# 基于微服务架构的智慧灯杆系统研究与应用

寇柏源，王少林

（山东建筑大学 信息与电气工程学院，山东 济南 250100）

**摘　要：**传统智慧灯杆系统实时数据的推送和获取需通过不断轮询，占用系统开销，网络带宽压力大，且高流量请求可能造成系统崩溃等问题。为解决上述问题，设计和实现了一种基于微服务架构的智慧灯杆数据实时推送系统。从业务实际需求出发，对系统技术架构与功能架构进行设计，明确4种技术模块与功能模块，并通过提取梳理模块、功能、实体以及资源结点的关系与属性，明确不同结点之间的权重，搭建图网络，采用图聚类算法将系统拆分为以实时数据推送服务为主的九大微服务。系统针对实时数据推送需求，采用双Redis数据库设计，分离实时数据与缓存数据，明确了3种实时数据存储结构以及数据点位表、转存配置表、系统三者之间的关系，启用Redis实时数据库键空间通知功能，监听特定键值变化，并通过Websocket实时推送到客户端。系统测试表明，该系统可以高效、准确地实现智慧灯杆实时数据推送以及历史数据的展示与存储，满足实时性与并发需求，其为智慧城市中的智慧灯杆数据可视化场景建设提供了完善的解决方案。

**关键词：**智慧灯杆；Redis；图聚类；实时推送；Websocket

**DOI：**10. 11907/rjdk. 231089　　开放科学（资源服务）标识码（OSID）：

**中图分类号：**TP399　　**文献标识码：**A　　**文章编号：**1672-7800（2024）004-0100-07

# Research and Application of Smart Lamppost System Based on Microservice Architecture

KOU Baiyuan，WANG Shaolin

（*School of Information and Electrical Engineering*，*Shandong Jianzhu University*，*Ji'nan* 250100，*China*）

**Abstract：**Due to the constant polling，the push and acquisition of real-time data in the traditional smart lamppost system takes up system overhead，which causes the large network bandwidth pressure，and high traffic requests may cause system collapse and other problems. To solve the above problems，a smart lamppost data real-time push system based on microservice architecture is designed and implemented. Based on the actual business needs，the system technical architecture and functional architecture are designed，and four technical modules and functional modules are clarified. By extracting and sorting out the relationship and attributes of modules，functions，entities and resource nodes，the weights between different nodes are clarified，and the graph network is built. The graph clustering algorithm is used to split the system into nine micro services based on real-time data push services. The system uses the dual-Redis database design to separate the real-time data and cache data according to the real-time data push demand，clarifies the three real-time data storage structures and the relationship between the data point table，the transfer configuration table，and the system，which enables the Redis real-time database key space notification function，monitors the specific key value changes，and pushes them to the client in real time through Websocket. The system test shows that the system can efficiently and accurately push the real-time data of the smart lamppost and display and store of the historical data to meet the real-time and concurrent requirements，which provides a perfect solution for the construction of the smart lamppost data visualization scenarios in the smart city.

**Key Words：**smart lamppost；Redis；graph clustering；real-time push；Websocket

**收稿日期：**2023-02-09

**基金项目：**山东省智慧住区项目（鲁财建指[2019]31号）

**作者简介：**寇柏源（1997-），男，山东建筑大学信息与电气工程学院硕士研究生，研究方向为建筑智能化与能效管理；王少林（1963-），男，博士，山东建筑大学信息与电气工程学院教授、硕士生导师，研究方向为建筑智能化与能效管理、智慧城市、智慧住区。

## 0　引言

随着智慧城市朝着更加深层次信息化发展，城市逐渐实现全面感知、资源互联共享、智能协同，最终具备智慧城市的服务与管理功能[1]。灯杆乘着5G与新基建的热潮，形成了“有网、有电、有杆”三位一体特点的智慧灯杆，成为智慧城市建设中必不可少的关键终端[2]。2021年4月6日，多部门联合发布《关于加快发展数字家庭 提高居民品质的指导意见》，提出鼓励建设智能视频监控、智能充电桩、智能灯杆、智能井盖等公共配套设施[3]。

智慧灯杆相较于传统灯杆已经有较大差别，除基础的照明功能外，智慧灯杆还包含智慧安防、信息屏幕发布、公共广播、环境监测、智能充电桩等服务功能。此外，在管理功能方面，智慧灯杆还需具备告警管理、运维管理、运管管理、能耗管理等功能。越来越多的功能叠加使得智慧灯杆系统对数据的分析处理及其实时性要求越来越高，传统的Restful Api接口已经无法满足数据实时性需求。各行业对实时数据都有推送需求，文献[4-7]所提出的系统都是借助Redis的订阅发布功能实现数据实时推送与控制的下发，但存在由于数据本身并没有存入到Redis中，若发布者未及时发布数据，会导致Web页面打开时接收不到实时数据，造成页面数据展示缺失，影响页面美观等问题。并且，随着智慧灯杆系统复杂性的增加，越来越需要对系统进行微服务拆分，微服务架构强调要尽可能地将系统划分为小型轻量级的服务[8-10]。目前，常用的微服务拆分方式主要有基于代码规模[11]、基于领域驱动设计[12-13]、基于相似度算法[14-15]以及基于图聚类算法[16-18]，从系统降耦合度、增内聚度考虑，图聚类算法具有明显优势。

本文针对智慧灯杆运维特点，并在已有研究基础上进行技术改进，设计了一款面向智慧灯杆的实时、美观、高效的系统。

## 1　智慧灯杆系统总体框架

### 1.1　设计目标

该系统设计的主要目标是将高频变化的智慧灯杆实时数据，高效便捷地推送给Web终端用户并进行服务拆分，因此对整体系统性能要求较高。实时数据不能通过轮询等高延迟传输方式，且为了系统界面美观并满足实时性，界面加载完成后必须立即有实时数据展示。同时，为了提高系统可移植性，方便对接不同城市的智慧灯杆系统，系统需具有集成标准开放协议的接入能力。由于智慧灯杆系统面向城市运维人员，在用灯高峰期需解决系统高并发的可能性。最后，由于智慧灯杆系统包含多个子系统，但是后期井盖、垃圾桶等子系统都需要整合到该系统，因此子系统解耦，各系统独立开发同样重要。

基于以上设计目标，该系统在实时性方面依靠监听Redis实时数据库键值对变化配合Websocket长链接的方式实现实时数据推送。在可移植性方面，该系统整合标准Restful Api接口，可随时与第三方接口对接。在满足系统解耦上，该系统通过配置微服务架构服务器，综合考虑结点关系，搭建结点关系图，采用图聚类算法，合理拆分系统服务。最终，基于Vue设计了满足智慧灯杆运维、监测、控制的可视化系统页面，满足智慧城市发展需求。

### 1.2　系统总体技术架构设计

本文设计的智慧灯杆系统技术架构如图1所示，整个系统主要分为4个部分：数据采集模块、数据存储与Web服务模块、用户请求转发模块以及服务与配置模块。数据采集模块智能网关实时接收各子系统的数据并完成协议转换，统一通过TCP/IP的方式维持与数据采集服务器长链接。数据存储与Web服务模块由Quartz作业调度分类获取实时数据与历史数据，并根据实际需求修改获取频率，实时数据放入Redis主(master)实时数据库，由主数据库完成数据复制，实现Redis高可用，将历史数据存入Mysql数据库，采用旁路缓存(Cache Aside)模式更新缓存，同时配置监听放置时序数据的Redis从(slave)数据库，并通过Websocket将变化的实时数据推送到客户端，借助Tomcat发布HTTP Web服务，Netty作为WebSocket服务，所有微服务接口统一通过微服务网关(Spring Cloud Getway)进行暴露，由微服务网关完成路由转发、限流熔断等措施，所有服务器日志统一交由日志服务器(ELK)进行存储方便系统维护与管理。用户请求转发模块主要使用Nginx作为反向代理并实现IP hash算法与Weight权重以实现系统负载均衡功能，同时解决Session不共享问题。服务与配置模块依靠Nacos搭建服务注册发现中心与配置中心，分别用于记录微服务地址以方便微服务网关查找，以及简化系统配置文件管理。

### 1.3　系统总体功能架构设计

智慧灯杆系统功能模块拆分为4个部分，如图2所示。

(1)用户管理。由于系统直接监控城市智慧灯杆，因此面向专业运维人员使用，所有登录用户注册完成后，统一由高级管理员分配权限，运维人员根据分配的权限监控相应的页面，实现权责分明。

(2)实时监控。提供智慧灯杆各子系统的实时监测数据，以智慧照明子系统为例，该系统可实现远程监测灯杆不同回路的电压、电流、功率因数等实时数据，并可根据实时环境信息智能分析，实现智慧灯杆的假日模式、智能模式、节能模式、普通模式的不同模式切换，达到智能调光目的。同时，用户亦可通过系统实现监控云平台角度调整、信息发布屏幕节目切换、公共广播播放与暂停等实时控制功能。

(3)设备管理。提供该监控单元的完整设备信息，包含设备地理位置、设备分组编码信息、设备所属灯杆、设备

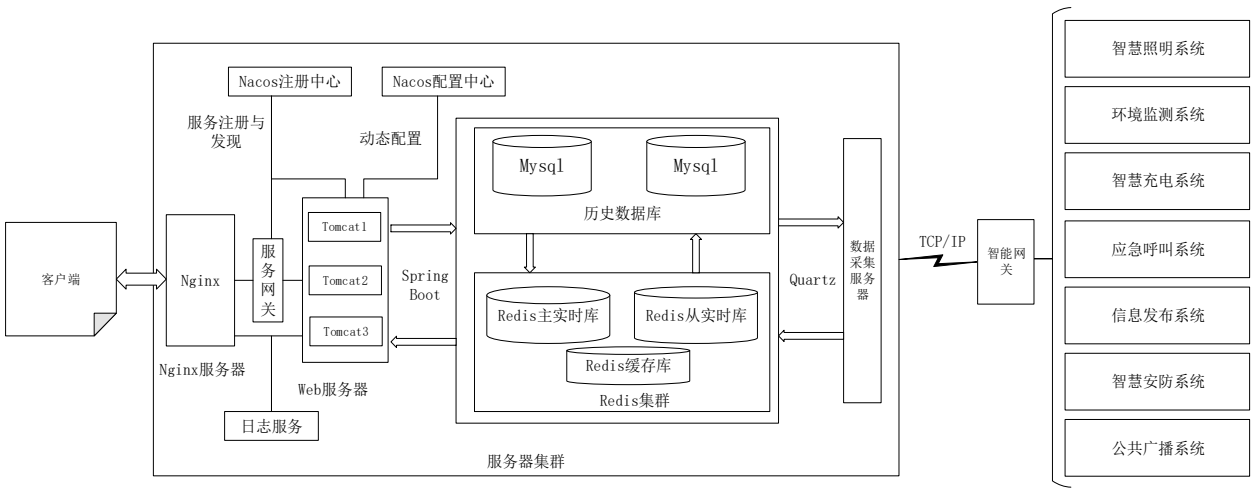


**Fig. 1 Technical architecture of intelligent lighting pole system**

**图1 智慧灯杆系统技术架构**

生产信息和设备安装时间等一系列设备资产信息管理,并提供基础的CRUD操作。

(4)运维管理。运维管理包含能耗管理、故障管理和运维工单管理3个子系统,其中能耗管理提供按小时、按天、按月、按年能耗数据查询并制作报表,同时根据近期能耗数据完成能耗数据预测;故障管理可将系统产生的实时故障与报警信息通过提醒的方式告知运维管理人员,并自动提交工单;运维管理系统可实时监控工单进度和巡检任务下发。

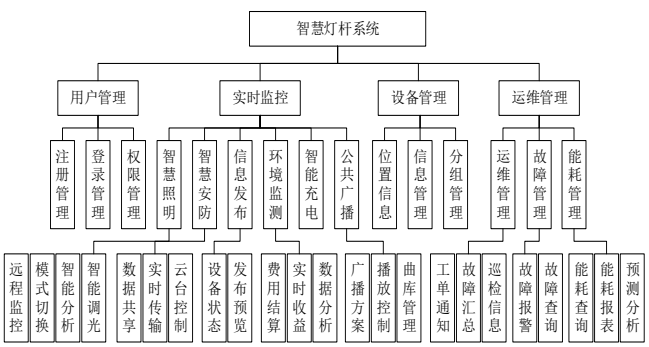


**Fig. 2 Function module of intelligent lighting pole system**

**图2 智慧灯杆系统功能模块**

### 1.4 系统实时数据架构设计

智慧灯杆系统实时数据种类与结构多样,如灯杆实时工作电压、电流、功率数据,以及温湿度噪声等环境数据、广播与信息屏实时节目数据、充电桩实时充电量数据、应急呼叫数据、故障报警数据等。基于智慧灯杆系统时序数据存储需求,在Redis数据库中分别采用String与Hash组合的方式存储基础实时数据并配合Zset[19]存储部分排名实时数据。智慧灯杆信息存储结构如图3所示,Redis键采用“:”分割不同的层次(命名空间),第一层(Location)表示设备所处区域,第二层(equipmentType)表示设备类型,第三层根据具体待定,键Location:equipmentType:count采用String结构记录该区域智慧灯杆个数;键Location:equipmentType.to.id采用Hash结构用于记录设备和id之间的映射关系;键Location:equipmentType:id同样采用Hash结构保存智慧灯杆设备的设备名、设备分组、设备所属灯杆、设备工作区域等。

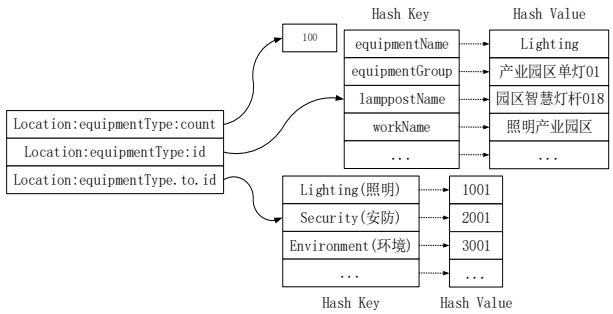


**Fig. 3 Intelligent light pole information data storage structure**

**图3 智慧灯杆信息数据存储结构**

环境监测噪声数据存储结构模型如图4所示,键Location:Environment:connt采用String结构记录该区域环境监测设备个数;键Location:Environment:noise采用Zset结构存储当前区域的噪声排名。其他智慧灯杆信息在Redis中的存储可参考图3、图4的存储结构。

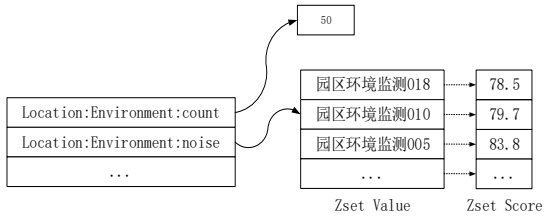


**Fig. 4 Environment monitoring noise data storage structure**

**图4 环境监测噪声数据存储结构**

针对智慧灯杆系统的实时数据查询与转存需求,图5展示了实时数据管理方式。该系统将Redis数据库分为缓存库与实时库,缓存库主要负责客户端高频查询的数据缓存,以提高系统响应时间,实时库负责智慧灯杆异构数据采集,数据转存以及Web接口发布。其中,数据点位表是Redis实时库集成、处理并转存智慧灯杆异构数据的依据,

数采程序按照数据点位表规定的异构数据转存频率和数据正常值范围等完成采集与校验，并将数据更新到Redis实时库。数据转存配置表是Redis实时库数据转存的管理依据，Redis实时库按照转存配置表的规定将不同点位的时序数据一部分转化为历史数据，一部分更新缓存，一部分通过Websocket直接推送到系统客户端，因此系统的数据来源主要有Redis实时库中的实时数据，Redis缓存库中的热点数据以及历史数据库中的历史数据。

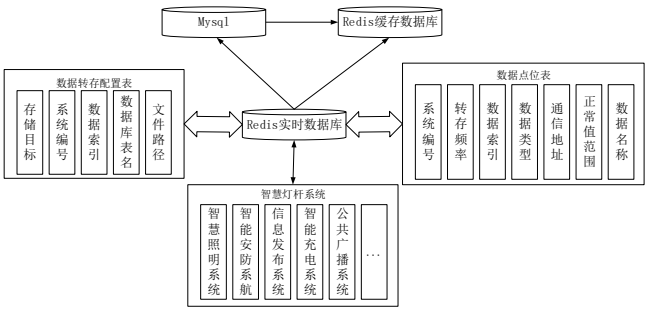


**Fig. 5　Design of time sequence database management**

**图5　时序数据库管理设计**

## 2　关键技术实现

### 2.1　基于图聚类算法的微服务拆分

基于智慧灯杆系统功能的复杂性，需要对其进行服务拆分，服务拆分后既可以将复杂业务简单化，又可以提高系统并发能力，同时突破单数据库性能瓶颈。但由于微服务拆分的粒度越细意味着维护成本越高，同时服务在运行过程中会引发分布式事务跨库查询等业界较难解决的问题，而拆分的粒度越粗就越容易产生单体应用的问题，因此服务粒度拆分后的内聚度与耦合度尤为重要。

考虑到目前微服务拆分的关键问题，图聚类算法对微服务拆分结果符合这一原则，将关联程度高的结点放到同一子图中，不同子结点关联程度低。同时，从系统本身的业务逻辑分析，充分利用系统需求说明提供的关键信息，根据数据库内置算法简洁高效地完成图架构的搭建，降低人为因素影响。步骤如下：①根据系统规范文档、系统API接口与功能说明、系统数据库中的实体表、系统所需硬件资源分别进行模块、功能、实体以及资源结点的提取，根据提取的4种结点反应系统组成；②构建结点之间的关系与权重，规定基础权重为1，如模块结点与功能结点之间、实体结点多对多的关系等为基础权重，实体结点一对多权重设置为3，一对一设置为5，以便让关联程度高的实体划分到同一服务，功能结点与实体结点的权重设置为完成该功能结点对实体结点的访问次数，资源节点与模块结点的权重本文主要考虑CPU、内存、硬盘、网络与各模块的关系，该关系可以通过Web性能测试工具获得，并将模块结点对不同资源节点的依赖程度由小到大进行排序，权重取值为数组的索引号；③选择Louvain算法对图数据进行聚类分析；④根据算法聚类分析结果进行微服务拆分。

模块结点和功能结点提取可通过系统设计文档与图1、图2获得，其中大模块结点4个，小模块结点（将实时监控模块分解）9个，功能结点63个，将两种结点存入Neo4j图数据库并建立权重关系。

实体结点提取可通过系统E-R图获得，智慧灯杆系统数据库主要包含用户、灯杆设备、工单、多媒体信息等几大主要实体，如图6所示。除主要实体外，还包含如“用户灯杆设备修改记录”“多媒体历史记录”等一系列中间实体。综合考虑所有实体，智慧灯杆系统共提取83个实体结点，并根据实体结点与功能结点的对应关系确定权重存入数据库。

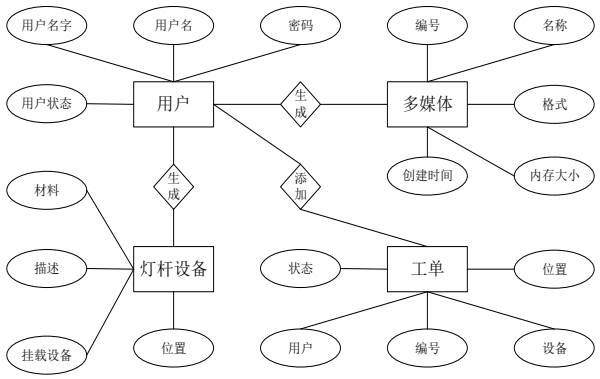


**Fig. 6　Main entity relationship diagram**

**图6　主要实体关系图**

资源节点提取主要针对系统9个模块节点进行9轮性能测试，比较4种性能指标下各模块的数据变化曲线，从而获得排序数组如表1所示，根据依赖程度数组确定模块与资源节点之间的权重并存入数据库。

**Table 1　Performance indicators and module dependency**

**表1　性能指标与模块依赖程度**

| 资源结点 | 性能指标 | 依赖程度数组（数值为模块序号） |
|---|---|---|
| CPU | CPU利用率 | [4,9,3,7,2,8,1,6,5] |
| 内存 | 内存使用率 | [2,5,9,4,8,7,3,1,6] |
| 硬盘 | IO读取速率 | [1,5,8,3,9,2,6,7,4] |
| 网络 | 网络负载 | [7,3,8,2,9,1,5,6,4] |

根据Neo4j中提供的Louvain算法对生成的图结构进行聚类分析，将智慧灯杆系统服务拆分为9个微服务，如图7所示。将其划分为两类，一类是单系统服务，另一类双系统服务。单系统包含实时数据推送、数据接收大型的微服务系统以及用户管理、运维管理、设备信息、智慧照明、环境监测小型微服务。从服务拆分上可以看出通过图聚算法结合实际系统业务逻辑将实时数据推送单独拆分出来，避免了实时数据分散在不同的服务中，造成用户实时数据存在时间差的问题。同时，将实时数据与其他数据剥离可在故障时实现隔离，以提高系统稳定性，降低开发难度。双系统服务包含安防与充电微服务、屏幕与广播微服务，其主要考虑两个系统的相互关联性较大或字段类型与数据接口类似，例如安防与充电微服务充分考虑两个子

系统的关联性,充电时需要调动相关的摄像头进行车辆识别和动态追踪,屏幕与广播微服务两个子系统从数据库结构到数据接口都极为相似,因此将两个系统合并开发可以降低开发难度。

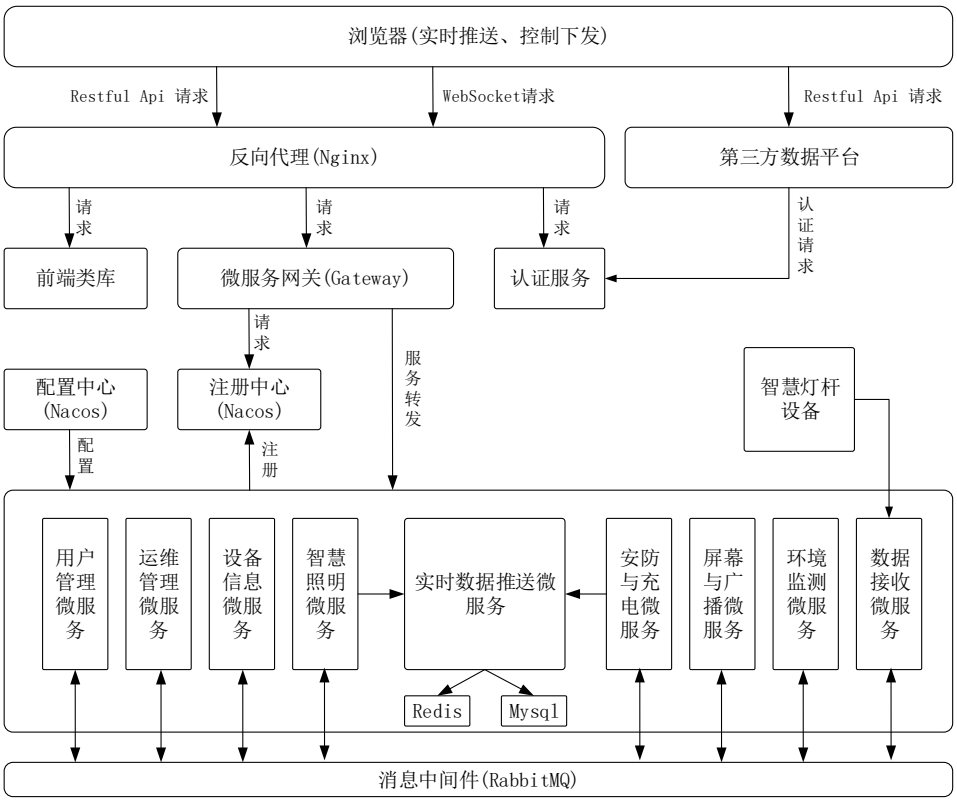


**Fig. 7 Smart lamppost microservice system architecture**

**图7 智慧灯杆系统微服务系统架构**

### 2.2 基于Redis的远程实时监控技术改进

图聚算法拆分后的服务将实时数据推送服务单独出来,进行单独开发,主要是考虑到Redis可以执行订阅/发布模式[20](Subscribe/Publish)。该模式可以将消息的发送者与接受者分离以实现消息解耦,使得程序具有更好的扩展性,但是当客户端执行订阅后,除基础命令外无法执行其他操作,客户端将一直处于阻塞状态,直至订阅通道消息的到来,并且发布的消息在Redis系统中不存储,必须先执行订阅,再等待消息发布,顺序颠倒客户端则无法接收到消息。此外,如果客户端同时在不同模式的通道,并且名称存在交集,则对于一个发布的消息,客户端将接收到两个消息。就上述问题,直接应用Redis的订阅/发布模式,在智慧灯杆系统上可能造成页面渲染实时数据短暂缺失、页面实时数据混乱等问题,为此该系统采用Redis的键空间通知(Keyspace Notification)功能,使得客户端可以通过监听特定键的方式,接收那些以某种方式改动了Redis数据集的事件。

默认配置下,键空间通知功能处于关闭状态,通过修改redis.conf文件notify-keyspace-events参数,开启此功能,该系统将参数字符串设置为“AE”,表示发送键事件所有类型的通知。基于智慧灯杆系统需要监听Redis中的特定Key的事件,在Spring Boot框架下继承KeyExpirationEventMessage Listener(监听当前key消失的事件)与KeyspaceEventMessageListener(监听所有的key事件)不太合适,需要手动创建监听类RedisKeyChangeListener并实现MessageListener接口及其onMessage方法。

借助WebSocket可以让Web端与服务端建立长链接,并可进行双向数据通信,这恰好满足智慧灯杆系统的需求,Redis监听到实时变化的特定键变化,通过WebSocket推送到Web端,完成实时数据刷新,WebSocket连接建立过程流程为:

步骤1:客户端发起连接请求。

步骤2:新建线程安全Map用来存储客户端Session。

步骤3:websocket.onopen = function(){…}//建立连接。

RedisKeyChangeListener Listener = new RedisKeyChangeListen er (RedisMessageListenerContainer, ” Environment”, session);//传输session与需要监听的键

RedisMessageListenerContainer. addMessageListener (Listener , listenerKeyName);//注册消息监听。

等待Environment键空间通知

websocket.sendMessage(message);//发送消息,便于页面及时渲染实时数据。

步骤4:websocket.onclose = function(){…}//关闭连接,移除消息监听,移除客户端对应Session。

至此,借助Redis键空间通知与WebSocket链接即可实现智慧灯杆系统实时数据推送需求,具体流程如图8

所示。

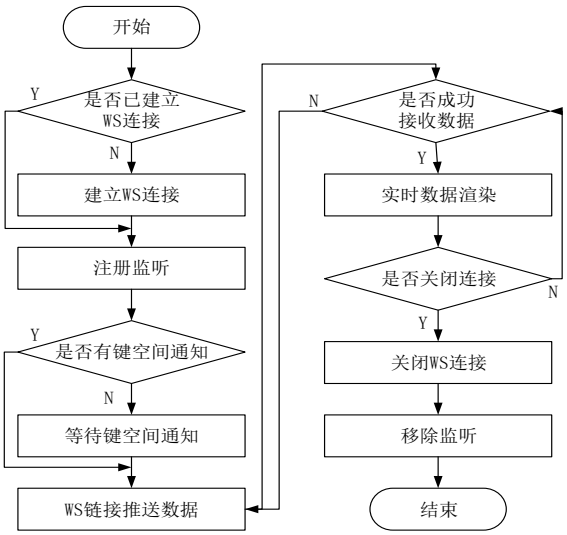


**Fig. 8　Real-time data push process**

**图 8　实时数据推送流程**

## 3　测试与展示

### 3.1　服务器性能测试

服务器性能测试是系统整体上线前必须具备的过程,通过压力测试获取特定硬件环境下网站服务能够承载的最大服务量,并获取网站最大服务量的阈值,同时解决网站潜在的问题。此次测试采用 Httpref 和 Autobench 两款 Web 服务器自动化测试工具,分别测试通过图聚算法拆分后的微服务系统与单体系统的压力测试以及图聚算法拆分后的微服务系统与简单拆分的微服务系统实时数据推送延迟测试。在云服务器搭建测试环境,单台虚拟机系统为 Centos7.6,处理器数为 2,内核数为 4,系统压力测试命令如下:

autobench --single_host --host1 x.x.x.x --uri1 /index.html --port1 xx --quiet --low_rate 50 --high_rate 500 --rate_step 50 --num_call 1 --num_conn 500 --timeout 5 --file testResult.tsv[21]

压力测试结果如图 9 所示,面对压力测试时,该系统搭建的微服务系统相较于单体系统具有较大优势,主要体现在响应时间与响应稳定性。响应时间方面,微服务系统一直处于较低水平,单体系统则在并发数达到 300 后有较大延迟;稳定性方面,单体系统存在较大波动。

实时数据推送延迟测试结果如图 10 所示,在并发数达到 400 后算法拆分的微服务系统相较于按照子系统进行微服务拆分出现明显优势,主要原因是通过图聚算法将实时数据推送服务单独出来,集中管理实时数据在提高效率的同时降低了异构时序数据间的时间差。图聚算法拆分的微服务架构和 Redis 的改进应用通过以上测试以搭建智慧灯杆系统满足系统实际需求。

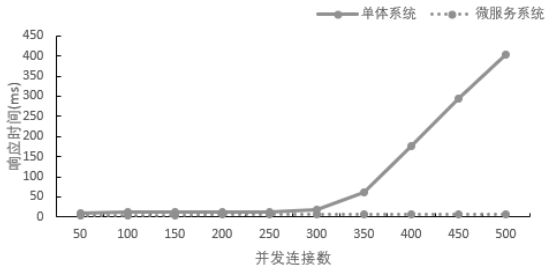


**Fig. 9　Comparison of response time between single system and microservice system**

**图 9　单体系统与微服务系统响应时间对比**

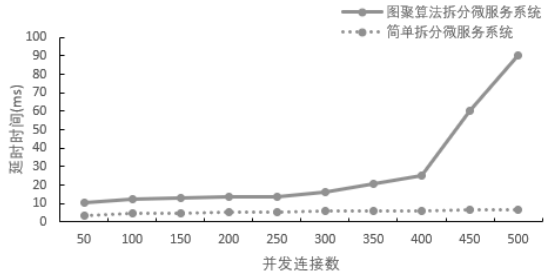


**Fig. 10　Comparison of real-time data latency between two microservice architectures**

**图 10　两种微服务架构实时数据延迟时间比较**

### 3.2　界面展示

智慧灯杆系统首页如图 11 所示,将系统中主要的数据如开关灯时间、控制模式、各种设备数量、能耗和故障报警进行集中展示。同时,环境等实时变化的数据也可得到快速刷新,通过首页能够大致了解智慧灯杆系统。

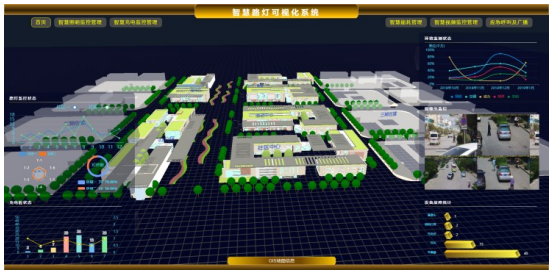

**Fig. 11　Home page of intelligent lighting pole system**

**图 11　智慧灯杆系统首页**

智慧灯杆系统每个子系统中间部分为 GIS 地图详细标记该设备位置信息,通过点击灯杆即可弹出该子系统详细信息,智慧照明界面如图 12 所示,包含智慧照明子系统运行情况、单系统能耗分析以及亮灯率与亮灯时长预测等。

**Fig. 12　Intelligent lighting interface**

**图 12　智慧照明界面**

## 4 结语

本文从实际系统需求出发，搭建系统模块结点、功能结点、资源结点和实体结点，采用图聚类算法对系统图结构进行聚类与服务拆分，将系统划分为实时数据推送等九大微服务系统。相较于简单按照子系统类型进行拆分的微服务，其在实时数据响应时间和异构数据时间差方面有较大优势。同时，本文针对实时数据推送需求设计并实现了双 Redis 数据库分离时序数据与缓存数据，并且对时序数据存储与管理进行设计，最后借助 WebSocket 推送时序数据，缓解了由原来仅依靠轮询方式带来的网络带宽压力较大等问题。最终将该系统应用于智慧灯杆系统，运维管理人员通过浏览器即可实现对城市智慧灯杆的实施监测与控制。

**参考文献：**


[1] XIAO H, LI W C, ZHU Y C, et al. Application research of multi-function intelligent lamp pole system[J]. Journal of Lighting Engineering, 2019, 30(4): 1-5.
肖辉，李文超，朱应昶，等. 多功能智慧灯杆系统应用研究[J]. 照明工程学报，2019，30(4)：1-5.

[2] LIU Q Y, WANG H B, WANG T X, et al. Research on the investment, construction and operation mode of urban multi-functional smart poles [J]. Smart City, 2018, 4(20): 1-3.
柳庆勇，王海滨，王天小，等. 城市多功能智能杆投资建设运营模式研究[J]. 智能城市，2018，4(20)：1-3.

[3] Ministry of Housing and Urban-Rural Development, et al. Guiding opinions on accelerating the development of digital households and improving the quality of residents (JB (2021) No. 28)[EB/OL]. https://www.gov.cn/zhengce/zhengceku/2021-04/17/content_5600311.htm.
住房和城乡建设部等. 关于加快发展数字家庭 提高居民品质的指导意见(建标(2021)28号)[EB/OL]. https://www.gov.cn/zhengce/zhengceku/2021-04/17/content_5600311.htm.

[4] GUI C J, ZENG X H. Real-time control technology of intelligent air conditioning mobile environment combined with Redis and WebSocket [J]. Software Guide, 2020, 19(1): 190-194.
桂成杰，曾献辉. 结合 Redis 与 WebSocket 的智能空调移动环境实时控制技术[J]. 软件导刊，2020，19(1)：190-194.

[5] KONG X Y, DAI Z H. Design of high-performance Web configuration system based on WebSocket and Redis[J]. Information Technology and Standardization, 2021(3): 51-56.
孔晓阳，代真虎. 基于 WebSocket 与 Redis 的高性能 Web 组态系统设计[J]. 信息技术与标准化，2021(3)：51-56.

[6] WEI J D, TU J H, LAI S D, et al. Design and implementation of moving target track tracking system based on WebSocket and Redis[J]. Computer Era, 2021(8): 76-79, 83.
魏江东，涂继辉，赖少东，等. 基于 WebSocket 和 Redis 的移动目标轨迹跟踪系统设计与实现[J]. 计算机时代，2021(8)：76-79，83.

[7] PAN F, WANG X T, SONG Y T, et al. Design and implementation of real-time battlefield situation push scheme based on Redis and WebSocket [J]. Software Guide, 2018, 17(7): 143-146.
潘峰，王笑天，宋钰涛，等. 基于 Redis 与 WebSocket 的战场态势实时推送方案设计及实现[J]. 软件导刊，2018，17(7)：143-146.

[8] XIN Y Y, NIU J, XIE Z J, et al. Overview of microservice architecture implementation framework [J]. Computer Engineering and Application, 2018, 54(19): 10-17.
辛园园，钮俊，谢志军，等. 微服务体系结构实现框架综述[J]. 计算机工程与应用，2018，54(19)：10-17.

[9] JAMSHIDI P, PAHL C, MENDONÇA N C, et al. Microservices: the journey so far and challenges ahead[J]. IEEE Software, 2018, 35(3): 24-35.

[10] ALMEIDA W H C, DE AGUIAR M L, HAZIN R R, et al. Survey on microservicearchitecture-security, privacy and standardization on cloud computing environment [C]// Athenas: International Conference on Sustainable Environment and Agriculture, 2017.

[11] ZHONG C X, LI S S, ZHANG H, et al. Micro-service granularity evaluation from the perspective of bounded context [J]. Journal of Software, 2019, 30(10): 3227-3241.
钟陈星，李杉杉，张贺，等. 限界上下文视角下的微服务粒度评估[J]. 软件学报，2019，30(10)：3227-3241.

[12] EVANS E. Domain-driven design reference: definitions and pattern summaries[M]. Alaska: Dog Ear Publishing, 2014.

[13] ZHOU J H. Research on key technologies of microservice architecture and implementation of general framework [D]. Guangzhou: Guangdong University of Technology, 2019.
周家昊. 微服务架构关键技术研究与通用框架实现[D]. 广州：广东工业大学，2019.

[14] CHEN X, QIU Z Z. Research on semantic similarity analysis based on Siamese-BIGRU-Attention [J]. Journal of Dalian Jiaotong University, 2022, 43(4): 113-116.
陈鑫，邱占芝. 基于 Siamese-BIGRU-Attention 的语义相似度分析研究[J]. 大连交通大学学报，2022，43(4)：113-116.

[15] LIAO Z F, ZHOU G E, LI J F, et al. Algorithm for french and semantic similarity of Chinese text[J]. Journal of Hunan University (Natural Science Edition), 2016, 43(2): 135-140.
廖志芳，周国恩，李俊锋，等. 中文短文本语法语义相似度算法[J]. 湖南大学学报(自然科学版)，2016，43(2)：135-140.

[16] GYSEL M, KÖLBENER L, GIERSCHE W, et al. Service cutter: a systematic approach to service decomposition[C]// Proceedings of European Conference on Service-Oriented and Cloud Computing, 2016: 185-200.

[17] MAZLAMI G, CITO J, LEITNER P. Extraction of microservices from monolithic software architectures[C]// 2017 IEEE International Conference on Web Services, 2017: 524-531.

[18] SHANG C Q. Research and application of single application service partition method in microservice scenario[D]. Wuhan: Huazhong University of Science and Technology, 2020.
尚晨琦. 微服务场景下单体应用服务划分方法的研究与应用[D]. 武汉：华中科技大学，2020.

[19] ZENG C Y, LI J X. Application of Redis in cache system[J]. Microcomputer and Application, 2013, 32(12): 11-13.
曾超宇，李金香. Redis 在高速缓存系统中的应用[J]. 微型机与应用，2013，32(12)：11-13.

[20] OUYANG W C. Design and implementation of distributed cache e-commerce platform based on Redis technology [D]. Nanchang: Nanchang University, 2018.
欧阳文臣. 基于 Redis 技术的分布式缓存电商平台设计与实现[D]. 南昌：南昌大学，2018.

[21] CHENG T T, CUI J D. Design and implementation of intelligent street lamp remote monitoring platform[J]. Computer Application and Software, 2018, 35(3): 93-97, 101.
程婷婷，崔佳冬. 智慧路灯远程监控平台的设计与实现[J]. 计算机应用与软件，2018，35(3)：93-97，101.


(责任编辑：孙 娟)