搜狐新闻客户端
后端架构演进和PUSH系统

秦启东
qidongqin@sohu-inc.com

搜狐新闻客户端

搜狐新闻 先知道

2013年10月

关于

秦启东

2004年加入搜狐Chinaren Team

2009年负责开发维护搜狐微博

2012年负责搜狐新闻客户端后端架构和基础设施

对分布式存储和高性能系统有浓厚兴趣

1. 后端架构演进

- 1.1 框架选型
- 1.2 缓存优化

2. Push系统实践


上线
性能
优化
容灾
建设

1.1 框架选型

- ✓ 基于netty的http路由层封装 + Ostrich
- ✓ Twitter Finagle
- ✓ Rest.li + Hystrix
- ✓ 其他基于thrift/pb的框架

Metric

二次开发工作:

1. 调整metric信息的输出思路
   原来：每个server提供http admin接口Stream输出
   现在：udp输出到控制中心

2. 开发新的Dashboard

1.1 缓存优化

1. 懒惰式缓存到直写式缓存
2. 细粒度化

   用户对象拆分：{安装信息，设置信息，订阅信息…}

出现的新问题：过多的memcached请求

1.1 缓存优化

过多的memcached请求

方案一：

同一用户对象的缓存到同一个memcached实例，用MultiGet减少请求次数

为spymemcache增加新的connection factory，key前缀相同的存到同一实例，如：123%info,123%install

方案二：本地缓存

解决各个server中本地cache的同步问题

1.1 缓存优化

本地cache的同步：通过消息系统同步数据

- ✓ Kestrel队列
- ✓ RabbitMQ
- ✓ Nsq，高性能、高可靠、无单点故障、发布订阅模式


OpenTSDB
hadoop
Tarantool
Scalding
rest.li
APACHE
HBASE
akka
redis
Kestrel
NSQ
BILLIONS AND BILLIONS
OF MESSAGES SERVED
HIVE
KAFKA
Scala
ZOO

2. Push系统实践

2. Push系统实践

- 定期轮询
  实现简单,非实时,查询时间过短则流量耗费多,耗电量大,后端压力大

- **XMPP**
  XML, ejabberd/openfire

- **MQTT**
  基于代理发布/订阅 模式的消息传输协议，适用于受限环境

2．Push系统实践

第一版

- Netty + MQTT
- 目标是200w
- 结果最大60w长连接
- 继续提高到100w时，FullGC严重

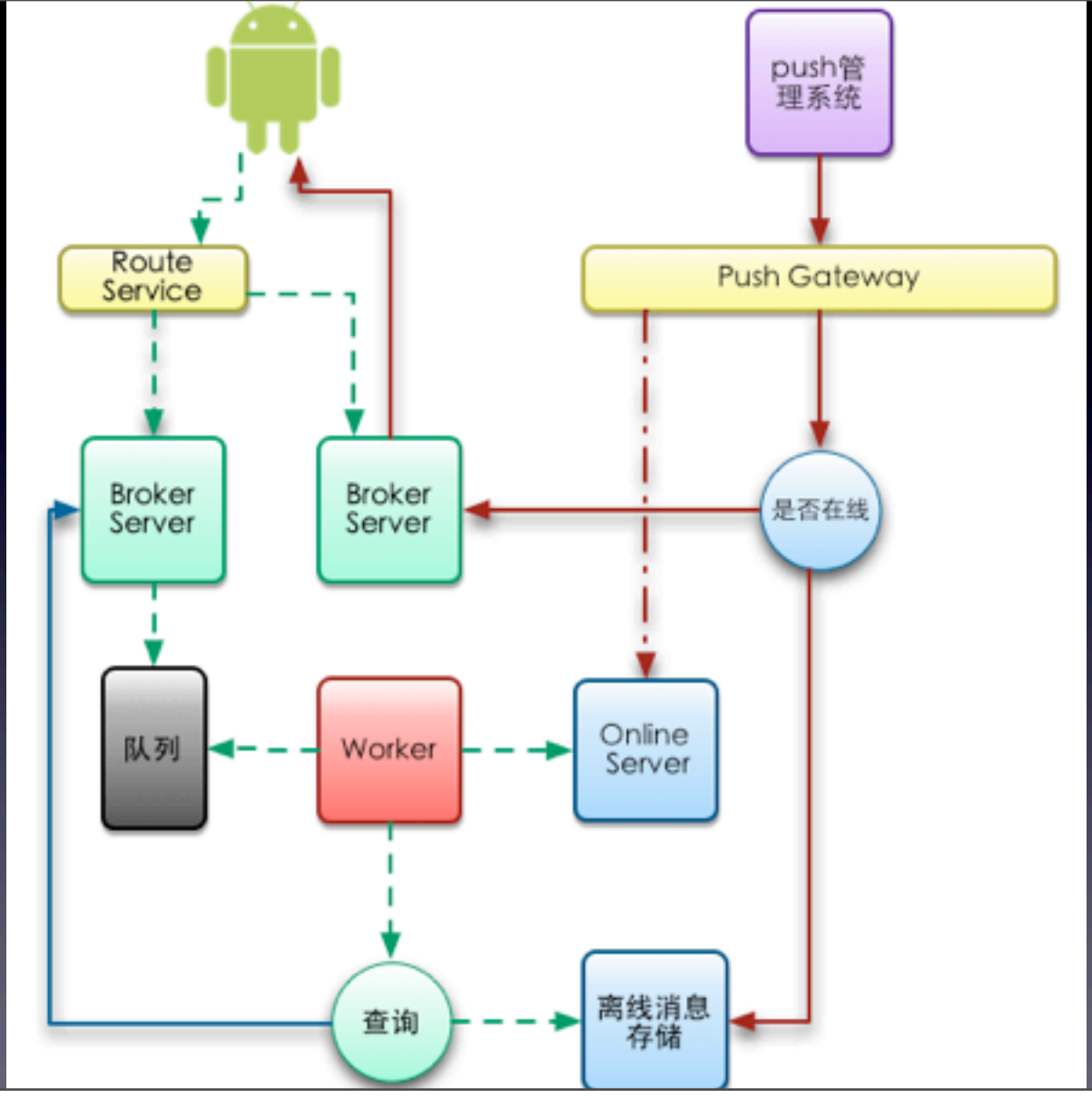
2．Push系统实践

第二版

- 改用golang
- Mqtt协议很复杂， 使用自定义协议

<sub> <topic> <shortId> <longId>\r\n

<set> <clientid:msgid> <_> <_> <bytes>\r\n


push管理系统
Route Service
Push Gateway
Broker Server
Broker Server
是否在线
队列
Worker
Online Server
查询
离线消息存储

2．Push系统实践

运行数据

- 每台100w在线，最大长连接200w
- 单机内存使用12G，12K/长连接
- 消息到达率为93%
- 最大发送时间2分钟

2. Push系统实践

golang实践总结

- 类C的语法学习成本较低
- 基于channel的消息传递机制
- go routine变并发编程为顺序编程
- 快速开发，性能也很高

谢谢大家