Computer Science and Application 计算机科学与应用, 2020, 10(6), 1209-1223
Published Online June 2020 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2020.106125

Hans汉斯

Design and Development of Remote Monitoring Platform Software for High-Load Electric Vehicles Based on 4G Communication

Cheng Wang, Fangjie Chen, Chenlong Kang, Yongfeng Zhang

Ordnance NCO Academy, Army Engineering University, Wuhan Hubei
Email: 939025387@qq.com

Received: May 27th, 2020; accepted: Jun. 5th, 2020; published: Jun. 12th, 2020

Abstract

In the context of the expanding market size of electric vehicles and the continuous increase in the penetration rate of pre-loaded connected equipment, it's an important part of forming an intelligent connected vehicle ecosystem that targeted developing a vehicle remote monitoring platform that integrates data monitoring and remote management functions. This paper designs and develops a remote monitoring platform for electric vehicles based on 4G communication technology and mainly uses the Go programming language to complete data processing. Starting from the overall platform architecture design, load balancing is implemented in combination with the hardware and software level during the development process, so that the platform can effectively deal with the simultaneous access of a large number of terminals and highly concurrent data processing and management, and realize remote monitoring of vehicle data. Finally, after joint software and hardware testing and verification, the overall operation of the platform is stable, which can realize remote monitoring of connected vehicles and provide support for the development of related remote services.

Keywords

Remote Monitoring Platform, Electric Vehicle, Load Balancing, Go Language

基于4G通信的高负载电动汽车远程监控平台软件设计与开发

王　成，陈方杰，康辰龙，张永峰

陆军工程大学军械士官学校，湖北　武汉

文章引用: 王成, 陈方杰, 康辰龙, 张永峰. 基于 4G 通信的高负载电动汽车远程监控平台软件设计与开发[J]. 计算机科学与应用, 2020, 10(6): 1209-1223. DOI: 10.12677/csa.2020.106125

Email: 939025387@qq.com

收稿日期：2020年5月27日；录用日期：2020年6月5日；发布日期：2020年6月12日

摘　要

在电动汽车的市场规模不断扩大的环境以及前装车联网设备渗透率持续增长的态势下，针对性开发集数据监控和远程管理功能于一体的车辆远程监控平台，是组建智能网联汽车生态圈中的重要一环。本文设计并开发了一种基于4G通信技术，并主要利用Go编程语言完成数据处理的电动汽车远程监控平台。从整体平台架构设计出发，开发过程中结合软硬件层面实施负载均衡，使平台可以有效应对大量终端的同时接入，并满足高并发数据处理及管理的需求，实现对车辆数据的远程监控。最后，经过软硬件联合测试验证，平台总体运行稳定，可对接入车辆实现远程监控，并为相关远程服务的开发提供支持。

关键词

远程监控平台，电动汽车，负载均衡，Go语言

1. 引言

目前，5G 通信技术仍处于加速架设的进程中，并未全面商用，车联网产业的发展受到一定程度的制约。尤其是对于车路协同、自动驾驶、智慧交通系统等前沿方向的研究，高度依赖高带宽、低时延的先进通信网络[1]。因此，现阶段车联网技术的推进，更多的是聚焦于车辆的“网联化”，在获取车辆数据的基础上，提供对网络状况无苛刻要求的相关远程服务。如今，基于车载终端等车联网硬件设备的前装，对于车辆的数据采集具有极大的促进作用，丰富的数据样本不仅能为车辆的研发提供实质性的参考依据，还能对实时监控的车辆提供故障诊断方案、售后服务等扩展功能。同时，基于充足样本的大数据挖掘应用，将会极大促进人机交互以及辅助驾驶方面的研究[2]。然而不论是从数据管理或是数据应用场景挖掘的角度来看，都需要以远程监控平台作为载体[3]。多终端的同时接入以及高并发的数据传输也将对平台产生巨大的压力，因此对平台在高负载环境下的负载均衡能力及稳定性提出了较高的要求。平台的扩展性及适应性也决定了其能否广泛运用于不同场景，将此特性贯穿于设计与开发的过程，也将为后续的开发提供便利。当前，诸如汽车分时租赁、远程车辆故障诊断等基于车辆监控数据的车联网服务，在多终端接入场景下的系统稳定性面临着较大的挑战。而能够较好应对高负载压力的服务平台，大多是以目标功能为导向的设计开发，整体系统可移植性有限，在未来进行功能扩展或二次开发时将增加开发成本及周期[4]。因此，在 5G 通信技术尚未完全铺开的环境下，本文基于成熟的 4G 通信技术设计开发的车辆远程监控平台是一种能满足对车辆数据进行精确采集，且可移植于不同应用服务系统的方案，在一定程度上提升开发的效率。

2. 电动汽车远程监控平台整体架构

目前，基于车联网的远程监控技术主要覆盖“端、管、云”3 个层面。“端”是指前装于车辆的车载终端及相关汽车电子；“管”包括诸如 4G、5G 车载蜂窝通信等无线通信技术；“云”则是基于数据

驱动的应用开发[5]。本文设计开发的电动汽车远程监控平台依据上述的技术范畴，从宏观角度将其架构划分为车载终端、数据处理系统、管理端及数据应用接口四个部分。其主要结构如图 1 所示。

1) 车载终端

出于成本的考虑，现今主流的电动汽车在整车数据通信部分依然是采用非常成熟的 CAN 总线系统，这也给原始数据的采集带来了较大的便利性。市面上的车载终端一般通过 OBD 接口与各 ECU 进行通信，以实现车辆在行驶、充电和驻车环境下的数据的获取，同时负责数据的转码、加密和组包。

2) 数据处理系统

数据处理系统主要由服务器和数据库组成。被车载终端处理完成的数据包通过 4G 无线网络，按照 TCP/IP 协议的要求，上传至服务器的对应端口。服务器端按照编译的流程，对数据包的字段进行解析和预处理，生成以时间为序列且可读的原始数据表，并最终交由数据库存储，以供后期数据调用。

3) 管理端

管理员可以运用跨平台的管理系统实现对远程监控平台的管理。管理端通过唯一预设的 API 与数据处理系统进行通信及执行数据调用，保证了整体平台的安全性。同时，具有权限的管理员均能通过身份验证的方式在管理端进行操作，提高了管理端的使用便利性。

4) 数据应用接口

在数据处理系统中预留数据应用接口，在管理端开放使用权限即可接入并获取车辆数据，以供不同场景下的应用及服务。

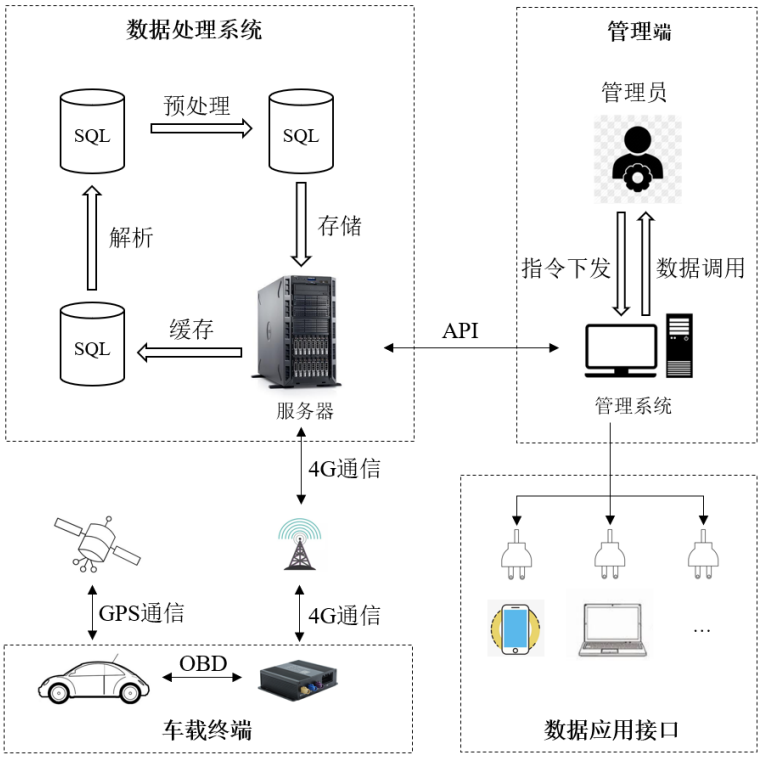

Figure 1. Architecture of remote monitoring platform for electric vehicles
图 1. 电动汽车远程监控平台架构

3. 电动汽车远程监控平台软件设计

为了提高远程监控平台的应用适应性，数据处理系统采用 B/S (Browser/Server)与 C/S (Client/Server)

混合型架构，即浏览器/服务器与客户端/服务器结合的模式，以 B/S 架构为主。B/S 架构的逻辑为：在前端进行的处理操作，主要的业务逻辑都交由后台来处理操作，前端仅负责请求、渲染等少量逻辑处理。其特点在于无需安装特定的客户端，只需要 web 浏览器的支持就可以在管理系统上完成一系列的操作，不受操作系统的限制，适应性强[6]。而对于云平台延伸功能，诸如移动终端(APP)显示、车载终端与平台交互等功能，由于其对丰富界面和操作以及快速响应的要求，则采用只有一层交互的 C/S 架构。

B/S 架构主要将系统划分为三个部分，分别为表现层、逻辑层和数据层。表现层负责处理客户端请求，完成与后台的通信，并显示输出处理结果[7]。逻辑层负责处理业务逻辑，与数据库完成数据交互。数据层负责接受请求，对数据内容进行运算处理，并返回处理的结果。由于云平台部分的整体系统架构主要采用的是 B/S 模式，因此本文以 B/S 架构的结构形式作为依据，对数据处理系统进行软件设计。

3.1. 表现层

表现层即远程监控平台中的管理系统，是访客或管理人员直接在系统进行操作的入口。对于表现层而言，其任务是将监控数据以功能模块的形式，直观的呈现给管理人员，并具备将指定数据以列表形式导出的功能。其主要功能可预设为管理账户登录与注册、车辆数据监控、故障报警、车载终端管理及数据查询等 5 个模块。关于表现层模块层级关系可由图 2 所示。

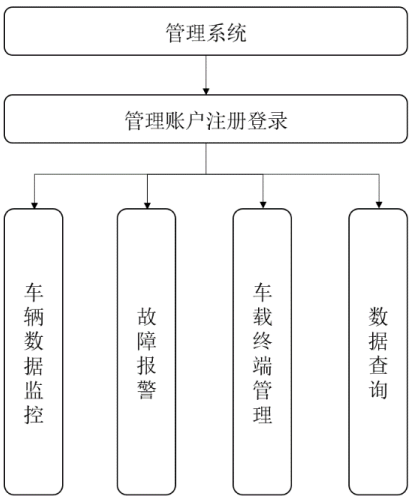

Figure 2. Design of presentation module
图 2. 表现层模块设计

3.2. 逻辑层

逻辑层部分主要是处理相关的业务逻辑。对于云平台而言，需要处理的业务逻辑分为两类，一类为管理系统即前端的各项操作指令，另一类为后台对上传数据处理。

对于处理前端操作指令的业务逻辑，主要是对数据库接口的调用。无论是车辆数据的查询或是相关信息的编辑管理，本质上是对数据库中存储的数据进行调用。而逻辑层需要完成的工作就是针对各个功能模块要呈现给客户端信息，对需要调用的数据信息做决策，最后从数据库开放的接口调用所需的数据，实现对采集数据和录入数据的增删查改。同时，对于访客的登录登出等操作，须完成用户信息的比对校验工作。

对于实时数据的处理属于默认的业务逻辑，同时也时数据处理系统最核心的功能。数据处理系统首先需要与车载终端建立通信连接，完成车载终端在对应端口上的登录和身份验证，随后开始接收上传的实时数据。当接收到上传数据后，第一步进行数据包的校验，查看数据包是否完整。若不完整则向车载终端返回数据补

发命令，并舍弃不完整数据包。校验成功的数据包则交由解析模块进行处理，按照 TCP/IP 协议内容进行解包，将数据包还原成可读形式的数据信息，再对数据进行预处理，筛去异常值，最后交由数据库进行管理。

而就逻辑层的整体需求来看，其最主要的就是处理高并发负载。云平台与多台车载终端连接，且车辆运行时数据上传频率较高，因此会造成服务器的高负载压力。逻辑层采用的编译语言、处理机制以及服务器架构都需要配合高并发的工作模式，保证整体系统的良好运行以及数据的精准管理。

3.3. 数据层

数据层即数据库部分，数据库主要的工作是对解析完成的实时数据进行存储、管理及调用。存储工作即将服务器端接收的数据存储至服务器端，为数据调用或云计算提供支持。管理工作则是依靠数据库管理系统，负责数据的组织、操作、维护、控制及保护，为数据库系统的核心部分。为了便于数据的调用，还需要对数据进行分类管理并建立索引目录。调用工作是为外部命令提供数据调用接口，以实时数据作为核心内容为客户端提供相应的服务。

综上所述，远程监控平台软件部分的核心就是逻辑层和数据层的设计，平台架构层面即表现为数据处理系统和管理端的开发。

4. 数据处理系统及管理端的开发

4.1. 整体架构

数据处理系统和管理端由硬件和软件两部分构成。硬件部分主要包括无线网络设备和服务器，软件部分则包括操作系统、编程语言和数据库。

考虑到数据处理系统需要同时处理平台与车载终端之间上下行的数据传输，且多终端的同时接入带来的高并发访问压力，仅通过软件层面的负载均衡会给开发带来较大的难度。为保证系统运行的稳定性以及后期扩展性，这里采用“一主多从”的服务器架构。在主服务器宕机时，可将任务切换到从服务器上进行处理，这种模式也将大大提高系统的可靠性[8]。其物理架构如图 3 所示。

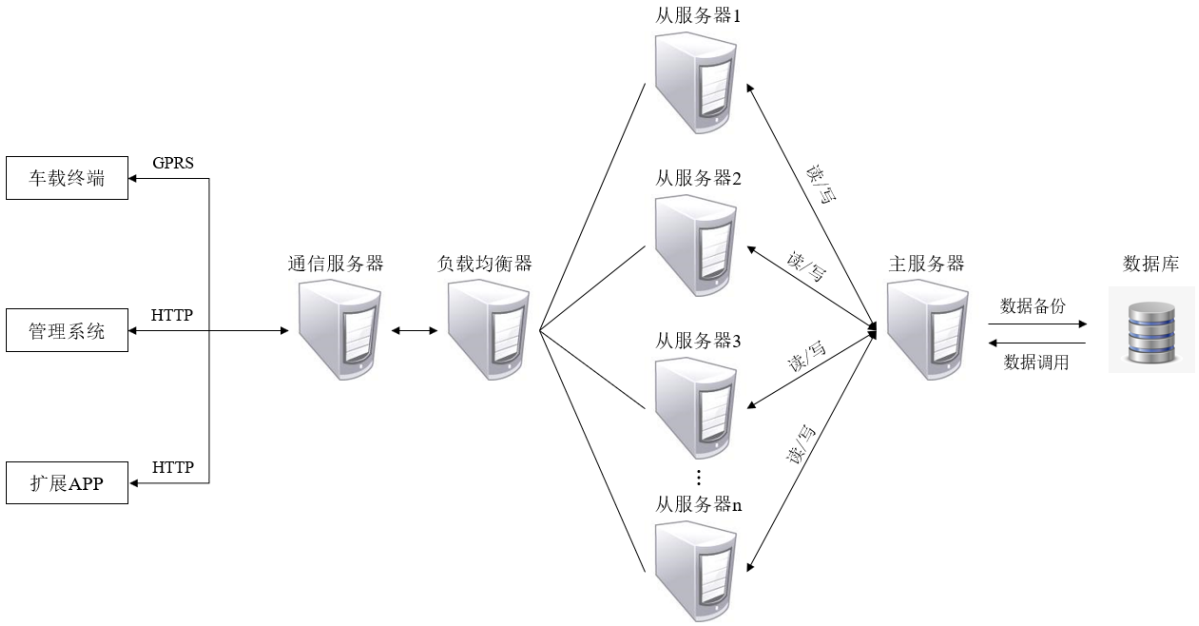

Figure 3. Server physical architecture
图 3. 服务器物理架构

软件部分开发技术及相关工具如表 1 所示。

Table 1. Software development technology and tools
表 1. 软件部分开发技术及工具

项目	采用技术或工具
软件流程设计工具	Microsoft Visio
软件开发工具	Sublime Text3/LiteIDE/CSS3
软件开发语言	PHP/Go/JavaScript
软件开发框架	Laravel
软件开发技术	HTML5
操作系统	Windows7/Linux
数据库	MySQL
Web 服务器	Apache
浏览器	Chrome

数据处理系统主要的功能以数据包解析、数据库管理为核心进行开发。因此，系统整体结构可分为数据解析和数据管理及调用两大主要板块。同时，数据解析部分是基础，是为数据管理及调用部分服务，整体系统的根本目的是实现对实时数据的管理及调用。远程监控平台系统主程序架构如图 4。

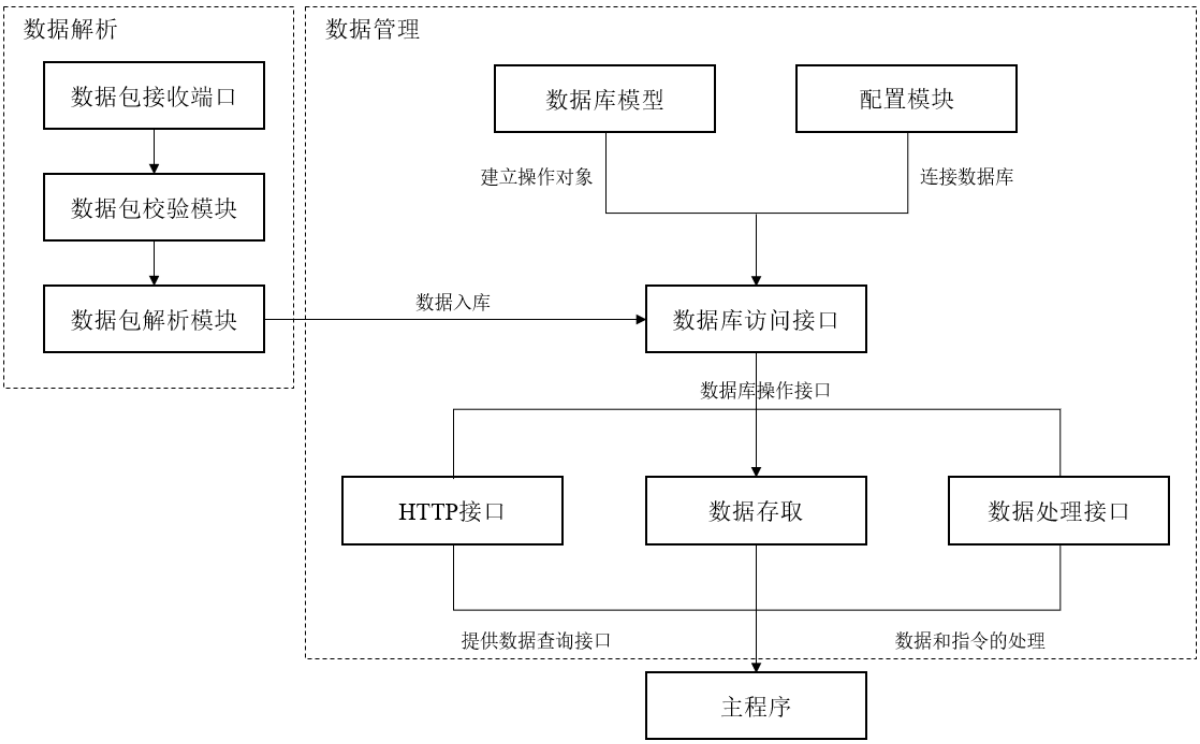

Figure 4. Main program framework of remote monitoring platform
图 4. 远程监控平台主程序架构

4.2. 主程序设计开发

数据处理系统及管理端均采用基于 PHP 编程语言的 Laravel 框架进行开发。Laravel 框架集成了极为

丰富的预置组件，只需要聚焦于需要实现的功能并对其进行编写。对云平台及管理系统主要结构的开发流程如下：

1) 配置 Laravel 框架开发环境

为了开发的便捷性，采用 WampServer 作为开发的主要工具。WampServer 集成了开发过程中关联的服务器及数据库工具，并自带一套 PHP 解释器，提供了完善的开发环境，省去了以传统方式配置 PHP 开发环境的复杂过程。完成 WampServer 软件的部署后，首先通过 Composer 工具箱，全局下载并安装 Laravel 开发框架，命令如下：

$ composer global require "laravel/installer=~1.1"

安装完成后将 c:/users/desktop/.composer/vendor/bin 路径导入系统环境变量的路径中，框架可控制执行的文件能在控制台命令行中检索到，后期开发过程中仅需在控制台中直接运行命令即可。

2) 创建平台主程序

在 Windows 系统下的 Powershell 中打开控制台终端，进入预先为项目预留且具有完全权限的文件夹，执行 laravel 的应用创建命令，建立以项目名称为索引目录的新应用文件框架。本文将云平台命名为 monitorcontrol，这里采用命令：

$ composer create-project laravel/laravel monitorcontrol--prefer-dist

生成名为 monitorcontrol 的文件夹及 Laravel 主程序，生成后进入该文件夹：

$ cd monitorcontrol

主程序运行于服务器端，主要处理云平台和管理系统部分的业务逻辑。monitorcontrol 主程序文件夹中很多都是自动生成的与框架相关且能自由调用的预置文件，根据系统需求，主要对 app 文件夹中各文件代码进行编写。文件框架中各主要文件及文件夹作用总结如表 2 所示。

Table 2. The main function of each file and folder in the file framework
表 2. 文件框架各文件及文件夹主要作用

文件或文件夹	作用
app	囊括项目中的 controllers，models，views 和 assets，存放应用程序的业务逻辑代码。
bootstap	存放支持系统启动时的必备文件。
config	存放用于个性化定制框架功能的配置文件，包括运行规则、数据库、中间件等。
database	数据库操作相关的文件。
public	存放可供非开发人员查看的文件内容，包含服务器目录等。此文件夹还能存放 css 等静态资源。
resources	应用资源，包含视图文件、原生资源文件和本地化语言文件。
routes	包含建立的应用程序所有的路由。
storage	存放开发框架中支持功能运行的临时文件，且此目录在 web 上可以写入，由 Laravel 维护管理。
test	用来存放测试文件，便于进行单元测试。若要进行批量单元测试，可借助编译语言中内置的 Artisan 工具。
vender	存放第三方代码及额外的预包装功能插件，Composer 依赖模块。
seed	存放关系型数据批量导入项目数据层表单的引导文件。
controllers	存放用于支持执行业务逻辑及完成处理模型、相关文件等加载功能的控制器。
auth.php	在程序中进行直接的身份验证操作。
session.php	控制 Laravel 管理诸如 session driver，session lifetime 等用户 session。

3) 开发 API

数据处理系统及管理端需要预先配置数据操作的 API 接口。在 laravel 框架中编写接口文件比较方便，在主程序目录下的 routes/api.php 文件中编写路由即可，编写方式与网页路由相同。在路由编写完成后，在 App/Http/Controller 目录下建立 api 文件夹，存放所有的接口文件。管理端对 API 的控制具有最高权限，用户对数据操作的前提是通过管理端的授权并完成身份验证。

本文创建的接口均为 RESTful API。RESTful API 遵循统一的接口原则，使用 HTTP 动词表示操作：GET——获取资源；POST——创建资源；PUT——更新资源；DELETE——删除资源[9]。RESTful 的无状态性决定了只需要维护资源的状态，而不需要考虑客户端状态，这样类型的 API 简单精炼，还能有效实施负载均衡。RESTful API 采用 URI 标识资源、使用“链接”关联相关的资源、运用统一的接口、使用标准的 HTTP 方法和无状态性的特点，是在本系统不同层级间构建标准交互接口的理想方案。

客户端在执行与当前功能相关的数据调用请求时，统一从预设的数据调用接口接入监控平台服务器，所有的数据交换均遵循通用标准，不存在运行于不同操作系统的客户端需要用不同的编程语言来实现数据交互的问题，以此大大简化了为实现监控数据在不同平台设备间调用的开发流程。

4.3. 云平台数据库设计

4.3.1. 数据库概念模型设计

对于数据处理系统及管理端而言，车辆监控数据库是整体系统的底层基础。而在数据库设计与开发的过程中，其概念模型的规划显得尤为重要，概念模型建立的本质是对数据库的存储结构进行设计。目前，大部分数据库设计均使用实体-关系模型，即 E-R 模型。E-R 模型从结构上分析拥有实体、属性、关系三个成分，分别用长方形、椭圆形、菱形来表示。就本平台而言，系统管理员及管理系统各功能模块充当的是实体，模块下的各信息列表则为属性，关于数据处理系统的数据库概念模型可由图 5 表示。

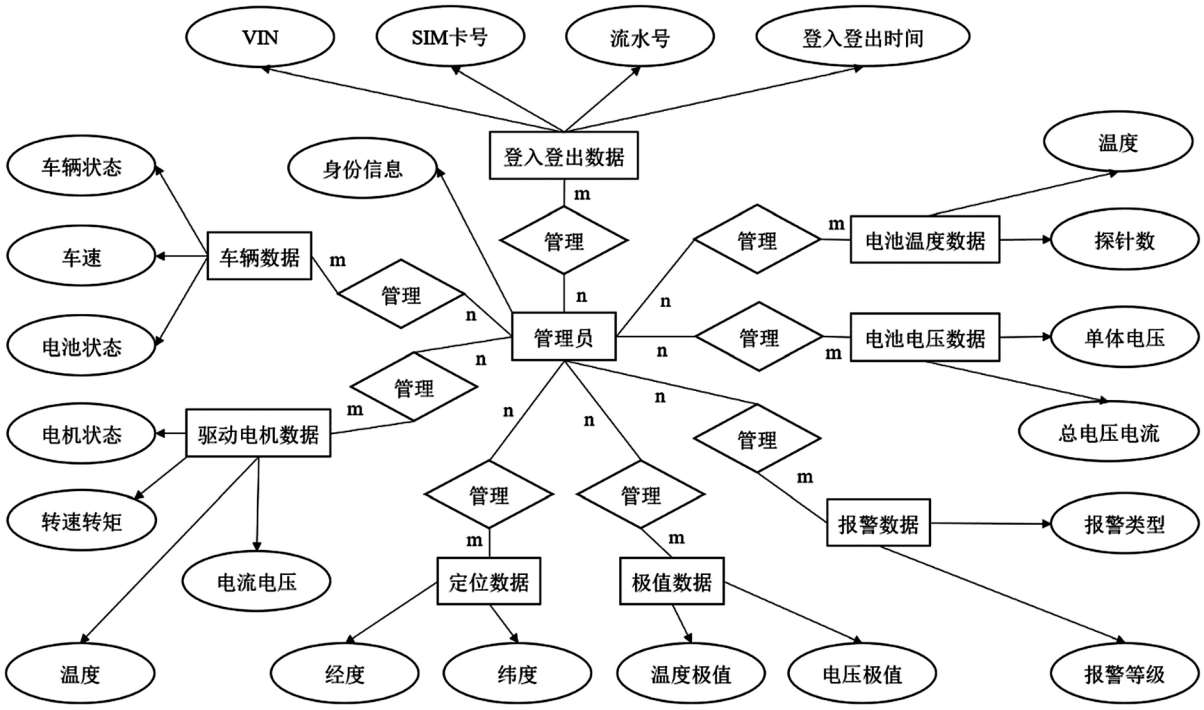

Figure 5. E-R diagram of remote monitoring platform database
图 5. 远程监控平台数据库 E-R 图

4.3.2. 数据库信息列表设计

从数据库 E-R 图出发，并综合《GB/T 32960 电动汽车远程服务与管理系统技术规范》的采集要求，为满足系统的监控功能须创建以下 9 个数据表：登入登出信息表(t_car_login_and_logout)；车辆数据表(t0x01)；驱动电机数据表(t0x02)；燃料电池数据表(t0x03)；车辆定位数据表(t0x05)；极值数据表(t0x06)；报警数据表(t0x07)；电池电压数据表(t0x08)；电池温度数据表(t0x09)。为便于开发过程中查阅及核对，将表单名称以与国标中类型编码匹配的 t0x0_形式命名。由于篇幅限制，下面仅对登入登出信息及车辆数据表的内容、数据类型及结构进行设计，其余表单设计不再赘述。

1) 登入登出数据表(t_car_login_and_logout)

登入登出信息表存放车辆在远程监控平台上登入登出数据，其结构与包含信息如表 3 所示。

Table 3. Login and logout information form
表 3. 登入登出信息表

序号	字段代号	字段含义	数据类型
1	VIN	车架号	char(17)
2	login_time	登入平台时间	timestamp
3	login_number	登入流水号	smallint(2)
4	ICCID	SIM 卡号	char(20)
5	energy_storage_sys_code	储能装置编号	text
6	logout_time	登出平台时间	timestamp
7	logout_number	登出流水号	smallint(2)

2) 车辆数据表(t0x01)

车辆数据表存放监控车辆的实时监控数据，包含车辆运行的各种信息，其结构与相关信息如表 4 所示。

Table 4. Vehicle data sheet
表 4. 车辆数据表

序号	字段代号	字段含义	数据类型
1	report_at	上报时间	timestamp
2	status_vehicle	车辆状态	char(1)
3	status_charge	充电状态	char(1)
4	speed	车速	smallint(6)
5	mileage	累计里程	int(11)
6	voltage	总电压	smallint(6)
7	electricity	总电流	mediumint(6)
8	SOC	SOC	tinyint(4)
9	status_dcdc	DC/DC 状态	char(1)
10	gear	档位	char(1)
11	accelerator	加速踏板行程	smallint(6)
12	brake	制动踏板状态	smallint(6)
14	resistance	绝缘电阻	mediumint(9)

至此，数据库的设计全部完成。

4.4. 实时数据报文解析入库模块开发

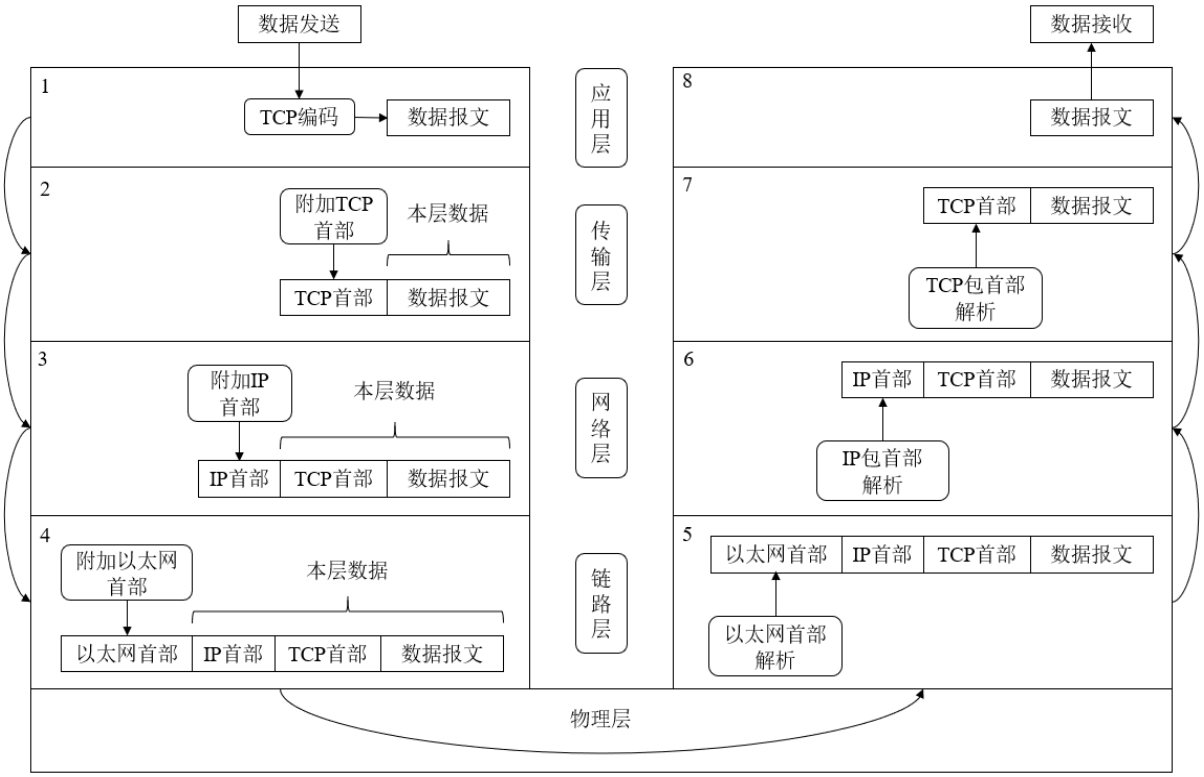

Figure 6. Data packaging and unpacking process
图 6. 数据封装及解包流程

车载终端采集的车辆实时数据是以 TCP/IP 协议报文的形式，通过 4G 无线网络传输至数据处理系统的服务器端。因此，对于报文的解析处理同样是依赖 TCP/IP 协议。TCP/IP 参考模型将协议分为四个层次，在实时数据传输的实际应用中，TCP/IP 协议族会对采集的数据按照这四个层次的层级按照顺序进行包装，每一层的包装操作如下：

1) 应用层：应用程序按照 TCP 协议对采集的数据进行编码处理。

2) 传输层：参照 TCP 协议中的规定，将上一层传送下来的字节流分割为报文字段，并在数据前端封装 TCP 首部。

3) 网络层：参照 IP 协议，将上一层传送下来的封装数据前端加装 IP 首部，确认接收 IP 包的主机。

4) 链路层：将等待发送的封装数据加装以太网首部，并进行 CRC 编码，最终生成的以太网数据包通过物理层进行发送[10]。

当实时数据完成上述四层封装后，将会传输到服务器的接收端口。由于其从上到下、按层包装的特点，接收端的解包流程则与发送端的包装流程互逆，数据处理及传输流程如图 6 所示。

至此，远程监控平台对实时数据的报文完成了初步解析，获得了完整的数据包。由于数据包是以数据帧的形式呈现在服务器端，属于不可读的信息，因此，在后台需要对数据帧进行二次解析，即对应 GB/T 32960 对数据包各字段的定义，将数据帧还原为可读的信息[11]。将数据帧还原的操作流程设计依据为数据包的结构，数据包结构如表 5 所示。

Table 5. Data package structure and definition
表 5. 数据包结构及定义

起始字节	定义		数据类型	描述
0	起始符		STRING	固定为 ASCII 字符‘##’，用“0x23，0x23”标识。
2	命令单元	命令标识	BYTE	命令发起方的唯一标识。
3		应答标识	BYTE	令接收方识别此包类型。
4	唯一识别码		STRING	传输数据时使用的车辆 VIN。
21	数据单元加密方式		BYTE	对 RSA 算法、AES128 位算法等加密方式进行标识。
22	数据单元长度		WORD	数据单元总字节数。
24	数据单元		—	车辆相关的各类型数据编码。
末位	校验码		BYTE	采用 BCC (异或校验)法。

根据其结构特点，平台接收到数据包后应立即进行识别，国标规定起始符为“0x23，0x23”数据包为有效数据，不符合规定起始符的数据包不予解析，并向车载终端回复补发命令；接着进行数据帧异或校验，未通过校验的数据包存在通讯错误，视为无效数据并返回未通过校验信息；之后进行完整性校验，即对数据单元长度标识和实际数据单元长度进行比对，若匹配则进行解析的下一步流程，若不匹配则舍弃当前数据包并向车载终端回复补发命令；完成各识别和校验操作后，即可对数据信息进行还原及入库的操作；由于为保障信息传输的安全性对数据进行了加密，因此首先需要针对加密类型进行解密，获得可操作数据帧；定位命令单元，解析命令标识及应答标识，确认其为车载终端与远程监控平台间交互的何种命令；寻址至数据帧第 24 位，将信息单元编码对照 GB/T 32960 中对各类型数据编码的定义，转换为可读信息。车辆 VIN 作为数据帧的识别码，本质上代表了当前数据包的归属对象，与可读的数据信息整合后，形成具有对象信息的实时数据。最后将可读数据按照其类型编码，填充入前文设计的数据库数据表单中，交由数据库进行存储与管理。实时数据报文在服务器端的解析入库流程可由图 7 所示。

由于云平台接入的车载终端数量较多，加之各车载终端的实时数据上报频率较高，云平台服务器端亟需解决的是应对高并发实时数据的接收与处理。前文针对此问题，采用了“一主多从”的分布式服务器物理架构，软件部分的程序编写同样需要对此进行优化。在此，本文选用 Go 语言，针对高并发的实时数据解析入库部分进行程序的编写。

Go 语言在并发编程方面的应用主要运用的是 goroutine。goroutine 是 Go 语言中处理并行任务的核心，其本质为比线程更小的协程。创建 goroutine 并不需要对原始代码进行复杂的调整，仅需在原始语句前增加“go”关键字，就可以创建并发执行单元。

报文解析入库部分的文件框架除主运行程序 main.go 外，其他各模块均存放于 tool 文件夹，方便主程序的调用。主程序负责设置 CPU 数量、开启日志服务、连接数据库及启用数据解析功能等方面工作。tool 文件夹中包含 dataParser、Logger、mysql、tcpServer 和 utils 5 个主要模块。dataParser 负责定义各数据的数据类型，并针对各类数据信息单元的不同结构对其解析规则进行制定；另外编写了对信息体数据解析流程，以及对查询指令的处理逻辑。Logger 对日志服务进行管理。mysql 负责数据库的连接及验证，利用 go 关键词建立 goroutine，实现实时数据的并行入库操作；同时将车辆 VIN 与实时数据绑定，运用 SQL 语句将数据按其类型归属插入到数据表单中。tcpServer 负责建立通讯连接，完成后开始接收上传数据。接着按照 TCP/IP 协议中各字段含义的映射规则，运用解析函数将报文中的字段转换为可读信息；同时负责对升级指令、控制指令等下行数据包的发送。utils 为数据计算工具，负责标准时间的换算、单双字节计算、数值异常判断及时间生成等数值计算方法的调用。后台主程序启动后各模块协同工作，整体程序按照设计的数据解析逻辑完成解析入库的处理，并在应对高并发场景中具备较好的负载均衡能力。

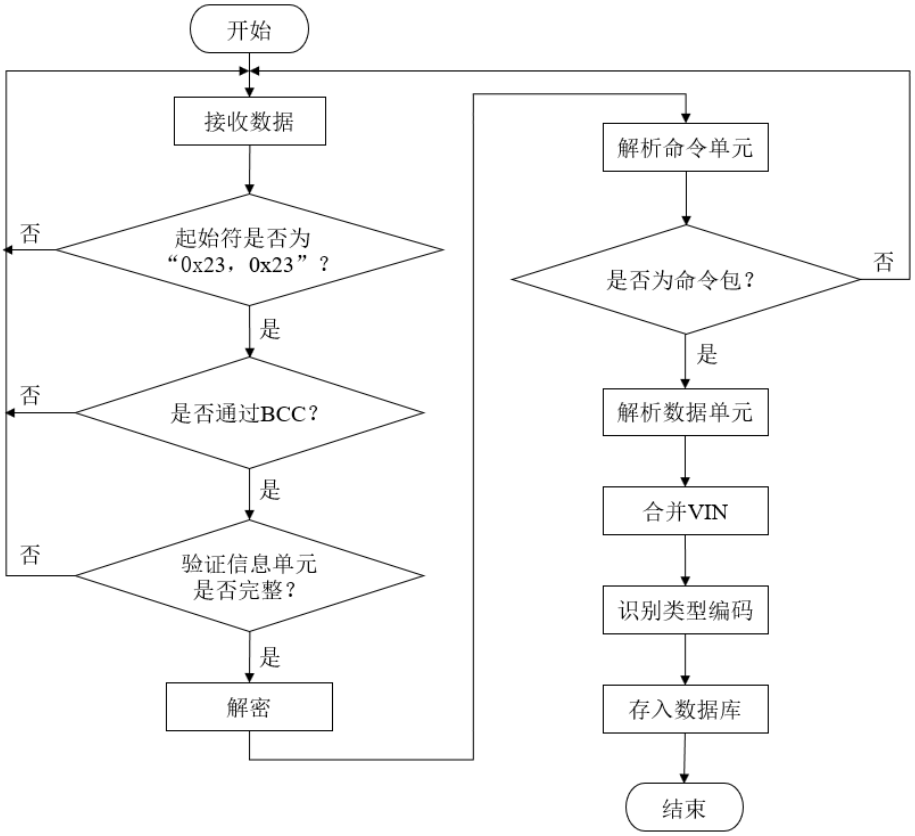

Figure 7. Data analysis and storage process
图 7. 数据解析入库流程

5. 平台软件系统联合测试

软件系统测试采用软硬件结合的方法，即将车载终端连接远程监控平台，针对车载终端软件部分及平台各功能模块进行黑盒测试。

为模拟从车辆实时数据的采集，到最终管理系统对数据库中数据调用的全部流程，同时考虑时间和成本，本文采用编写测试脚本的方法来完成功能性测试。测试脚本根据车辆运行场景分为两部分，其一为行驶场景下的数据收发，另一为驻车充电场景下的数据收发。整体系统测试的原理为：利用 CAN 总线分析仪作为 CAN 总线数据的收发设备，在 CANTest 软件中选择不同运行场景下测试脚本，向车载终端发送模拟数据，模拟车载终端采集车辆实时数据及向云平台发送数据的工作流程。

测试环境硬件部分包括由广成科技有限公司推出的型号为 USBCAN-II Pro 的 CAN 总线分析仪，本文设计开发的车载终端，可调电源以及 PC。软件部分包括负责转发 CAN 数据的 CANTest 应用软件和本文设计开发的电动汽车远程监控平台。测试环境硬件连接如图 8 所示。

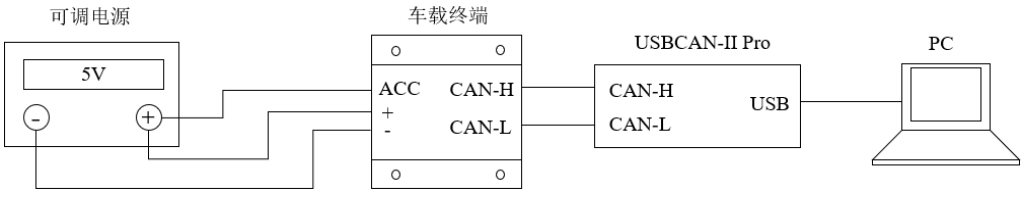

Figure 8. Test hardware wiring diagram
图 8. 测试硬件接线图

软件部分的测试流程如下：登录电动汽车远程监控平台，在管理系统中录入车辆身份信息及车载终端(车机)身份信息，并完成两者的绑定操作；启动车载终端，完成与远程监控平台的通讯连接；开启CANTest 软件并初始化参数，利用测试脚本文件向车载终端发送模拟数据；进入远程监控平台管理端各功能模块，验证数据显示是否正常以及数据调用是否正确。其具体操作流程可由图 9 所示。

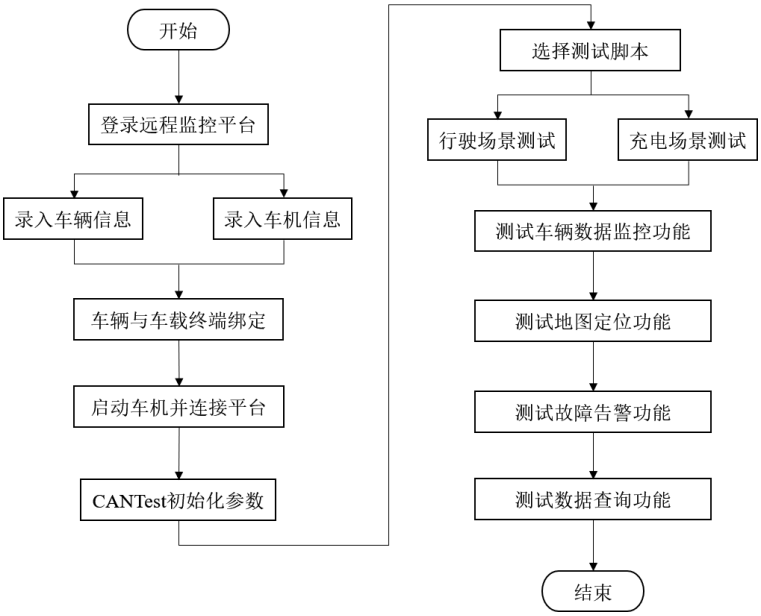

Figure 9. System testing process
图 9. 系统测试流程

登录界面如图 10 所示，车辆数据监控界面如图 11 所示。

通过上述方案对远程监控平台的软件测试，根据表现层各模块的功能反馈，整体平台运行良好。其测试结果可由表 6 所示。

Figure 10. Login interface
图 10. 登录界面

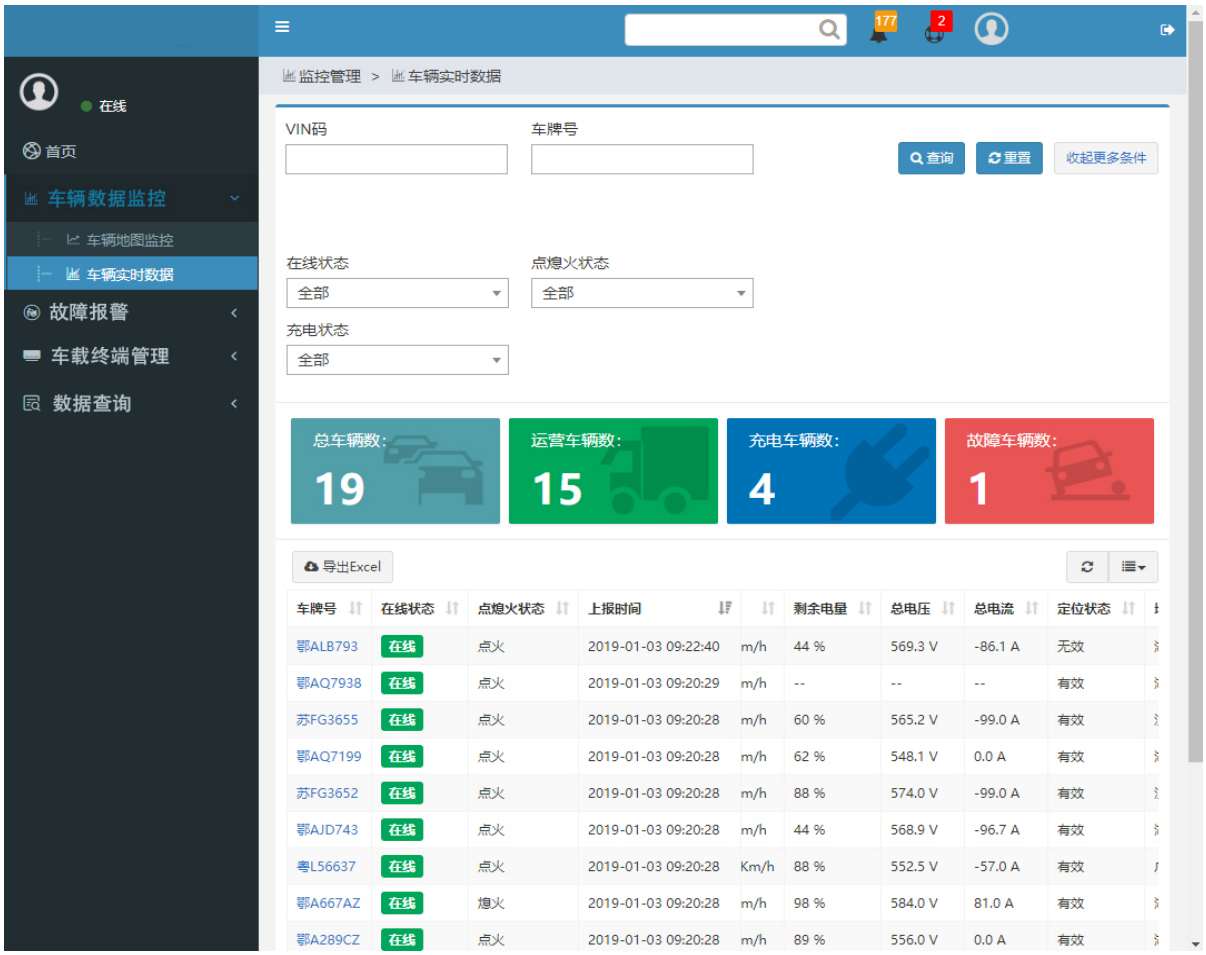

Figure 11. Vehicle data monitoring interface
图 11. 车辆数据监控界面

Table 6. Results of platform software test
表 6. 平台软件测试结果

测试项目	功能反馈	
	成功	失败
车载终端登录平台	√	
车辆信息录入	√	
车载终端信息录入	√	
车辆与车载终端绑定	√	
管理端登录验证	√	
实时监控数据显示与更新	√	
实时定位数据显示与更新	√	
故障报警上报(等级)	√	
故障信息上报(故障发生前后 30 s)	√	
历史数据数据查询	√	
历史数据列表批量导出	√	

最后，将本文开发的电动汽车远程监控平台与维特讯新能源汽车数据监测平台及上汽通用五菱汽车分时租赁系统进行功能性对比。维特讯新能源汽车数据监控平台专注于采集车辆运行数据，而五菱分时租赁系统则是典型的基于车辆数据的车联网远程服务系统，其对比结果如表 7 所示。可以看出，与两典型系统相比，本文设计的电动汽车远程监控平台既满足了作为数据平台应对高负载压力的需求，又在系统层面具有丰富的扩展性和较强的可移植性，可在很大程度上降低二次开发的工作量，凸显了本系统的优势。

Table 7. System functional comparison
表 7. 系统功能性对比

项目 \ 平台	电动汽车远程监控平台	维特讯数据监测平台	五菱分时租赁系统
应用场景	基于车辆监控数据的各类服务	新能源汽车运行数据采集与管理	汽车分时租赁
运行环境	网页端	Windows	Windows，Android
最大可接入终端数	取决于服务器负载极限，不设上限	取决于服务器负载极限，不设上限	1500 辆
可扩展性	通用 API 扩展接口	仅支持数据类型扩展	仅支持客户端功能扩展
可移植性	可移植于任何基于电动汽车监控数据的应用服务系统	不可移植	不可移植

6. 结束语

本文基于 4G 网络环境设计开发了一套用于电动汽车的远程监控平台，其具备较高的可移植性和拓展性，并针对多终端同时接入场景能有效实施负载均衡，软件系统经过测试与验证后能稳定运行。该平台能实现对电动汽车远程监控数据的跨平台管理，可运用于基于车辆监控数据扩展的应用与服务，不仅能提升远程监控管理水平，还能为后续的二次开发提供技术支撑，减少成本开销。

参考文献

[1] 杨立波. 车联网产业发展分析[J]. 中兴通讯技术, 2020, 22(3): 105-113.

[2] 张井玲, 刘婷婷, 陈琦. 车联网通信技术的发展现状及未来趋势分析[J]. 现代信息技术, 2019, 15(3): 185-187.

[3] 吴晶晶. 纯电动汽车车载信息的采集与远程监测系统的研发[D]: [硕士学位论文]. 南昌: 南昌大学, 2011.

[4] 顾文琰. 5G 时代车联网的发展和机遇挑战[J]. 科技创新, 2019(3): 1-3.

[5] 吴海波. 纯电动汽车运行状态参数的远程监测系统设计[J]. 长春大学学报, 2018, 28(4): 13-16.

[6] 林伟婷. C/S 与 B/S 架构技术比较分析[J]. 信息技术, 2018(13): 15-16.

[7] Dou, X., Yang, Y. and Chen, X. (2017) Design of Tourism Information System with B/S and C/S Architecture Based on Android and Web Platform. *Proceedings of Science*, **12**, 129-135.

[8] 任峥峥, 叶桦, 孙晓洁. 基于 M2M 平台的智能车载终端通信研究[J]. 东南大学学报, 2012, 42(I): 146-151.

[9] RUNOOB. RESTful 架构详解[Z/OL]. https://www.runoob.com/w3cnote/restful-architecture.html

[10] 胡宇鸿, 高鸿峰, 盛瑞琨. IPv6 车联网数据云服务基础架构的研究[J]. 信息与电脑, 2018(10): 54-57.

[11] 中华人民共和国国家质量监督检验检疫总局, 中国国家标准化管理委员会. GB/T 32960-2016 电动汽车远程服务与管理系统技术规范[S]. 北京: 中国标准出版社, 2016.