

深入解读JIMDB—京东分布式缓存与高速KV存储

系统技术部@云平台

- ❖ JimDB简介
- ❖ JimDB的发展历程
- ❖ JimDB架构概述
- ❖ 管理端平台
- ❖ 监控与报警
- ❖ 迁移与扩容
- ❖ 冷热数据及持久化
- ❖ JimDB S-自主研

❖ JimDB-Jingdong in memory database

- JimDB从缓存发展而来，目前服务于京东的几乎所有的业务系统，包括很多重要的业务系统，例如，前台的商品详情页，交易平台，广告，搜索，即时通讯..... 后台的订单履约，库存管理，派送和物流.....

商品介绍 规格参数 包装清单 商品评价(6769) 售后

⚠ 如果您发现商品信息不准确，欢迎纠错

品牌	苹果 (Apple)
型号	iPhone 6 A1586
颜色	银色
上市年份	2014年
上市月份	9月
输入方式	触控
智能机	是
操作系统	苹果 (IOS)
操作系统版本	IOS
CPU品牌	苹果
我的关注	最近浏览的
处理器	

猜你喜欢



浩酷 轻薄透明手机壳保护套 适用于苹果6/iPhone6 透

¥35.00

加入购物车



RTAKO【全屏全覆盖】苹果6/iPhone6plus钢化玻璃

¥48.00

加入购物车

推广商品



洛克 (ROCK) 卡尼系列金属边框透明手机壳保护套 适用于苹果iPhor

¥56.00



萝莉 手机壳透明保护套适用于iPhor e6手机壳/苹果6 plus 透明5.5英寸



购买了该商品的用户还浏览了



苹果 (Apple) iPhone 6 Plus (A1524) 16GB 银色 移动联通电信4G手机

¥5988.00



苹果 (Apple) iPhone 6 (A1589) 16GB 银色 移动4G手机

¥5188.00



JimDB的发展历程

❖ JimDB 1.0

- 采用官方Redis作为单节点服务
- 客户端一致性Hash + Presharding技术
- 管理，监控和报警

❖ Jimdb 2.0

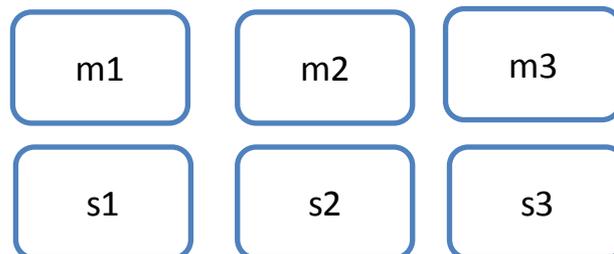
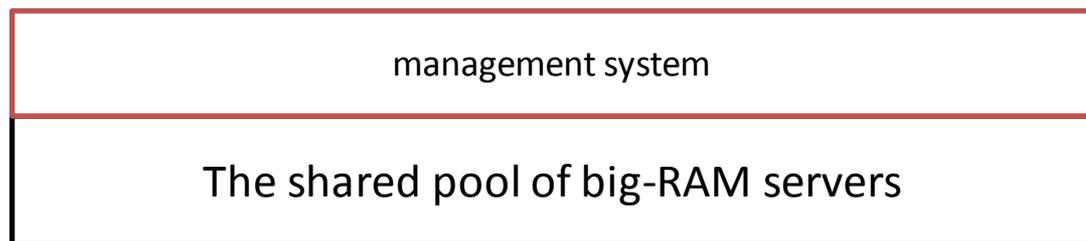
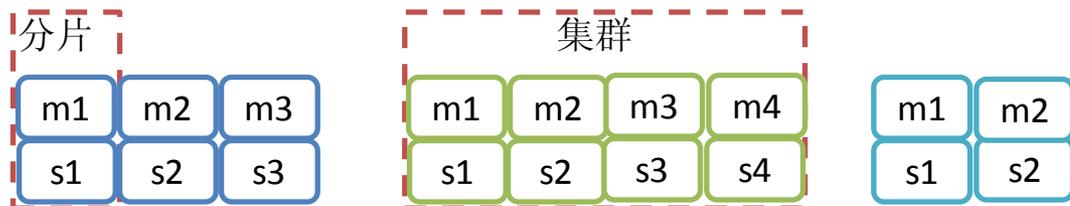
- 故障检测和自动切换
- 平滑纵向扩容和平滑横向扩容
- 基于内存+SSD的两级存储结构和自主研发存储引擎

❖ Jimdb 3.0

- 自助接入和自动部署
- 容器化
- 全自动弹性调度

JimDB架构概述

❖ Redis实例(万)、服务器(千)、集群(千)



❖ 支持大容量缓存

将缓存数据分摊到多个分片（每个分片上具有相同的构成，比如：都是一主一从两个节点）上，从而可以创建出大容量的缓存。

❖ 数据的高可用性

支持异步复制和同步复制，目前可以达到等同于mysql级别的数据可用性。

❖ 支持多种I/O策略

针对读操作可分为“主优先”、“从优先”、“随意挑选”等方式；不同的I/O策略，对数据一致性的影响也不同，应用可以根据自身对数据一致性的需求，选择不同的I/O策略。

❖ 哨兵服务和故障自动切换

通过选举算法实现的哨兵服务能够自动判断实例的不存活状态，通知 Failover服务进行主从自动切换，切换时间在秒级，以保证服务的7*24小时不间断运行。

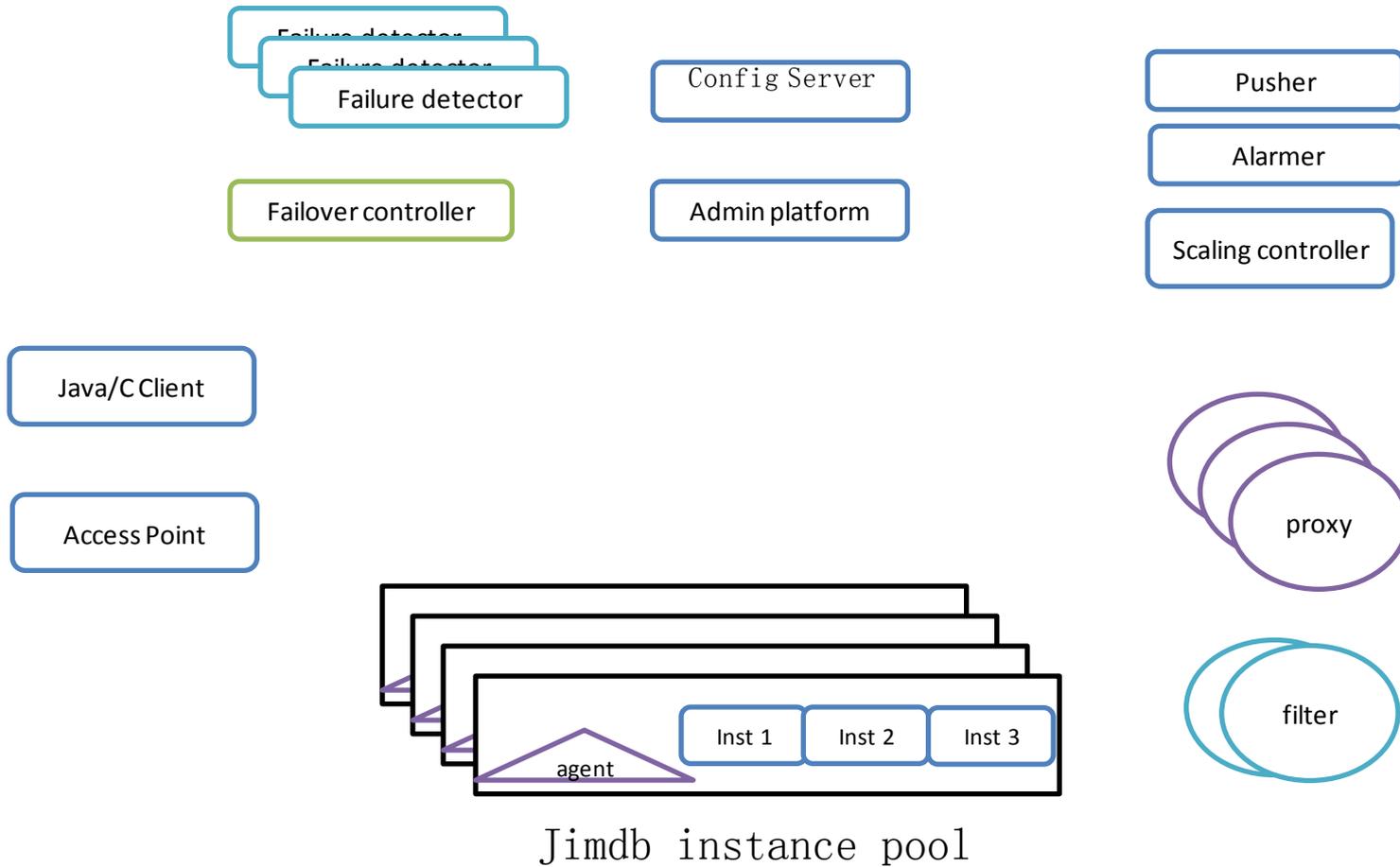
❖ 支持动态扩容

可通过多种途径实现动态扩容：

第一种形式，通过在单个节点上预留内存，然后需要扩容时直接使用预留内存的方法达到扩容的目的；

第二种形式，通过数据迁移来实现扩容。（平滑纵向扩容）

第二种形式，通过增加分片数来实现扩容。（平滑横向扩容）



- Jimdb instance pool

独立部署一系列jimdb instance或者将现有已经部署的jimdb instance纳入管理，形成逻辑上统一的缓存资源池。

- Failure detector(哨兵)

每机房一套哨兵，通过分布式投票判断实例是否存活。

- Failover controller (自动切换控制器)

接受来自哨兵的投票结果，执行自动切换

- Pusher (信息采集)

部署一系列采集应用节点，对所有的jimdb实例进行信息采集，存储到时间序列数据库，必要的时候发送报警信息。

- Alarm(规则报警器)

对采集传输的数据进行实时分析，使用预定义的一系列规则进行报警

- Scaling controller(扩容控制器)

控制各分布式组件(proxy, filter) 协调工作，进行平滑纵向切换和水平扩容。

- Config Server

负责提供配置和元信息服务，以便应用端的Client SDK来polling变更。

- Client SDK

Client SDK为应用提供操作jimdb集群数据的API。利用polling到的集群配置元信息（节点拓扑、路由和读写策略等），访问后端的jimdb节点。SDK负责监控配置信息变更（比如故障切换、服务节点迁移等），动态重启底层的连接池。对应用透明。

- Admin Platform (管理平台)

Jimdb统一管理入口，提供集群管理（创建、销毁、更改参数配置），实例管理（起停、部署、更改运行配置、数据查询），报警监控管理（报警规则管理、报警及异常记录管理、实例及集群状态曲线展示），报表管理，客户端配置管理等功能。同时，权限管理将用户划分为超级管理员，集群管理员，集群用户三种角色，对每种角色的操作权限作严格的权限控制。

管理平台



-> 了解集群拓扑, 及有哪些实例, 做到心中有数

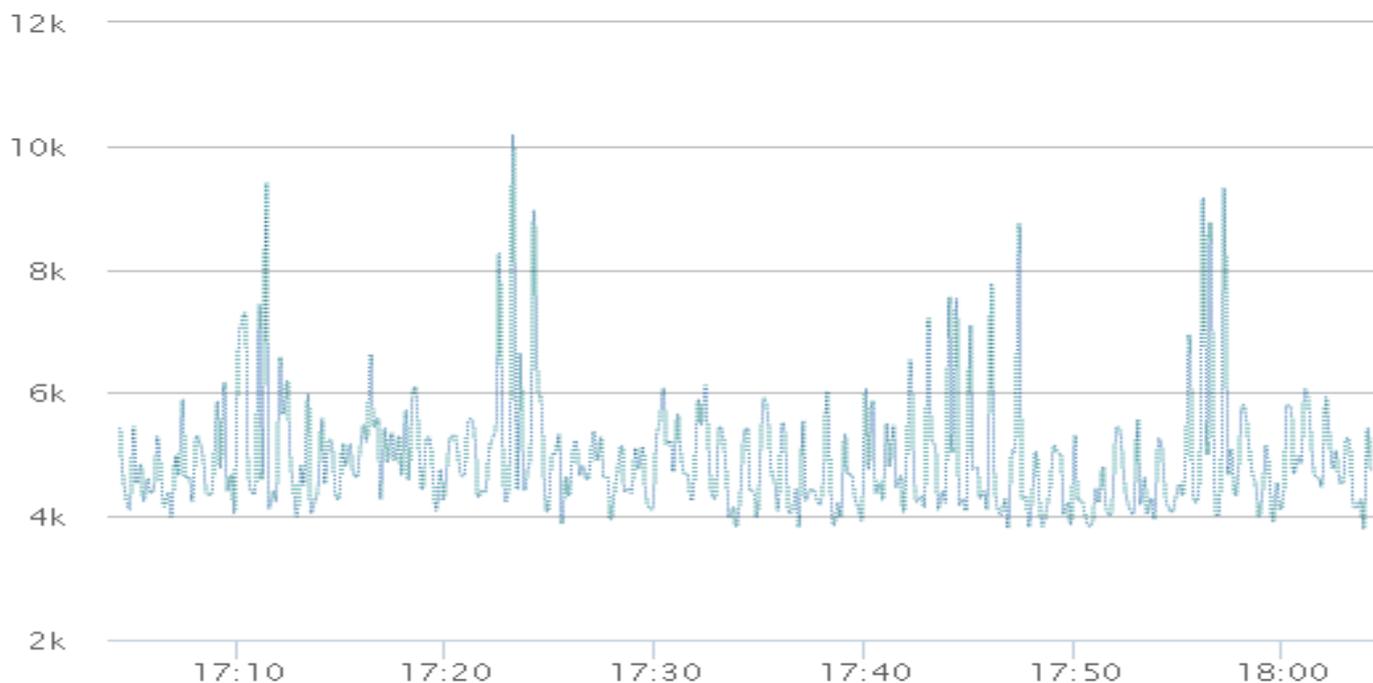
每秒操作数

最近一小时

最近一天

分片1

instantaneous_ops_per_sec (172.17.51.72:6508)



-> 了解集群每个实例Ops

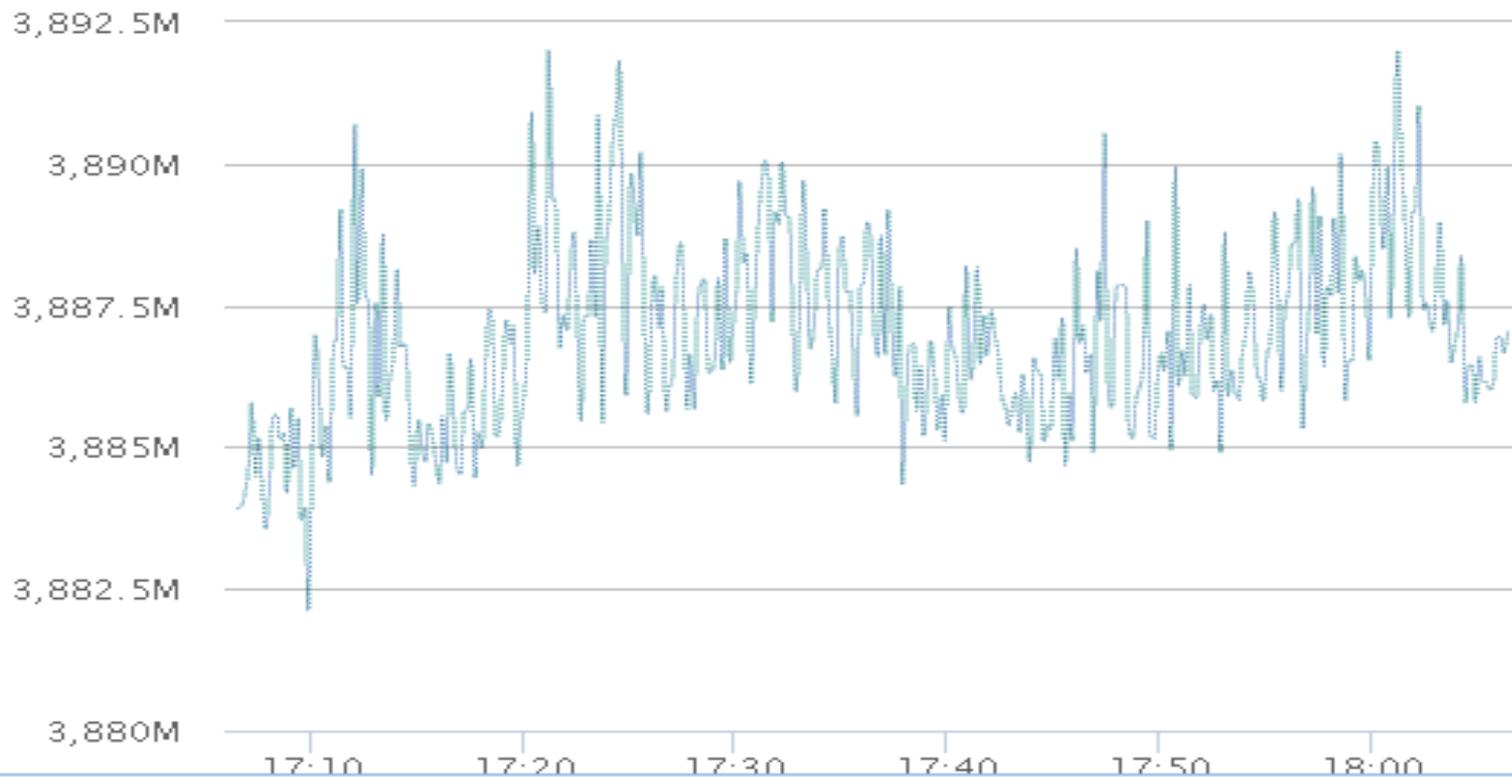
已使用内存量(byte) ▼

最近一小时

最近一天

分片1

used_memory (172.17.51.72:6508)



-> 了解集群每个实例的内存使用情况

查看客户端配置及列表

configId: /redis/cluster/513

Token: 1413009276797

+ 添加客户端配置

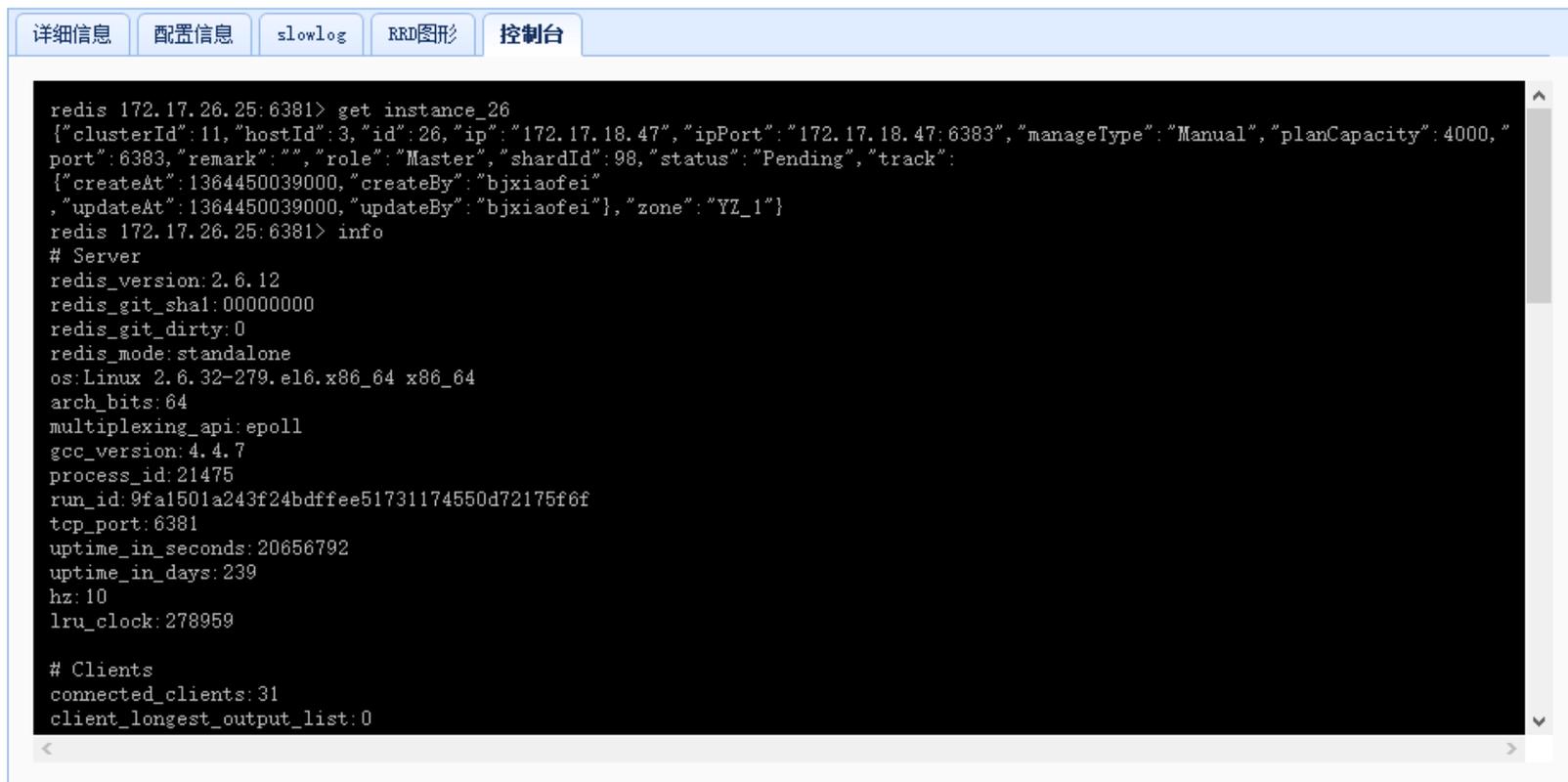
ID	创建时间	服务器版本号	操作
863	2014-10-11 14:34:36	1	查看 下发 启用
875	2014-10-14 10:57:03	2	查看 下发 当前启用中

客户端列表

刷新

客户端标识	IP	端口	已同步版本	正使用版本
10641@A01-R02-D1403-I17-94. JD. LOCAL[10/14/14 17:40:53.807]	127.0.0.1	61665	2	2
3750@A01-R02-D1403-I24-21. JD. LOCAL[10/14/14 18:24:36.187]	127.0.0.1	61379	2	2
982@A01-R02-D1403-I24-21. JD. LOCAL[10/14/14 18:19:31.377]	127.0.0.1	61252	2	2
9573@A01-R02-D1403-I24-23. JD. LOCAL[10/14/14 20:00:10.810]	127.0.0.1	63031	2	2
27433@A01-R02-D1403-I37-126. JD. LOCAL[10/14/14 18:25:07.836]	127.0.0.1	41980	2	2
8731@A01-R02-D1403-I17-94. JD. LOCAL[10/14/14 17:34:47.926]	127.0.0.1	42784	2	2
26586@A01-R02-D1403-I37-126. JD. LOCAL[10/14/14 18:24:09.500]	127.0.0.1	62361	2	2
15829@A01-R02-D1403-I17-96. JD. LOCAL[10/14/14 17:40:56.525]	127.0.0.1	64146	2	2
9933@A01-R02-D1403-I17-93. JD. LOCAL[10/14/14 17:33:23.272]	127.0.0.1	41904	2	2
13354@A01-R02-D1403-I37-125. JD. LOCAL[10/14/14 17:30:23.257]	127.0.0.1	61950	2	2

提供类似于redis命令窗口的web控制台，禁止危险命令，严格控制写命令和一些运维相关命令，适当放开查询命令。



The screenshot shows a web-based Redis console with a navigation bar at the top containing tabs for '详细信息', '配置信息', 'slowlog', 'RRD图形', and '控制台'. The '控制台' tab is active, displaying a terminal window with the following text:

```
redis 172.17.26.25:6381> get instance_26
{"clusterId":11,"hostId":3,"id":26,"ip":"172.17.18.47","ipPort":"172.17.18.47:6383","manageType":"Manual","planCapacity":4000,"port":6383,"remark":"","role":"Master","shardId":98,"status":"Pending","track":{"createAt":1364450039000,"createBy":"bjxiaofei","updateAt":1364450039000,"updateBy":"bjxiaofei"},"zone":"YZ_1"}
redis 172.17.26.25:6381> info
# Server
redis_version:2.6.12
redis_git_sha1:00000000
redis_git_dirty:0
redis_mode:standalone
os:Linux 2.6.32-279.el6.x86_64 x86_64
arch_bits:64
multiplexing_api:epoll
gcc_version:4.4.7
process_id:21475
run_id:9fa1501a243f24bdf5e51731174550d72175f6f
tcp_port:6381
uptime_in_seconds:20656792
uptime_in_days:239
hz:10
lru_clock:278959

# Clients
connected_clients:31
client_longest_output_list:0
```

监控与报警

❖ Pains

- 网络不佳的情况下可能发生误判
- Redis单线程执行，在进行长任务时可能发生误判

❖ Solution

- 在机房中不同区域部署多个Failure Detector
- 多个Failure Detector之间采用分布式选举算法，判断Redis实例的死活
- 连接健康度不佳时，验证端口是否通畅

集群首页	集群分片	集群信息	集群主机	授权	报警规则	报警任务	异常记录	监控	报表	管理任务	分组	客户端
+ 添加新报警规则												
报警规则名称	实例类型	指标类型	类型	阈值	故障持续时间(秒)	报警最小间隔期	创建时间	创建人	操作			
内存使用率过高	redis_instance	used_memory_ratio	大于	90	20	3600	2014-05-31 14:03:55	bjliangqiushi	✔ 启用			
实例存活性	redis_instance	alive	不等于	alive	60	3600	2014-03-04 15:55:49	bjliangqiushi	<input type="text"/>	↕	⊘ 禁用小时	
主从复制延迟	redis_slave	master_last_io_seconds	大于	30	30	3600	2014-01-03 17:25:04	bjliangqiushi	<input type="text"/>	↕	⊘ 禁用小时	
master_port发生变化	redis_slave	master_port	不等于	#{master_port}	30	43200	2013-06-06 22:39:51	bjliangqiushi	<input type="text"/>	↕	⊘ 禁用小时	
master_host发生变化	redis_slave	master_host	不等于	#{master_host}	30	43200	2013-06-06 22:37:28	bjliangqiushi	<input type="text"/>	↕	⊘ 禁用小时	
Slave角色发生变化	redis_slave	role	不等于	slave	30	43200	2013-06-06 22:33:51	bjliangqiushi	<input type="text"/>	↕	⊘ 禁用小时	
Master角色发生变化	redis_master	role	不等于	master	30	43200	2013-06-06 22:31:19	bjliangqiushi	<input type="text"/>	↕	⊘ 禁用小时	
主从故障	redis_slave	master_link_status	不等于	up	30	3600	2013-05-30 12:39:24	bjxiaofei	<input type="text"/>	↕	⊘ 禁用小时	
20 ▾	⏪ ⏩	第 1 共 1 页	⏪ ⏩	🔄	显示1到8, 共							

报警:

- 基于规则
- 可设定全局规则
- 和局部规则

- 可设定规则排除和例外

监控对象类	指标类型	警告内容	状态	创建时间	发送时间
redis_inst	alive	Redis集群store-queue(id:516)实例172.22.202.64:6666(id:11363)触发实例存活性规则: ali	SENT	2014-10-14 15:	2014-10-14 15:0
redis_inst	master_l	Redis集群store-queue(id:516)实例172.22.202.65:6666(id:11364)触发主从故障规则: maste	SENT	2014-10-14 15:	2014-10-14 15:0
redis_inst	alive	Redis集群store-queue(id:516)实例172.22.202.64:6666(id:11363)触发实例存活性规则: ali	SENT	2014-10-14 14:	2014-10-14 14:0
redis_inst	master_l	Redis集群store-queue(id:516)实例172.22.202.65:6666(id:11364)触发主从故障规则: maste	SENT	2014-10-14 14:	2014-10-14 14:0

监控对象类型	监控对象Id	指标类型	异常值	创建时间
redis_instance	6670	used_memory_ratio	90.18413552	2014-10-14 19:45:34
redis_instance	6669	used_memory_ratio	90.61903125333333	2014-10-14 19:45:34
redis_instance	2088	used_memory_ratio	90.73997797333332	2014-10-14 19:45:25
redis_instance	6669	used_memory_ratio	90.61894858666668	2014-10-14 19:45:19
redis_instance	2089	used_memory_ratio	90.30214058666667	2014-10-14 19:45:16
redis_instance	6670	used_memory_ratio	90.18400928	2014-10-14 19:45:16
redis_instance	2088	used_memory_ratio	90.73435738666666	2014-10-14 19:45:15
redis_instance	2089	used_memory_ratio	90.30529765333334	2014-10-14 19:45:15
redis_instance	6670	used_memory_ratio	90.18400517333333	2014-10-14 19:45:14
redis_instance	6669	used_memory_ratio	90.61918143999999	2014-10-14 19:45:14

- 报警记录可查
- 异常记录可查



2014/10/14 (周二) 18:31

jimstore

Jimdb存活报警

收件人

广告播放 2(非托管)->172.17.17.73:16579 不存活(2014-10-14 18:49:38)
第一联系人:bjjscuitao/18612527839/cuitao@jd.com



2014/10/14 (周二) 19:34

jimstore

Jimdb规则报警

收件人

Jimdb 集群 chenxi-详情(id:146)实例 172.17.38.166:6401(id:2088)触发内存使用率过高规则:
used_memory_ratio 指标大于阈值 90 连续 20 秒, 当前值为 90.736384373333334。(2014-10-14 19:33:54)

最近一小时

最近一天

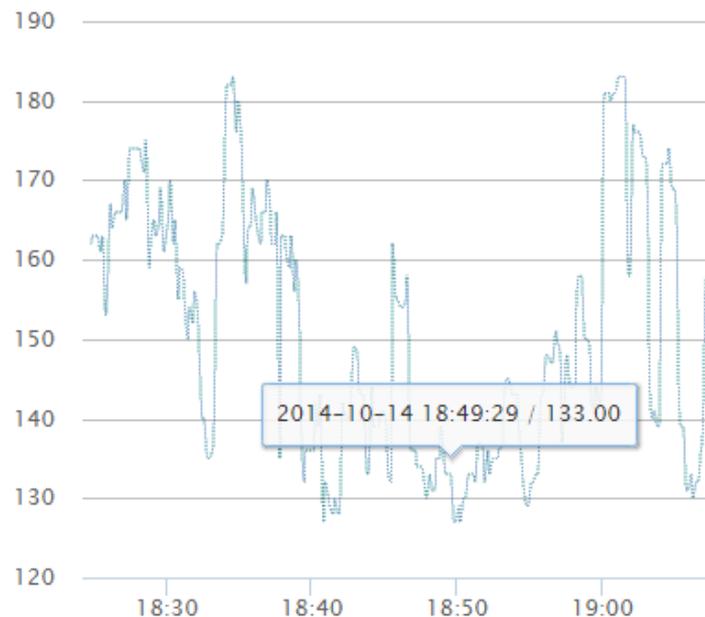
最近一周

最近一月

最近一年

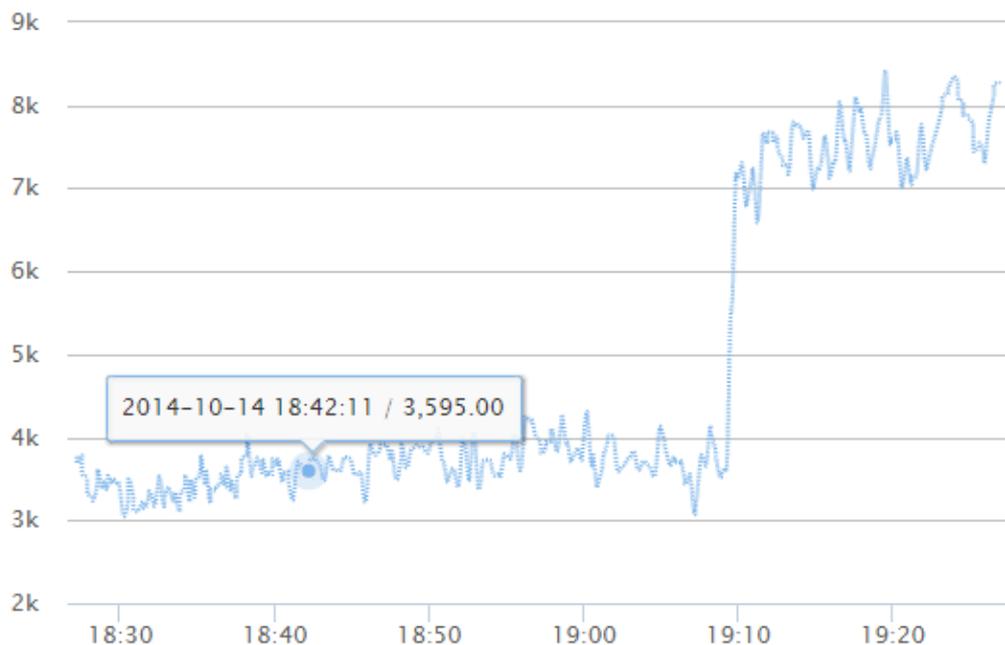
分片1

connected_clients (172.17.23.197:16380)



分片2

instantaneous_ops_per_sec (172.17.23.197:16381)



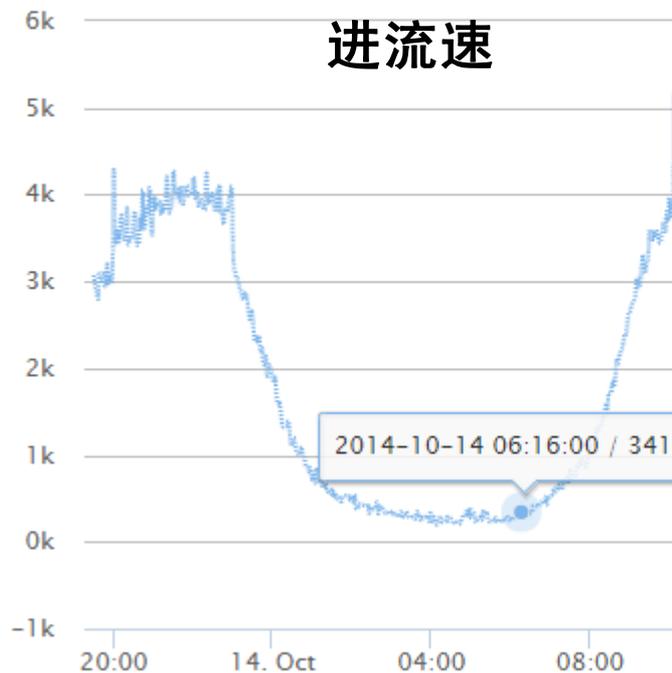
监控

- 多指标维度
- 实时
- 小时、天、周、月、年

分片1

net_io_in_per_sec (172.17.51.71:6701)

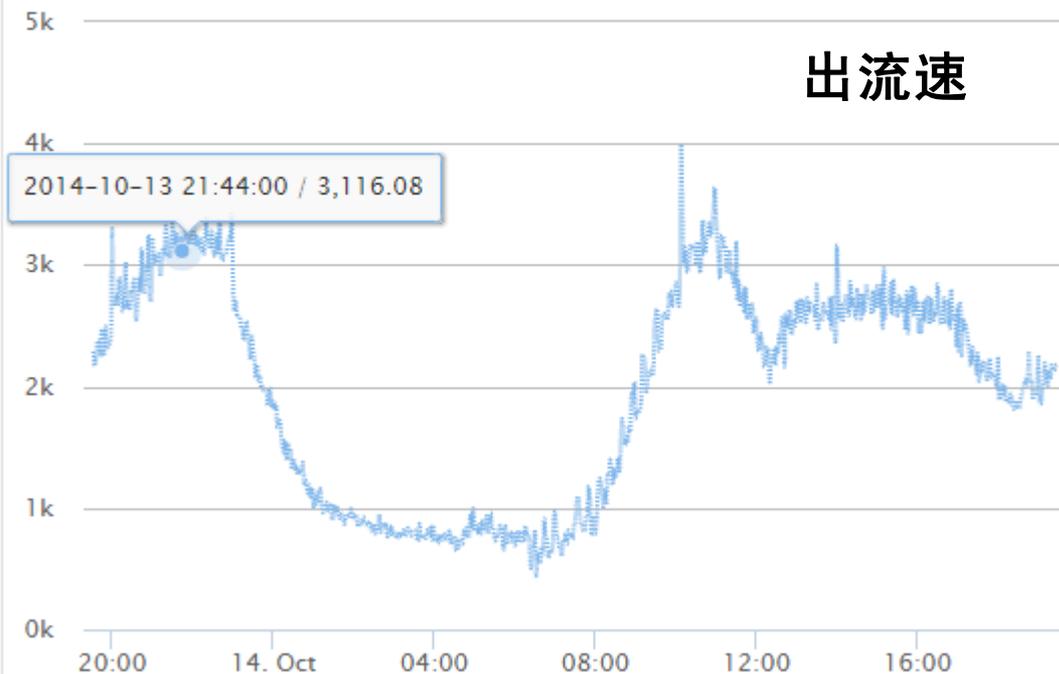
进流速



分片1

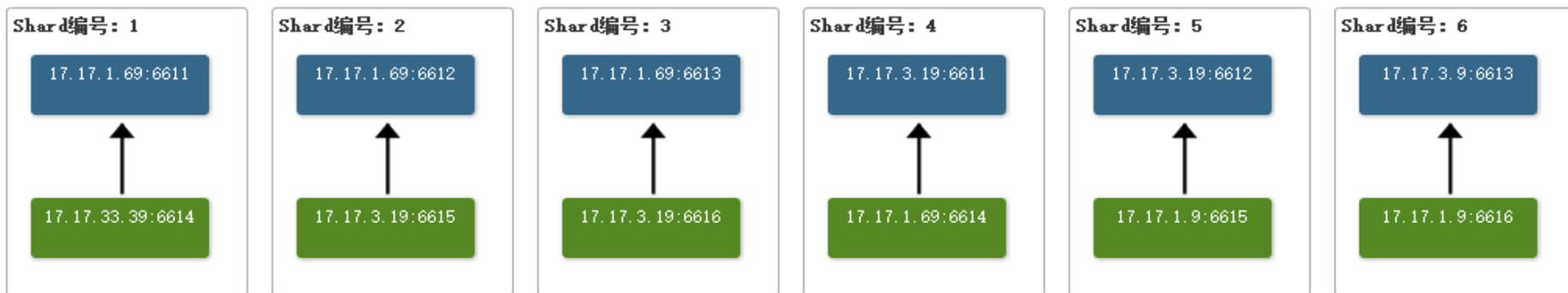
net_io_out_per_sec (172.17.51.71:6701)

出流速



迁移与扩容

一主一从

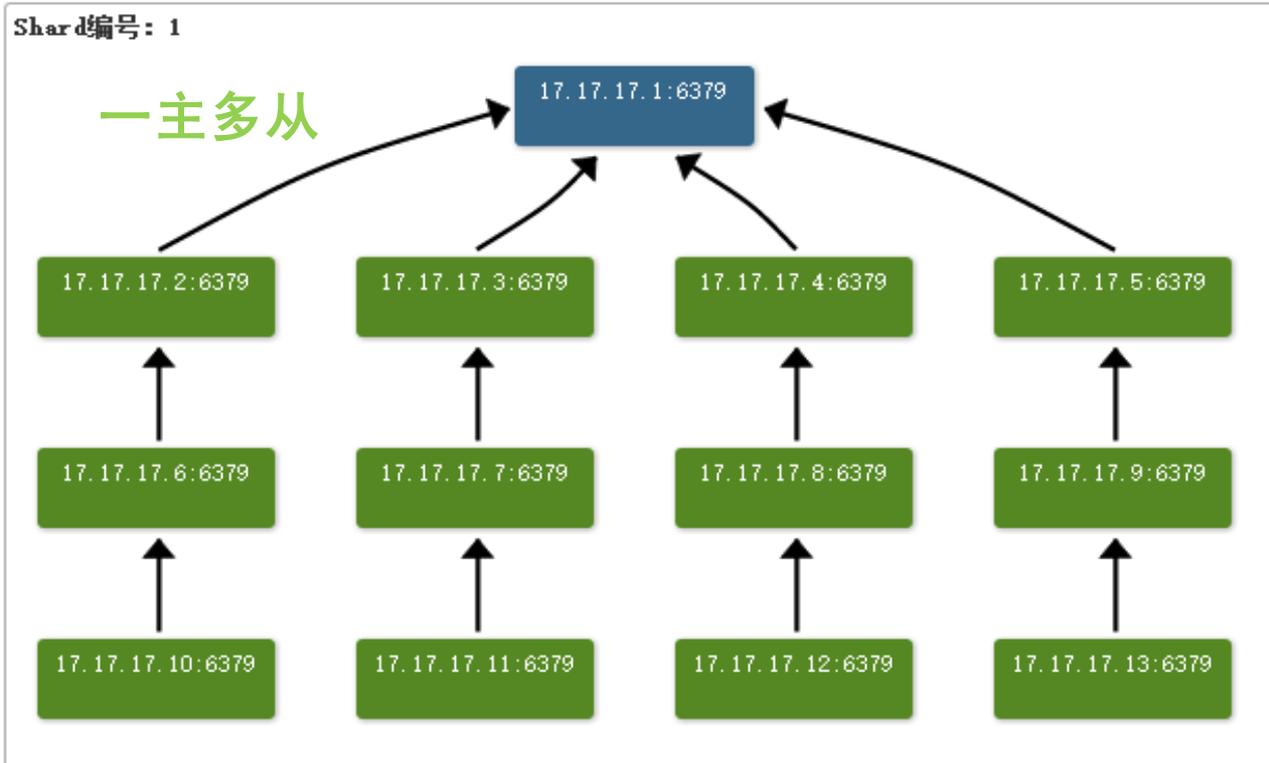


分片间:

- 平滑横向扩容
- 分摊压力, 流量

分片内:

- 平滑纵向扩容
- 主从灾备防止单shard不可用
- 主从读写分离, 分摊读压力, 流量

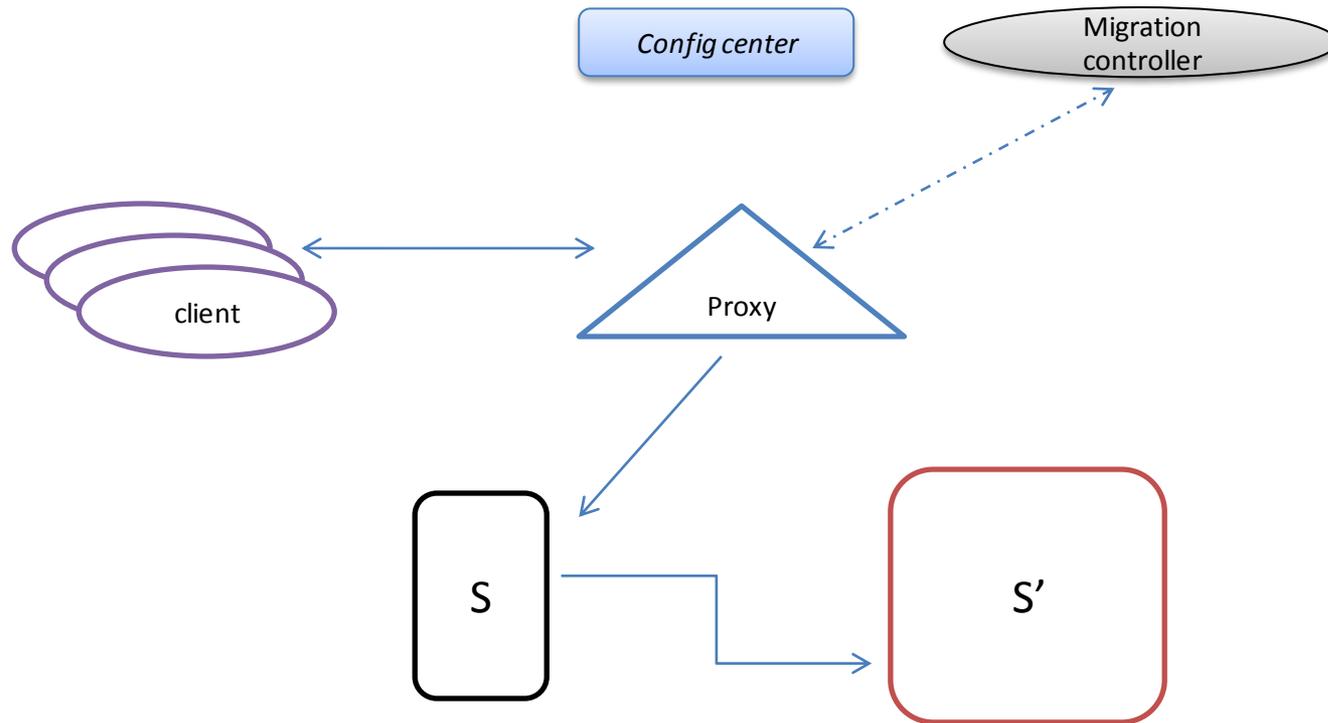


❖ Scaling Up – 纵向扩容

- 在内存不够，需要增加内存时首先考虑的是纵向扩容，即增加每个分片的主、从节点的内存。
- 纵向扩容时如果Redis实例所在计算机物理内存不够，就需要进行数据迁移
- 数据迁移的同时，服务不能暂停

❖ Scaling Out – 横向扩容

- 单一分片的内存是不能无限扩容（纵向）的，太大了会影响复制的效率
- 在纵向扩容无法进行的情况下（单一分片内存已经很大，或者流量压力很大），就需要进行横向扩容，即增加集群的分片数
- 横向扩容的同时，服务不能暂停



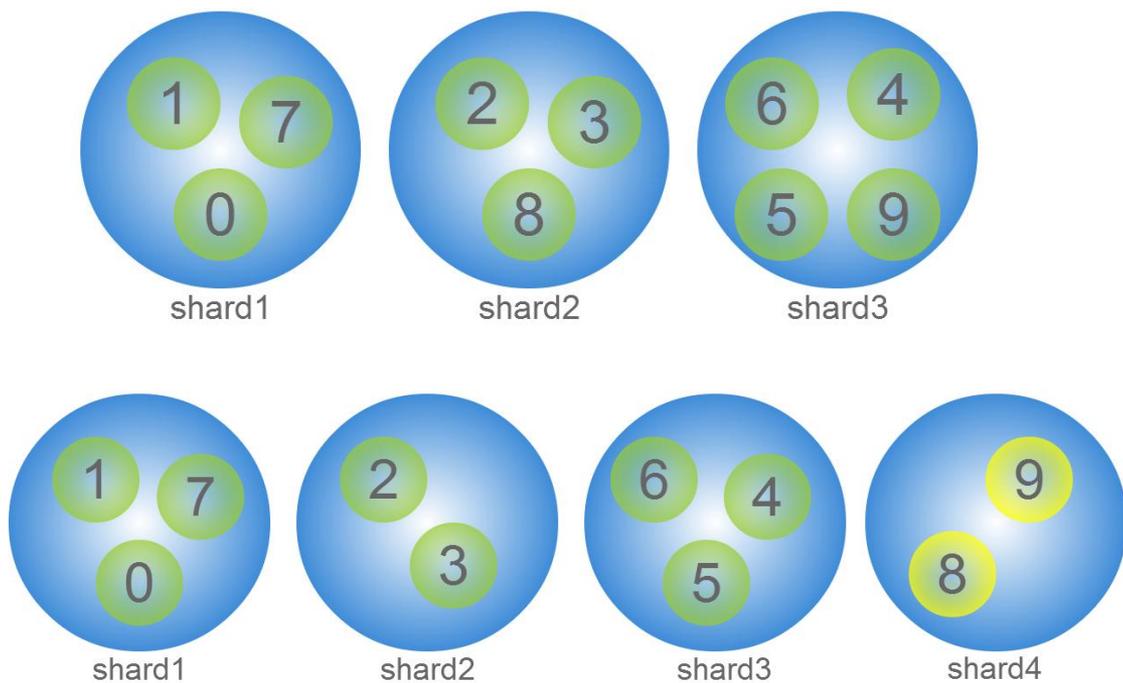
❖ Pains

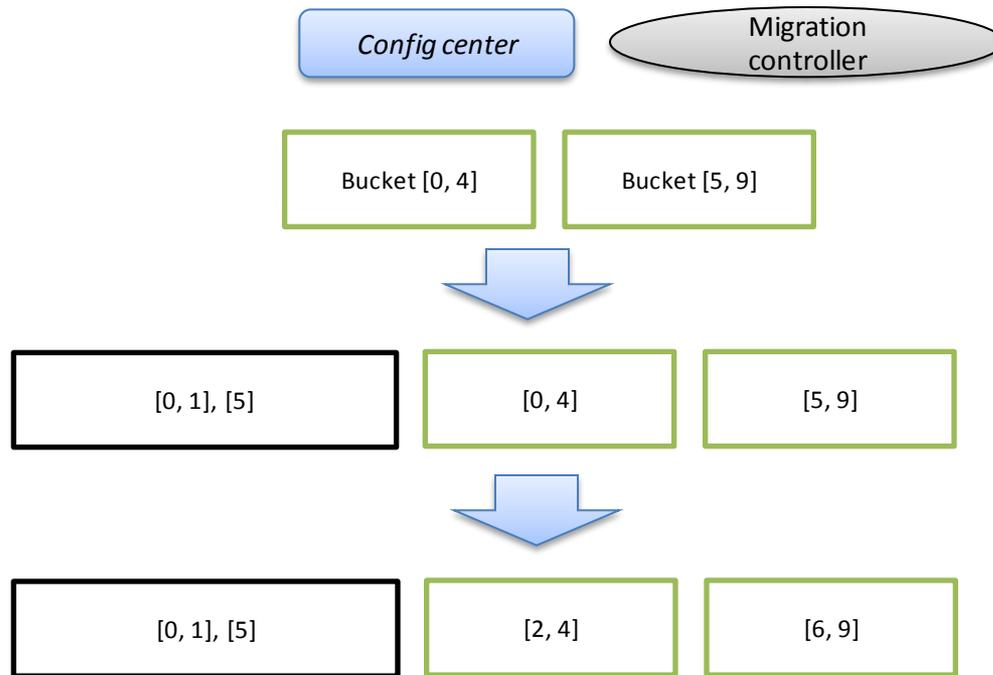
- 垂直扩容并不增加分片数，简单修改jimdb实例运行时参数可提高该实例可用内存上限，但在机器内存吃紧时，若要提高该分片内存上限，需要将该实例平滑迁移至一台内存资源更加充沛的机器。
- 流量打满或者出现热点时，需要加分片分散压力。机器内存不够时，有时也需要加分片
- Pre-sharding的方式，在不影响服务的情况下增加分片有难度。
- 可以通过定制开发引入bucket来进行横向扩容，但线上还有2.4，2.6，2.8等既有版本，也有加分片的需求
- 避免主从数据不一致
- 服务不能暂停，平滑不影响业务

❖ Solution

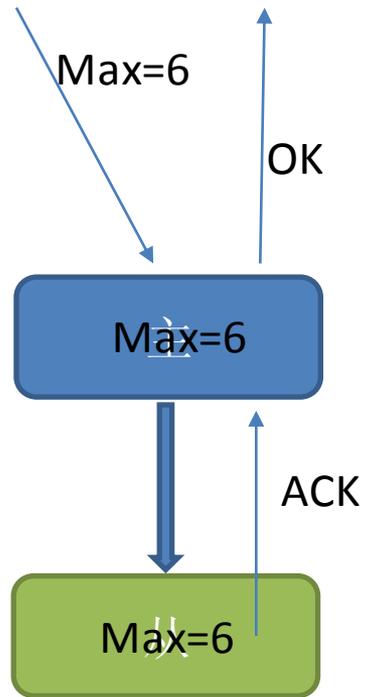
- 通过Filtered replication，实现某个结点的分裂(split)
- 开发一个支持Hold和Split的Proxy, 并通过一个流程控制器来协调客户端，服务结点，Proxy，等相关各方

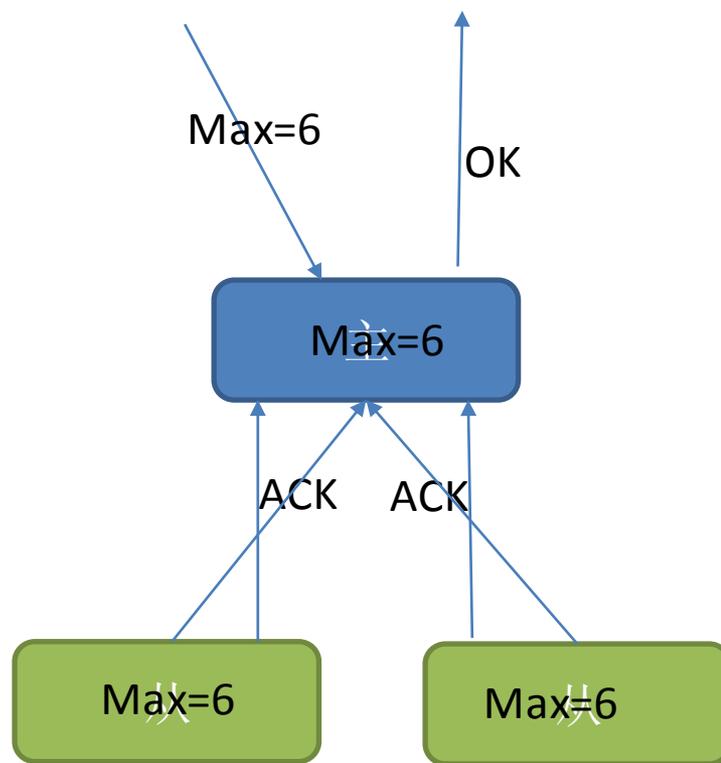
对于水平扩容，则是依赖于bucket来解决的。每个jimdb实例内部都含有若干个buckets，和上述第一类扩容相似，水平扩容也是通过对数据进行平滑迁移类实现的，但迁移的粒度不再是整个实例，而是针对集群中的这些buckets。扩容前后如下图所示：

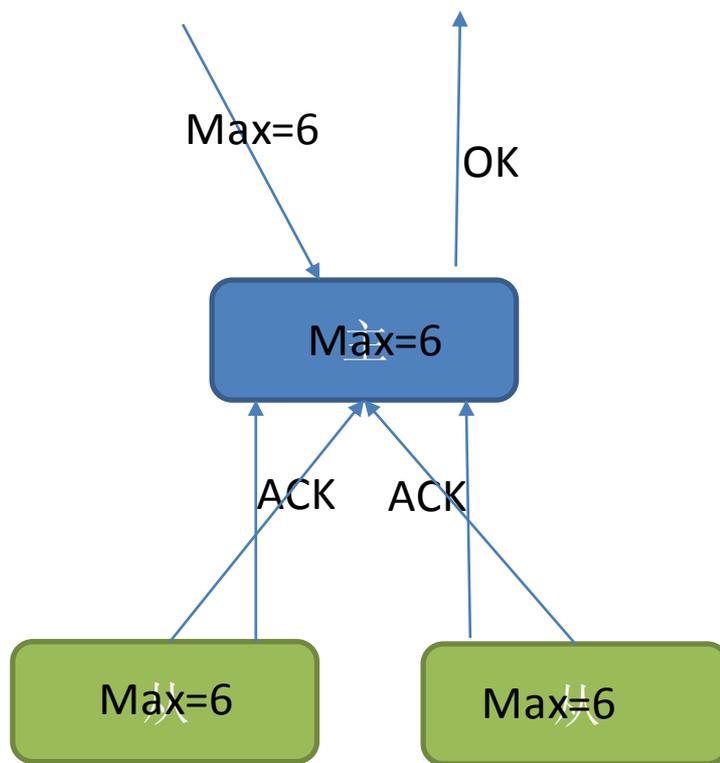




同步复制







冷热数据及持久化

❖ Pains

- Redis完全依赖于内存，往往内存不够使用
- Redis启动时需要把全部数据加载到内存,在数据量大时启动速度慢
- 规划总是赶不上业务发展，内存总量不断被突破，不断陷入扩容，再扩容...的梦魇

❖ Solution

- 引入RAM + SSD/HDD两级存储，在内存中存储热点数据，冷数据被自动交换到磁盘，解决了内存不足的问题
- 启动时并不把所有数据加载入内存，而是在运行时根据需要加载，解决了启动速度慢的问题
- 因为引入了二级存储，存储容量通常比较大，所以不需要频繁的扩容了

- ❖ 大容量
- ❖ 结合主从同步复制，提供数据库级别的可靠性
- ❖ 接近纯内存redis的读性能
- ❖ 稳定的写性能和低延迟
- ❖ 使用SSD存储介质

❖ 存储

JFS : 京东文件系统 , 提供统一的文件/对象/块存储服务

JIMDB : 京东分布式缓存与高速KV存储 , 兼容Redis协议

❖ 中间件

SAF : 服务框架 , SOA之基石

JMQ : 消息队列 , the Datacenter Pipes!

❖ 弹性计算

JDOS : 软件定义计算单元 (VM & Container) + 软件定义网络 (自研SDN) + 软件定义存储 (JFS & JimDB) = 软件定义数据中心

CAP: 弹性调度平台

谢谢!

袁航@JIMDB团队

系统技术部@云平台

www.jd.com

