基于 Netty 通信的消息推送系统的设计与实现

Design and Implementation of Message Push System Based on Netty Communication

魏井辉 吕 明 (南京理工大学自动化学院,江苏 南京 210094)

摘要:当下公共交通工具是至关重要的社会组成部分,基于它们存在的车载广告成为重要的广告传输媒介。以车载广告项目为背景,主要解决推送系统中大量广告终端连接带来的性能问题,设计出一种可行性的方案,来保证消息推送的低延迟性和推送服务的可扩展性。首先介绍了系统的总体架构的设计,然后分功能模块进行详细阐述,最后介绍了系统的关键技术的实现。

关键词:Netty;负载均衡;路由管理中心;车载广告

Abstract:At present,public transport is an important part of society,and vehicle advertising based on them has become an important advertising transmission medium.Based on the background of vehicle advertising project,this paper mainly solves the performance problems caused by the connection of a large number of advertising terminals in the push system, and designs a feasible scheme to ensure the low delay of message push and the scalability of push service.Firstly,this paper introduces the design of the overall architecture of the system,then elaborates on the functional modules in detail,and finally introduces the implementation of the key technology of the system.

Keywords:Netty,load balancing,routing management center,car advertising

在车载广告推送系统中,项目需求有大量的车载广告终端,云端服务器会根据业务需求定向的推送广告信息,包括广告视频、天气信息、即时文本消息等消息类型,项目实现中会有不同的解决方案。方案一采用客户端定时轮询,定时向服务端发起 http 请求拉取消息;方案二采用 http 协议维持客户端与服务端的长连接,进行长轮询;方案三采用 socket 技术维护长连接,进行消息的推送。以上方案都有利弊,方案一请求中大半是无用的,浪费带宽和服务器资源,后两种方案都存在难于管理和维护长连接的问题。本设计采用路由 Netty 集群式方案,解决消息即时性和长连接集中管理和分派的问题。

1 系统整体结构设计

在广告推送面临终端过多的情况下,单一的服务推送会受到网络带宽和服务器性能的影响,会导致终端接收存在时延或者服务器性能下降等问题[1]。本广告推送系统采用集群的思路,提高系统的可用性,整体架构框图如图 1 所示,广告系统服务层部署多个服务,服务之间可以通过 RPC 相互调用[2],每个服务都可以作为消息的生产者,推送到消息分流层。

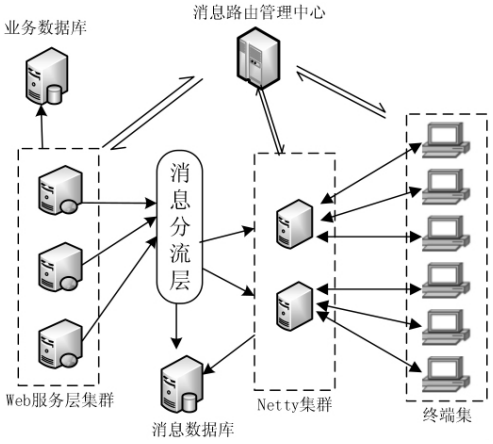

图 1 系统整体架构图

消息分流层访问消息路由管理中心,查找对应的路由信息,将消息定向发送特定的 Netty 服务端,Netty 服务端发送消息给对应的终端。整个项目集群数据库层面分为业务数据库与消息数据库,分库建表,以缓解单一数据库的性能问题。

2 系统模块设计

2.1 消息路由管理中心设计

消息路由管理中心可以作为服务连接的注册中心,作用是保存和定时更新 Netty 服务端与终端的连接映射关系,保存和更新某主题消息下的终端表,为其他服务模块提供路由信息[4]。

连接路由映射设计如图 2,通过 hash 表保存 Netty 服务端与终端的对应关系,终端 ID 作为 hash 的 key 值,可供服务方高效的查询路由信息。

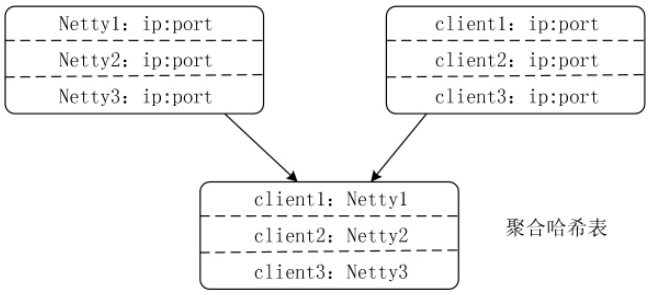

图 2 路由映射示意图

主题消息终端映射保存某主题消息类型下的终端集合,当特定类型的消息产生时,推送到该主题类型下的所有终端设备,映射关系设计如图 3。通过 hash 表保存消息主题与终端的对应关系,topic 作为 hash 的 key 值,终端集合作为 hash 的 value。发

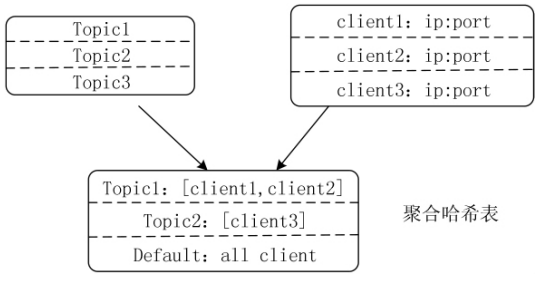

图 3 主题消息与终端映射示意图

送主题消息时,首先会查映射表中 topic 对应的客户端集合,然后再查连接路由映射,找到对应的 Netty 服务方,最后发送消息。

2.2 消息分流层设计方案

Web 服务层集群作为消息的生产者,根据业务需求来生成消息,对于广告系统而言,消息可分为:广告视频、广告图片、天气信息、即时文本消息等其他的消息类型,消息在业务层会确定消息的消费方,可设置固定端口消息和主题消息两种类型。主题消息的推送比较灵活,可以实现区域性推送和服务主体推送等业务需求,根据业务在消费路由管理中心灵活配置。

上述属于消息分流层的查找分派 Netty 服务的功能,消息分流层是消息生产者和 Netty 服务端之间的协调者,另外,消息分流层应具备消息缓存和消息持久化日志的功能,应在此层实现,消息缓存可解决大量不同种类消息同时发送造成的网络拥塞、服务器性能下降的问题,可使用缓存队列来暂存消息,使用线程池并发处理缓存队列中的消息,达到线程复用、节省性能的效果。消息持久化日志可作为一种保障机制,保存到数据库中,事务消息可结合消息日志来实现,日志可实现错误定位和数据分析,对于生产环境,日志持久化是必不可少的。

2.3 终端连接注册流程

终端请求消息路由管理中心,路由中心返回 Netty 服务器的 IP 与 port;

终端向消息路由管理中心发起 http 请求;

服务器获得请求后,通过负载均衡算法分派 Netty 服务器;

更新路由映射表;

服务器返回响应,断开 TCP 连接,关闭资源;

终端获得到 IP 与 port,主动连接 Netty 服务器,建立长连接;

终端发起连接请求;

Netty 服务器监听到连接事件,注册连接通道;

服务器返回连接成功的响应信息;

终端与 Netty 服务端维持长连接;

终端连接失败则重新发起 Netty 连接请求,达到一定次数之后,重新请求消息路由管理中心,重复以上步骤。

终端离线时,先发送请求更新消息路由管理中心的路由表,再关闭 Netty 连接通道。

2.4 Netty 服务端设计流程

消息转发到 Netty 服务端之后,首先进行消息日志的存储,设置为 Netty 服务端已接收状态,然后将消息发送到特定的终端,终端获得消息之后返回消息响应状态到 Netty 服务端,Netty 服务端进行消息的状态更新,设置为已消费状态。

为保障消息的不丢失性,需要设计相应的异常保障机制,Web 服务端定时更新消息状态,如果消息未在定时范围内持久化到数据库,则由服务端重新发送消息,如果终端未在定时范围内返回成功的响应,则由 Netty 服务端重新发送。Netty 服务端需与终端之间维持长连接,服务端定时进行心跳检测,来清除异常的连接或者使用断线重连机制维护连接[3]。

3 关键技术

3.1 一致性 hash 分派算法

消费路由管理中心保存了 Netty 服务端与终端的映射关系,当有新的连接请求时,为了实现负载均衡的能力,尽可能平均的将请求分发到各个 Netty 服务端上,本设计采用一致性 hash 算法来实现。

一致性 hash 算法是集群系统中常用的算法。数据分派算法主要有取模 hash、划段和一致性 hash。取模和划段最大的弊端是节点数固定才能实现均衡的分派,扩展性差,一旦有新的节点加入或者旧的节点宕机,需要所有的数据根据新的规则进行重新分派,而一致性 hash 解决了这个问题,扩展性好,需要迁移的数据较少。

一致性 hash 算法,对节点和数据都要进行 hash 运算[5],如式(1)所示。得到节点和数据的 hash 值之后,将节点和数据映射到 hash 环上,以顺时针的方向计算,将数据分派到离自己最近的节点中,如图 4 所示。

$$hash(ip:port)=id(ip:port)\%(2^{32}-1) \quad (1)$$

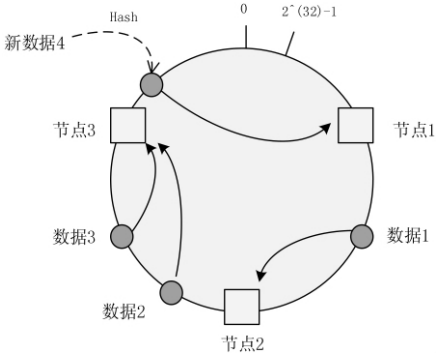

图 4 一致性 hash 示意图

一致性 hash 算法满足了负载均衡的特性以及扩展性,但是存在平衡性的问题,多个数据可能被分派到同一个节点中,一致性 hash 算法并不能保证相对平衡的分派,本设计采用加入虚拟节点的一致性 hash 算法解决这个问题[6],如图 5 所示。

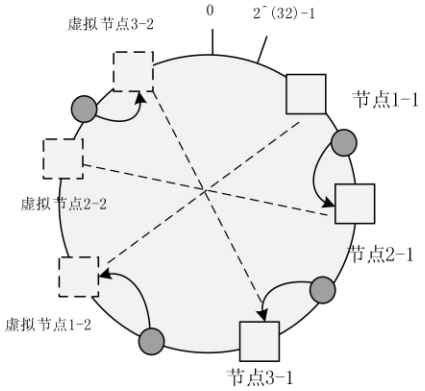

图 5 加入虚拟节点的一致性 hash 示意图

加入虚拟节点的一致性 hash 算法核心代码实现如下:

```
//FNV1_32_HASH 算法计算 HASH
private static int getNodeHash(String str){
    final int node = 16777619;
    int hash = (int)2166136261L;
    for (int i = 0; i < str.length(); i++)
        hash = (hash ^ str.charAt(i)) * node;
    hash += hash << 13;
    hash ^= hash >> 7;
    hash += hash << 3;
    hash ^= hash >> 17;
    hash += hash << 5;
    if (hash < 0)
        hash = Math.abs(hash);
    return hash;
}
```

3.2 Netty 心跳检测与断线重连机制的实现

Netty 服务端需要与终端维持长连接,长连接需要心跳信号来维持,一个连接如果长时间不用,防火墙或者路由器就会断开

该连接。心跳检测机制，即终端与服务端之间定时发送心跳信号，通知 Netty 服务端连接正常，仍处于连接状态，来维持传输层 TCP 长连接的有效性。

终端使用 Netty 提供 IdleStateHandler 机制来实现心跳，终端连接到 Netty Server 端后，会开启一个定时线程，每隔一段时间向服务端发送心跳信号。发送超时失败时，终端则认为 Server 端已主动断开连接。核心代码实现如下：

```
private void ping(Channel channel) {
    int second = Math.max(1, random.nextInt(baseRandom));
     ScheduledFuture < ? > future = channel.eventLoop ().
schedule(new Runnable() {
        @Override
        public void run() {
            if (channel.isActive()) {
                channel.writeAndFlush(ClientIdleStateTrigger.
HEART_BEAT);
            } else {
                System.err.println("The connection had broken.");
                channel.closeFuture();
                throw new RuntimeException();
            }
        }
    }, second, TimeUnit.SECONDS);
```

服务端需实现断连触发器，使用 Netty 的 userEventTriggered 机制来实现。如果在 10s 内没有收到来自客户端的任何数据包(包括但不限于心跳包)，将会主动断开与该客户端的连接。

断线重连机制是指由于网络拥塞或终端进程故障造成服务端和客户端断开连接，待网络恢复之后，客户端重新发起连接，连接后服务器尝试恢复到上次断开时的状态和数据。

实现流程是终端在监测到与 Netty 服务端的连接断开后，使用重连机制进行重连，直到重新建立连接，或者达到重试次数阈值。连接是否断开，可以通过 Netty 提供的 channelInactive 机制来实现，结合 Netty 提供重连策略进行重连。本设计可以使用默认的重连策略 ExponentialBackOffRetry 来实现。

3.3 消息分流层实现

消息分流层设计流程如下：

1)消息分流层收到消息请求之后，加入线程池处理[7]；

2)判断有无可用的线程，若有可用的线程，继续执行，若无，加入缓存队列，等待空闲的线程产生[8]；

3)判断是否为主题消息类型，如果是则通过消息主题向路由中心发请求，如果不是，则通过 clientID 向路由中心发请求；

4)获得 Netty 服务和终端地址后，通知 Netty 服务端发送消息；

5)消息持久化到数据库，设置为新增状态；

6)等待后续消息状态通知。

4 结束语

随着互联网技术的不断发展，物联网平台推送消息的能力要求越来越高。本文以广告推送平台为项目背景，主要设计了一种海量广告消息推送的后台服务系统，以路由管理平台为中心，消息分流层、Netty 服务集群、终端集群协调运作，具有负载均衡、高实时性和可扩展性的特点。目前该架构已经部署在车载广告机项目中，在运营期间达到了预期的效果。

参考文献

[1]胡利军.Web 集群服务器的负载均衡和性能优化[D].北京：北京邮电大学，2010

[2]王伟，余利华.RPCI:面向互联网的 RPC 框架[J].计算机工程与应用，2013，49(21)：106-110

[3]魏莹.基于 Netty 框架的智能终端与服务器通信的研究[D].西安：西安电子科技大学，2017

[4]刘琰翔.面向海量微服务的高可用服务注册中心的研究与实现[D].广州：华南理工大学，2019

[5]王康，李东静，陈海光.分布式存储系统中改进的一致性哈希算法[J].计算机技术与发展，2016，26(7)：24-29

[6]苏跃明，李晨，田丽华.基于分片一致性哈希负载均衡策略与应用[J].计算机技术与发展，2017，27(11)：62-65，70

[7]盛琳阳，盛芳圆.Java 多线程技术在网络通信系统中的应用[J].数字技术与应用，2019，37(8)：108，110

[8]杨开杰，刘秋菊，徐汀荣.线程池的多线程并发控制技术研究[J].计算机应用与软件，2010，27(1)：168-170，179

[收稿日期：2020.9.16]

(上接第 56 页)

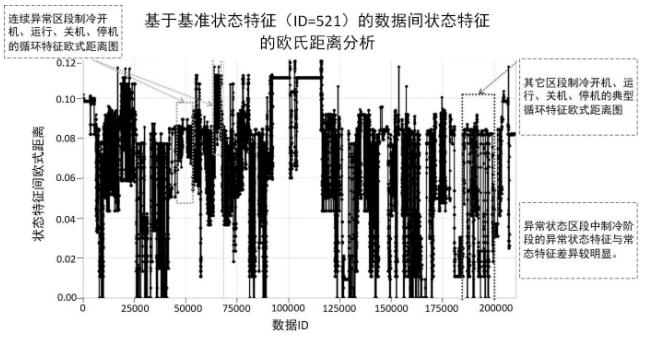

图 6 数据状态特征的欧氏距离

程中，重点、难点体现在中央机组的运行参数多、故障类型多、多种故障可能同时发生、不同故障对数据产生的影响不同等方面；但经过采用边缘层数据的清洗、预处理功能，准确地实现了数据的故障特征提取，顺利地建立了模型，进行了故障的预测。

6 结束语

设备健康预测管理的目标是通过实现备品备件的适时更换，来优化修复性维护与周期性维护之间的平衡，同时结合边缘层的数据采集、数据清理功能来实现数据特征的精准提取。设备健康预测管理最终是要和 ERP、OA、WMS 等系统集成，实现产品的全流程管理，从而为企业降低生产成本。

参考文献

[1]年夫顺.关于故障预测与健康管理技术的几点认识[J].仪器仪表学报，2018，12(23)：244-245

[2]郭阳明，米琪，张双，等.基于故障预测与健康管理的 DIMA 动态重构技术综述[J].计算机测量与控制，2019，14(10)：233-234

[3]张佳乐，赵彦超，陈兵，等.边缘计算数据安全与隐私保护研究综述[J].通信学报，2018，39(3)：1-21

[4]祁兵，夏琰，李彬，等.基于边缘计算的家庭能源管理系统：架构、关键技术及实现方式[J].电力建设，2018，39(3)：33-41

[5]李永浩.探究航空电气设备的故障预测与健康管理[J].冶金与材料，2018

[收稿日期：2020.9.4]