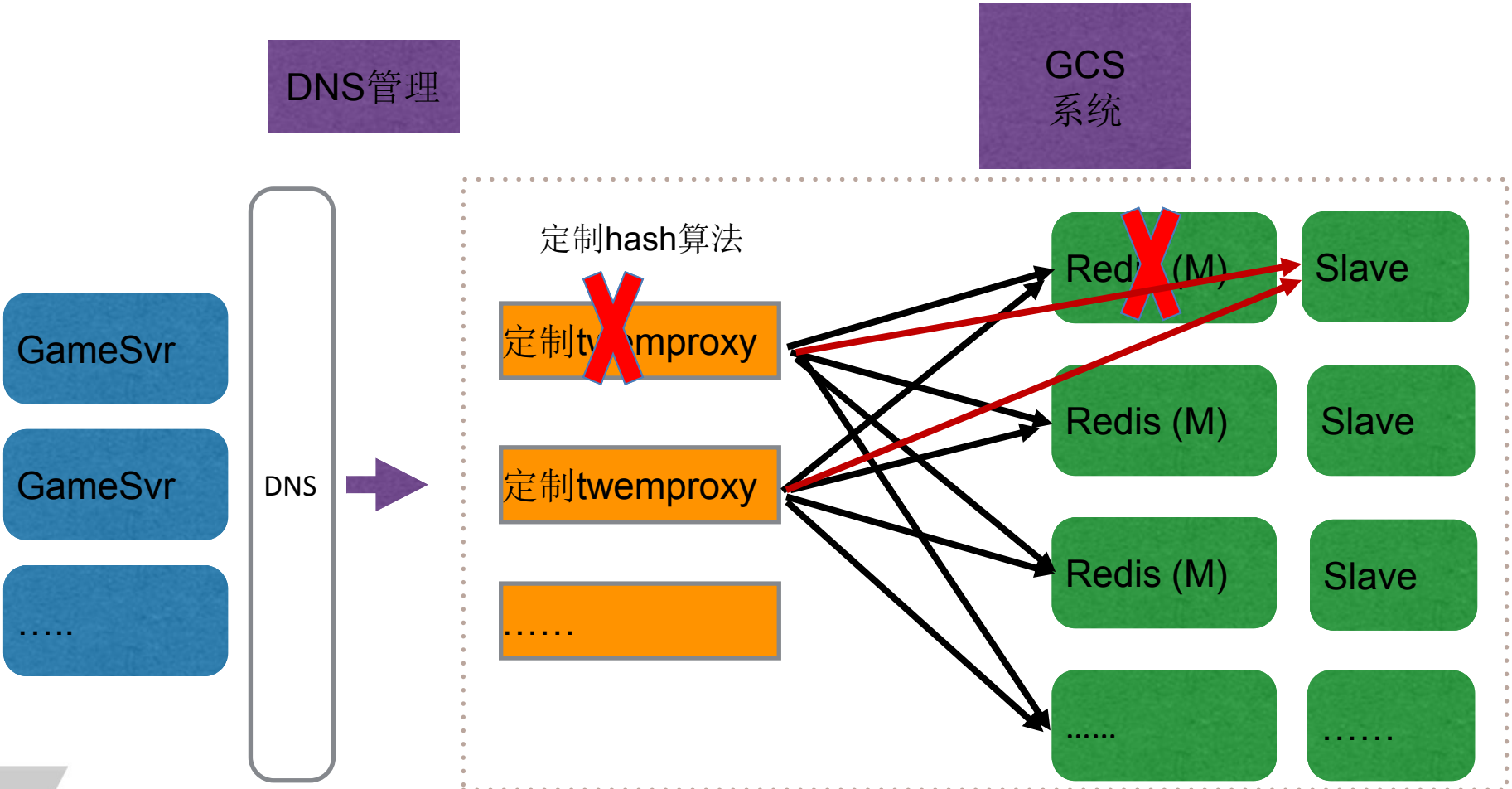
TRedis集群定制

- TRedis支持分片管理，key状态管理
  - 42w分片
- Twemproxy支持自定义分片算法
- Twemproxy支持Redis节点切换
  - change name ip1:port1 ip2:port2
- Twemproxy支持增加Redis节点
  - add name ip:port seg_start seg_end

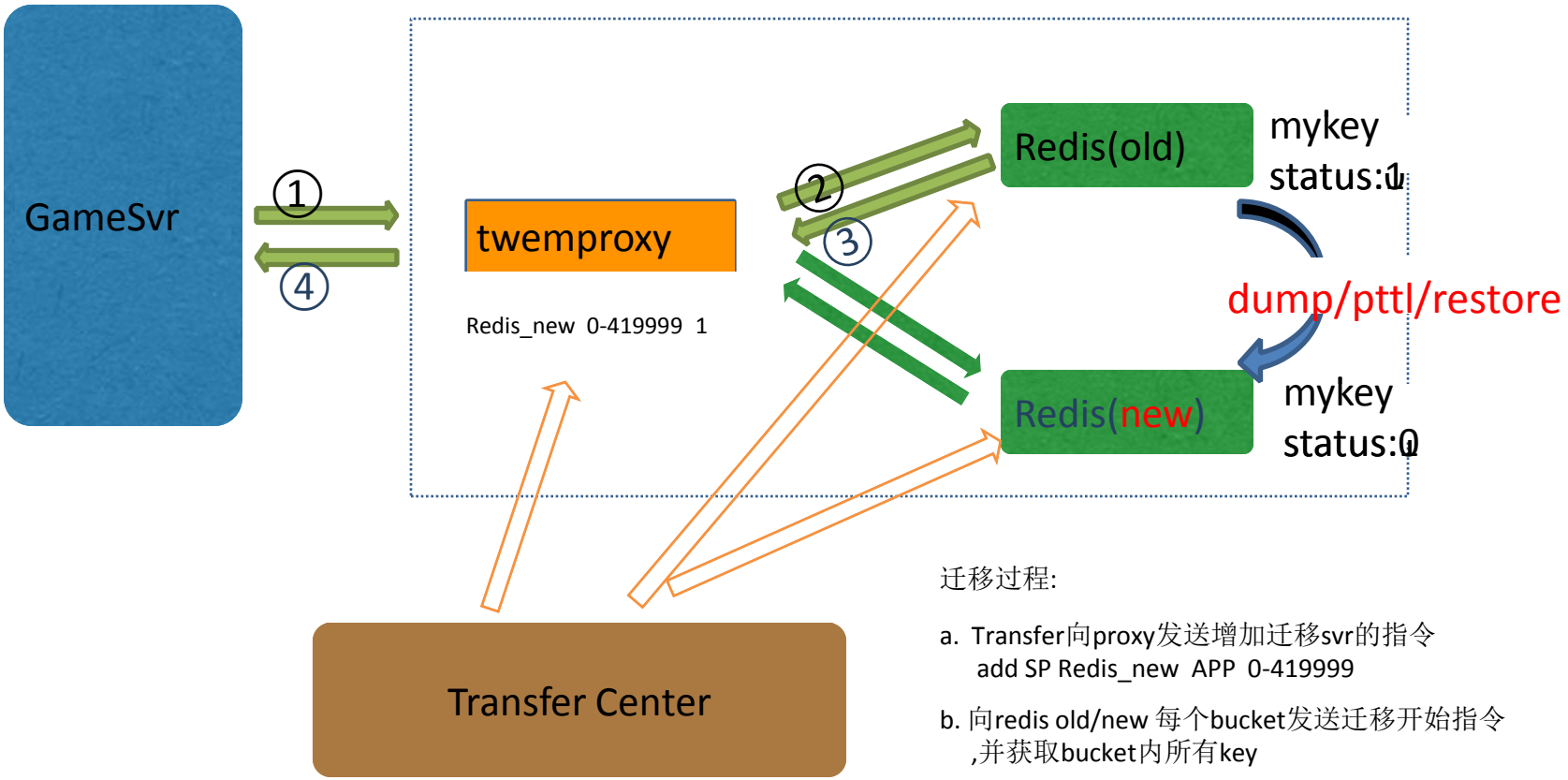
TRedis集群-高可用解决

- Redis集群通过DNS+port访问，透明
- twemproxy故障,直接剔除
  - GCS系统会将twemproxy从DNS中剔除掉故障IP,业务访问重试即可恢复.*（需重连）*
- Redis实例故障，切slave
  - GCS系统会将所有访问该redis节点的twemproxy全部指向redis的slave.
  - 切换操作预计 1分钟左右完成.

TRedis集群-高可用解决



集群-在线迁移流程



迁移过程:

a. Transfer向proxy发送增加迁移svr的指令
add SP Redis_new APP 0-419999

b. 向redis old/new 每个bucket发送迁移开始指令,并获取bucket内所有key

c. 在old/new上锁定一个key,开始搬迁

d. 搬迁key(dump/pttl/restore)

e. 释放new上的key锁定,删除old key

f. Bucket迁移完成

g. 迁移完成,发送adddone命令到proxy

Redis/TRedis遇到过的一些问题

- 单key很大
  - 8kw个item的hash/1.1亿的list
  - 4.5G内存的单key
- 单实例很大
  - 一个redis实例50G内存
- 主从同步在业务峰值时失败
- AOF rewrite失败不断重写

TRedis在腾讯游戏的应用实践

- 数据量、业务量
  - 实例数 **2k+**，数据量 **34T**
- 集群情况
  - 16个集群
  - 单集群最大**480**个节点，峰值可以提供150Wqps
- 后续发展方向
  - 游戏核心数据Cache化
  - 用户画像数据存储

TRedis未来

- 支撑工具建设
  - 完善Tredis相关支撑平台及工具建设
- 定制&优化
  - 贴近业务定制及优化
- 开源
  - 将Tredis及相关组件开源，与开源社区共同成长

腾讯智营 Intelligent Operation

智能，从懂你开始！

- 为游戏全方位护航
- 智能服务，智慧运营
- 懂你，更懂用户

运维智能决策

蓝鲸，是一套由腾讯游戏运营部沉淀多年，并承载着数百款业务支撑运营系统的基于PaaS的技术解决方案；它提供了完善的前后台开发框架、调度引擎、公共组件等模块，帮助业务的产品和技术人员得以快速、低成本、免运维的构建支撑工具和运营系统。

GCS，是互娱运营部沉淀多年的贴合游戏生命周期的数据存储及“一站式”管理平台，它提供了数据缓存层及持久化层的技术解决方案，帮助业务快速接入的同时降低存储成本，减少硬件故障及常规DB维护导致的停机时间，助力公司游戏业务长期发展。

蓝鲸

游戏云储存平台 Game Cloud Storage

- 数据分析挖掘
- 营销开发闭环
- 用户分层管理
- 运营渠道延伸

腾讯游戏在游戏运营领域具备十余年的经验，在游戏运营的中后端，已形成智能化、多维度、可信赖的一站式运营服务体系，包括运维智能决策、数据分析挖掘、营销开发闭环、用户分层管理、运营渠道延伸等。

THANKS

SequeMedia 盛拓传媒 IT168.com ChinaUnix ITPUB