Computer Science and Application 计算机科学与应用, 2019, 9(9), 1689-1695
Published Online September 2019 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2019.99188

Hans汉斯

Research on Implementation of Network State Monitoring Based on Heartbeat Mechanism

—Real-Time Network Monitoring

Mingming Wang

School of Information, Renmin University of China, Beijing
Email: 610212129@qq.com

Received: Aug. 27th, 2019; accepted: Sep. 11th, 2019; published: Sep. 18th, 2019

Abstract

In order to ensure that the applications deployed on the client computer node can communicate with the sever normally, network state monitoring system is designed. In this paper, a scheme is provided to monitor the network state based on the heartbeat packet mechanism of the computer network node. The client node reports the heartbeat data, analyzes the results, quantifies the network state at the level of the excellent and poor, and visually presents the network state in the monitor center. The research content of this paper mainly includes the data communication protocal between the heartbeat packet of the client computer node and the service monitor center. Around the data model of the detection task and the whole task life cycle, the communication model is introduced in detail and the engineering demonstration is carried out. This monitor framework is more flexible. The results show that the model has achieved good results, improved availability and reliability of the entire network infrastructure, and provides a good guarantee for the stability of the business system.

Keywords

Heartbeat, Network Monitoring, Message Queue

基于心跳机制的网络状态监测实现研究

——实时网络监测

王明明

中国人民大学信息学院，北京
Email: 610212129@qq.com

文章引用: 王明明. 基于心跳机制的网络状态监测实现研究[J]. 计算机科学与应用, 2019, 9(9): 1689-1695.
DOI: 10.12677/csa.2019.99188

收稿日期：2019年8月27日；录用日期：2019年9月11日；发布日期：2019年9月18日

摘　要

为了保证部署在客户端计算机上的应用程序能够稳定的与服务器进行数据交互，保证业务正常运行，设计了一套网络状态监控系统。本例中提供一种基于客户端网络节点心跳包机制监测网络状态的方案，客户端节点上报心跳数据，监控中心进行分析，将结果以优、良、差的级别对网络状态进行量化并在监控中心进行可视化呈现。本文研究内容主要包括客户端节点心跳包与服务监控中心的数据通讯机制，围绕检测任务的数据模型及整个任务生命周期进行，详细介绍了通讯模型并进行了工程示范。这种监控框架更加灵活。研究结果表明，该模型达到了较好的效果，提升了整个网络基础设施的可用性和可靠性，为业务系统的稳定性提供了很好的保障。

关键词

心跳，网络监测，消息队列

1. 引言

随着互联网技术的发展，大量公司的业务都在进行数字化转型，网络通信状态作为基础设施，业务闭环的开展必须依赖于稳定的网络通信状态。针对网络状态的监控是信息化服务管理者需要关注的核心问题，同时也是衡量服务稳定性的重要指标[1]。心跳检测是目前已广泛用于网络通信服务、故障检测，连接维持等领域的一项技术[2]。

2. 研究目的

为了保证部署在客户端计算机上的业务程序能够正常的与服务器进行数据交互，保证业务正常运行，设计一套监控系统。该监控系统通过在服务器与客户端之间传递心跳包数据，应用一定的算法判断服务器与客户机之间的连接状况，来监测客户端网络节点的通信状况，网络通信状态作为业务数据交换的基础设施，需要监控系统的稳定性和高可用性来保障整个通信的通畅性，可用性及故障快速感知。

2.1. 业务背景

对于某集团下分布在全国区域连锁商店内的网络状态进行监控，监控中心可以实时了解商店节点的网络状态，当网络出现异常时(如网络抖动造成的用户数据丢失，查询失败[3])触发监控机制，通知相关的工作人员及时处理和修复。对于网络异常，可以尽早发现并修复，预防大面积断网等极端情况的经济损失和其它负面影响，确保业务正常进行。

2.2. 设计目标

1) 通过消息队列组件解耦业务与数据存储，通过 Redis 存储需要处理的业务对象模型，实现数据库与业务数据隔离。

2) 监控中心的分析服务能够稳定执行，支撑万余节点的线上使用场景。

3. 研究方法

监控中心的网络检测依赖于心跳包的传送，心跳包的设计思路来源于分布式系统中网络节点数据同步的实现方式。

心跳一般是指某端(绝大多数情况下是客户端)每隔一定时间向对端发送自定义指令，以判断双方是否存活，因其按照一定间隔发送，类似于心跳，故被称为心跳指令[4]。

Redis Sentinel 是 Redis 高可用的实现方案。Sentinel 称为哨兵，是一个管理多个 Redis 实例的工具。这种模式下，Sentinel 会不断的检查主服务器和从服务器是否正常运行。

默认情况下，每个 Sentinel 节点会以每秒一次的频率对 Redis 节点和其它的 Sentinel 节点发送 PING 命令，并通过节点的回复来判断节点是否在线。

如果在持续的几毫秒之内，Sentinel 没有收到目标节点的有效回复，则会判定该节点为主观下线。

我们把分布在全国各地的连锁商店抽象成网络拓扑中的一个个节点，同时拥有一个集中式的监控中心，监控中心就是中央节点，总体采用星型拓扑结构。

要达到查看各个节点的网络状态，需要各个分节点通过心跳包的方式定时上报各自状态。服务监控中心对于每个节点的心跳包进行单独存储，分析，并汇总结果。

3.1. 系统结构组成

系统主要由客户端节点，云端监控服务中心，监控中心控制台三部分组成，见图 1。主要采用 PHP 语言开发，基于 LAMP 环境进行部署和生产运行。

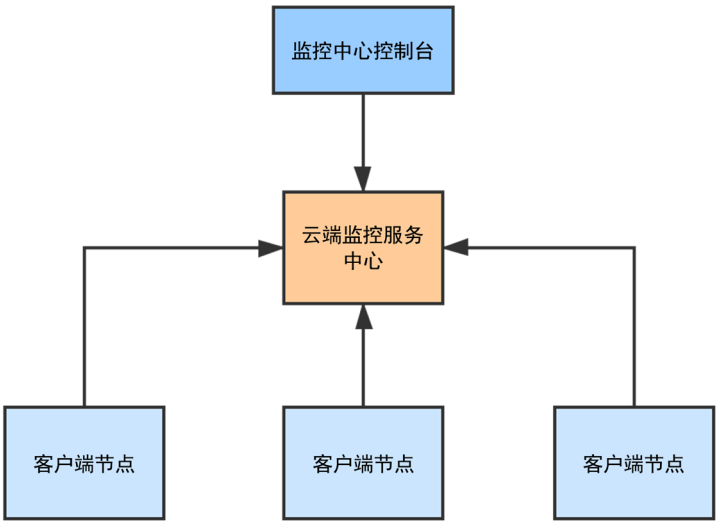

Figure 1. Structure of system
图 1. 系统结构

客户端节点依靠本身应用程序间隔 5 s，上传数据到监控中心的 Redis 服务器。

监控中心服务按照每个节点对数据包进行分组，实时进行分析，如果在持续的两分钟之内，没有接收到节点的心跳包，则认为此节点网络中断。

监控中心控制台是一个 UI 系统，用于业务人员查看节点网络状态。

3.2. 实现原理

以往的心跳检测技术大多是基于 Socket 编程接口实现[2]。

完整的套接字格式{protocol, src_addr, src_port, dest_addr, dest_port}。这常被称为套接字的五元组。其中 protocol 指定了是 TCP 还是 UDP 连接，其余的分别指定了源地址、源端口、目标地址、目标端口。TCP/IP 协议集提供了可供程序员网络开发所用的接口，即 Socket 编程接口。

本中的检测主要依赖 httpapi 接口传递，在数据包采集阶段客户端和服务端不需要保持长连接和强关联关系。

系统将需要处理的业务场景抽象成任务，以商店为维度，为任务编号，通过实时处理任务集合的方式实现商店网络状态监控。采用 NoSql 数据库 Redis 提供的消息模型作为基础，实现消息队列。

一个是使用生产者消费模式模式，另外一个方法就是发布订阅者模式。一个 List 或者 sort 集合就是一个队列，本文中我们采用生产者消费模式实现消息队列。出于避免客户端与服务器端建立长连接需要额外网络消耗的考虑，本文中采用生产消费实现轻量级的消息队列。

消息队列中每一个消息都会被分配唯一标记。唯一标记以业务组件名称加时间戳的生成规则，确保系统唯一。

在客户端节点数据包采集阶段，只需要各个客户端节点安装一个小型常驻代理服务，定时向服务中心发送 http 接口请求，时间间隔为 5s，以 json 为数据格式，节点编号 mall_id 作为 key，值为当前格式化时间，如(2019-08-02 17:44:30)。采集到的数据点在服务监控中心异步进入 Redis 消息队列，等待消费程序按照规则分析，从而得出网络状态。

假设服务端域名为 center.com,http 客户端请求报文如下，见表 1。

Table 1. Package message
表 1. 数据包报文

```
POST    http://center.com/netstate/info HTTP/1.1
Content-Type: application/json
{
        “key”: “mall_101”
        “time_stamp”: 1565938239
}
```

数据点队列按照节点编号做隔离，以节点编号区分不同的队列，为了降低 Redis 队列的存储消耗，每个队列仅保存最近 20 条数据点。

3.3. 数据模型

客户端节点心跳包数据结构，见表 2。

Table 2. Heartbeat package message
表 2. 心跳包数据结构

键	值
mall_id	2019-08-02 17:44:30

客户端节点编号结构，请见图 2。

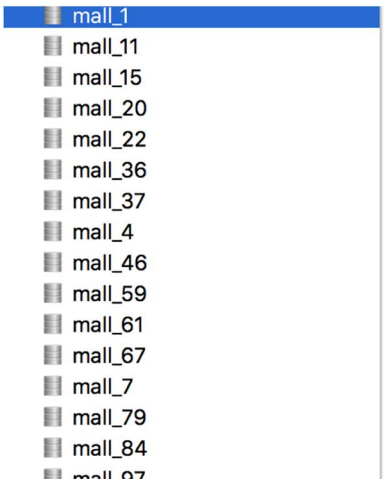

Figure 2. Client node id
图 2. 客户端节点编号

4. 主要研究内容

本文的研究内容主要围绕心跳包数据收集，监控中心数据存储设计，检测任务的数据模型及整个任务生命周期进行。上文提到的数据模型中，客户端节点心跳包在 redis 数据库中以 string 类型存储，检测任务以 sortset 和 string 结合的方式存储。消息的生成与流转涉及到的基本概念和任务生命周期在本节中详述。

4.1. 基本概念

- Job

需要异步处理的任务，每一个网络节点的网络状态分析会构建成一个任务，Job 是延迟队列里的基本单元。与具体的 Topic 关联在一起。

- JobId

任务编号，由业务使用方决定的，一定要保证全局唯一性。唯一标记以业务组件名称加时间戳的生成规则，确保系统唯一，即 realtime_netstate_timstamp。

- Topic

业务主题，一组相同类型 Job 的集合(队列)。供消费者来订阅或者消费。本文中的实时网络监控就是一个 Topic，我们定义为 realtime_netstate。

Topic 具有很强的业务属性，本文中业务涉及连锁零售集团下的所有实体店面的网络状态，网络状态可以认为是一个业务，而分析的领域对象是所有的对象。

- DelayJob

需要延迟的时间。单位：秒。(服务端会将其转换为绝对时间)

- TTR

(time-to-run)，Job 执行超时时间。单位：秒。

- Body

Job 的内容，供消费者做具体的业务处理，以 json 格式存储，可以根据不同的情况进行调整和扩充，不同的 topic 的 body 是不一致的。

- JobPool

一组需要处理的任务集合统称为任务池，见图 3。

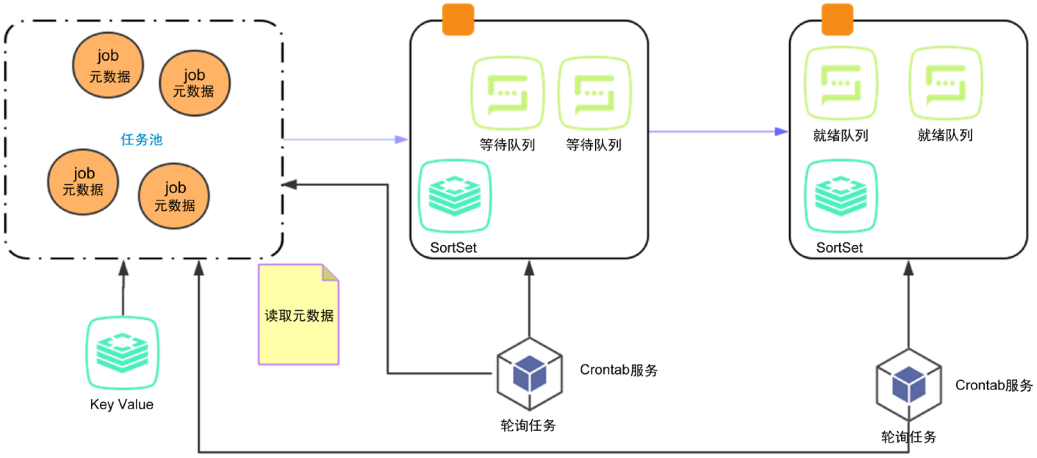

Figure 3. Message queue data structure
图 3. 消息队列数据结构

4.2. 任务存储结构

本系统中的云端监控服务中心，可以认为是一个小型的消息中间件，消息中间件对消息队列进行相应的封装操作，为用户提供了消息发送与消费的接口[5]。客户端节点通过消息发送接口上报数据，监控服务中心通过消费接口分析数据，以辅助快速定位问题，实现灵活可扩展的配置告警[6]。系统中涉及到的任务存储结构见表 3。

Table 3. Redis storage structure list
表 3. Redis 存储结构清单

TopicBucket	描述	数据结构
netstate_notice	网络监控最近异常任务等待队列	sortset
netstate_ready	网络监控最近预警任务就绪队列	sortset
realtime_netstate_notice	实时网络监测任务池主题	key/value
stop_strategy	实时网络监测停止策略就绪队列	sortset

4.3. 任务生命周期

任务的生命周期是指从任务到创建(init)，等待执行(waiting)，就绪(reday)，完成销毁(finished，deleted)的过程，服务负责跟踪，维护和执行。

1) 创建任务元素进入等待队列，存储任务节点

云端监控服务中心按照网络节点编号对客户端上报的原始数据点进行分组和排序，网络节点进行单独分析，任务的最小单位以网络节点纬度进行。也就是说在监控服务中心进行一次分析时，假设有 5000 个网络节点，那么就会分配 5000 个任务，每个任务包含任务编号和任务实体数据，一并存入元数据中，进入等待队列。

为保证实时性要求，包的大小应尽可能小，以减少带宽的占用[7]，同时也减少了 redis 存储数据文件的大小。

2) 消费者执行消费由等待队列状态转移到就绪队列，等待消费

服务端会定时统一执行消费程序，时间间隔是 5 s，对等待队列和就绪队列中的任务进行消费。根据

每个网络节点的原始数据上报数据的最后时间与消费时间做时间差，在持续的 120 s 之内，稳定的接收到数据上报，认为网络状态优，大于 120 s 未接收到数据，认为网络状态差，否则是良。

3) 销毁分发发送邮件任务，删除任务节点，当次任务结束

对上一步网络监测结果良和差的网络节点，将联系人信息写入发送邮件队列，进行发送邮件通知。

4.4. 监控服务部署

本文中涉及到的工程实践包括客户端节点，云端监控服务中心，监控中心控制台三部分组件程序，以 PHP 语言开发，基于 LNMP 环境部署和生产实践，同时定时服务借助 CentOs 系统的 Crontab 服务。

当服务器轮询程序对任务队列进行消费时，会遇到同一个任务被多次执行消费的情况，为了解决这一问题，本例中在任务消费过程中使用了 Redis 分布式锁，确保同一个任务只能被执行一次，实现了消息服务消费的幂等性。

4.5. 结论

本文通过对待监控的销售商店的程序抽象，提出一种基于 API 接口上报和 Rediis 数据库结合的网络监测模型，借助消息队列服务组件对业务数据处理和元数据存储进行解耦，深入地使用了 Redis 中 Sortset，List，string 等数据结构，很好地完成了网络状态的实时监测分析。

系统经过半年的线上运行，结果记录如下：

1) 系统应用 Redis 存储需要处理的业务对象模型，通过消息队列组件解耦业务与数据存储，很好地实现了原始数据与业务分析的程序分层隔离。

2) 监控中心的分析服务能够稳定执行，完成一次全量网络节点，数据量 7000 左右的网络状态分析，结果控制在 1 分半内，能够满足业务方的业务。

5. 后续优化建议

5.1. 队列组件

本例中采用的是 Redis 作为消息队列中间件，优势是基于 Redis 的消息队列模型实现比较轻量级，不足是队列中的元素出队列即为消费完成，Redis 本身不提供元素的消费确认机制，需要程序自身完成，增加了程序复杂性。消息队列中间件，后续可以由 RabitMQ 来代替 Redis，消费的确认交给 RabitMQ 完成。

5.2. 消费模型升级

消息的处理模式由消息队列模型转为发布订阅模式，替换监控服务中心的定时轮询机制，提高消息的处理时效，降低轮询对服务器资源的无谓消耗。

参考文献

[1] 彭天舒. 微服务架构下基于特征规则的异常检测研究[D]: [硕士学位论文]. 武汉: 华中科技大学, 2017.

[2] 朱建淼, 贾波, 华金晶, 汪圣华. 基于心跳包的 Socket 通信在危险源无线监测中的应用研究[J]. 安全, 2018(4): 12-14.

[3] 谭营军, 董俊磊, 李翠霞. 云计算环境下网络抖动状态的检测方法研究[J]. 计算机仿真, 2016, 33(1): 438-441.

[4] 汪琳霞, 鄢锋. 心跳包技术应用于高可靠工业网络中的研究[J]. 控制工程, 2015(5): 1048-1052.

[5] 刘子晨. 基于 Tair/Redis 的延迟队列系统的设计与实现[D]: [硕士学位论文]. 大连: 大连理工大学, 2017.

[6] 刘一田, 刘士进, 郭伟, 何翔. 柔性微服务监控框架[J]. 计算机系统应用, 2017(10):141-145.

[7] 马继伟, 何佳洲, 丁春山. 一种基于心跳检测的网络时间同步方法[J]. 指挥控制与仿真, 2017(1): 116-121.