Redis在唯品会的应用实践

——架构演进与功能定制

申政
高级数据库工程师

主办单位：杭州 Alibaba Group 阿里巴巴集团
战略合作伙伴：intel

目 录
content

- Redis集群架构演进
- 一些经验
- 二次开发
- Q&A

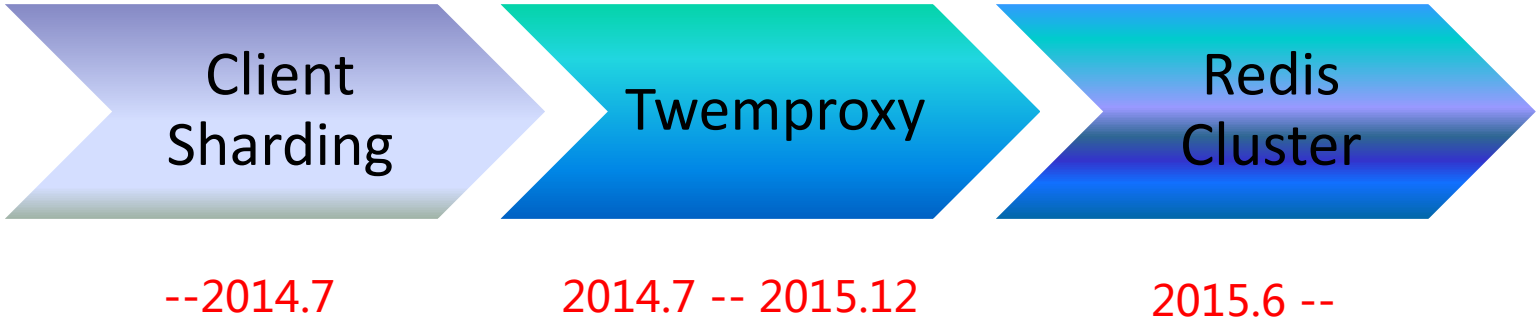
扫码观看大会视频

Redis 使用情况

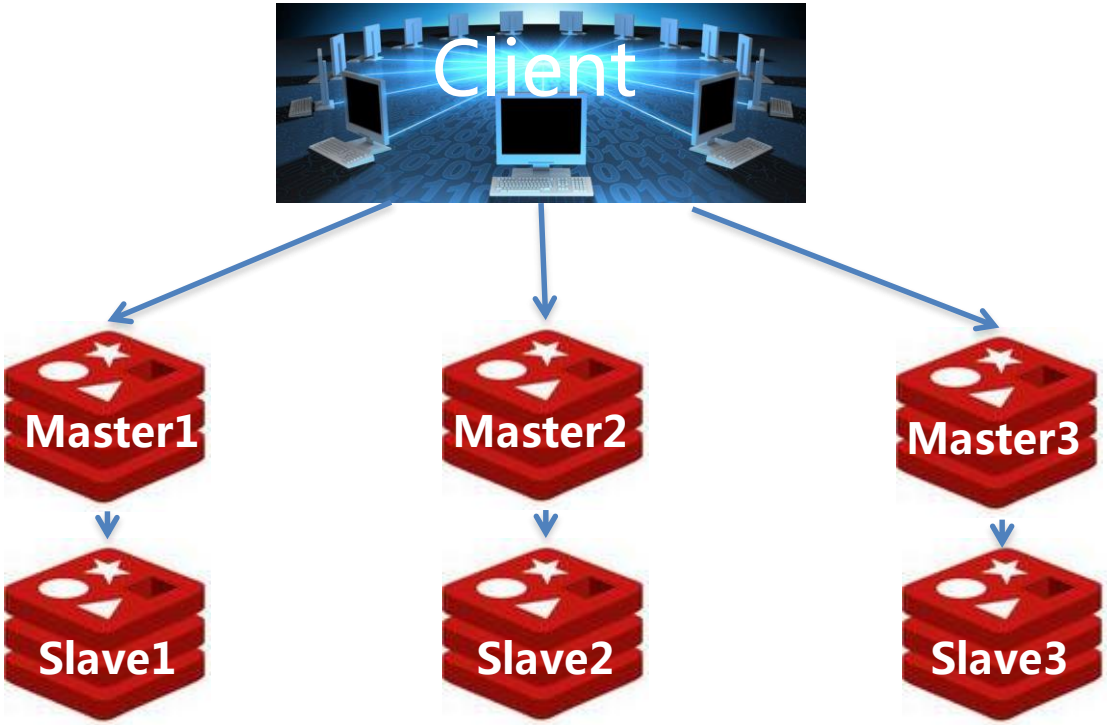
- 8000个实例
- 1000台物理机
- 500个应用

一、Redis集群架构演进

演进历史



Client Sharding



- 自定义key分布
- 增加业务开发难度
- 无在线扩容能力
- failover困难

Twemproxy

Lvs keepalive Lvs

Twempoxy Twempoxy

Master1 Master2 Master3

Slave1 Slave2 Slave3

HA

Sentinel1

Sentinel2

Twemproxy 优点

- 数据自分片，一致性hash
- 降低业务开发复杂度
- 支持redis/mc协议
- 支持pipeline
- 支持mget/mset操作
- 自身扩容简单

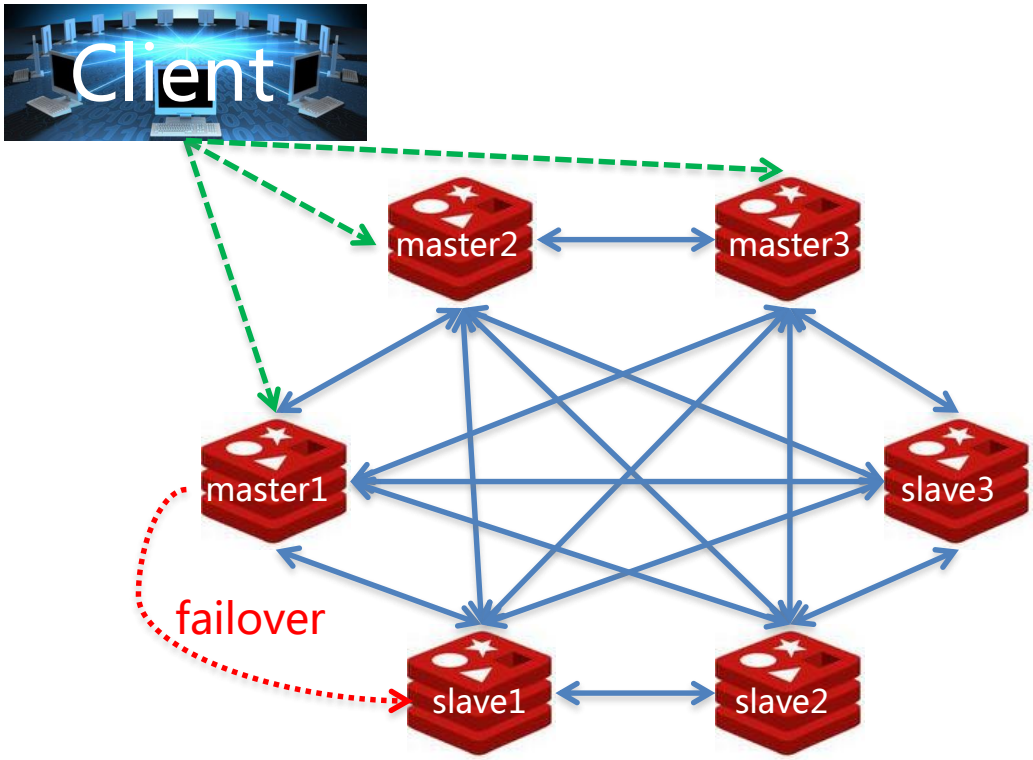
Twemproxy 缺点

- 架构复杂，机器成本高
- redis/mc扩容难
- 每一个环节都可能成为瓶颈
- 请求链路长，响应时间长
- 运维成本高

Twemproxy 使用情况

- 2000个实例
- 800台物理机
- 150个应用

Redis Cluster



- 无中心框架
- 在线扩容缩容
- 自动failover
- 相对Twemproxy架构，节省一半的机器
- 单层框架，响应时间短
- 需要智能客户端支持
- 对mget/mset支持不友好
- 对pipeline支持不友好

扫码观看大会视频

二、一些经验

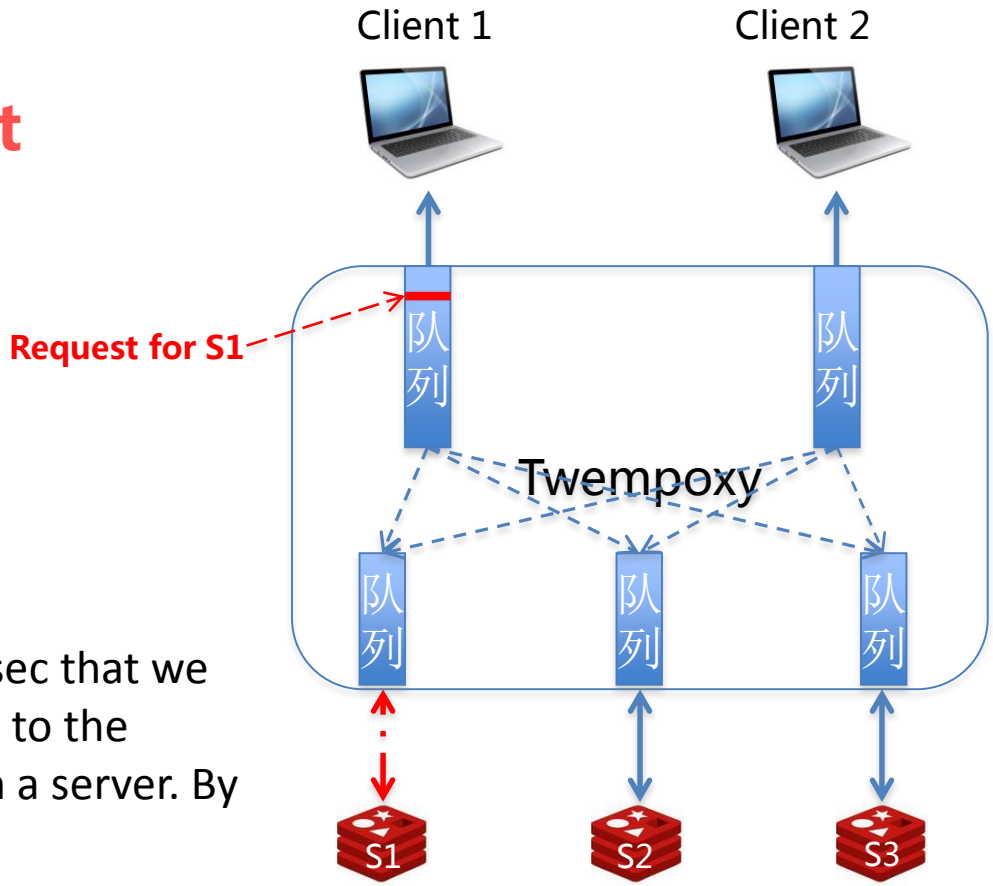
Twemproxy Pipeline

- 一次pipeline过大，内存暴增
- 一次性接收完所有数据，再转发
- 内存只增不减

一次pipeline包含的命令数量适中，最好不要超过1000

Twemproxy Timeout

- 卡住所有请求
- 内存飙升



timeout: The timeout value in msec that we wait for to establish a connection to the server or receive a response from a server. By default, we wait indefinitely.

Twemproxy 剔除

- redis/mc没有挂，却发生了eject
- server_connections与server_failure_limit

server_connections < server_failure_limit

server_connections: The maximum number of connections that can be opened to each server. By default, we open at most 1 server connection.

server_failure_limit: The number of consecutive failures on a server that would lead to it being temporarily ejected when auto_eject_host is set to true. Defaults to 2

Twemproxy 连接不释放

- LVS + Twemproxy + Redis/Mc
- Twemproxy端连接数远大于Client端
- LVS的expire机制，15min

增加tcpkeepalive功能

Twemproxy 内存泄漏

- mset命令引起
- *https://github.com/twitter/twemproxy/pull/486*

扫码观看大会视频

Redis Cluster

- cluster-require-full-coverage 建议 no
- cluster-node-timeout 建议 适当增大
- jedisCluster 注意捕获异常 MaxRedirectionsException
- 不建议使用mget/mset等multi-key命令
- 尽量使用官方redis-trib脚本管理集群，不要轻易使用Cluster命令

三、二次开发

Twemproxy 高可用

- 增加replace_server命令
- 当redis master挂了后，sentinel调用脚本通过replace_server命令把Twemproxy中的master替换成slave

Twemproxy 配置中心

- 集成zookeeper
- Twemproxy集群的配置信息统一由zookeeper管理，分发

Twemproxy Mc

```
master_pool:
  listen: 127.0.0.1:22122
  redis: false
  replication_mode: 1
  write_back_mode: 1
  penetrate_mode: 1
  servers:
  - 127.0.0.1:23401:1 master1
  - 127.0.0.1:23401:1 master2
  - 127.0.0.1:23401:1 master3

slave_pool:
  listen: 127.0.0.1:22123
  redis: false
  replication from: master pool
  servers:
  - 127.0.0.1:23402:1 slave1
  - 127.0.0.1:23402:1 slave2
  - 127.0.0.1:23402:1 slave3
  - 127.0.0.1:23402:1 slave4
```

增加Replication pool概念

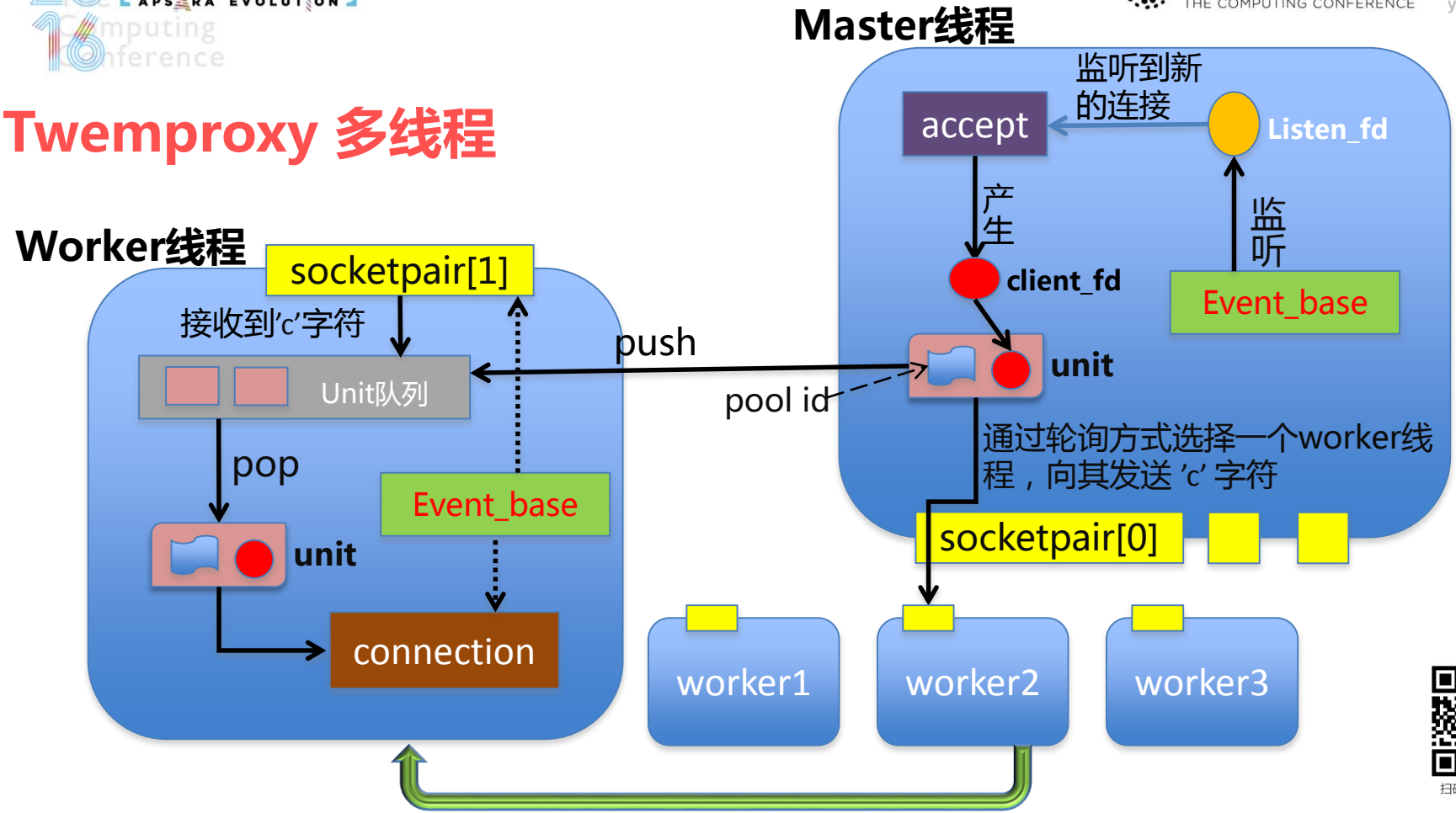
Twemproxy Slowlog

- 借鉴redis，增加slowlog功能，停留时间
- 增加简单统计信息，一天分成几个时间段，每个时间段记录slowlog次数

Twemproxy 多线程

- 借鉴mc多线程模型 master线程+N个worker线程
- master线程负责监听端口，接收新的客户端连接
- worker线程负责处理客户端请求
- 开6个线程，qps可以达到35万/s
- 节约千万服务器成本

Twemproxy 多线程



Redis Cluster Failover

- flushall删不掉数据
- master存活，网络也正常，但是Cluster却执行了failover
- Redis单线程，在执行时间复杂度为O(n)命令时，线程被阻塞，无法响应其他节点的ping命令，造成假死现象，主从切换
- 增加额外线程，extra-port
- 超过cluster-node-timeout/2时间，ping请求没有收到回复，主动向extra-port发送ping请求

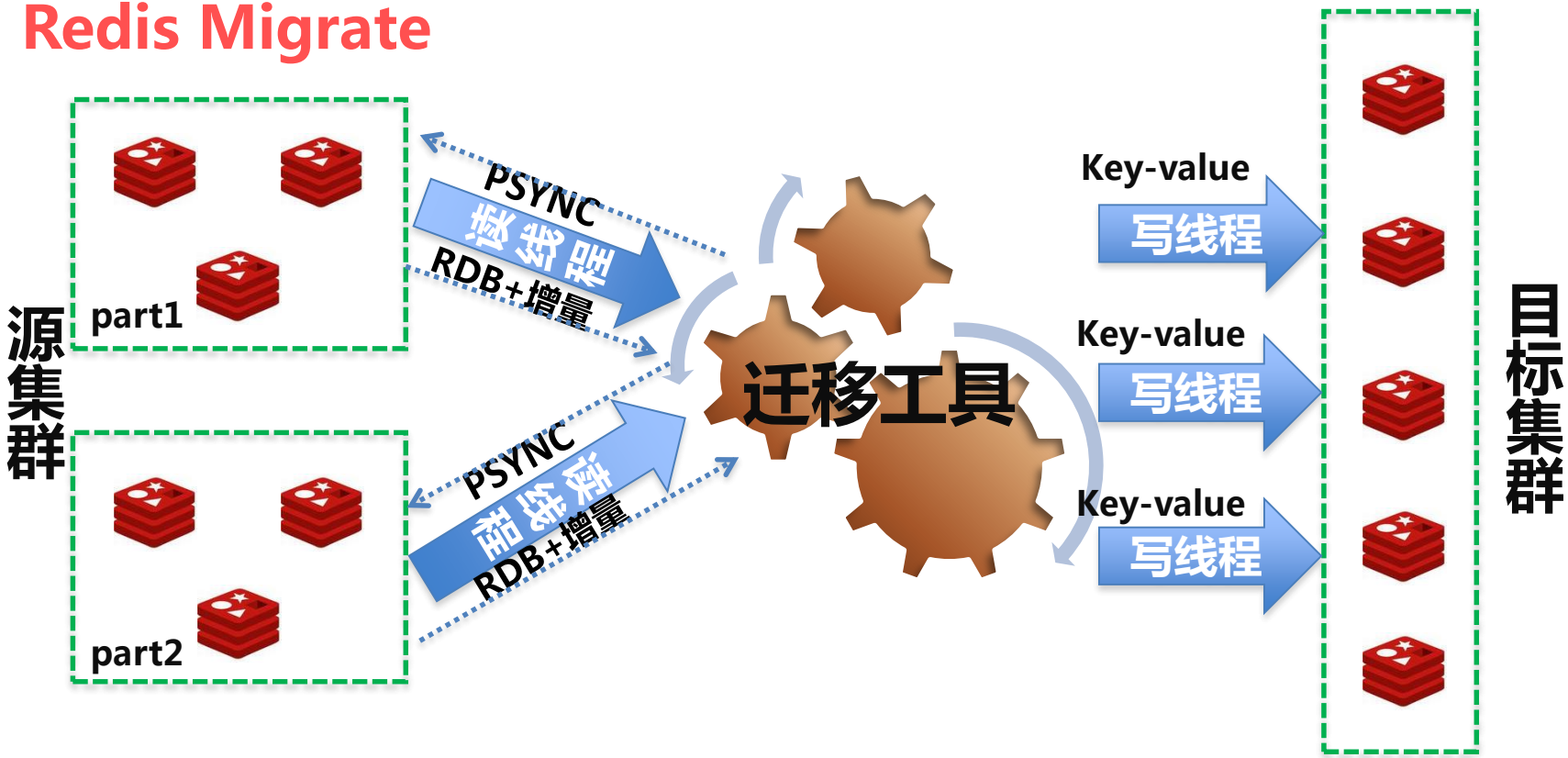
Redis Cluster 客户端

- Hiredis支持集群模式
- 解析并更新路由表
- 处理moved/ask错误
- Max redirect机制
- 支持mget/mset
- 支持pipeline
- 支持异步API

Redis Migrate

- Redis集群之间的数据迁移
- 多线程
- 基于redis复制
- 迁移过程中，老集群正常对外提供服务
- 异构迁移，支持Twemproxy，redis cluster，rdb文件 和 aof文件
- 迁移状态查看
- 数据抽样校验

Redis Migrate



Redis Migrate 步骤

```
[source]
type: twemproxy
hash: fnv1a_64
hash_tag: "{}"
distribution: ketama
servers:
 - 127.0.0.1:6379
 - 127.0.0.1:6380
 - 127.0.0.1:6381
 - 127.0.0.1:6382

[target]
type: redis cluster
servers:
 - 127.0.0.1:7379

[common]
listen: 0.0.0.0:8888
```

- 建立新集群
- 填写迁移工具配置文件
- 运行迁移工具，开始迁移
- 通过迁移工具的info命令查看迁移状态
- 当全量数据全部迁移完后，把老集群设置成readonly
- 通过迁移工具的校验机制校验数据是否一致
- 业务同学进行变更，切换到新集群
- 迁移完成

Redis 多线程

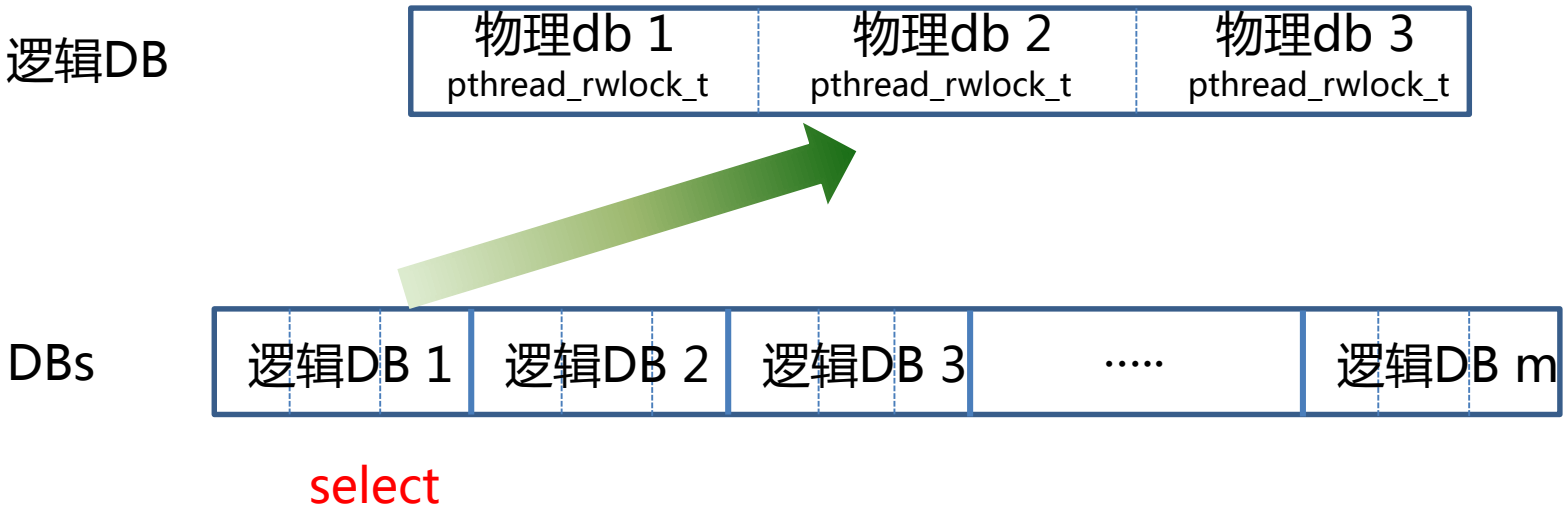
- Master+N个worker多线程模型
- 客户端由worker线程管理
- 单独后台线程处理过期key和dict维护
- DB级别读写锁
- 数据结构的实现代码移植于Redis3.2.0
- 支持五大数据结构（string/list/hash/set/zset）
- 支持100个Redis命令，包括管理命令

Redis 多线程-DB

- 为了降低DB锁竞争，引入逻辑DB
- 一个逻辑DB包含N个真实的物理db
- 用户使用的是逻辑DB
- 一个逻辑DB的所有key分散在各个物理db
- 每个物理db拥有一把读写锁-pthread_rwlock_t

Redis 多线程-DB

为了降低DB锁竞争，引入逻辑DB，一个逻辑DB包含N个真实的物理db，用户看到的是逻辑DB

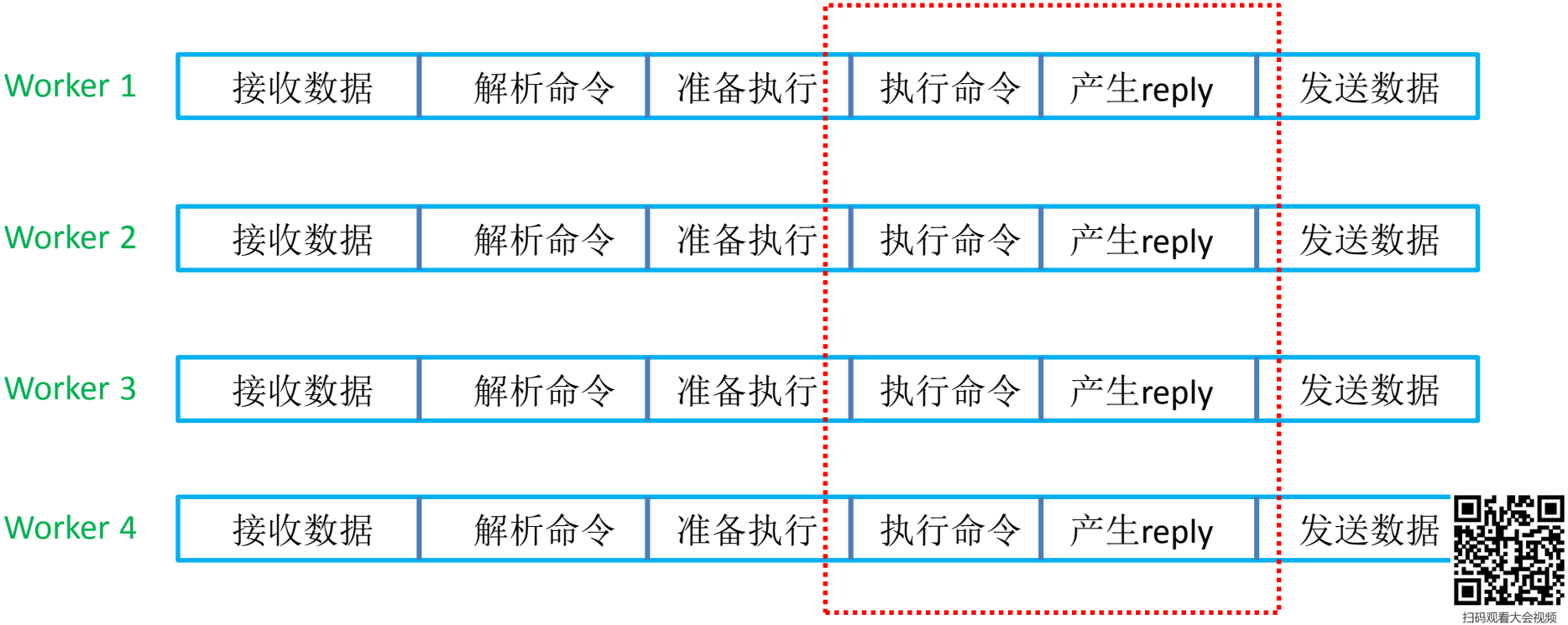


Redis 多线程-DB锁竞争

Redis处理一个客户端命令过程：

1. 接收数据 ：readQueryFromClient ()
2. 解析命令 ：processInputBuffer ()
3. 准备执行 ：processCommand ()
4. 执行命令 ：call ()
5. 产生reply：addReply()
6. 发送数据 ：handleClientsWithPendingWrites()

Redis 多线程-DB锁竞争



Redis 多线程-性能

测试环境：

- 4个worker线程
- OS: Centos6 Linux 2.6.32-504.23.4.el6.x86_64
- Cpu Model Name: Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz
- Cpu Processors: 24
- Network: 10000Mbps
- Memory: 128G

Redis 多线程-性能

测试用例：

- Key长度为16字节，1000万个随机key
- Value长度为20字节
- 测试命令：set/get

```
tests/vire-benchmark -h [redacted] -p 55555 -n 20000000 -r 10000000 -c 400 -T 12 -d 20 -t set,get -q
```

```
SET: 343023.75 requests per second
GET: 409483.66 requests per second
```

Redis 多线程-热key

只写一个key

```
tests/vire-benchmark -h [redacted] -p 55555 -n 20000000 -c 400 -T 12 -d 20 -q set key:__init__ xxx
```

```
set key:__init__ xxx: 301959.72 requests per second
```

只读一个key

```
tests/vire-benchmark -h [redacted] -p 55555 -n 20000000 -c 400 -T 12 -d 20 -q get key:__init__
```

```
get key:__init__: 436328.72 requests per second
```

有效改善热key问题

Redis中国用户组 http://www.redis.cn

Q&A

VIP开源 https://github.com/vipshop

2016 The Computing Conference

THANKS

扫码观看大会视频