

# 缓存弹性云平台架构及原理

--- *Kubernetes*网络选型实战与那些踩过的坑

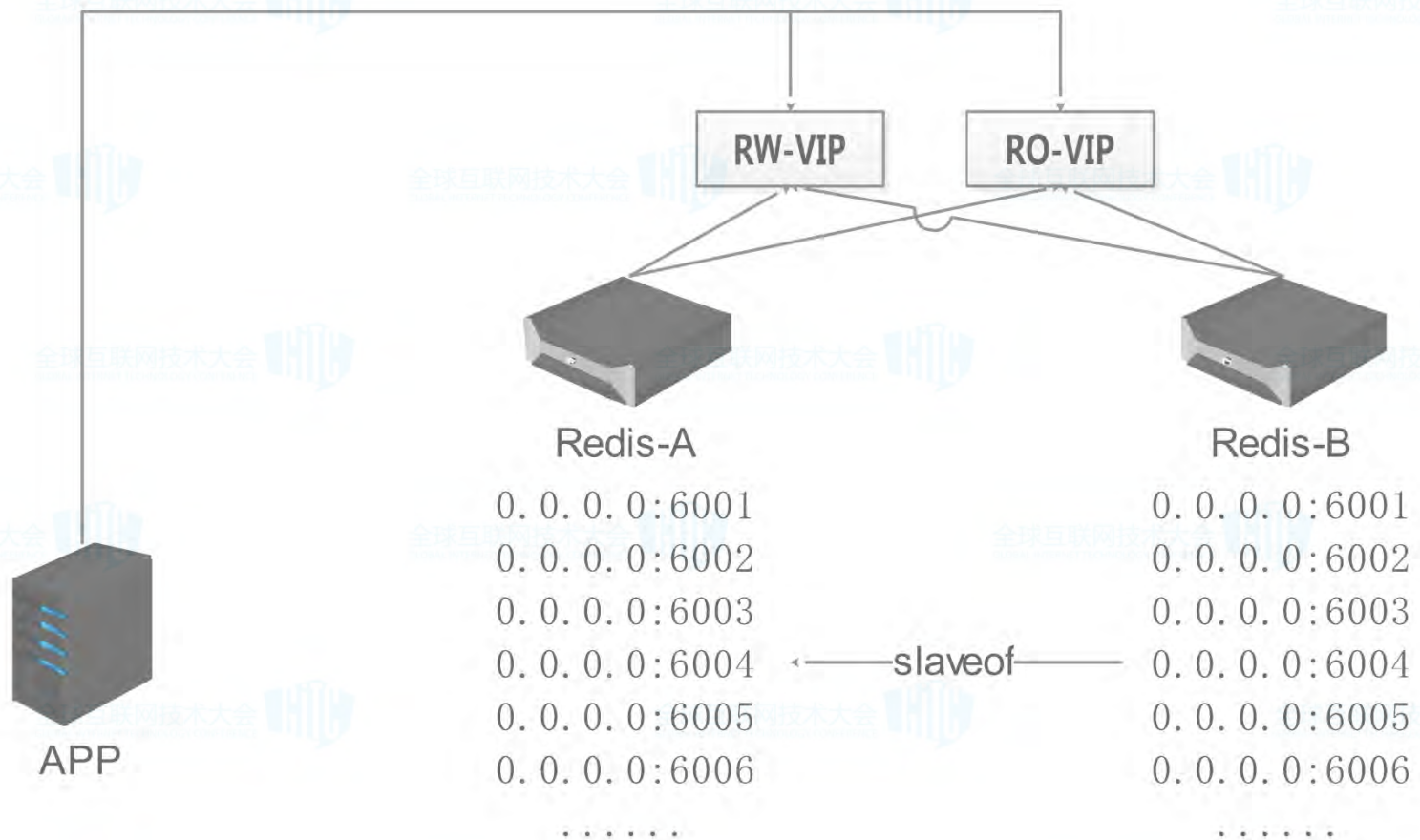
OPS: 王勇敬

2016-11-25

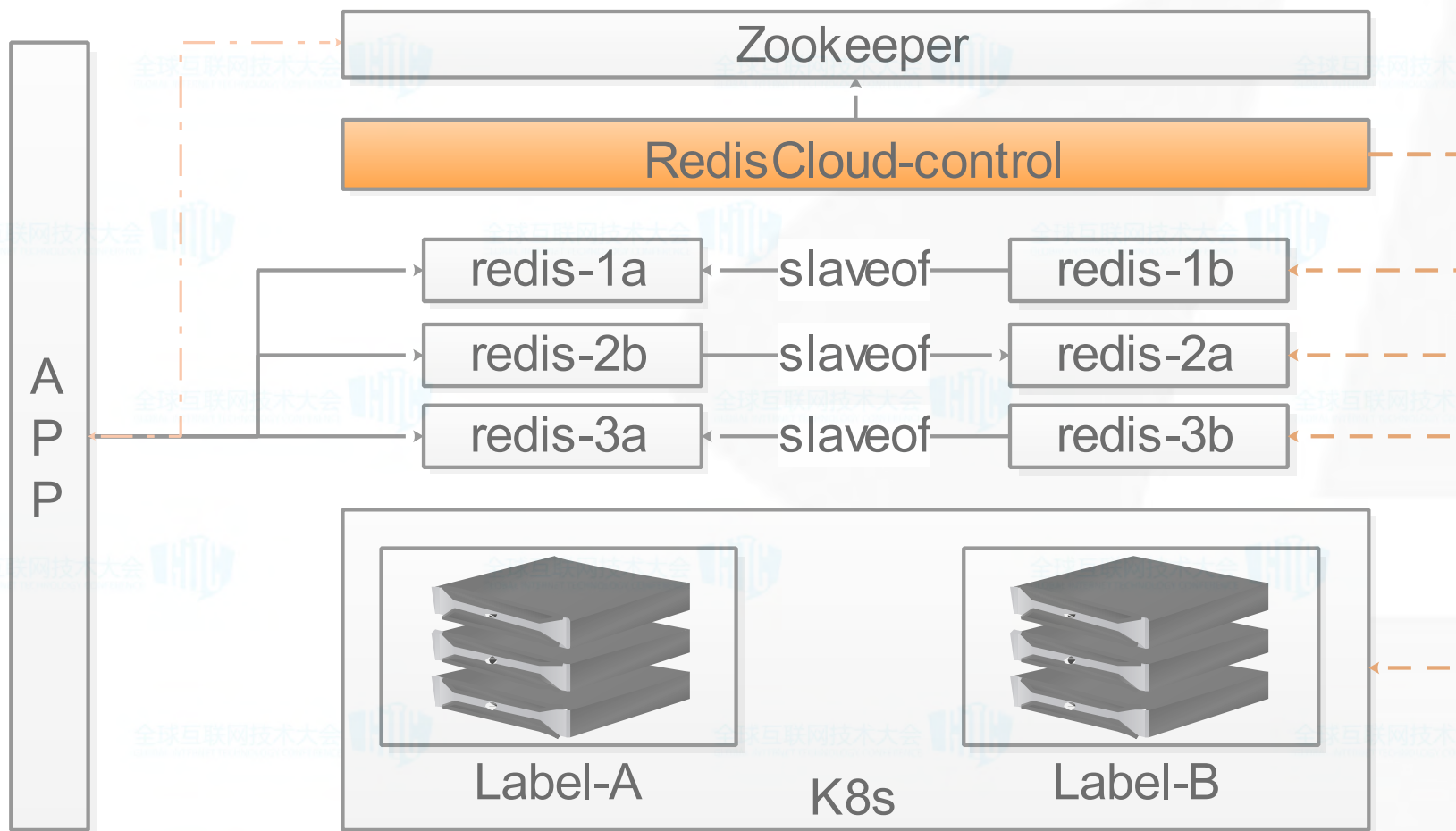
1. 持久化存储！= 有状态

2. 不持久化的有状态服务

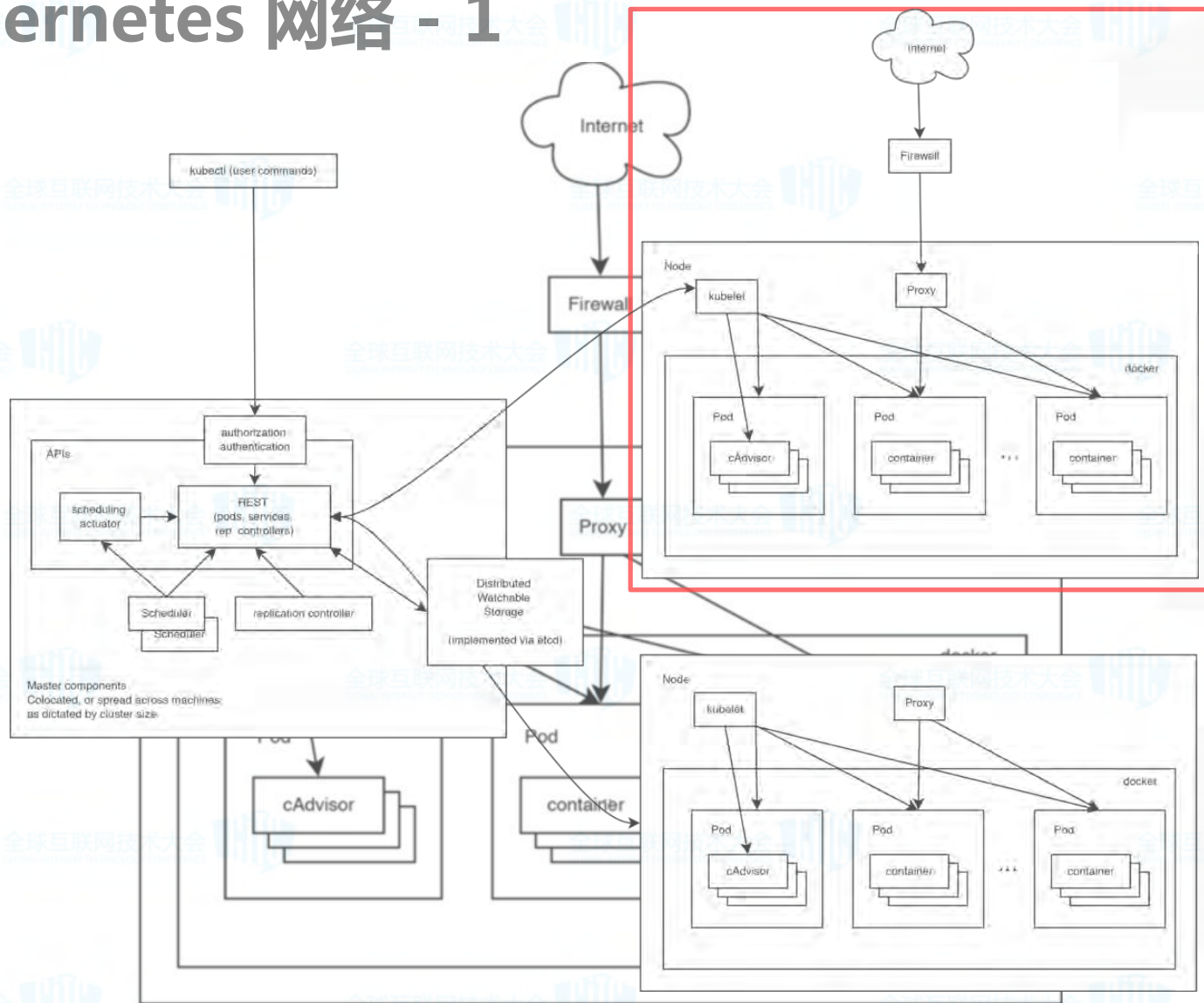
# Legacy Redis



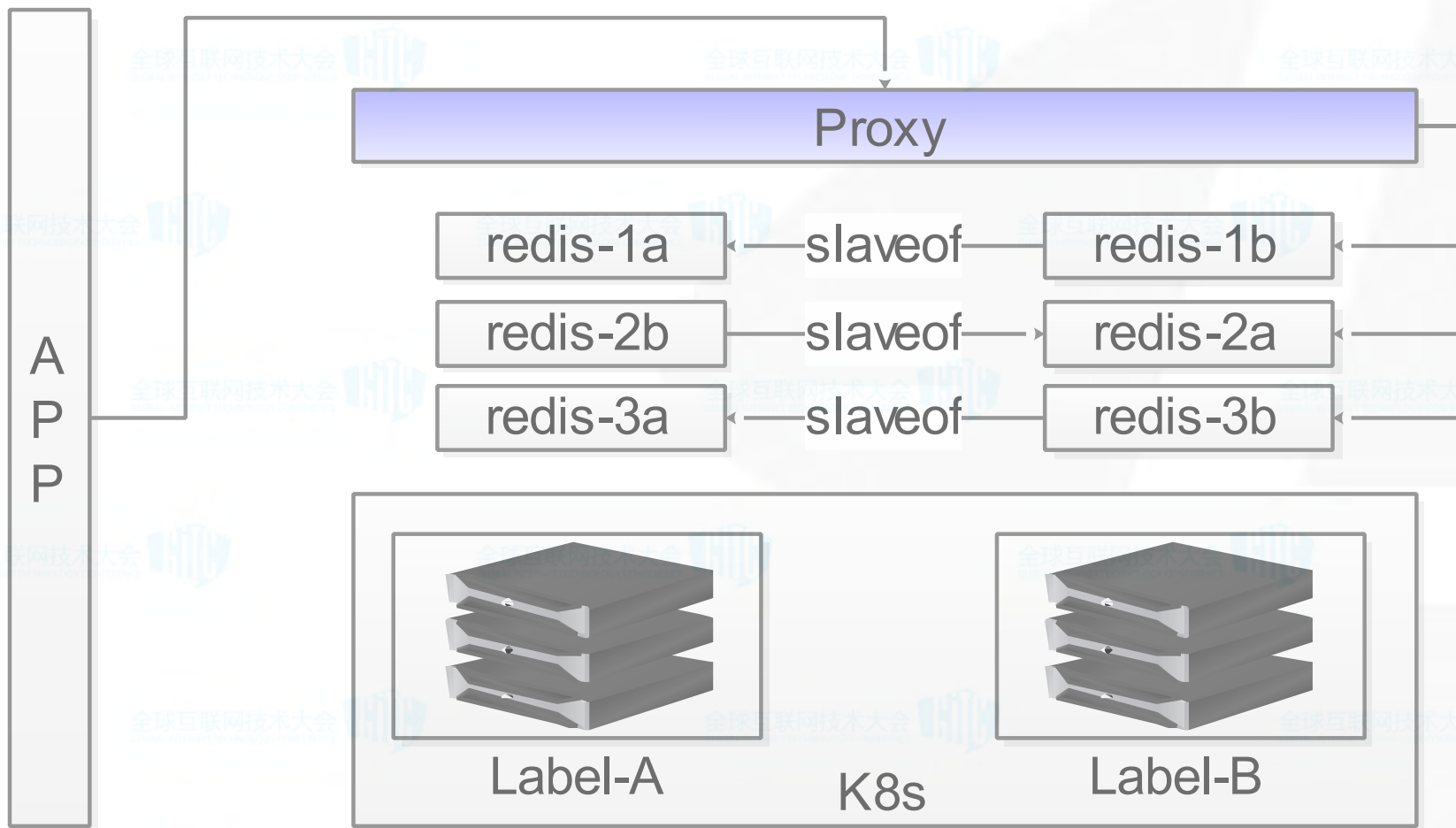
# Kubernetes + Redis ( 预期 )



# Kubernetes 网络 - 1

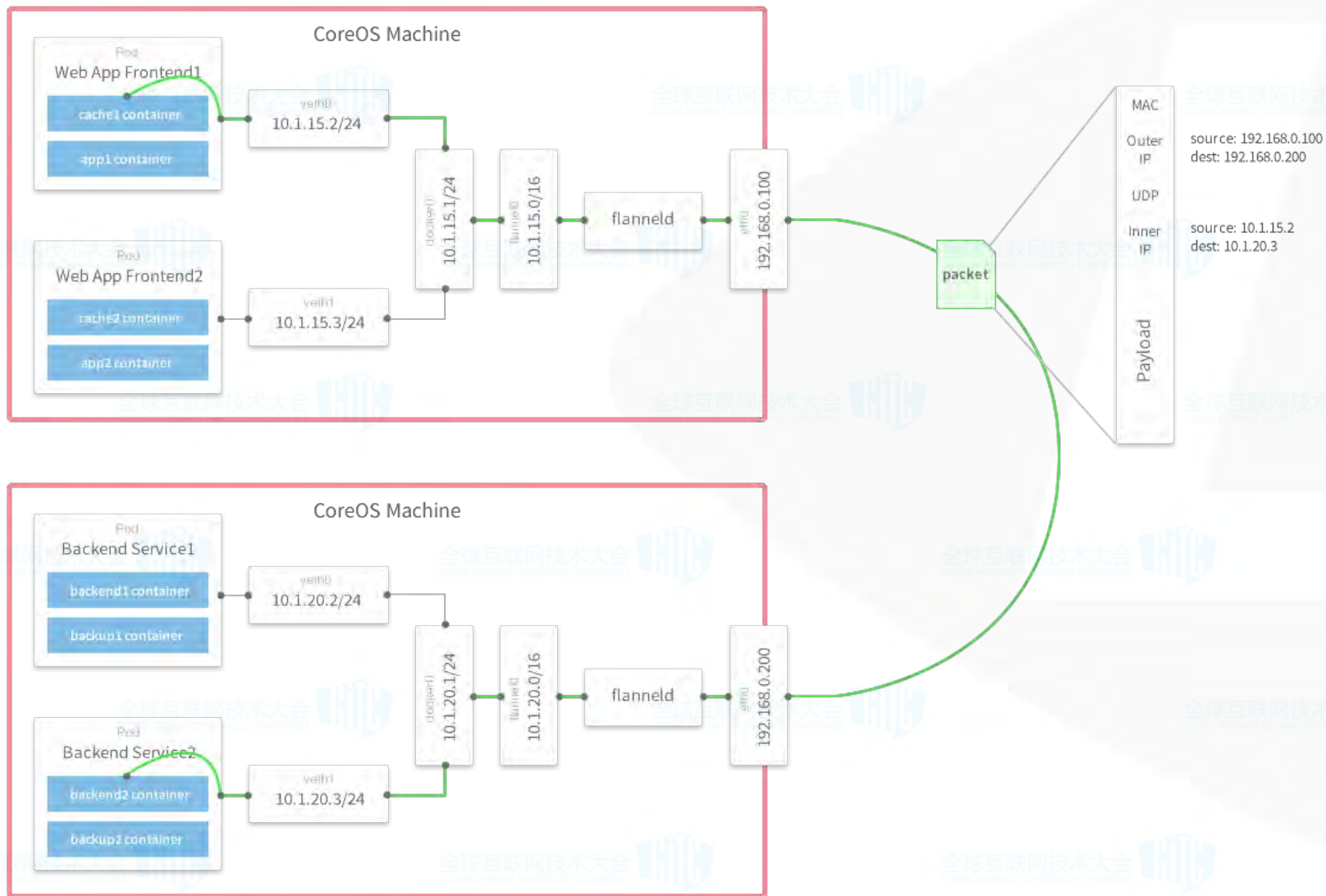


# Kubernetes 网络 - 2

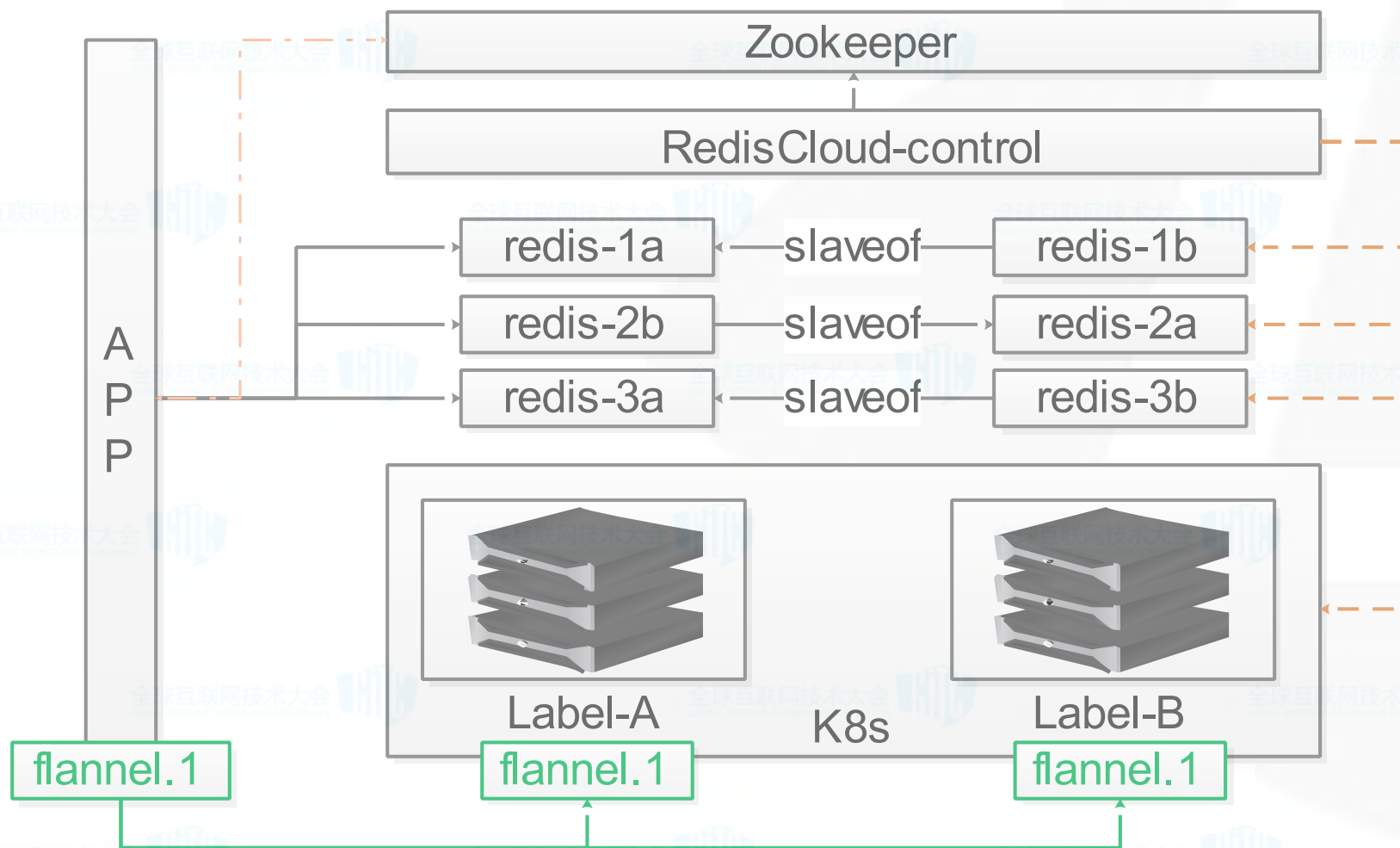




# Kubernetes 网络 - 3

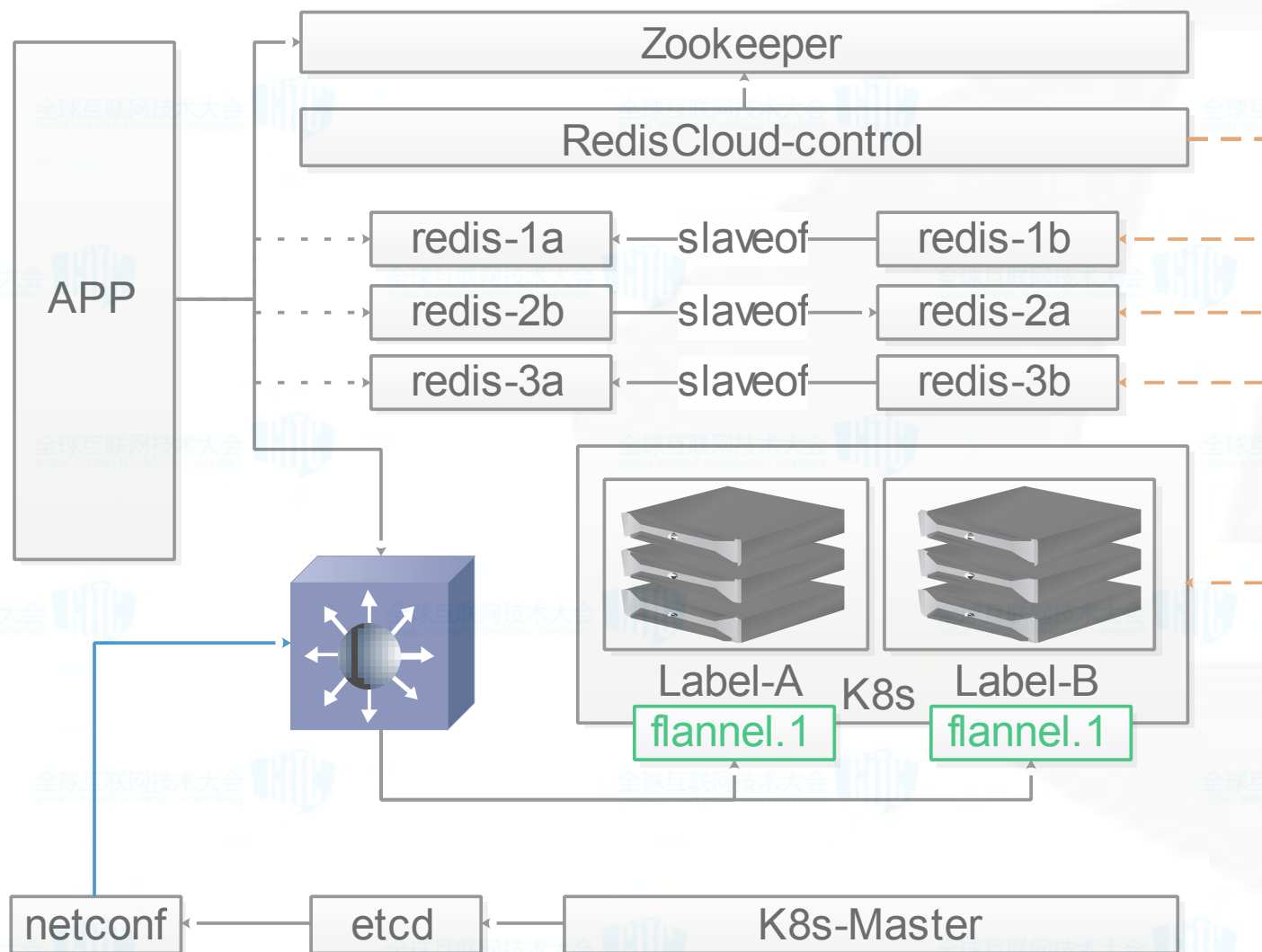


# Kubernetes 网络 - 4





# Kubernetes 网络 - 5



# Kubernetes 运维 – 1

## 1. 部署

a. CMDB

b. Ansible

## 2. 监控&报警

a. Open-Falcon

# Kubernetes 运维 – 2

## 3. 遇到的问题

- a. Flannel节点不通
- b. nf\_conntrack: table full, dropping packet
- c. Too many open files
- d. inotify cannot be used, reverting to polling: Too many open files

To be continued



# 缓存弹性云平台架构及原理

--- Redis/Memcached 弹性伸缩、自动路由、无损迁移

万韬

2016-11-25

# 缓存弹性云平台

1

## 缓存云平台的由来及特点

- 一个有温度&有思想的PaaS平台

2

## 缓存云平台核心架构

- 弹性伸缩&服务自动发现与路由

3

## Redis服务综合治理方案

- 多维度数据采样&分析&展示&报警

4

## Redis数据在线无损迁移揭秘

- 数据二阶段迁移&客户端快速平滑切换



# 1

# 缓存云平台的由来

## 待解决的痛点

- 运维人员

- 故障处理
- 维护成本
- 无场景化

- 开发人员

- 资源浪费
- 监控报警
- 弹性伸缩

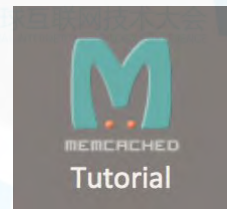


# 1

# 缓存云平台的特点

## 基于公司的私有云开发

- 物理资源统一管理，运维人员不再需要运维单个物理机和单个实例，只需关心整个资源池的管理。
- 平台实时反馈缓存服务的状态，缓存平台依据真实有效的数据随时调整缓存的节点数和空间。
- 该平台即为一个PaaS平台，提供了缓存服务的自动扩容、迁移、管理、监控、运维，以及使用情况和数据的自动治理。



## 2

# 缓存云平台核心架构

## 基于Kubernetes开发的私有云平台

猎聘云  
Cloud

视图大盘

容器管理

项目管理

集群管理

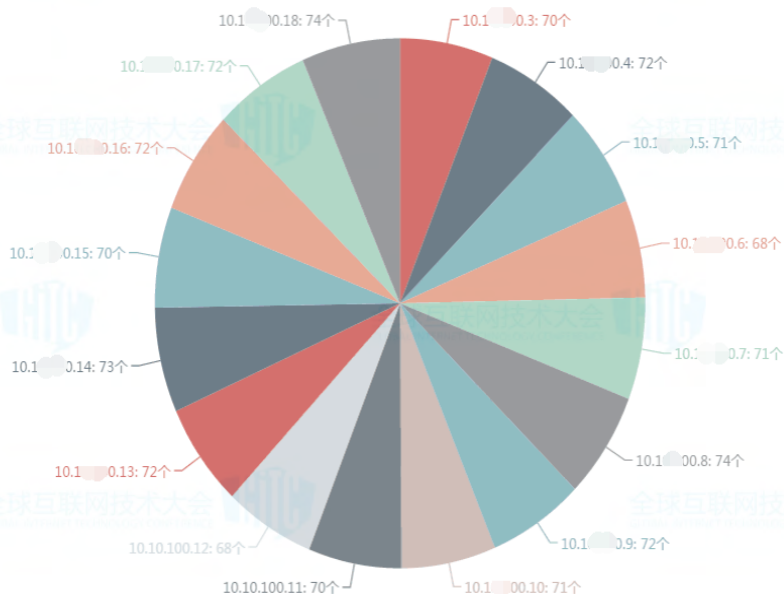
镜像管理

服务策略

日志查询

Redis, 总节点数: 16, 总实例数: 1140

节点实例数



Redis集群总负载情况: CPU总占用

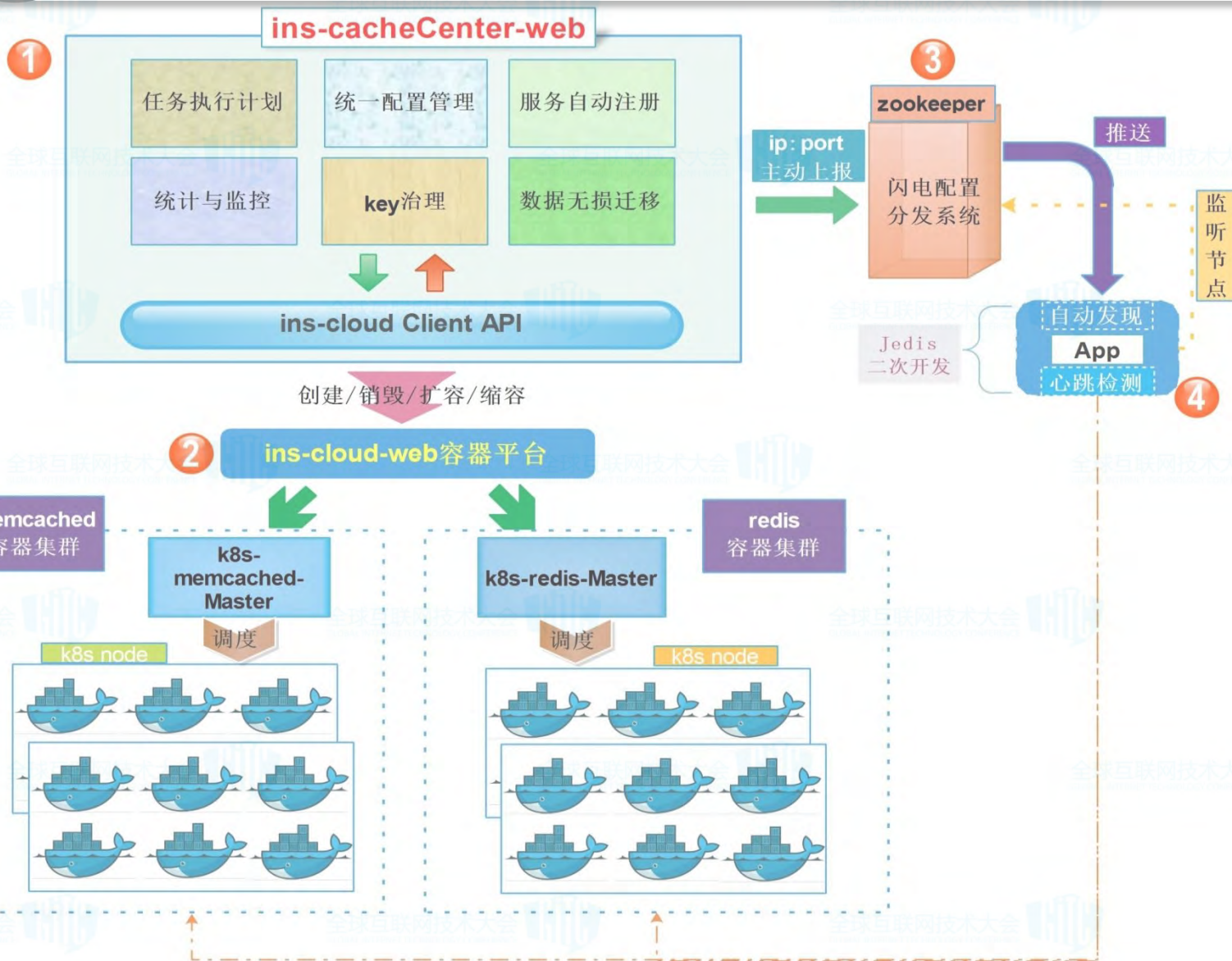


Redis集群总负载情况: 内存总占用



## 2

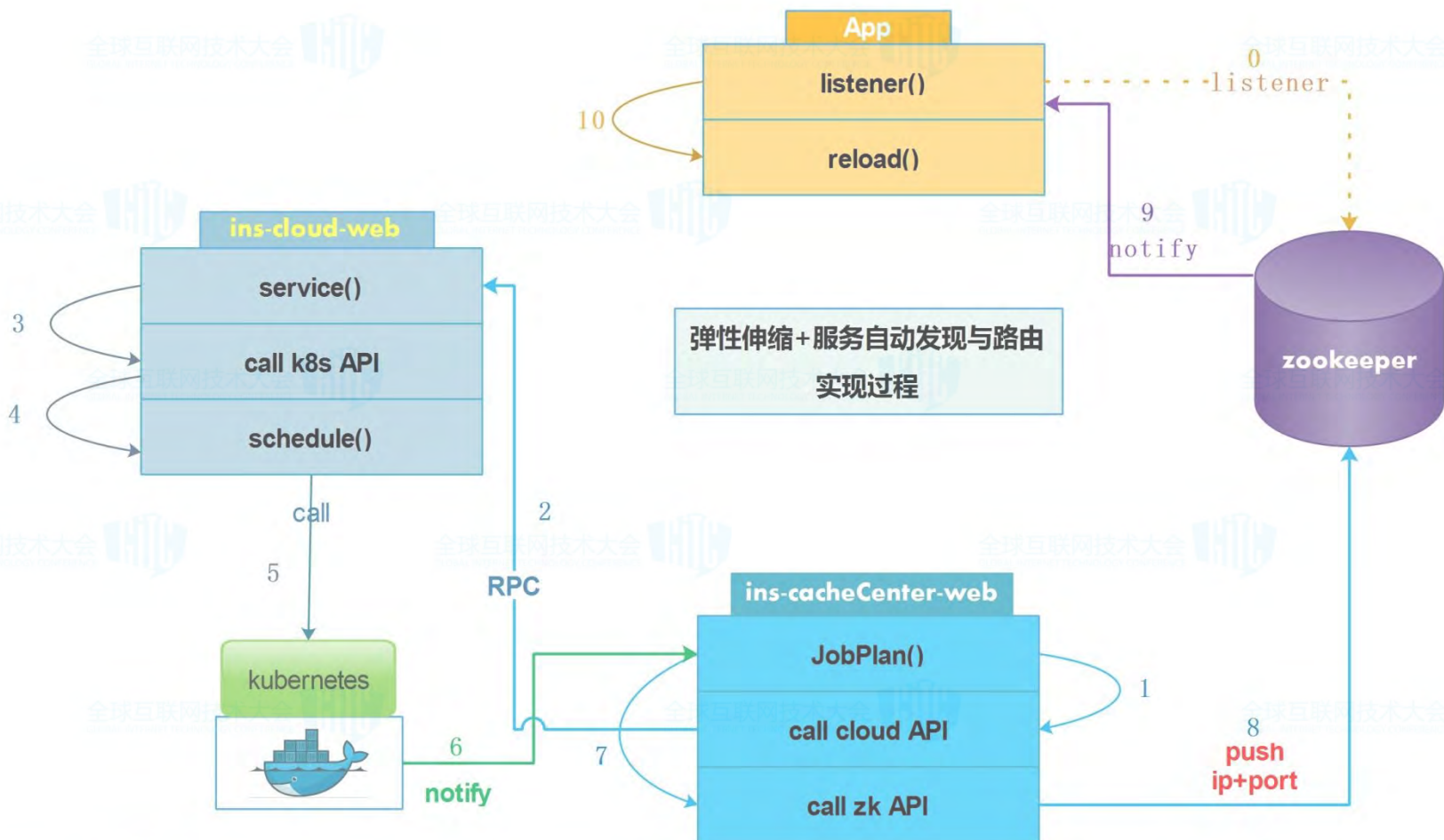
## 缓存云平台核心架构





## 2

## 缓存云平台核心架构



## • Redis服务核心治理思想

### — 方便运维人员管理集群、处理故障

- 集群容量配额统计及报警
- 容器平台时刻维护缓存实例数
- 容器平台自动转移故障节点

### — 为开发人员提供多维度的数据采集、分析与展示

- 峰值QPM环比
- 内存使用情况
- 网络流量统计
- 慢查询统计分析
- 命中率环比展示
- 全局Top统计



## 3

## Redis服务综合治理方案

- 资源统计分析

## 全局资源统计

| redis集群数 | redis实例数 | 已接入项目数 | 内存使用                     |
|----------|----------|--------|--------------------------|
| 202      | 1140     | 186    | 2662.4G Used/4096G Total |

## 资源阈值报警

| 项目名 | 内存使用                 | 对象数    | 连接数 | 命中率    | 节点数 | 操作         |
|-----|----------------------|--------|-----|--------|-----|------------|
|     | 2.2G Used/3.0G Total | 349738 | 146 | 96.23% | 4   | 快速扩容<br>扩容 |
|     | 1.5G Used/2.0G Total | 18513  | 233 | 99.97% | 4   | 扩容         |

## 资源分配不合理

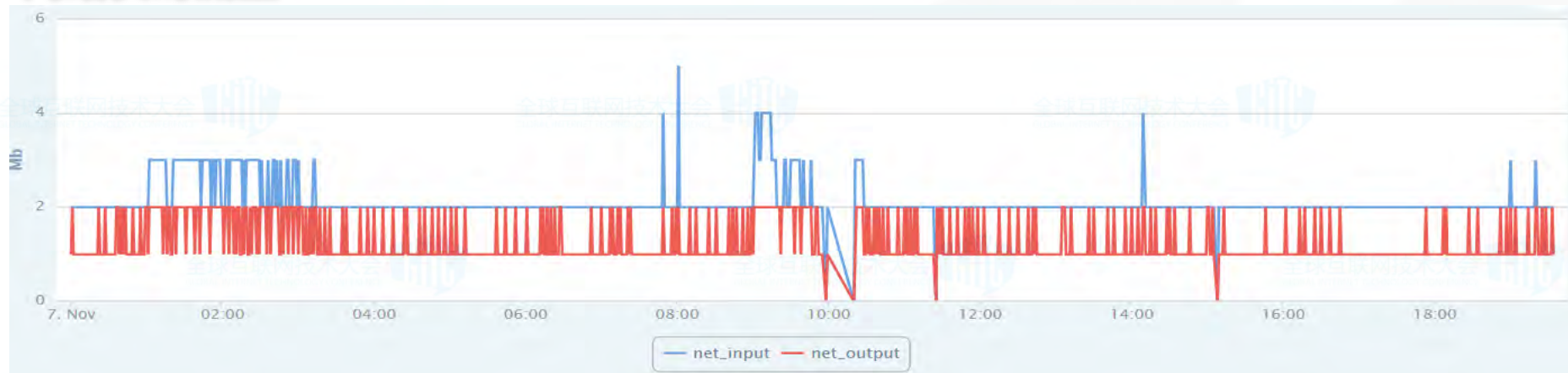
| 项目名 | 内存使用                | 对象数    | 连接数 | 命中率    | 节点数 | 操作         |
|-----|---------------------|--------|-----|--------|-----|------------|
|     | 60M Used/1.0G Total | 1903   | 71  | 95.03% | 2   | 一键缩容<br>缩容 |
|     | 94M Used/2.0G Total | 595178 | 33  | 90.55% | 2   | 缩容         |

## 3

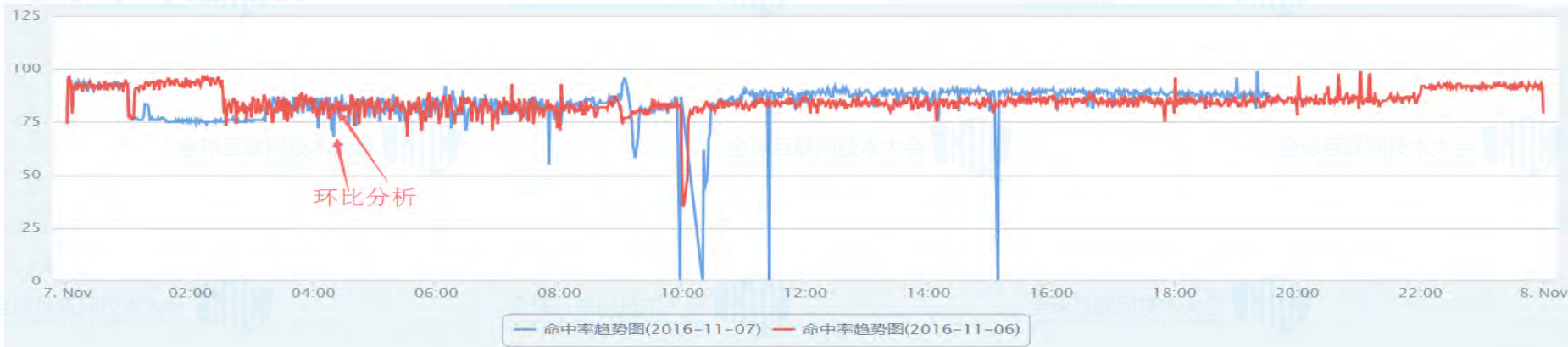
# Redis服务综合治理方案

## 请求访问实时/环比分析

### 网络实时流量



### 命中率环比分析



## 3

## Redis服务综合治理方案

## • TOP统计分析

## QPM访问最高Top

| 项目名 | 内存使用                 | QPM    | 对象数      | 连接数 | 命中率    |
|-----|----------------------|--------|----------|-----|--------|
| ... | 95M Used/2.0G Total  | 114061 | 600572   | 33  | 90.55% |
| ... | 1.6G Used/5.0G Total | 104251 | 12000433 | 240 | 81.55% |
| ... | 1.1G Used/6.0G Total | 66952  | 6134376  | 108 | 85.22% |
| ... | 359M Used/3.0G Total | 20395  | 8513     | 38  | 99.97% |

## 资源使用率最低Top

| 项目名 | 内存使用                | 对象数 | 连接数 | 命中率    |
|-----|---------------------|-----|-----|--------|
| ... | 3M Used/16.0G Total | 15  | 7   | 65.38% |
| ... | 5M Used/3.0G Total  | 3   | 4   | 86.8%  |
| ... | 3M Used/2.0G Total  | 0   | 2   | 0.0%   |

## 命中率最低Top

| 项目名 | 内存使用               | 命中率  | 对象数 | 连接数 |
|-----|--------------------|------|-----|-----|
| ... | 3M Used/2.0G Total | 0.0% | 0   | 2   |
| ... | 1M Used/1.0G Total | 0.0% | 0   | 1   |
| ... | 5M Used/3.0G Total | 0.0% | 0   | 27  |

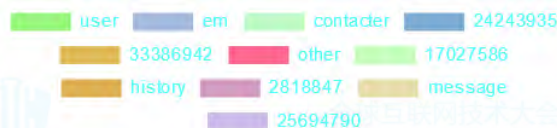
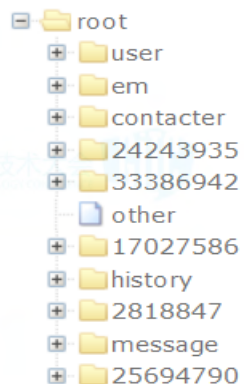
## 3

## Redis服务综合治理方案

## 缓存key治理

| key       | 总数      | 当天      |        | 三天内     |        | 一周内     |        | 一月内     |       | 三月内     |      | 三月外 |    |
|-----------|---------|---------|--------|---------|--------|---------|--------|---------|-------|---------|------|-----|----|
|           |         | 数       | 比      | 数       | 比      | 数       | 比      | 数       | 比     | 数       | 比    | 数   | 比  |
| root      | 5135819 | 1162964 | 22.64% | 1976892 | 38.49% | 4103716 | 79.9%  | 4719735 | 91.9% | 5135819 | 100% | 0   | 0% |
| user      | 79448   | 79445   | 100%   | 79444   | 99.99% | 79445   | 100%   | 79448   | 100%  | 79448   | 100% | 0   | 0% |
| em        | 1238226 | 113255  | 9.15%  | 188280  | 15.21% | 360412  | 29.11% | 822154  | 66.4% | 1238226 | 100% | 0   | 0% |
| contacter | 174317  | 61498   | 35.28% | 83967   | 48.17% | 122278  | 70.15% | 174309  | 100%  | 174317  | 100% | 0   | 0% |

key使用结构信息









## 3

## Redis服务综合治理方案

- Web SSH终端&日志

```
cacheGroupId: 48> set gitc hello
OK
cacheGroupId: 48> get gitc
hello
cacheGroupId: 48>
```

| 操作者   | 操作类型 | 操作内容           |
|---|------|----------------|
|  | 执行命令 | get gitc       |
|  | 执行命令 | set gitc hello |

|   |      |  |
|---|------|--|
| 系统  | 修改节点 | 修改节点,节点名称:12.a-o799i,ip:10.2.1.110,newIp:10.2.1.110        |
| 系统  | 修改节点 | 修改节点,节点名称:12.a-q8rko,ip:10.2.1.110,newIp:10.2.1.110        |
|  | 执行命令 | zrange message.dialogs.list.key.1.2.28723837.5864484 0 300 |
|  | 执行命令 | zrange message.dialogs.list.key.1.2.28723837.5864484 0 300 |

# 4

## 缓存数据无损迁移揭秘

### 缓存数据在线无损迁移关键步骤



### • 实现的重点及难点

- 支持Redis v2.8及以上版本的所有数据类型
- 数据迁移分为二个阶段，需要分别解析RDB和command
- 客户端“零”感知平滑切换



# 4

## 缓存数据无损迁移揭秘

### 无损迁移的思路 --- Redis主从复制原理

#### How Redis replication works

If you set up a slave, upon connection it sends a PSYNC command.

If this is a reconnection and the master has enough *backlog*, only the difference (what the slave missed) is sent. Otherwise what is called a *full resynchronization* is triggered.

When a full resynchronization is triggered, the master starts a background saving process in order to produce an RDB file. At the same time it starts to buffer all new write commands received from the clients. When the background saving is complete, the master transfers the database file to the slave, which saves it on disk, and then loads it into memory. The master will then send all buffered commands to the slave. This is done as a stream of commands and is in the same format of the Redis protocol itself.

You can try it yourself via telnet. Connect to the Redis port while the server is doing some work and issue the SYNC command. You'll see a bulk transfer and then every command received by the master will be re-issued in the telnet session.

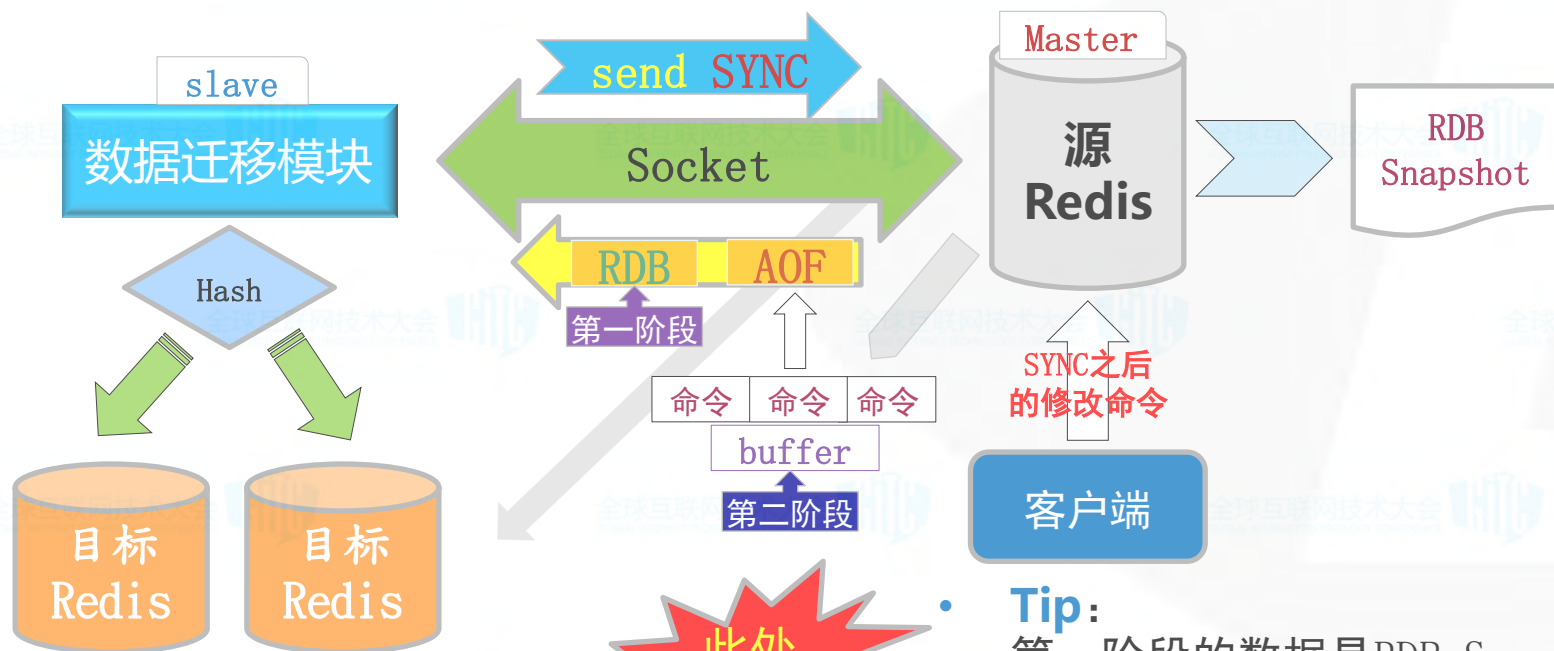
Slaves are able to automatically reconnect when the master-slave link goes down for some reason. If the master receives multiple concurrent slave synchronization requests, it performs a single background save in order to serve all of them.

- 1、只要是给Master发送SYNC命令，就等同于“Slave”
- 2、数据迁移分为两个阶段:RDB快照和buffer增量数据

## 4

## 缓存数据无损迁移揭秘

## 扩容Redis+数据迁移过程



- **Tip:** 第一阶段的数据是RDB Snapshot，返回的是二进制流，需要按RDB格式依次解析；
- 第二阶段属于增量数据，返回的是命令，解析时按行读取。
- 必须先迁移RDB快照再迁移增量数据

## 4

# 缓存数据无损迁移揭秘

## Redis缓冲区限制

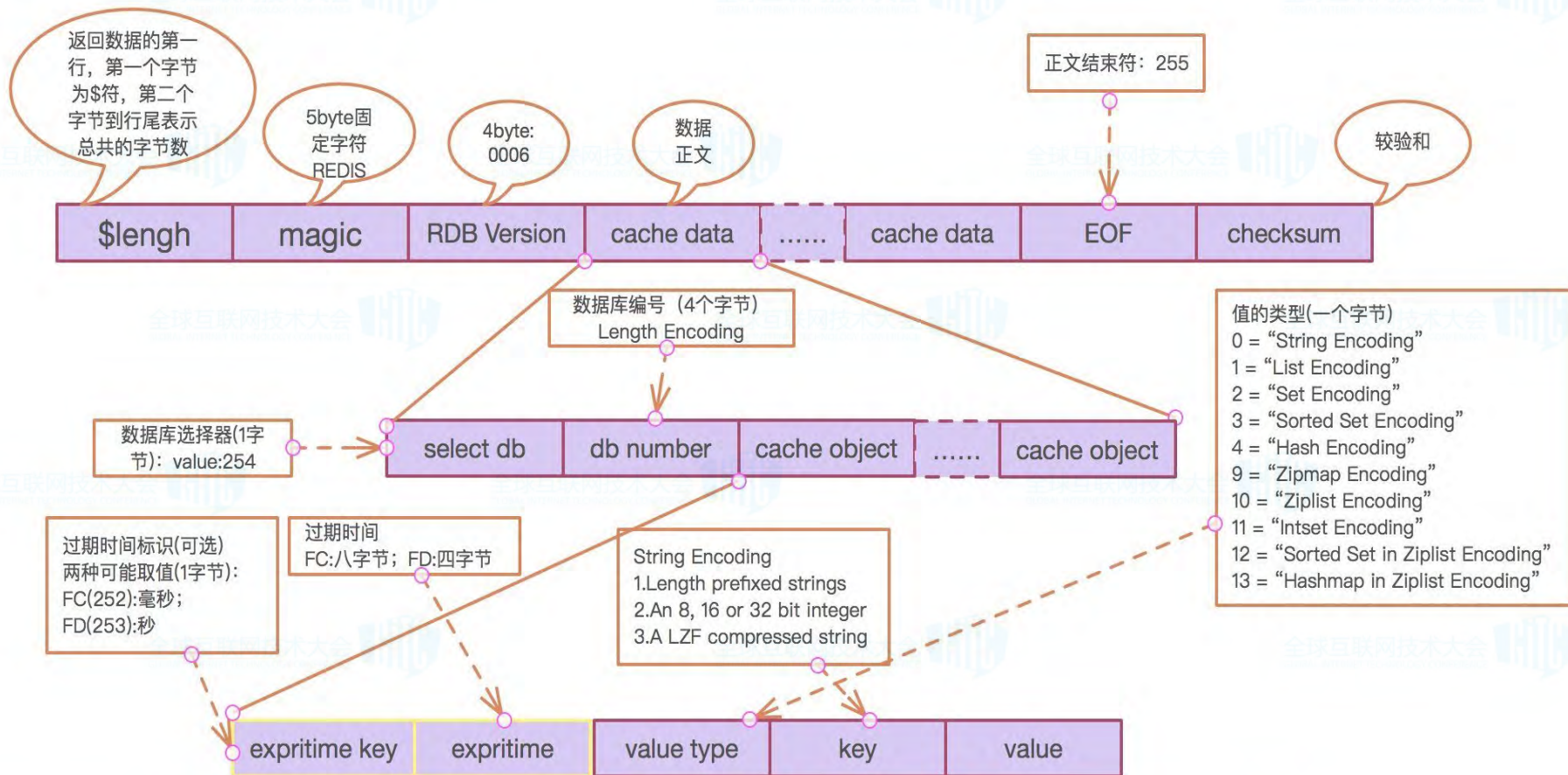
```
client-output-buffer-limit normal 0 0 0
client-output-buffer-limit slave 256mb 64mb 60
client-output-buffer-limit pubsub 32mb 8mb 60
```

- **趟过的坑：**在RDB数据未传完之前，Master接受到的指令不会立刻发送给slave，先会放在output buffer中。重点关注client-output-buffer-limit，含义是如果buffer超过256m或者连续60秒超过64m，连接就会被立刻强行关闭！！！！
- **解决方案：**
- 限制单个Redis实例的内存不要超过8G
- 迁移时先把RDB文件持久化至本地磁盘后，立即取走buffer中的增量数据，防止buffer区占满

## 4

## 缓存数据无损迁移揭秘

## Redis RDB文件格式

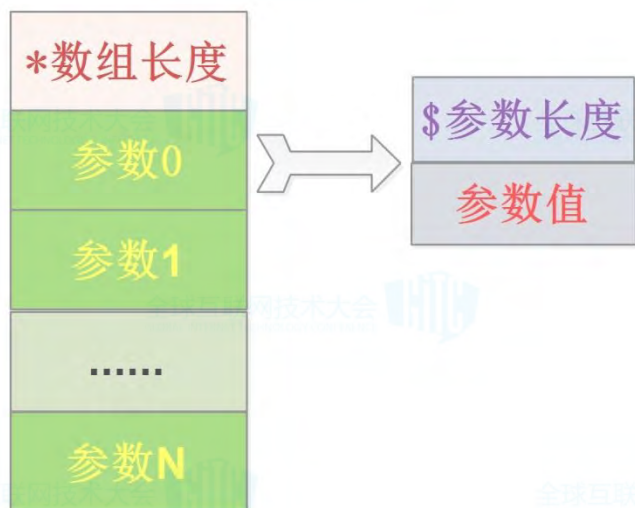




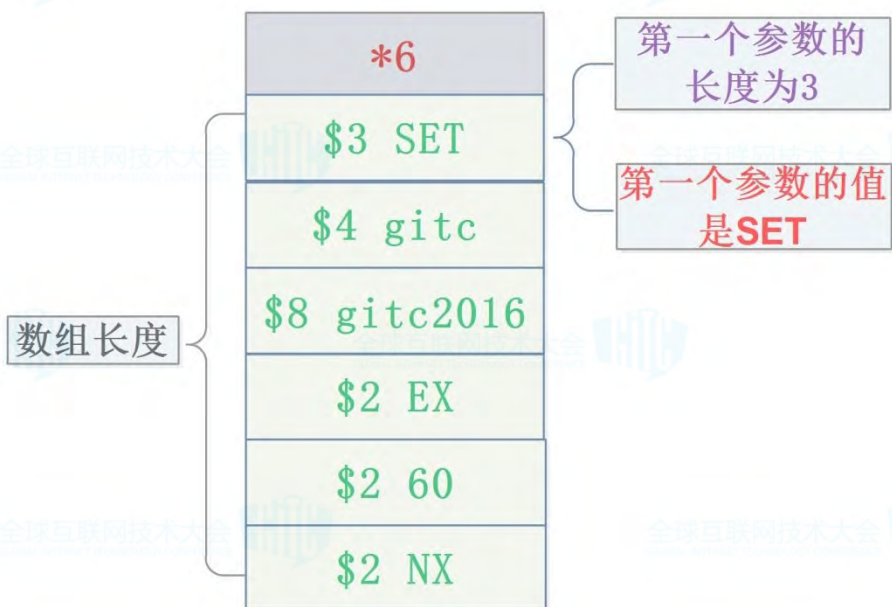
## 4

## 缓存数据无损迁移揭秘

## Buffer增量数据格式



```
SET gitc gitc2016 EX 60 NX
```



全球互联网技术大会  
GLOBAL INTERNET TECHNOLOGY CONFERENCE



全球互联网技术大会  
GLOBAL INTERNET TECHNOLOGY CONFERENCE



全球互联网技术大会  
GLOBAL INTERNET TECHNOLOGY CONFERENCE



全球互联网技术大会  
GLOBAL INTERNET TECHNOLOGY CONFERENCE



全球互联网技术大会  
GLOBAL INTERNET TECHNOLOGY CONFERENCE



全球互联网技术大会  
GLOBAL INTERNET TECHNOLOGY CONFERENCE



全球互联网技术大会  
GLOBAL INTERNET TECHNOLOGY CONFERENCE



全球互联网技术大会  
GLOBAL INTERNET TECHNOLOGY CONFERENCE



全球互联网技术大会  
GLOBAL INTERNET TECHNOLOGY CONFERENCE



全球互联网技术大会  
GLOBAL INTERNET TECHNOLOGY CONFERENCE



全球互联网技术大会  
GLOBAL INTERNET TECHNOLOGY CONFERENCE



大会



全球互联网技术大会  
GLOBAL INTERNET TECHNOLOGY CONFERENCE



全球互联网技术大会  
GLOBAL INTERNET TECHNOLOGY CONFERENCE



全球互联网技术大会  
GLOBAL INTERNET TECHNOLOGY CONFERENCE



全球互联网技术大会  
GLOBAL INTERNET TECHNOLOGY CONFERENCE



全球互联网技术大会  
GLOBAL INTERNET TECHNOLOGY CONFERENCE



大会



全球互联网技术大会  
GLOBAL INTERNET TECHNOLOGY CONFERENCE



全球互联网技术大会  
GLOBAL INTERNET TECHNOLOGY CONFERENCE



全球互联网技术大会  
GLOBAL INTERNET TECHNOLOGY CONFERENCE



# THANKS

