SDCC 2016
中国软件开发者大会
SOFTWARE DEVELOPER CONFERENCE CHINA

同程旅游缓存演进

嘉宾：王晓波

同程旅游缓存概况

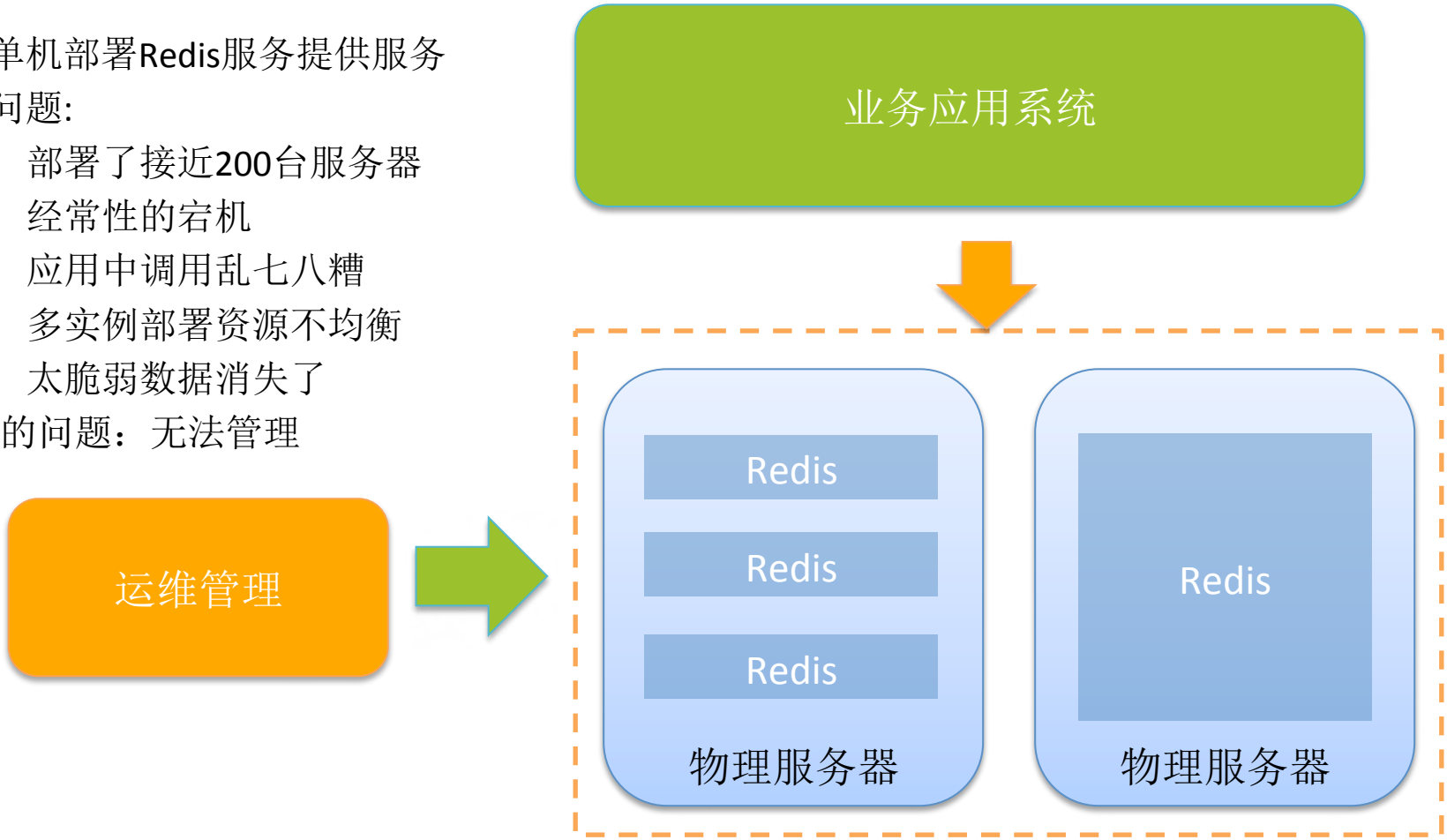
- 从memcache开始使用缓存
- 再从memcache转到Redis
- 从单机Redis升级到集群Redis
- 从简单的运维升级到全自动运维
- 自研开发Redis客户端
- 开发Redis调度治理系统
- Redis部署全面Docker化

我们也是一路坑过来的

单点部署

- 单机部署Redis服务提供服务
- 问题:
  - 部署了接近200台服务器
  - 经常性的宕机
  - 应用中调用乱七八糟
  - 多实例部署资源不均衡
  - 太脆弱数据消失了

最大的问题：无法管理



初步改善部署

- 开启主从+keepalived模式
- 开启数据落盘

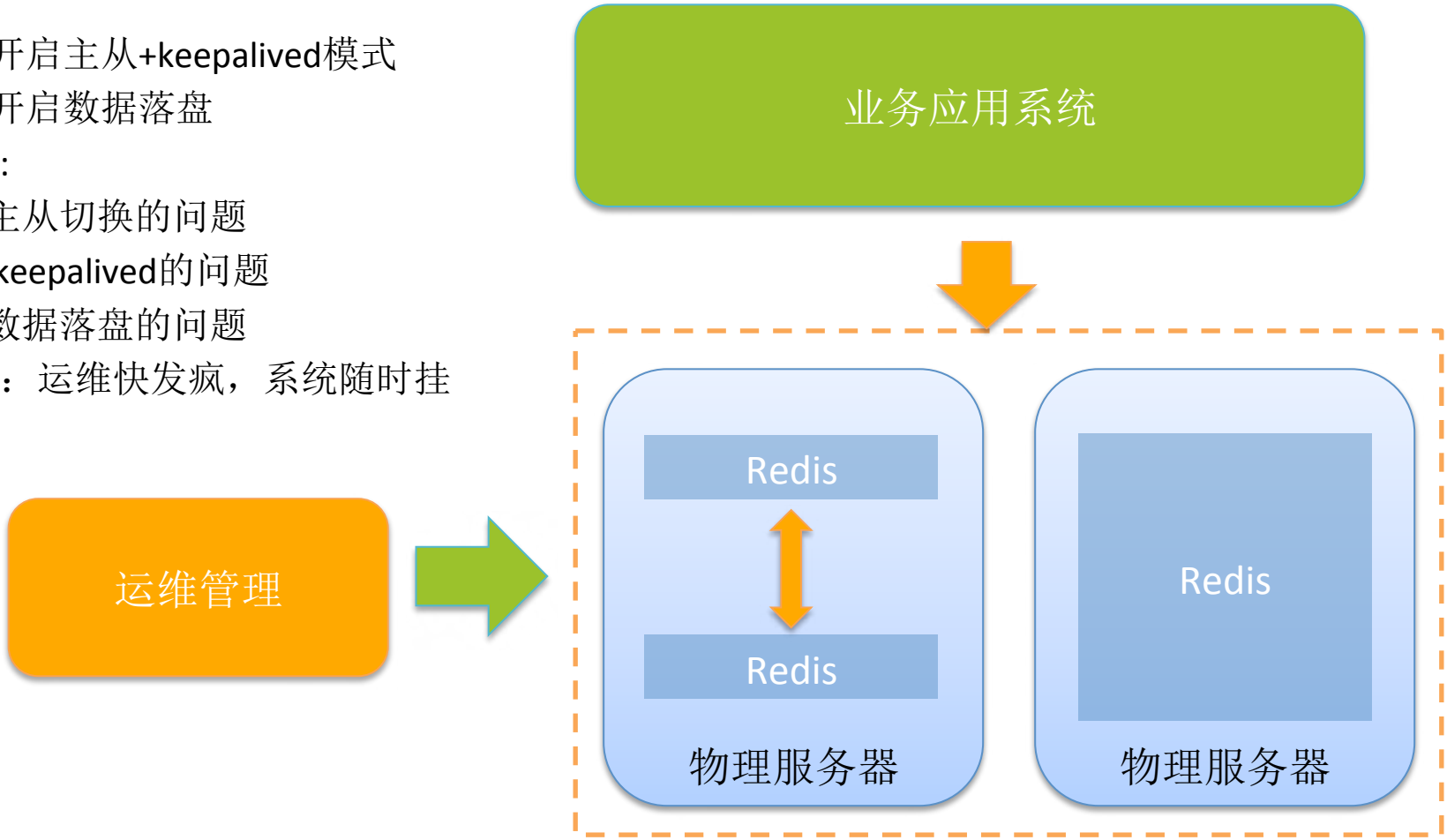
问题:

主从切换的问题

keepalived的问题

数据落盘的问题

总结：运维快发疯，系统随时挂



初步的监控

- connected_clients : 已连接客户端的数量
- client_longest_output_list : 当前连接的客户端当中，最长的输出列表
- client_longest_input_buf : 当前连接的客户端当中，最大输入缓存
- blocked_clients : 正在等待阻塞命令（BLPOP、BRPOP、BRPOPLPUSH）的客户端的数量
- used_memory_human : 以人类可读的格式返回 Redis 分配的内存总量
- used_memory_rss : 从操作系统的角度，返回 Redis 已分配的内存总量（俗称常驻集大小）。这个值和 top 、 ps 等命令的输出一致。
- replication : 主/从复制信息

- instantaneous_ops_per_sec : 服务器每秒钟执行的命令数量。

使用者的问题

- 下面一段运维与开发者的对话

开发:我们的redis为啥不能访问了?
运维:刚刚服务器内存坏了,服务器自动重启了啊
开发:为什么我们的redis延迟这么大?
运维:大哥,不要在Zset里放几万条数据,插入排序会死人啊
开发:我写进去的key呢,为什么不见了?
运维:你的redis超过最大大小了啊,不常用key的都丢了啊
开发:刚刚为啥读取全部失败了啊
运维:网络临时中断了一下,从机全同步了,在全同步完成之前,从机的读取全部失败的啊
开发:我的系统,我需要800G的redis,什么时候能准备好?
运维:大哥,我们线上的服务器最大就256G,不能玩这么大啊
开发:我的redis慢的像驴，你们的服务器有问题吧
运维:大哥千万级的KEY,用keys*。慢是一定了。

一个架构师的总结

从来没想过,一个小小的Redis还有这么多新奇的功能。就像在手上有锤子的时候,看什么都是钉子。渐渐的,开发规范倒是淡忘了,新奇的功能却接连不断的出现了,基于redis的分布式锁,日志系统,消息队列,数据清洗,等等等等,各种各样的功能不断上线,从而引发各种各样的问题。运维天天疲于奔命,到处处理着redis堵塞,网卡打爆,连接数爆表….

缓存的故障频发

- 一个个Redis的故事（只能担心）

坑的总结

- 使用的者的乱用，滥用，懒用。
- 运维数千台毫无使用规则的缓存服务器
- 运维不懂开发，开发不懂运维
- 缓存在无设计无控制中被使用
- 开发人员能力各不相同
- 太多的服务器资源被浪费
- 懒人心理（应对变化不够快）

需要一个什么样的缓存

- 缓存要解决问题(解决了快但没解决烂)

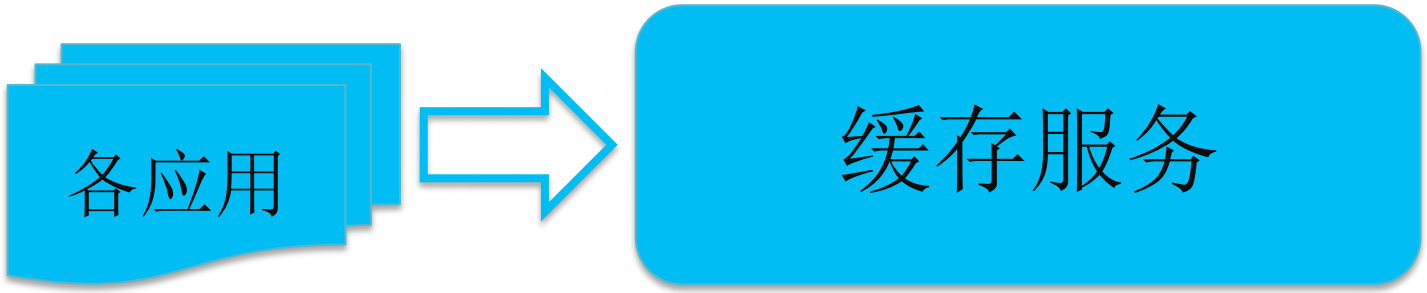
  应用对缓存大少需求就像贪吃蛇一般

  一堆孤岛般的单机服务器缓存服务运维是个迷宫

  现在缓存中的数据许多并不是永远的热数据

  还有许多的开发人员对缓存技术了解有限，胡乱用的情况很多

  起初估算的不足到用时发现瓶颈了



想要的是个百变魔术箱，不管它有多大总变出一堆神奇

直接使用开源方案

- 找个开源方案直接使用
- 只是跑起来挺好看的
- 问题:

  许多方案很美但我们用起来就不美了

  单一个方案解决不了全部的问题

开源方案真的很好，但无法与我们产生感情

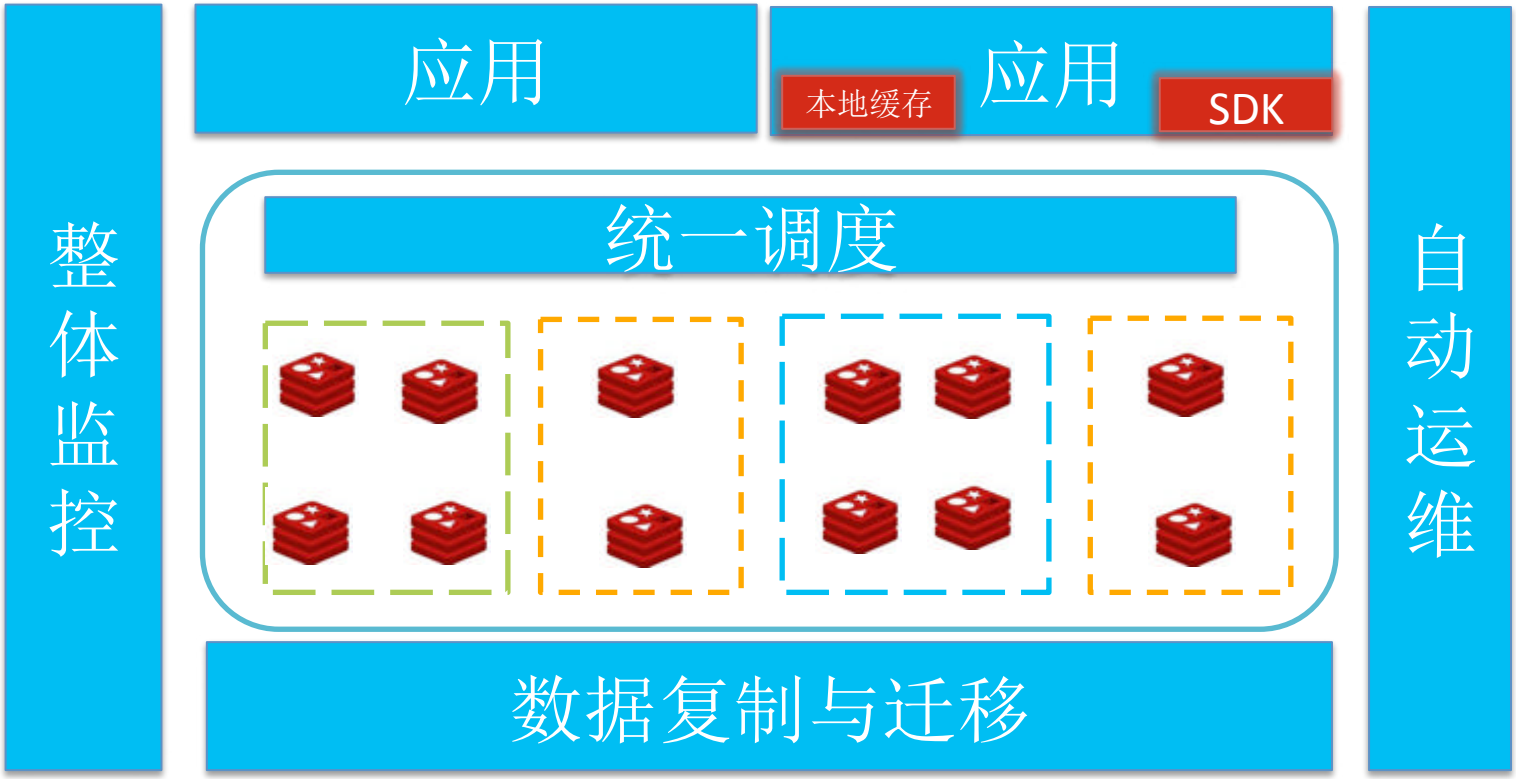
- Cachecloud：（部署与运维少了点）
- codis：（我们不想把集群做大）
- pika：（部署方式少了点）
- Twemproxy：（代理很强）

一个统一缓存平台

我们自己做一个（凤凰涅槃） **phoenix**

开始设计

- 在应用中能根据场景拆分（应用透明）
- 能从客户端调用开始全面监控
- 能防止缓存的崩塌
- 动态扩容缩容



做一个我们自己客户端

- Redis本身的服务不动
- 在客户端我们一些特性
- 应用的场景监控切换本地缓存

还要做一个代理层

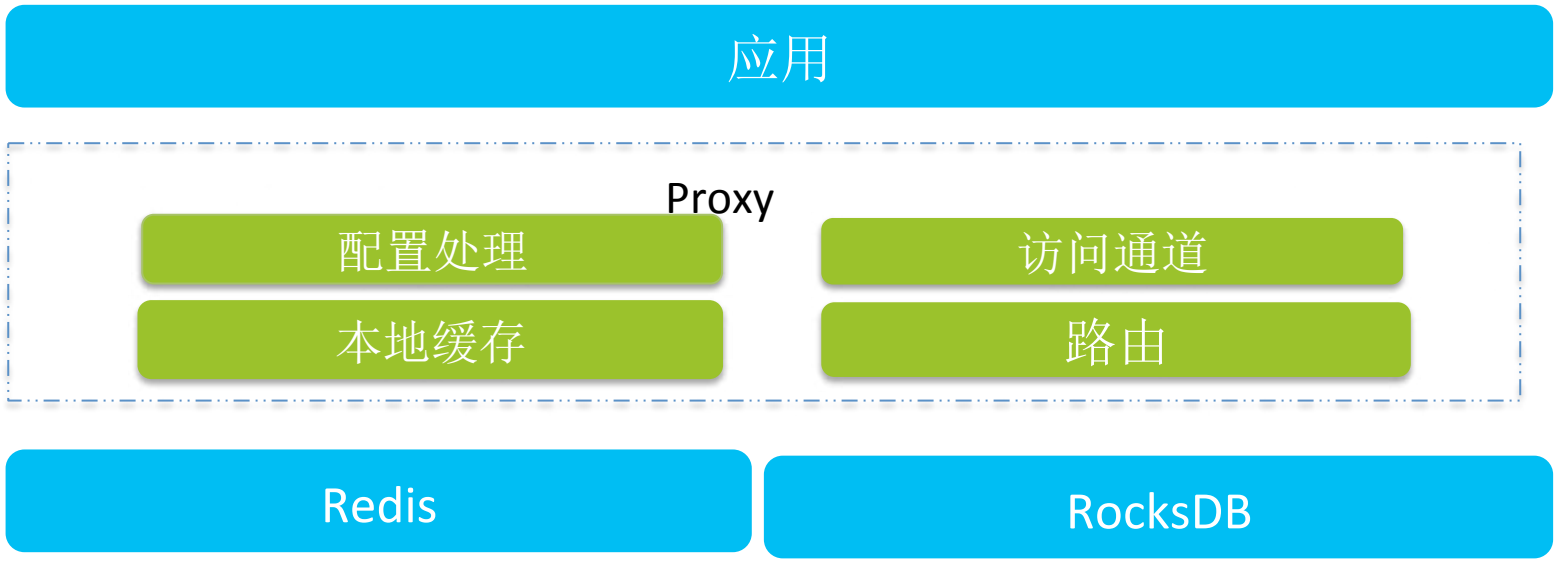
- 代理本身就是一个Redis，加入我们的实现

客户端多语言开发时间成本太大

客户端在应用中的升级是个大问题

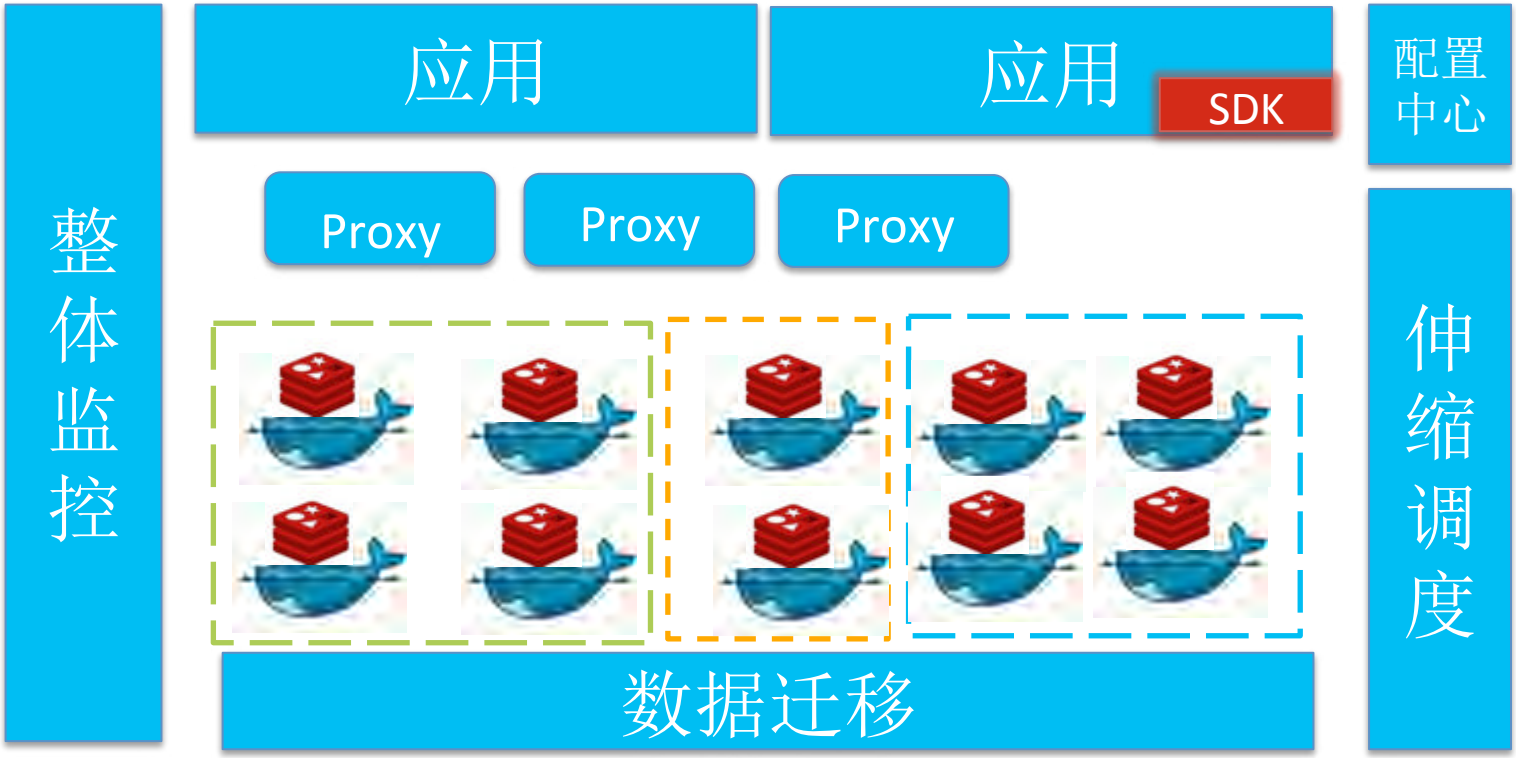
本地缓存的控制更好

部分使用频率不高的可以使用磁盘做缓存



缓存服务的架构设计

- 多个小集群+单节点
- 以场景划分集群
- 实时平衡调度数据
- 动态扩容缩容

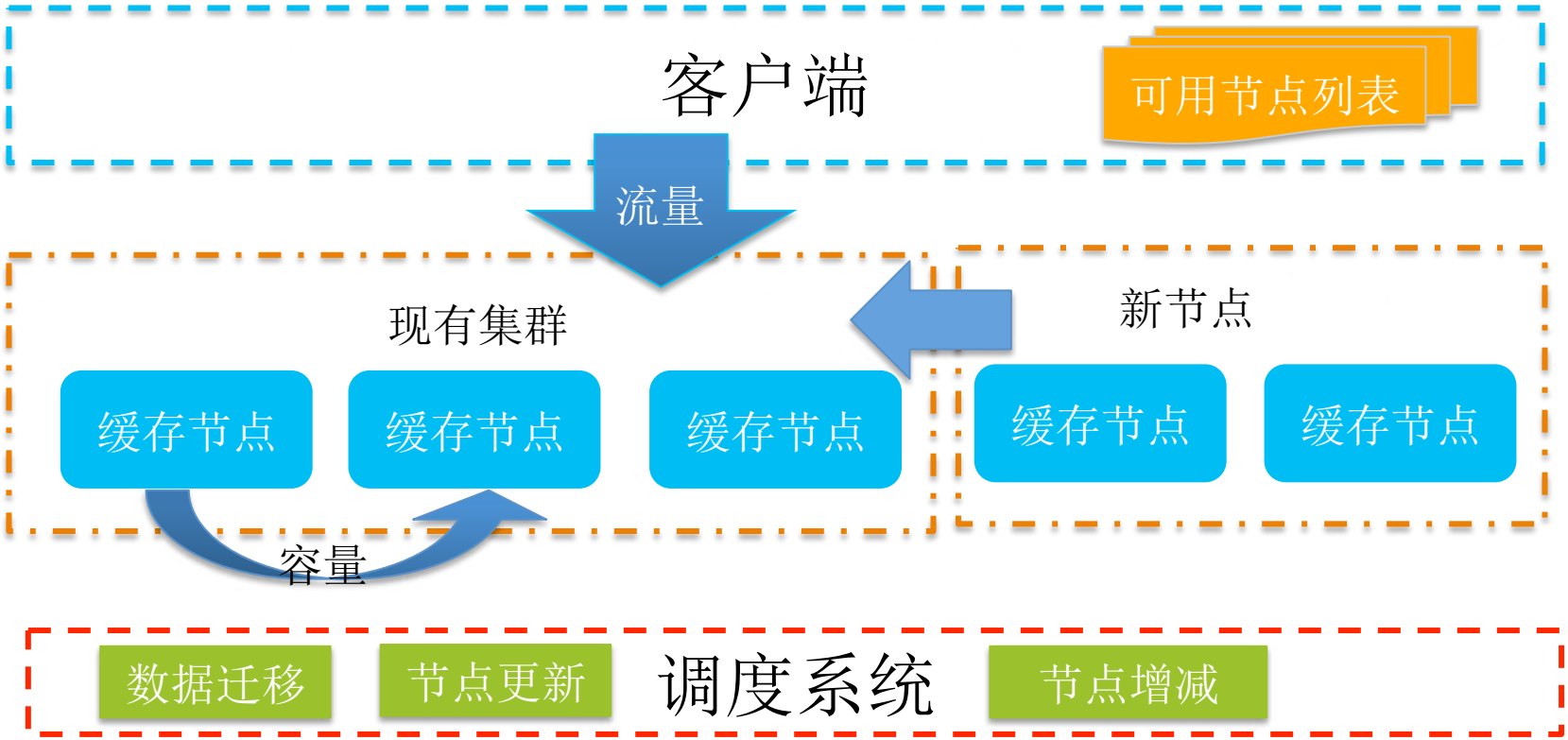


缓存服务的扩容问题

- 流量的快速增加必然带扩容的需求与压力
- 容量与流量的双扩容
- 如何做到平滑的扩容

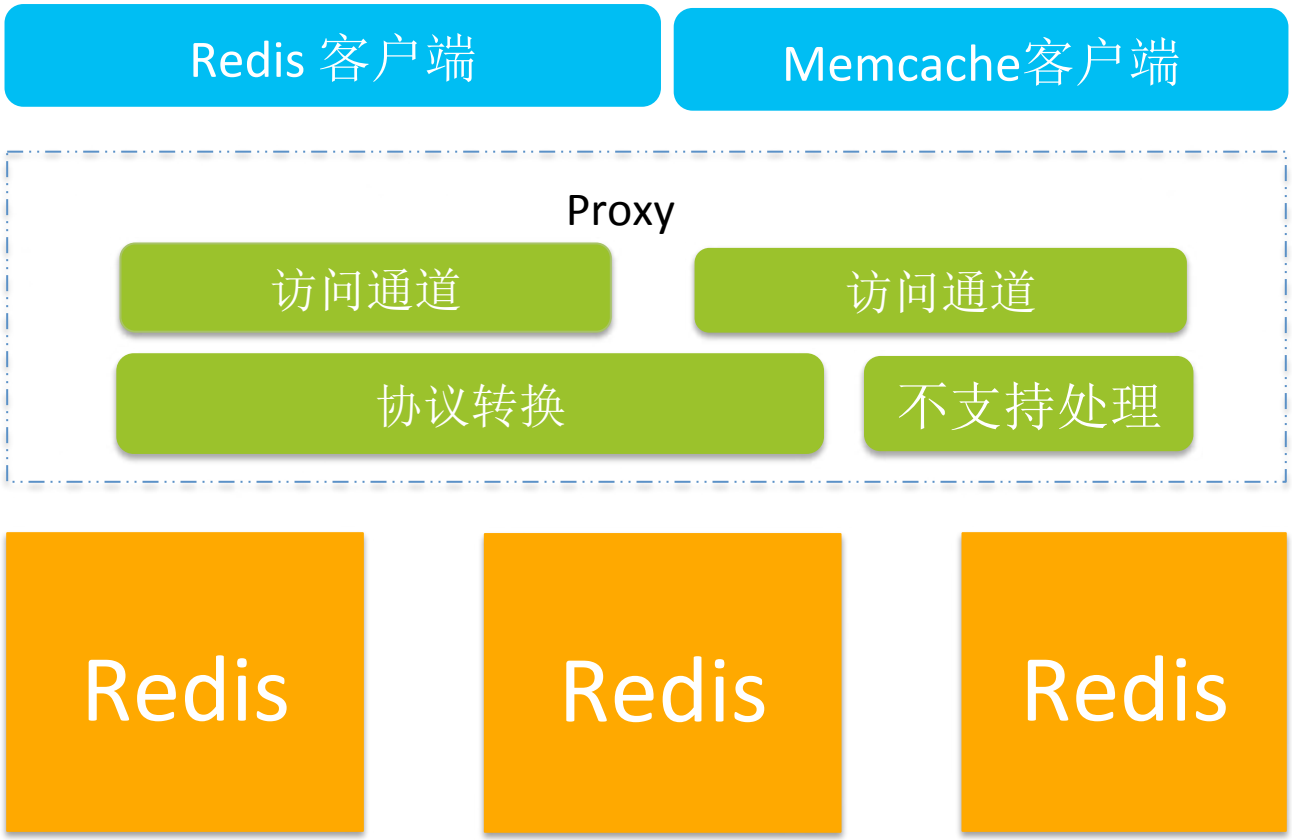
  容量动态的数据迁移(集群内部平衡，新节点增加)

  流量超出时的根据再平衡集群

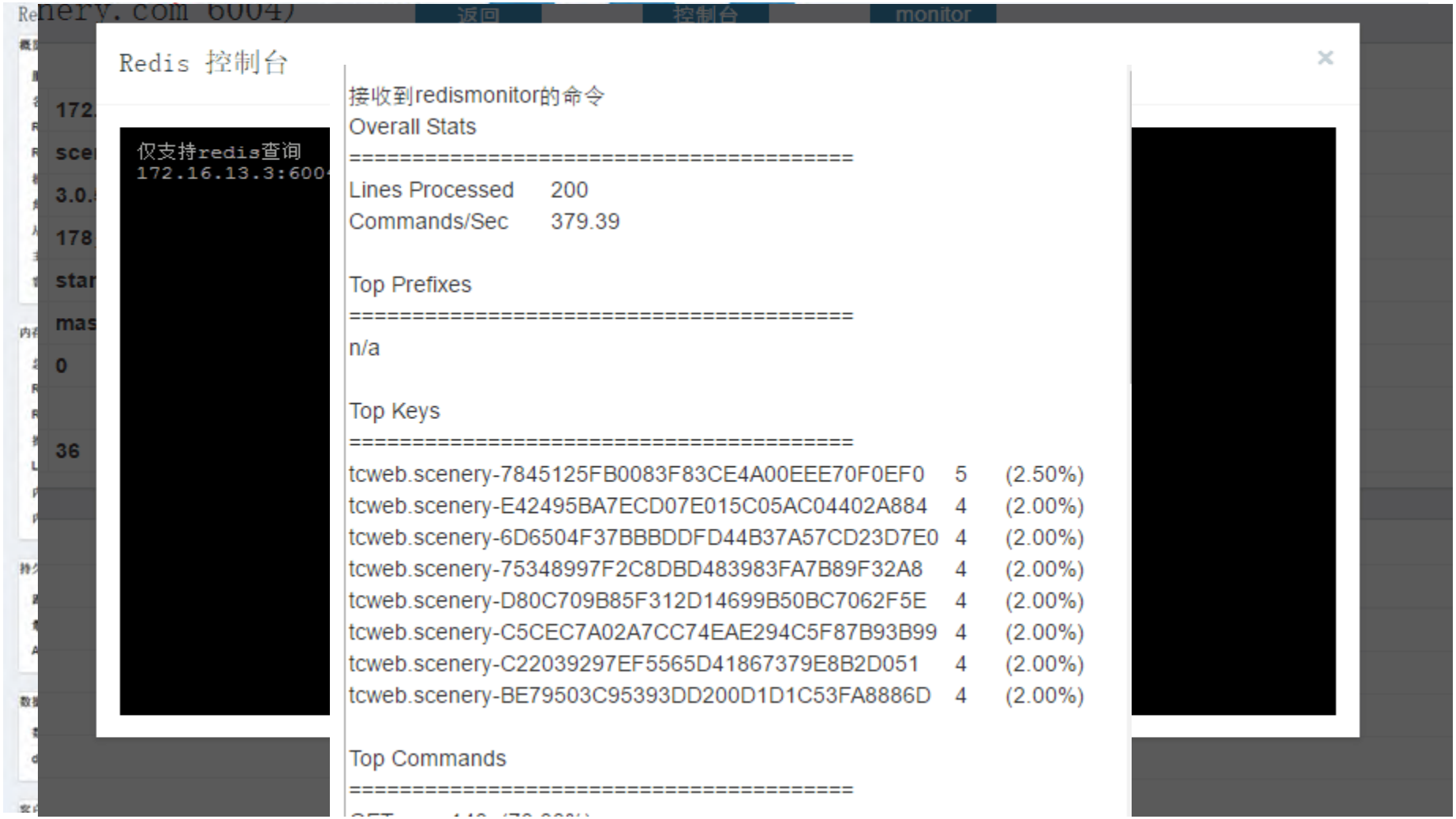


缓存服务多协议访问

- 老项目的更新代价太大
  使用代理做不同客户端的协议转换
  目前仅支持基本缓存动作



管理与监控


返回
控制台
monitor
Redis 控制台
仅支持redis查询
172.16.13.3:600
接收到redismonitor的命令
Overall Stats
==========
Lines Processed 200
Commands/Sec 379.39
Top Prefixes
==========
n/a
Top Keys
==========
tcweb.scenery-7845125FB0083F83CE4A00EEE70F0EF0 5 (2.50%)
tcweb.scenery-E42495BA7ECD07E015C05AC04402A884 4 (2.00%)
tcweb.scenery-6D6504F37BBBDDFD44B37A57CD23D7E0 4 (2.00%)
tcweb.scenery-75348997F2C8DBD483983FA7B89F32A8 4 (2.00%)
tcweb.scenery-D80C709B85F312D14699B50BC7062F5E 4 (2.00%)
tcweb.scenery-C5CEC7A02A7CC74EAE294C5F87B93B99 4 (2.00%)
tcweb.scenery-C22039297EF5565D41867379E8B2D051 4 (2.00%)
tcweb.scenery-BE79503C95393DD200D1D1C53FA8886D 4 (2.00%)
Top Commands
==========

一些业务应用场景

- 1元景点门票:需要缓存快速扩展，主要是写的量会快增加。
- 假日前火车票系统：读的量大，数据时效性大。
- 酒店的实时价格计算：秒出的价格。

SDCC2016

中国软件开发者大会

SOFTWARE DEVELOPER CONFERENCE CHINA

谢 谢！