

云智未来<sup>9th</sup>

第九届中国系统架构师大会  
SYSTEM ARCHITECT CONFERENCE CHINA 2017

# 大容量redis存储方案--Pika

陈宗志

360基础架构组技术经理

SACC  
2017

北京·新云南皇冠假日酒店

IT168.com

ChinaUnix

ITPUB

## 简介

- 13年入职360 基础架构组
  - Bada
  - Pika
  - Zeppelin
  - Mario, Pink, slash, floyd
- <https://github.com/Qihoo360>

SACC2017

# 概要

- 存在问题
- 分析问题
- 解决问题
- Pika vs redis

SACC2017

# Introduction

- Pika 是DBA 和 基础架构团队一起设计开发的大容量redis的解决方案
- 完全兼容redis 协议, 用户不需要修改任何代码进行迁移

## Pika User

- Redis实例数量：6000+个
- 日访问量：5000+亿
- Pika数据数量：1000+个
- 日访问量：1000+亿
- 覆盖率：80%以上业务线
- 单份数据体积：6.8T

## UserList



More

## Pika 定位

Pika 的出现并不是为了替代 Redis , 而是 Redis 的场景补充。

Pika 力求在完全兼容 Redis 协议、继承 Redis 便捷运维设计的前提下通过持久化存储的方式解决 Redis 在大容量场景下的问题

# Redis 问题

- 恢复时间长
- 一主多从, 主从切换代价大
- 缓冲区写满问题
- 成本问题

# Redis 问题

- 恢复时间长
  - 50G redis 回复时间70分钟
  - 同时开启aof 和 rdb

# Redis 问题

- 一主多从, 主从切换代价大
  - 主库挂掉后升级从库, 所有的从库全部重传数据

# Redis 问题

- 缓冲区写满问题

- 内存是昂贵资源, 缓冲区一般设置2G
- 网络原因很容易将数据堵死, 那么就会发生大量数据重传

# Redis 问题

- 内存太贵

- 线上使用的redis 机器是 64G, 96G. 只使用 80% 的空间.
- 如果一个redis 的实例是50G, 那么基本一台机器只能运行一个redis 实例. 特别的浪费资源

# Redis 问题



戴尔 (DELL) 原盒服务器工作站ECC内存条 8G 16G 32G DDR4 2400 RDIMM  
32G DDR4 2400MHz

原盒包装 正品行货 7天无理由退换货 因商品批次不同, 产品外观有所差异, 请以实物为准

原价: **¥2899.00** 降价通知 累计评价 40+

优惠券: **满2000减50** **满1000减30** **满500减20** 更多

增值业务: **以旧换新, 闲置回收**

配送至: 北京朝阳区四环到五环之间 **有货** 支持 送运费险 **由地升戴尔专营店** 从江苏南京市发货, 并提供售后服务

选择版本: UDIMM 8G DDR4 2133MHz | RDIMM 8G DDR4 2400MHz | RDIMM 16G DDR4 2400MHz | RDIMM 32G DDR4 2400MHz

增值服务: 服务送鼠标 ¥59 | 换新保1年 ¥189 | 换新保1年 ¥149

白条分期: 不分期 | ¥980.83×3期 | ¥497.66×6期 | ¥135.29×24期

**90/GB VS 2.6/GB**

**30倍的差距**



三星(SAMSUNG) 850 EVO 250G SATA3 固态硬盘

高速读写, 质保五年, 三星V-NAND技术, 快速开机! 十年质保, 企业级SSD>>>

京东价: **¥668.00** 降价通知 累计评价 8.7万+

**¥645.00** PLUS PLUS会员专享价 银牌及以上用户开通PLUS可享限时特惠 >>

促销 **限制** 此价格不与套装优惠同时享受

增值业务: **以旧换新, 闲置回收** **礼品包装**

配送至: 北京朝阳区四环到五环之间 **有货** 支持 99元免基础运费(20kg内) 货到付款 京准达 | 夜间配

由 **京东** 发货, 供应商提供售后服务。11:10前下单, 预计今天(09月27日)送达。

重量 0.1kg

选择颜色: 750 EVO | **850 EVO** | 850 EVO | 850 PRO | 960 EVO | 960 PRO

选择版本: SATA-3 | M.2 | M.2 NVMe | MSATA

容量: 4 | 5 | 120-128G | **250-256G** | 500-512G | 1TB | 2TB | 4TB

# 问题分析

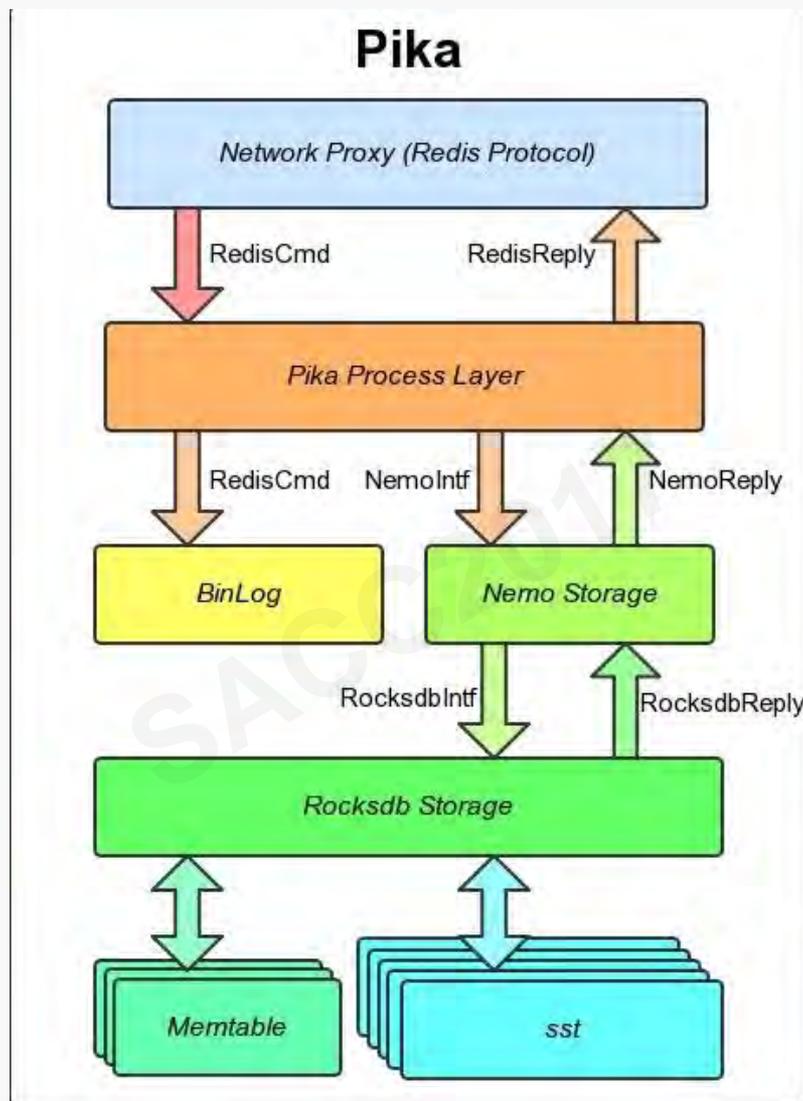
- 成本问题
- 可用性问题
- 同步问题
- 易用性问题

SACC2017

## 问题分析

- 尽可能兼容redis 协议
- 使用基于磁盘的存储引擎rocksdb 实现多数据接口接口
- 网络库
- 添加binlog 模块

# Pika 整体结构



## 网络模块--Pink

- 基础架构团队开发网络编程库, 支持pb, redis, pg, http等协议.
- 抽象各种不同类型线程
  - DispatchThread
  - WorkThread
  - BGThread
- <https://github.com/Qihoo360/pink>

SACC2017

## 网络模块--Pink

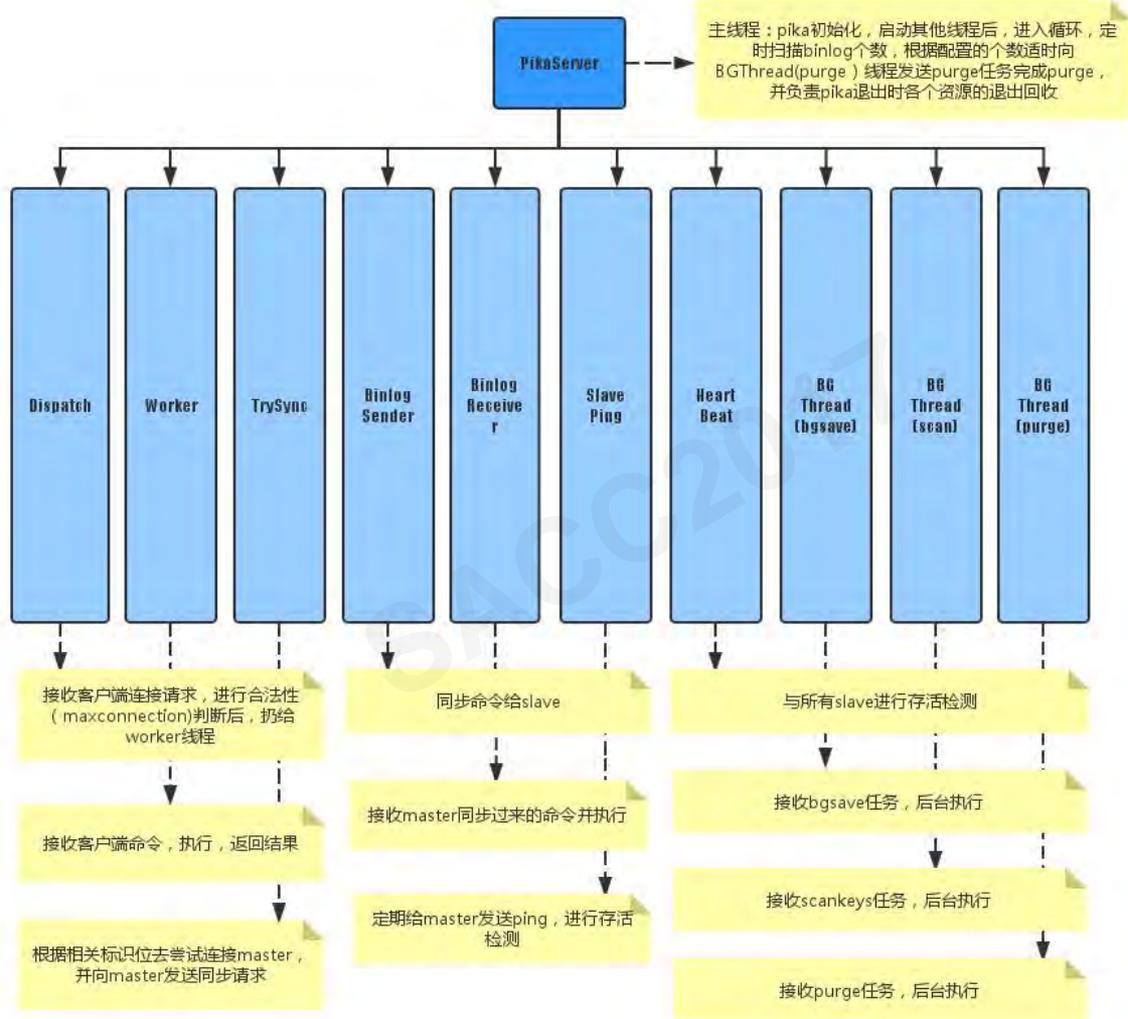
- 稳定行, 在各个项目中使用4年多
- 易用性
- 高性能

SACC2017

# 网络模块--Pink

```
class MyPbConn : public pink::PbConn {
Public:
    MyPbConn(int fd, std::string ip_port, pink::Thread* self_thread_ptr = NULL) : pink::PbConn(fd, ip_port) {
        res_ = dynamic_cast<google::protobuf::Message*>(&message_);
    }
    ~MyPbConn() {}
    int DealMessage() {
        message_.ParseFromArray(rbuf_ + cur_pos_ - header_len_, header_len_);
        message_.set_name("hello " + message_.name());
        uint32_t u = htonl( message_.ByteSize());
        memcpy(static_cast<void*>(wbuf_), static_cast<void*>(&u), COMMAND_HEADER_LENGTH);
        message_.SerializeToArray(wbuf_ + COMMAND_HEADER_LENGTH, PB_MAX_MESSAGE);
        set_is_reply(true);
    }
}
```

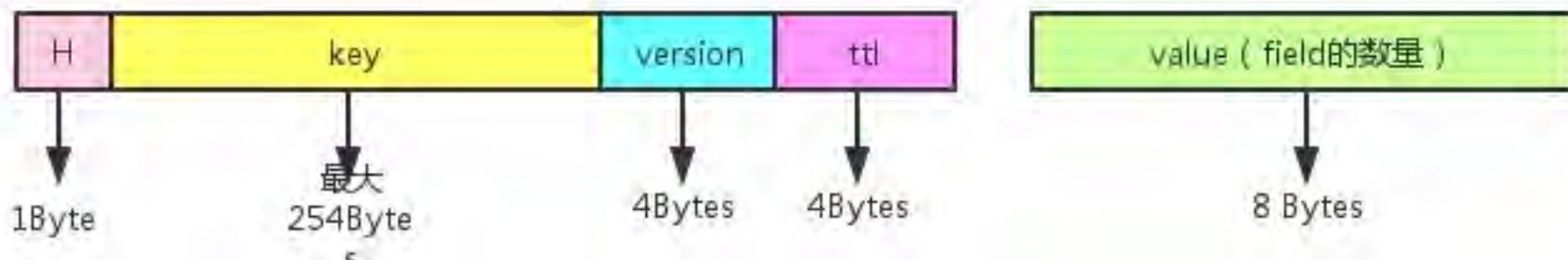
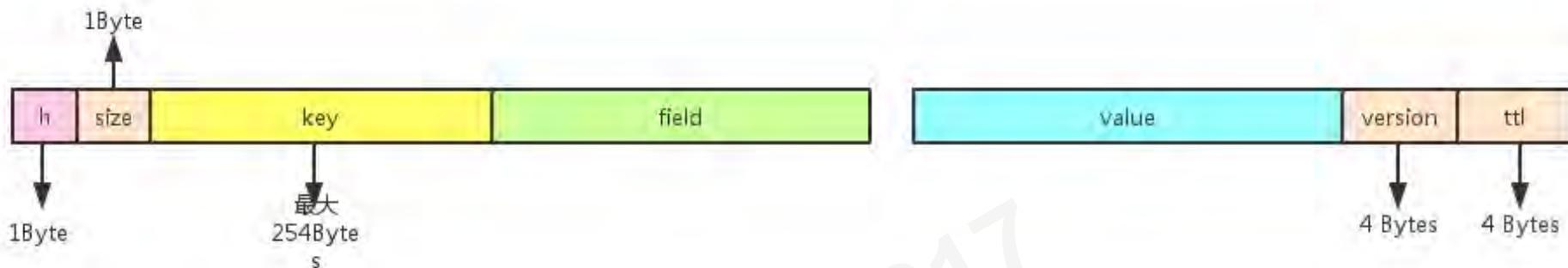
# 网络模块--Pink



# 存储引擎--Nemo

- Nemo
  - Pika 的存储引擎, 基于Rocksdb 实现. 实现了Hash, List, Set, Zset 等数据结构
  - Rocksdb 启动只需要加载log 文件
  - Rocksdb 使用的本地硬盘, 对SSD 盘友好
  - <https://github.com/Qihoo360/nemo>

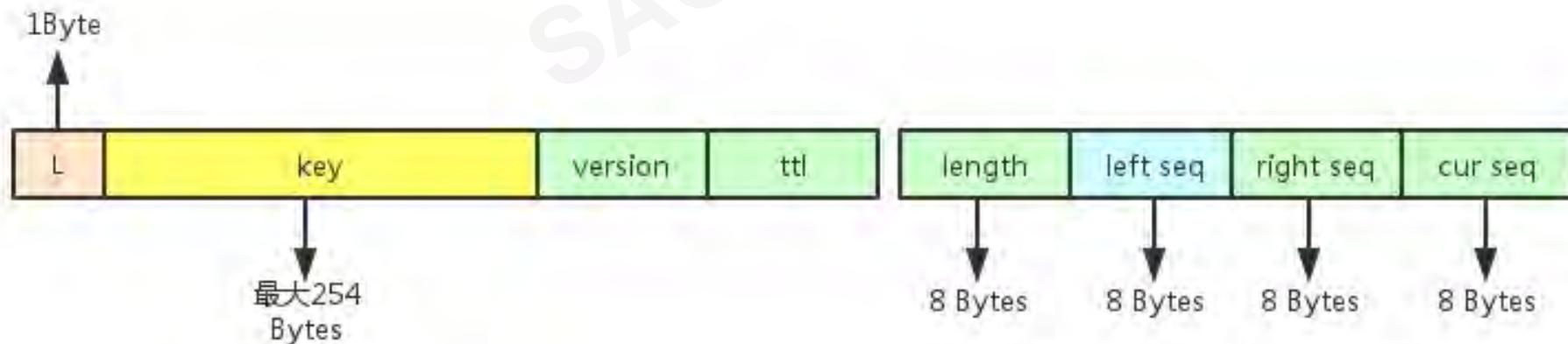
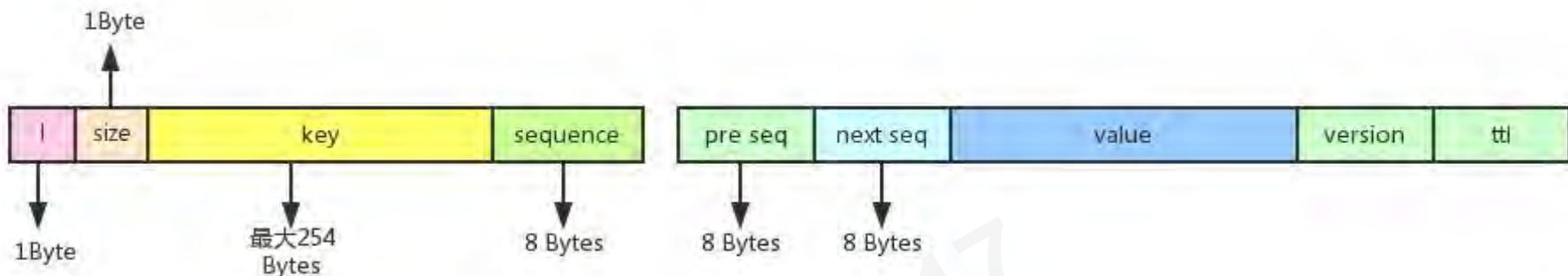
# 存储引擎--Nemo



# 存储引擎--Nemo

- HSET myhash field1 "Hello"
  - DB->Put(wop, h6myhashfield1, Hello01477671118)
  - DB->Put(wop, Hmyhash11477671118, 6)

# 存储引擎--Nemo



# 存储引擎--Nemo

- LPUSH mylist "world"
  - DB->Put(wop, 16mylist6, 57world01477671118)
  - DB->Put(wop, Lmyhash11477671118, 6071)

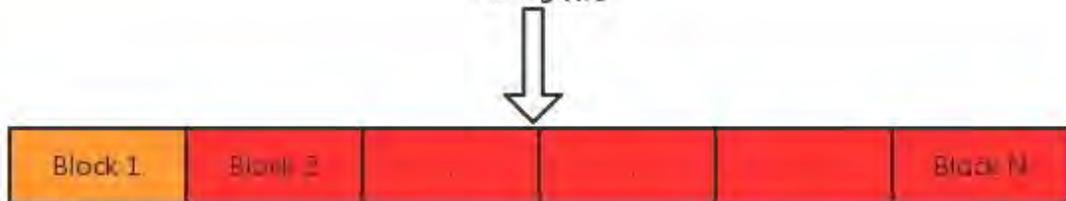
# 日志模块--Binlog

- Binlog

- 顺序写文件, 通过Index + offset 进行同步点检查
- 解决了缓冲区小的问题
- 支持全同步 + 增量同步

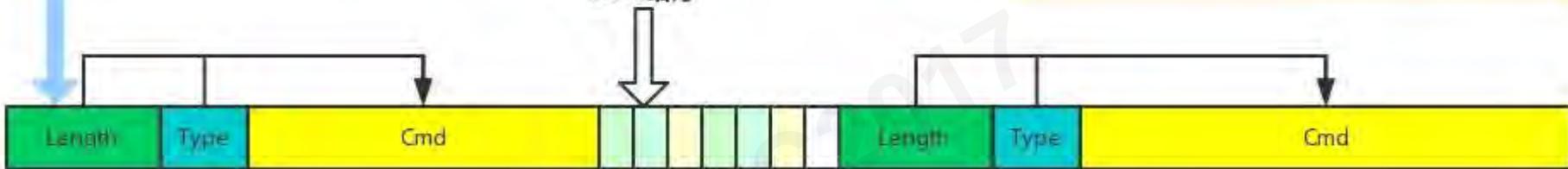
# 日志模块--Binlog

Binlog结构



1. Binlog按大小切分, 每个100M
2. Binlog由若干Block组成

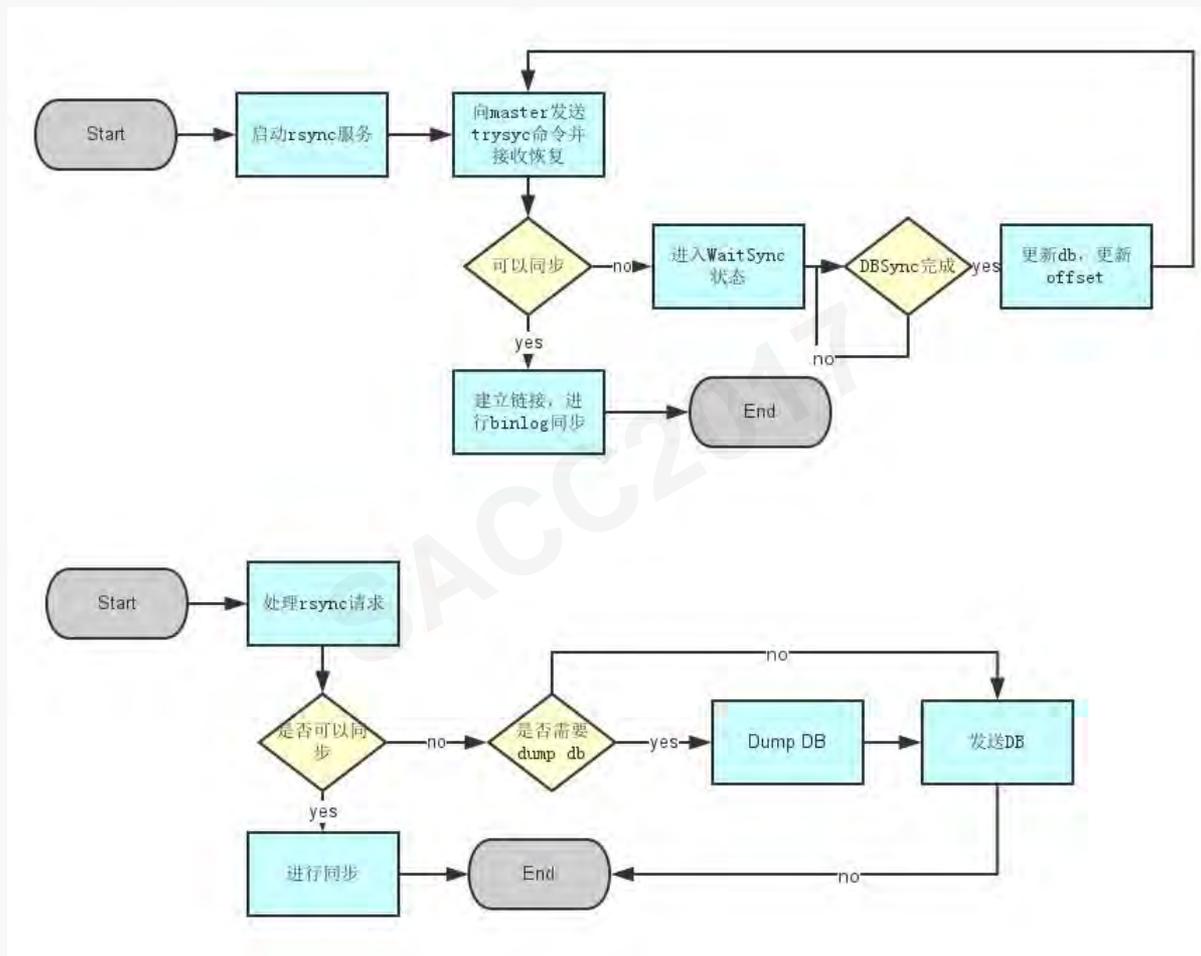
Block结构



1. Block每个定长64K
2. 一条消息包含三个字段: Length(3字节), Type(1字节), Cmd(Length字节)
3. 如果某条消息可以在当前block剩余空间中全部放下, 则Type为kFullType, 否则就要对消息进行拆分, 放到2+个Block当中, Type类型依次为kFirstType, 若干kMiddleType和kLastType

- Type
- kZeroType
  - kFullType
  - kMiddleType
  - kFirstType
  - kLastType
  - kEof
  - kBadType
  - kOldType

# 主从同步-- slaveof



# 主从同步-- slaveof

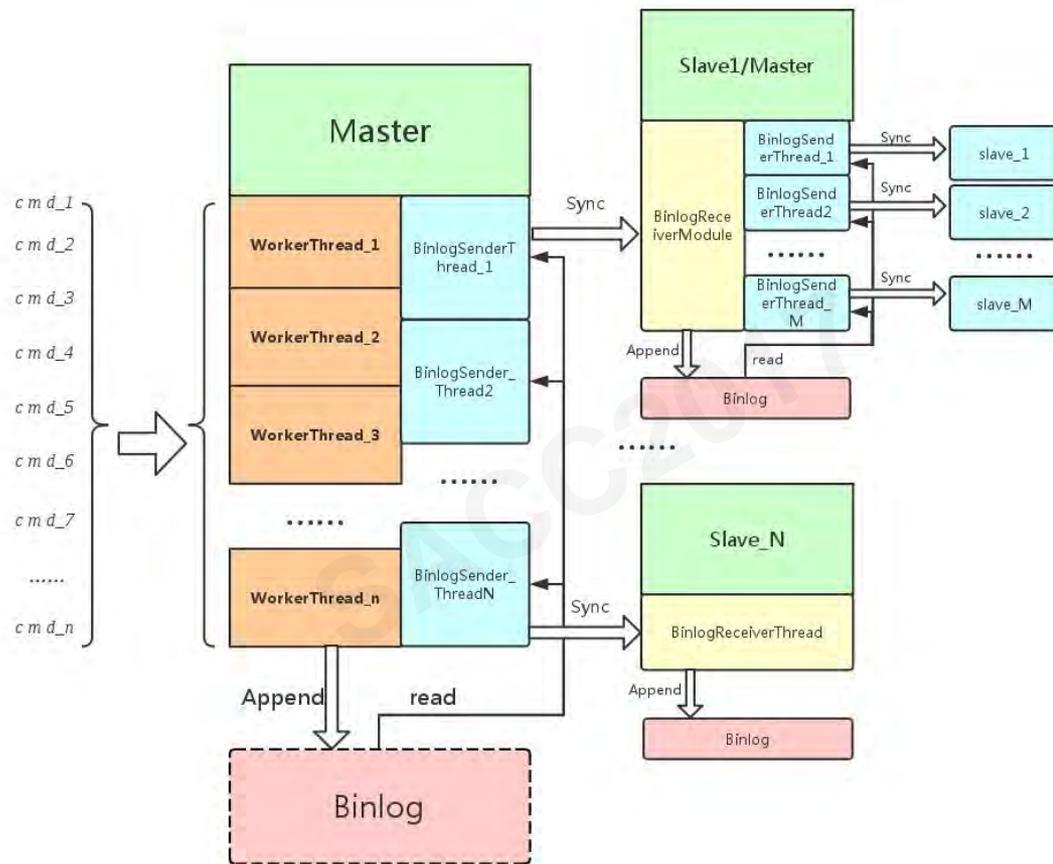


图1 Pika的主从命令同步框架图

# Pika 遇到问题

- 秒删

- 通过修改Rocksdb, 增加 version, timestamp 字段.删除只需要修改metadata
- 支持亿级别数据秒删

## Pika 遇到问题

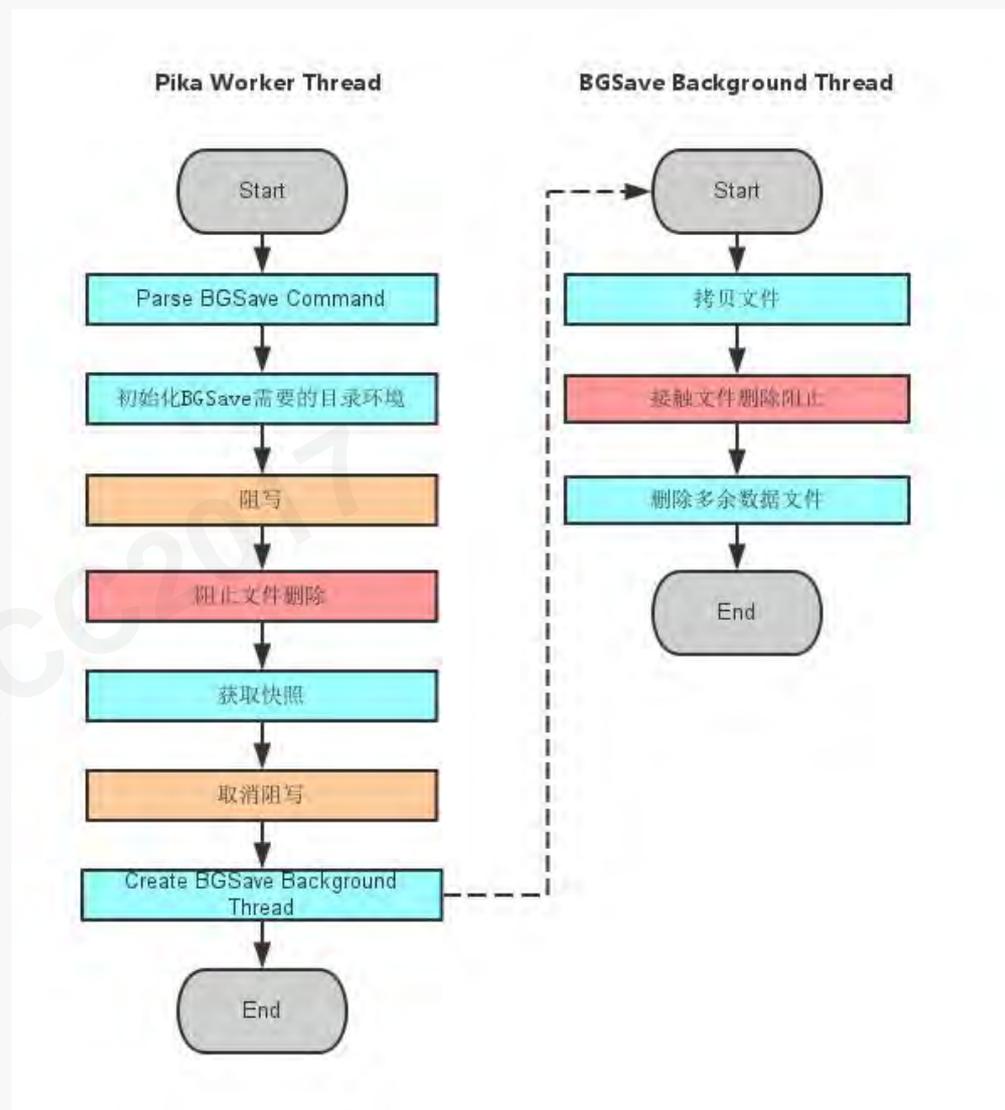
- 数据compact

- 修改Rocksdb manual compact 策略, 支持低优先级的 manual compact
- 根据机型调整rocksdb 配置, compac线程, memtable 个数
- 晚上定期执行

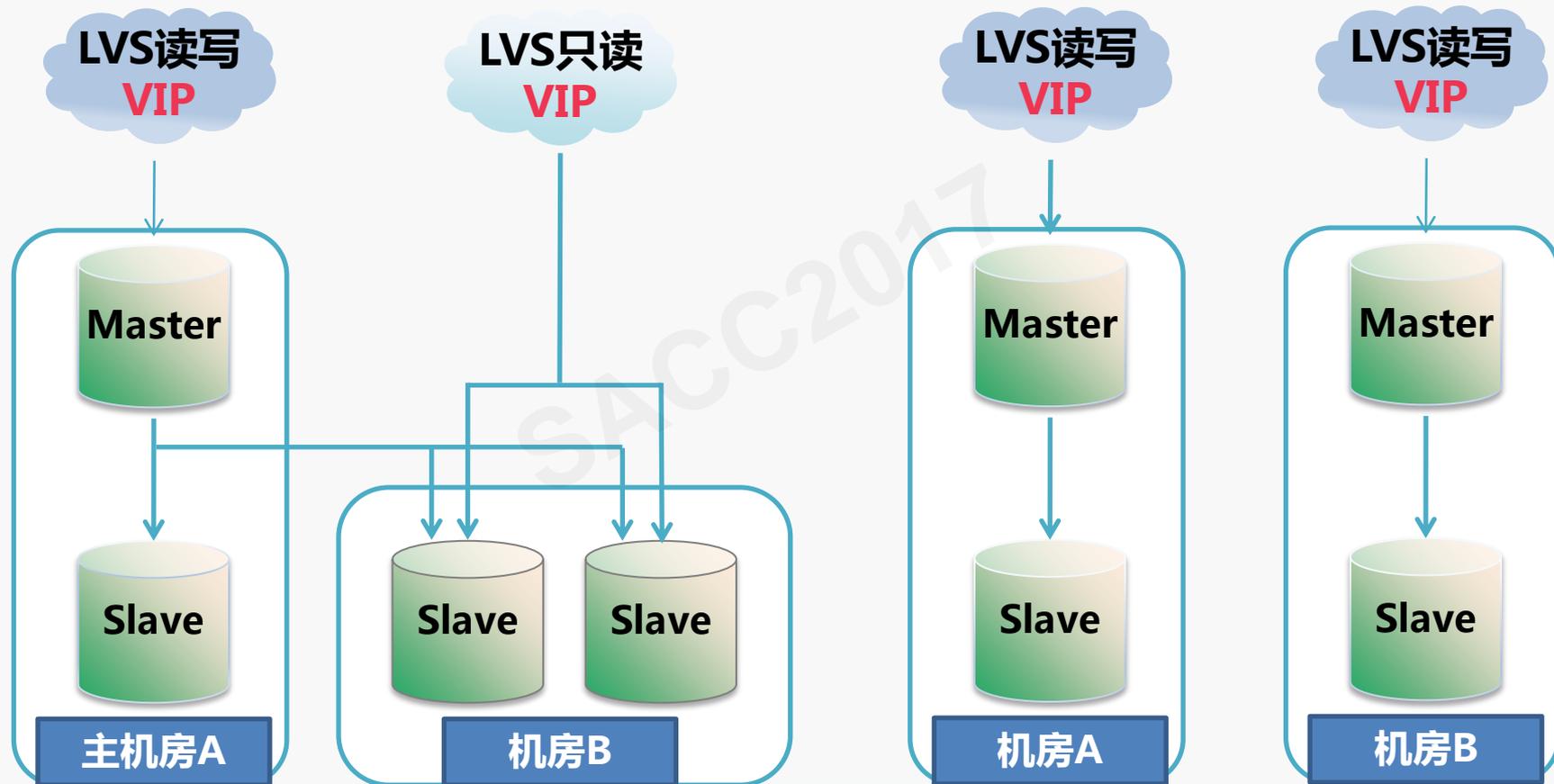
# Pika 遇到问题

- 数据备份

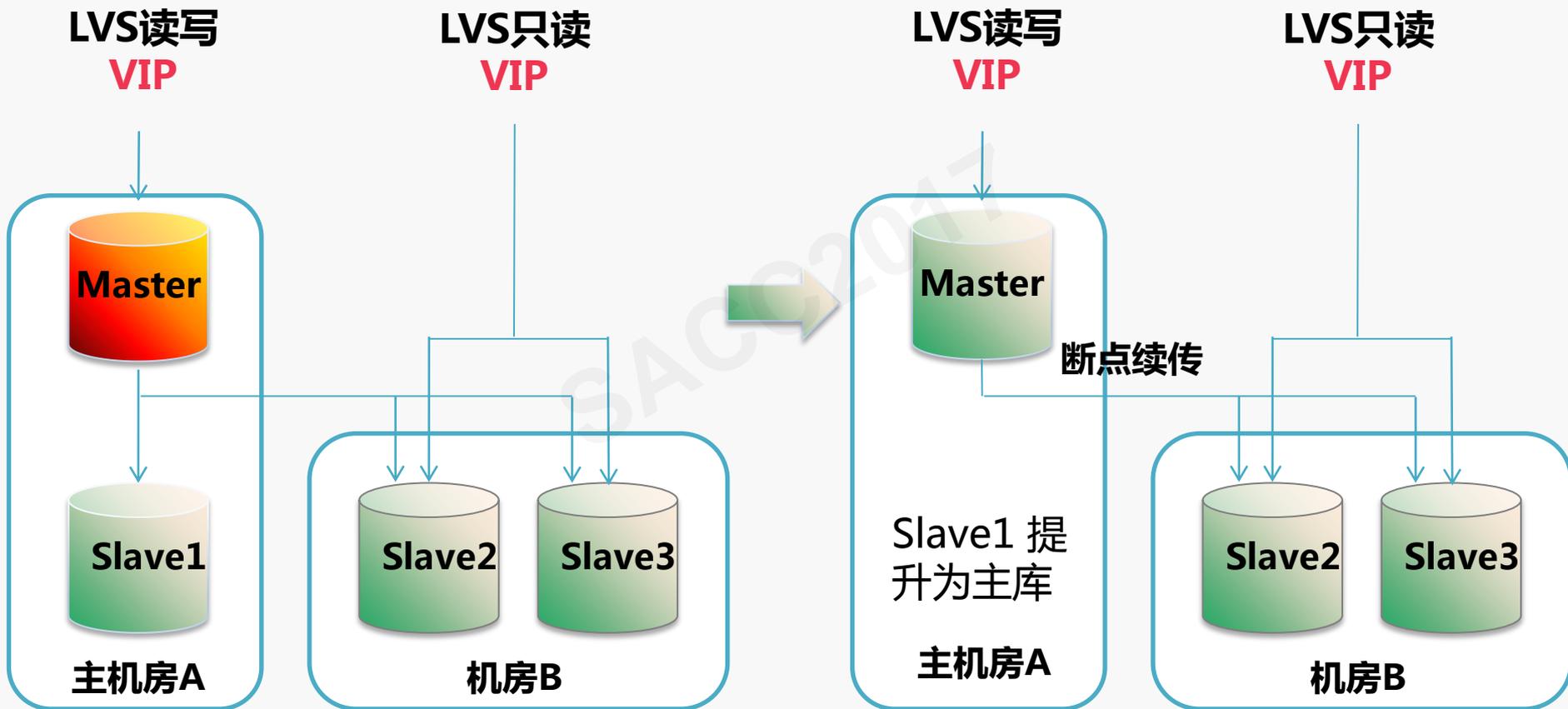
- 需要rocksdb 和 Binlog 配合



# Pika 运维 - 线上架构



# Pika 运维 - 线上架构



# Pika 运维 – 迁移工具

## – Redis\_to\_pika

- 将redis数据迁移到pika，基于aof，能全量+增量方式同步数据（Note关闭aof重写）

## – Pika\_to\_redis

- 业务增长过快，pika逐渐难以支持性能，将pika迁回redis，支持增量数据同步

## – Ssdb\_to\_pika

- 将ssdb数据迁移到pika，目前不支持增量同步

## Pika 运维 – 案例一

### 消息推送服务部分redis迁移到pika

- 迁移前：
  - SET数据结构为主
  - 5套30G左右的redis主从，占用300G内存
- 迁移后：
  - 1套50G左右的pika主从，占用100多G磁盘

## Pika 运维 – 案例二

### 数据分析业务redis迁移到pika

迁移前：

业务数据量增长迅速，上线不到1周数据量增长到40G

迁移后：

1套100G+ Pika主从

## Pika 开发现状

- Pika团队目前有2个主力开发维护，2个DBA做需求分析讨论、性能测试、bug跟踪、回归测试。积累1700+个测试用例
- 产品经理汇总github问题和交流群用户反馈,帮用户问题解决和需求排期开发
- 一月一个小版本, 二月一个大版本

# Pika 开发现状

- 双主支持
- Pika\_hub 提供多机房写入支持
- 支持sentinel
- 支持codis

SACC2017

# Pika 总结

- 恢复时间长
- 一主多从, 主从切换代价大
- 缓冲区写满问题
- 内存昂贵问题

# Pika vs redis

- 劣势

- 由于Pika是基于内存和文件来存放数据, 所以性能肯定比Redis低一些

- 优势

- 容量大
- 加载db速度快
- 备份速度快
- 对网络容忍度高
- 性价比高

SACC2017

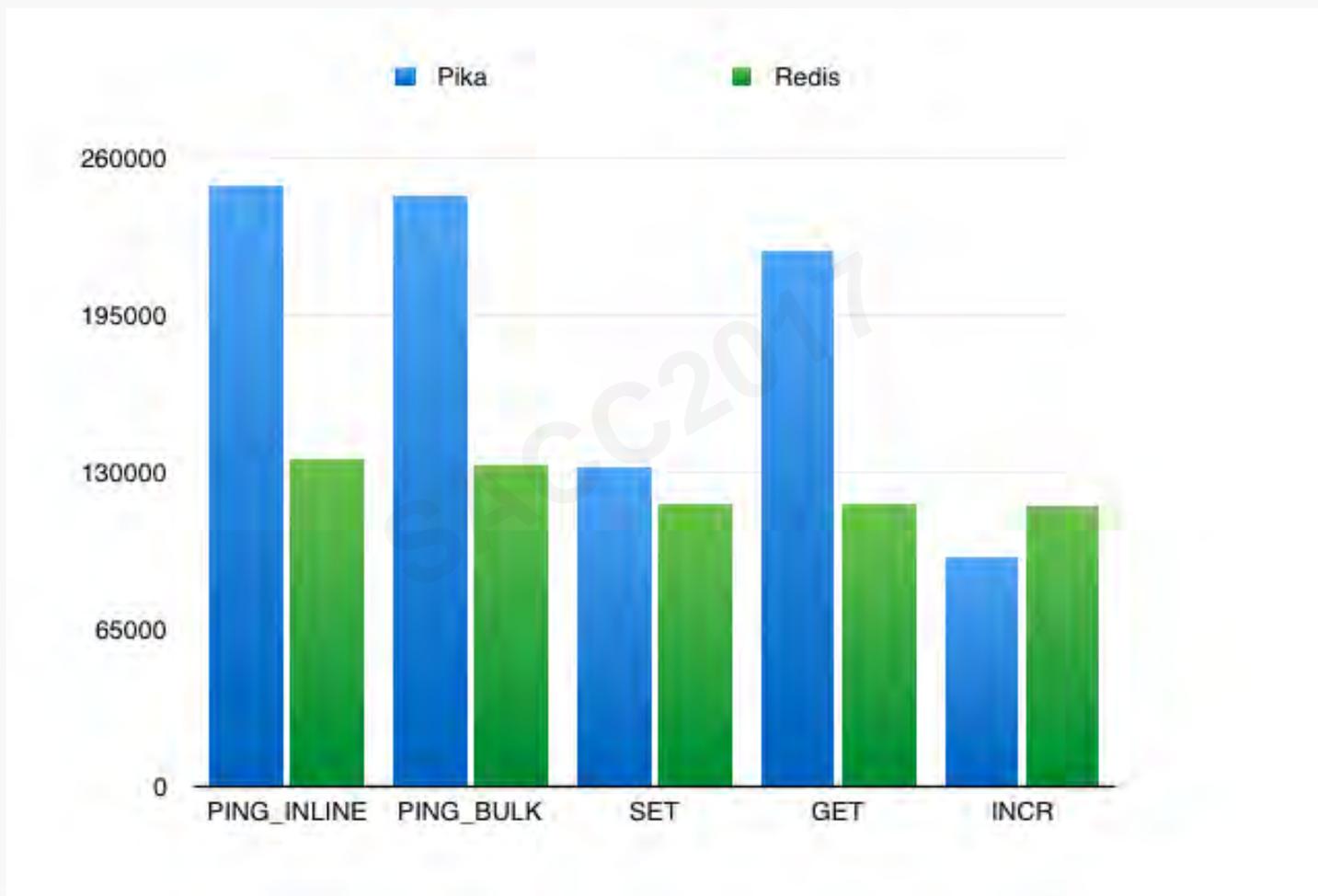
# Pika vs redis

- CPU: 24 Cores, Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz
- MEM: 165157944 kB
- OS: CentOS release 6.2 (Final)
- NETWORK CARD: Intel Corporation I350 Gigabit Network Connection

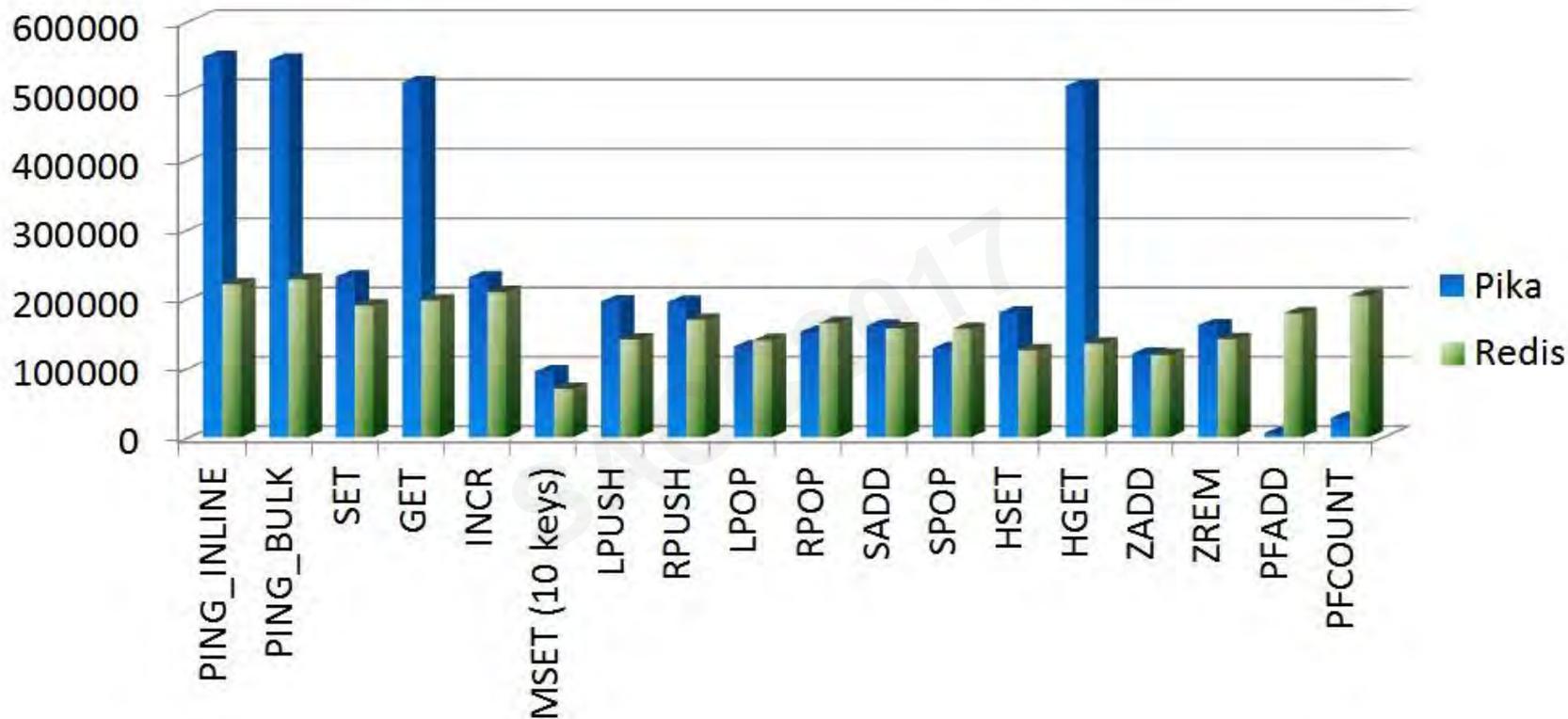
# Pika vs redis



# Pika vs redis



# Pika vs redis 来自vip 的测试



<https://github.com/Qihoo360/pika>



pika 技术交流

扫一扫二维码，加入该群。

THANKS

The background features a dark, almost black space filled with numerous small, bright blue particles. These particles are arranged in several distinct, curved paths that sweep across the frame from the bottom left towards the top right. A bright, white-to-blue gradient light source is positioned behind the word 'THANKS', creating a lens flare effect and illuminating the nearby particles.