目录

1.背景

1.1 数据可视化的兴起

随着大数据浪潮的到来，除了以 Hadoop 为首的大数据分析技术蓬勃发展外，以 Google Charts / Baidu Echarts 为代表的开源 JavaScript 框架也得到了飞速发展。优秀的可视化图形库将使用户对数据的理解更直观、更深刻，极大的提高数据分析的效率，提高商务决策的准确性。

1.2 Web 框架技术飞速发展

Web 前端这几年进入前所未有的高速发展期，传统的 HTML + CSS + ES5 开发已经逐渐消失，取而代之的是更加模块和高效化的前端框架，如 Google 推出的 Angular, Facebook 推出的 React。大型前端框架的出现简化了前端开发，降低了开发大型项目的难度。

Web 后端方面， Spring 以其独特的依赖注入解耦的方式征服了大多数开发者，使得以 Java Spring MVC + Spring + Mybatis 的三大框架组合（即 SSM 框架）基本占据 Java Web 服务器端开发主流，极大的简化了后端开发流程，提高了后端开发效率

所以，本项目中将以 React 框架和 Echarts 图形库为核心来开发前端，以 SpringMVC + Spring + Mybatis 三大框架开发后端。

1.3 分布式计算的兴起

随着计算技术的发展，分布式计算越来越普遍，逐渐发展成主流的计算模式而取代集中式的大型计算机：

现在分布式系统具有比集中式系统更好的性能价格比。你不要花几十万美元就能获得高效能计算。多数应用本身就是分布式的。如工业企业应用，管理部门和现场不在同一个地方。运用分布式计算更适合应用本身。分布式系统具有高可靠性。冗余不仅是生物进化的必要条件，而且也是信息技术。现代分布式系统具有高度容错机制，控制核反应堆主要采用分布式来实现高可靠性。分布式系统可扩展性。买一台性能更高的大型机，或者再买一台性能相同的大型机的费用都比添加几台 PC 的费用高得多。分布式系统具有高度灵活性。能够兼容不同硬件厂商的产品，兼容低配置机器和外设而获得高性能计算。

1.4 基于内存架构的 spark 的崛起

1.4.1 轻量级快速处理

大数据处理中速度往往被置于第一位，Spark 允许传统 hadoop 集群中的应用程序在内存中以 100 倍的速度运行，即使在磁盘上运行也能快 10 倍。Spark 通过减少磁盘 IO 来达到性能的提升，它们将中间处理数据全部放到了内存中。

Spark 使用了 RDD（Resilient Distributed Datasets）数据抽象，这允许它可以在内存中存储数据，只在需要时才持久化到磁盘。这种做法大大的减少了数据处理过程中磁盘的读写，大幅度的降低了运行时间。

1.4.2 易于使用

Spark 支持多语言。Spark 允许 Java、Scala、Python 及 R（Spark 1.4 版最新支持），这允许更多的开发者在自己熟悉的语言环境下进行工作，普及了 Spark 的应用范围，它自带 80 多个高等级操作符，允许在 shell 中进行交互式查询，它多种使用模式的特点让应用更灵活。

1.4.3 支持复杂查询

除了简单的 map 及 reduce 操作之外，Spark 还支持 filter、foreach、reduceByKey、aggregate 以及 SQL 查询、流式查询等复杂查询。Spark 更为强大之处是用户可以在同一个工作流中无缝的搭配这些功能，然后对数据进行实时 SQL 查询或使用 MLlib 库进行系统推荐，而且这些复杂业务的集成并不复杂，因为它们都基于 RDD 这一抽象数据集在不同业务过程中进行转换，转换代价小，体现了统一引擎解决不同类型工作场景的特点。

1.4.4 实时的流处理

对比 MapReduce 只能处理离线数据，Spark 还能支持实时流计算。Spark Streaming 主要用来对数据进行实时处理，当然在 YARN 之后 Hadoop 也可以借助其他的工具进行流式计算。

1.4.5 与已存 Hadoop 数据整合

Spark 不仅可以独立的运行（使用 standalone 模式），还可以运行在当下的 YARN 管理集群中。它还可以读取已有的任何 Hadoop 数据，这是个非常大的

优势，它可以运行在任何 Hadoop 数据源上，比如 Hbase、HDFS 等。如果合适的话，这个特性让用户可以轻易迁移已有 Hadoop 应用。

2. 目标

2.1 接受 Wi-Fi 探针上传的探测数据

2.2 缓存数据到分布式文件系统

2.3 缓存数据到消息队列

2.4 确保数据完整性

2.5 数据采集并发大于 1000 时确保系统正常运行

2.6 离线计算，读取存储与 hive 中的原始数据，进行分析计算。

2.6 实时计算，读取 Kafka 中的实时流数据，进行分析计算

2.7 机器学习分析运行机器学习算法，对数据进行分析处理。

2.8 以友好的界面实时展示采集得到的数据

2.9 提供用户查询的入口，支持用户按照年、月、日等时间范围进行查询历史数据

2.10 用户切换到不同页面时检查用户 session 状态，保证用户合法性，从而保护数据

2.11 监控所有探针状态

2.12 提供自动生成报表的功能，方面用户得到纸质版数据

2.13 所有页面状态刷新支持异步刷新

2.14 保证数据存储和数据传输的安全性

3. 所需技术栈简介：

3.1 大数据分析：

3.1.1 Hadoop

Hadoop 是一个由 Apache 基金会所开发的分布式系统基础架构。用户可以在不了解分布式底层细节的情况下，开发分布式程序。充分利用集群的威力进行高速运算和存储。

Hadoop 实现了一个分布式文件系统（Hadoop Distributed File System），简称 HDFS。HDFS 有高容错性的特点，并且设计用来部署在低廉的（low-cost）硬件上；而且它提供高吞吐量（high throughput）来访问应用程序的数据，适合那些有着超大数据集（large data set）的应用程序。HDFS 放宽了（relax）POSIX 的要求，可以以流的形式访问（streaming access）文件系统中的数据。

Hadoop 的框架最核心的设计就是：HDFS 和 MapReduce。HDFS 为海量的数据提供了存储，则 MapReduce 为海量的数据提供了计算。

3.1.2 Spark

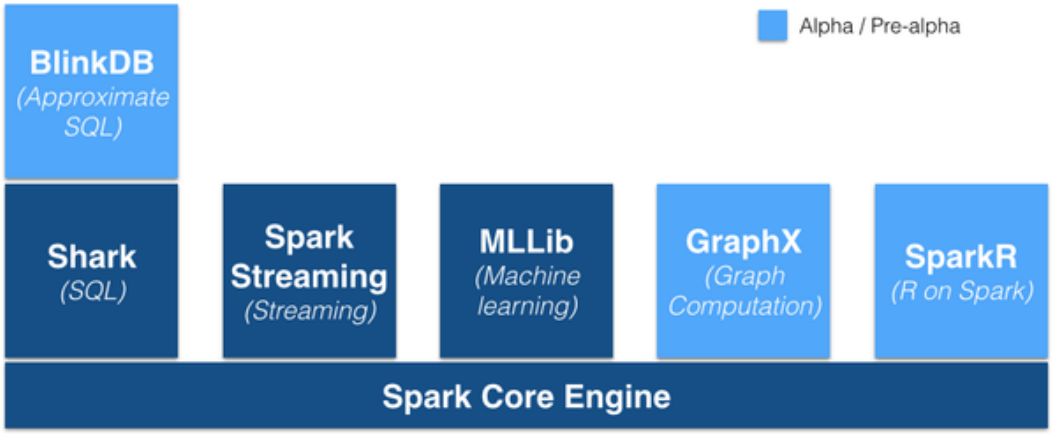

图 1 spark 架构图

Spark 是 UC Berkeley AMP lab (加州大学伯克利分校的 AMP 实验室)所开源的类 Hadoop MapReduce 的通用并行框架，Spark，拥有 Hadoop MapReduce 所具有的优点；但不同于 MapReduce 的是 Job 中间输出结果可以保存在内存中，从而不再需要读写 HDFS，因此 Spark 能更好地适用于数据挖掘与机器学习等需要迭代的 MapReduce 的算法。

Spark 是一种与 Hadoop 相似的开源集群计算环境，但是两者之间还存在一些不同之处，这些有用的不同之处使 Spark 在某些工作负载方面表现得更加优越，换句话说，Spark 启用了内存分布数据集，除了能够提供交互式查询外，它还可以优化迭代工作负载。

Spark 是在语言中实现的，它将 Scala 用作其应用程序框架。与 Hadoop 不同，Spark 和 Scala 能够紧密集成，其中的 Scala 可以像操作本地集合对象一样轻松地操作分布式数据集。

3.1.3 Hive

Hive 是基于 Hadoop 的一个数据仓库工具，可以将结构化的数据文件映射为一张数据库表，并提供简单的 sql 查询功能，可以将 sql 语句转换为 MapReduce 任务进行运行。 其优点是学习成本低，可以通过类 SQL 语句快速实现简单的 MapReduce 统计，不必开发专门的 MapReduce 应用，十分适合数据仓库的统计分析。

3.1.4 Kafka

Kafka 是一种高吞吐量的分布式发布订阅消息系统，它可以处理消费者规模的网站中的所有动作流数据。 这种动作（网页浏览，搜索和其他用户的行动）是在现代网络上的许多社会功能的一个关键因素。 这些数据通常是由于吞吐量的要求而通过处理日志和日志聚合来解决。 对于像 Hadoop 的一样的日志数据和离线分析系统，但又要求实时处理的限制，这是一个可行的解决方案。Kafka 的目的是通过 Hadoop 的并行加载机制来统一线上和离线的消息处理。有如下特性：

1）通过 O(1)的磁盘数据结构提供消息的持久化，这种结构对于即使数以 TB 的消息存储也能够保持长时间的稳定性能。

2）高吞吐量：即使是非常普通的硬件 kafka 也可以支持每秒数十万的消息。

3）支持通过 kafka 服务器和消费机集群来分区消息。

4）支持 Hadoop 并行数据加载。

在本项目中，通过使用 kafka 来为服务器集群提供实时消息，以分析 Wi-

Fi 探针数据。

3.1.5 Zokeeper

ZooKeeper 是一个分布式的，开放源码的分布式应用程序协调服务，是 Google 的 Chubby 一个开源的实现，是 Hadoop 和 Hbase 的重要组件。它是一个为分布式应用提供一致性服务的软件，提供的功能包括：配置维护、域名服务、分布式同步、组服务等。

ZooKeeper 的目标就是封装好复杂易出错的关键服务，将简单易用的接口和性能高效、功能稳定的系统提供给用户。

ZooKeeper 包含一个简单的原语集，提供 Java 和 C 的接口。

3.2 Web 前端：

3.2.1 React 框架

React 是 Facebook 推出的前端框架，其目标是 “构建数据会随时间变化的大型应用”，其主要包含两个特点：

1.简单

简单的表述任意时间点你的应用应该是什么样子的，React 将会自动的管理 UI 界面更新当数据发生变化的时候。

2 声明式

在数据发生变化的时候，React 从概念上讲与点击了 F5 一样，实际上它仅仅是更新了变化的一部分而已。React 是关于构造可重用组件的，实际上，使用 React 你做的仅仅是构建组建。通过封装，使得组件代码复用、测试以及关注点分离更加容易。

本项目中，将使用 React 搭建前端架构，并严格按照 React 规范进行开发。

3.2.2 Redux

Redux 是一种状态管理工具，适用于多交互和多数据源的情况。在本项目中，Redux 将负责前后端的通信工作。

3.2.3 React-Router

Router 在本项目中将用于管理前端路由。

3.2.4 NodeJS

前端服务器

3.2.5 NPM

NPM 是随同 NodeJS 一起安装的包管理工具，能解决 NodeJS 代码部署上的很多问题，常见的使用场景有以下几种：

允许用户从 NPM 服务器下载别人编写的第三方包到本地使用。

允许用户从 NPM 服务器下载并安装别人编写的命令行程序到本地使用。

允许用户将自己编写的包或命令行程序上传到 NPM 服务器供别人使用。

3.2.6 Webpack

Webpack 可以看做是**模块打包机**：它做的事情是，分析项目结构，找到 JavaScript 模块以及其它的一些浏览器不能直接运行的拓展语言（Scss, TypeScript 等），并将其打包为合适的格式以供浏览器使用。

3.2.7 Less

Less 是一门 CSS 预处理语言，它扩充了 CSS 语言，增加了诸如变量、混合（mixin）、函数等功能，让 CSS 更易维护、方便制作主题、扩充。

3.2.8 WebSocket

WebSocket 协议是基于 TCP 的一种新的网络协议。它实现了浏览器与服务器全双工(full-duplex)通信——允许服务器主动发送信息给客户端。本项目将使用 WebSocket 实现前后端实时通信。

3.2.9 Echarts

Echarts 是一个开源的、功能强大的数据可视化库，提供丰富的可视化图表，社区也非常完善。本项目汇中将使用它绘制图形和表格

3.2.10 Ant Design

Ant Design 是一种 UI 设计语言，同时也提供了一套完整的 UI 开发规范。本项目将使用它作为 UI 组件。

3.2.11 Fetch

Fetch 是一种新的与远程资源进行通信的技术，未来将全面替代 XMLHttpRequest。使用 Fetch 可以更好的与 React/Redux 兼容。

3.3 Web 后端

3.3.1 Spring

Spring 是一个开源框架，Spring 是于 2003 年兴起的一个轻量级的 Java 开发框架，由 Rod Johnson 在其著作 Expert One-On-One J2EE Development and Design 中阐述的部分理念和原型衍生而来。它是为了解决企业应用开发的复杂性而创建的。Spring 使用基本的 JavaBean 来完成以前只可能由 EJB 完成的事情。然而，Spring 的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何 Java 应用都可以从 Spring 中受益。 简单来说，Spring 是一个轻量级的控制反转（IoC）和面向切面（AOP）的容器框架。

3.3.2 Spring MVC

Spring MVC 属于 SpringFrameWork 的后续产品，已经融合在 Spring Web Flow 里面。Spring 框架提供了构建 Web 应用程序的全功能 MVC 模块。使用 Spring 可插入的 MVC 架构，从而在使用 Spring 进行 WEB 开发时，可以选择使用 Spring 的 SpringMVC 框架或集成其他 MVC 开发框架.它有如下优点：

3.3.2.1 清晰的角色划分

控制器(controller)、验证器(validator)、命令对象(command obect)、表单对象(form object)、模型对象(model object)、Servlet 分发器(DispatcherServlet)、处理器映射(handler mapping)、试图解析器(view resoler)等等。每一个角色都可以由一个专门的对象来实现。

3.3.2.2 可适配、非侵入

可以根据不同的应用场景，选择何事的控制器子类(simple 型、command 型、from 型、wizard 型、multi-action 型或者自定义)，而不是一个单一控制器(比如 Action/ActionForm)继承。

3.3.2.3 可定制的绑定(binding)和验证(validation)

比如将类型不匹配作为应用级的验证错误，这可以保证错误的值。再比如本地化的日期和数字绑定等等。在其他某些框架中，你只能使用字符串表单对象，需要手动解析它并转换到业务对象。

3.3.2.4 灵活的 model 转换

在 Springweb 框架中，使用基于 Map 的键/值对来达到轻易的与各种视图技术集成。

本项目中，将使用 SpringMVC 架构，并严格按照其规范进行开发。

3.2.3 Mybatis

MyBatis 本是 apache 的一个开源项目 iBatis, 2010 年这个项目由 apache software foundation 迁移到了 google code，并且改名为 MyBatis 。MyBatis 是一个基于 Java 的持久层框架。iBATIS 提供的持久层框架包括 SQL Maps 和 Data Access Objects（DAO）MyBatis 消除了几乎所有的 JDBC 代码和参数的手工设置以及结果集的检索。MyBatis 使用简单的 XML 或注解用于配置和原始映射，将接口和 Java 的 POJOs（Plain Old Java Objects，普通的 Java 对象）映射成数据库中的记录。

3.2.4 MySQL

MySQL 是一个关系型数据库管理系统，由瑞典 MySQL AB 公司开发，目前属于 Oracle 旗下产品。MySQL 是最流行的关系型数据库管理系统之一，在 WEB 应用方面，MySQL 是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件。

MySQL 是一种关系数据库管理系统，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

3.2.5 Maven

Maven 项目对象模型(POM)，可以通过一小段描述信息来管理项目的构建，报告和文档的软件项目管理工具。

Maven 除了以程序构建能力为特色之外，还提供高级项目管理工具。由于 Maven 的缺省构建规则有较高的可重用性，所以常常用两三行 Maven 构建脚本就可以构建简单的项目。由于 Maven 的面向项目的方法，许多 Apache Jakarta 项目发文时使用 Maven，而且公司项目采用 Maven 的比例在持续增长。

Maven 在本项目中将用于整个后端项目的配置和管理

3.4 系统结构图

3.4.1 系统总结构图

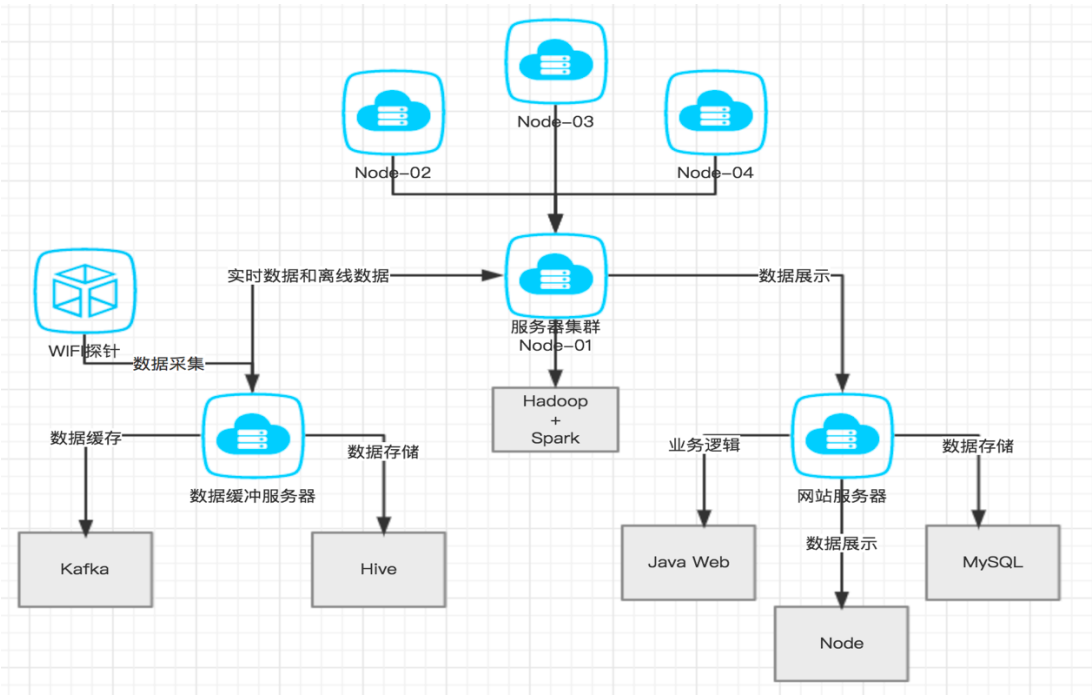

图 3.4.1 系统架构图

3.4.2 数据缓存服务器结构图

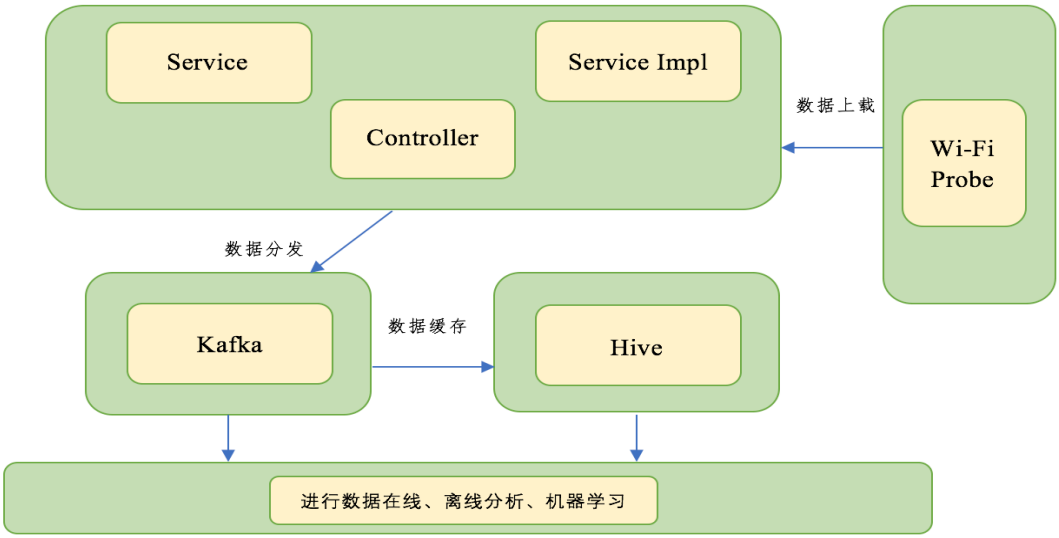

图 3.4.2 数据缓存服务器结构图

3.4.3 Web 端系统结构图：

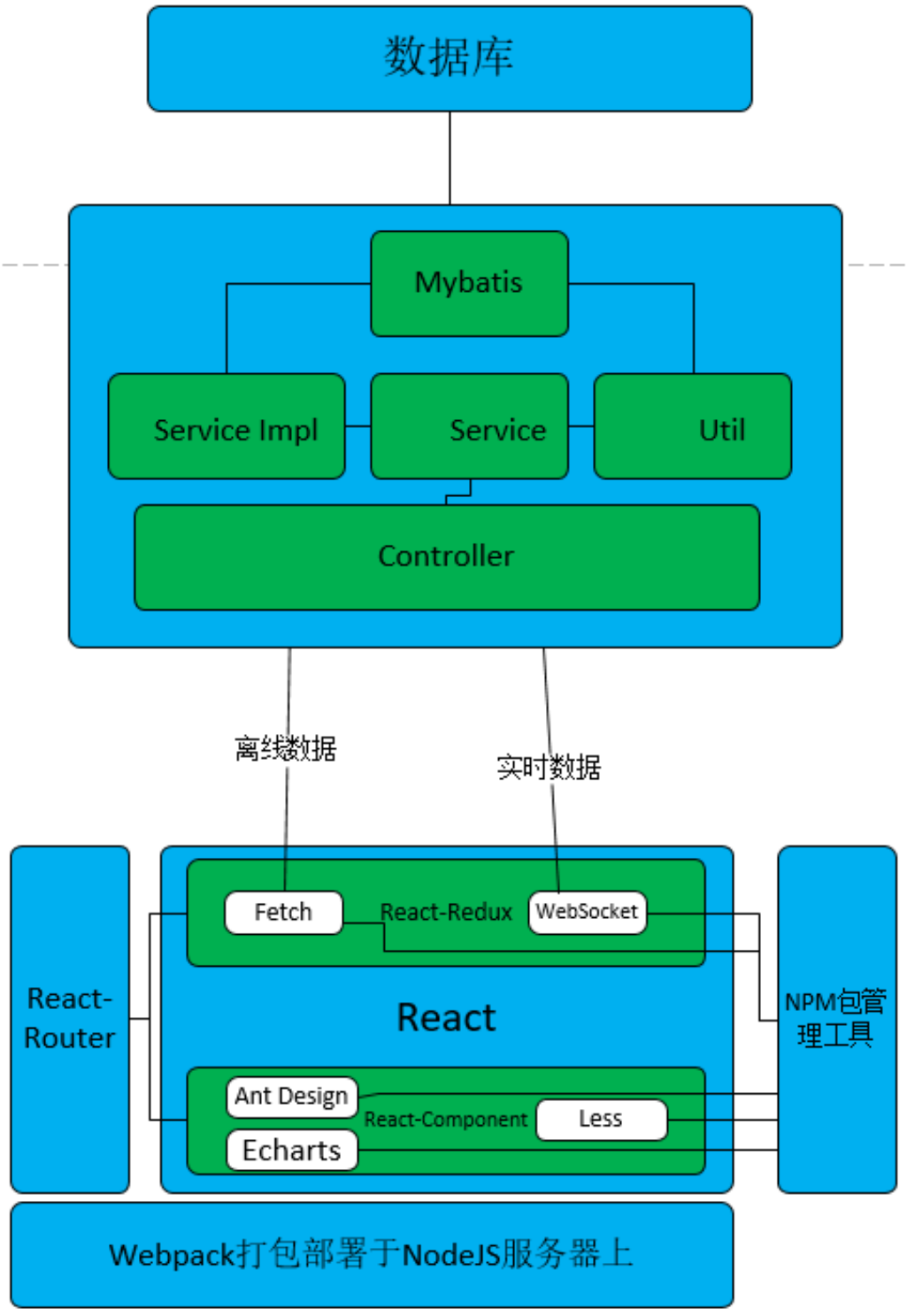

图 3.4.3 Web 端系统架构图

3.5 开发环境

3.5.1 Web 端开发环境

表 3.4.1 Web 端开发环境

前端	**编程语言**	**EmacsScript 6**
	样式语言	Less
	操作系统	MacOS X / CentOS 7
	开发工具	Jetbrain WebStorm
	运行环境	NodeJS
服务器端	编程语言	Java （JDK 18）
	操作系统	Mac OSX/CentOS 7
	开发工具	Jetbrain IDEA
	数据库	Mysql 5.6
	服务商	阿里云
文档\设计图	Microsoft Office 2016	
美术设计	Adobe Illustrator、Adobe Photoshop	
版本控制	Github/SourceTree	

3.5.2 数据缓存服务器开发环境

表 3.4.2 数据缓存数据器开发环境

服务器端	**编程语言**	**Java （JDK 18）**
	操作系统	Mac OSX/CentOS 7
	开发工具	Jetbrain IDEA
	服务商	阿里云
测试环境	服务器集群	
文档\设计图	Microsoft Office 2016	

版本控制	Github/SourceTree

3.5.3 数据分析服务器开发环境

表 3.4.3 数据分析服务器开发环境

服务器端	**编程语言**	**Java （JDK 18）/ Scala**
	操作系统	Mac OSX/CentOS 7
	开发工具	Jetbrain IDEA
	服务商	阿里云
测试环境	分布式集群	
文档\设计图	Microsoft Office 2016	
版本控制	Github/SourceTree	

4. 算法与系统设计实现细节描述

本节开始详细描述算法与系统设计实现细节

4.1 数据缓存模块

该部分主要用来接收探针上传数据，预处理数据，并将数据推入分布式消息订阅系统 Kafka，以及缓存如数据仓库 Hive 中。

4.1.1 模块编号与中文注释

W-01 /*接收并处理上传数据*/

4.1.2 功能描述与性能描述

4.1.2.1 功能描述

表 4.1.2.1. 功能描述

编号	名称	描述	优先级
001	建立探针与服务器的通信连接并获取数据	探针通过给定接口将数据上载到服务器，服务器需要读取出上载字段中的实际可用的 json 数据	高
002	数据预处理	对上载的数据重新编码并且数据分析系统要求数据格式的一致性，需要对缺失的字段进行补全	高
003	数据分发	预处理后的数据需要分发到分布式消息订阅系统 Kafka 中	高
004	数据离线缓存	预处理后的数据同时也需要离线缓存下来，为以后的机器学习，离线分析提供样本数据支撑	高

4.1.2.2 性能描述

(1) 满足多线程访问

(2) 使用 Kafka 分布式消息订阅系统解决高吞吐、高并发问题

(3) 数据离线存储

4.1.3 接口定义

4.1.3.1 后端接口

表 4.1.2.1.后端接口

名称	路径	注解
数据上载路径	/upload	@RequestMapping @ResponseBody

代码定义如下:

```
    @RequestMapping(value = "/upload", method = RequestMethod.POST,produces
="application/json;charset=utf-8")
```

探针将数据以 post 的方法上传如数据缓存服务器，服务器接收到数据后，进行 JSON 预处理，包括 JSON 重新编码，缺失字段补全等。

4.1.3.2 数据格式

接收到的 json 格式如下：

表 4.1.3.2 后端数据

名称	字段
探针唯一编号	id
探针 wifi 的 mac 地址	mmac
上传频率	rate
探针连接的 wifi 的 ssid	wssid
探针连接的 wifi 的 mac	wmac
时间戳	time
纬度（未采集到默认无此字段）	lat
经度（未采集到默认无此字段）	lon
探针地址（未采集到默认无此字段）	addr
采集到的设备的 mac 地址	mac
采集到的设备的信号强度	rssi
采集到的设备与探针的距离	range
采集到的设备所连接的 wifi 的 ssid（未采集到默认无此字段）	ts

采集到的设备所连接的 wifi 的 mac 地址（未采集到默认无此字段）	tmc
采集到的设备是否连接路由器（未采集到默认无此字段）	tc
采集到的设备是否随眠（未采集到默认无此字段）	ds
路由器的 ssid	router

4.1.4 算法流程图

4.1.4.1 整体流程图

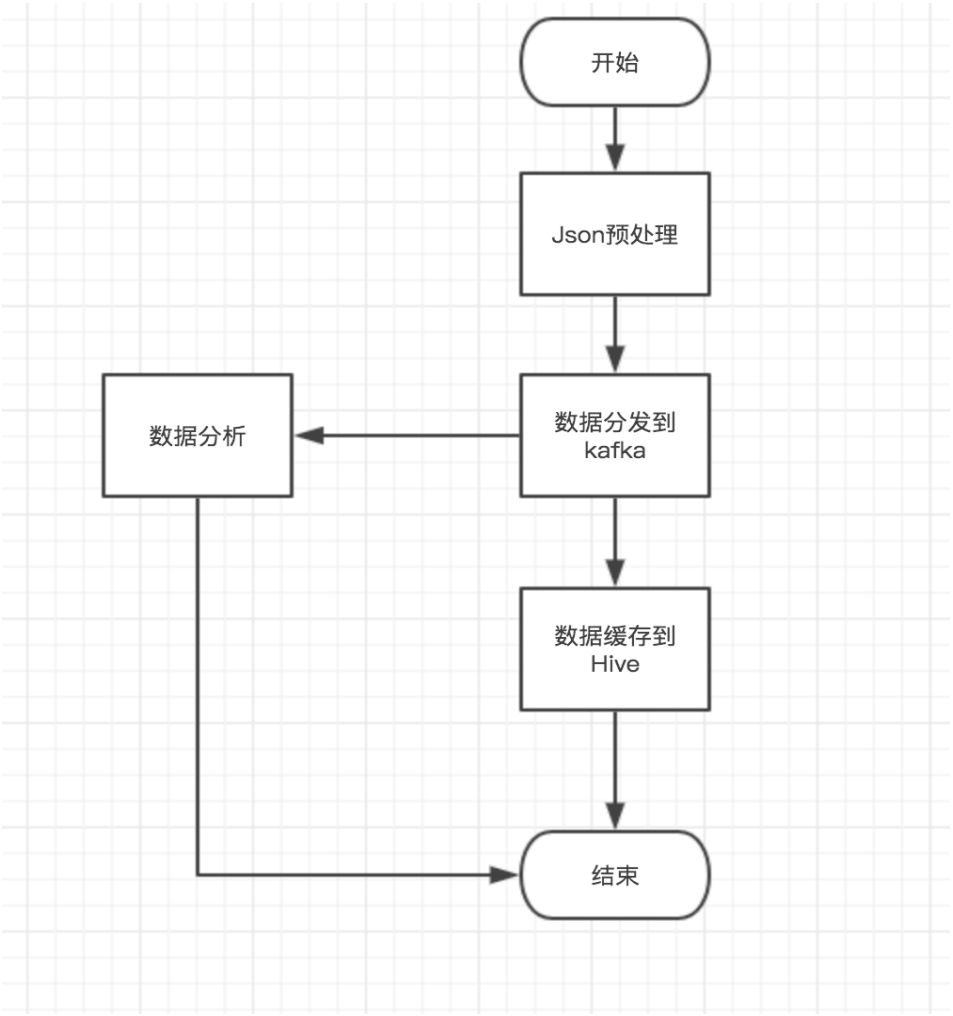

图 4.1.4.1.整体流程图

4.1.4.2 Json 预处理流程图

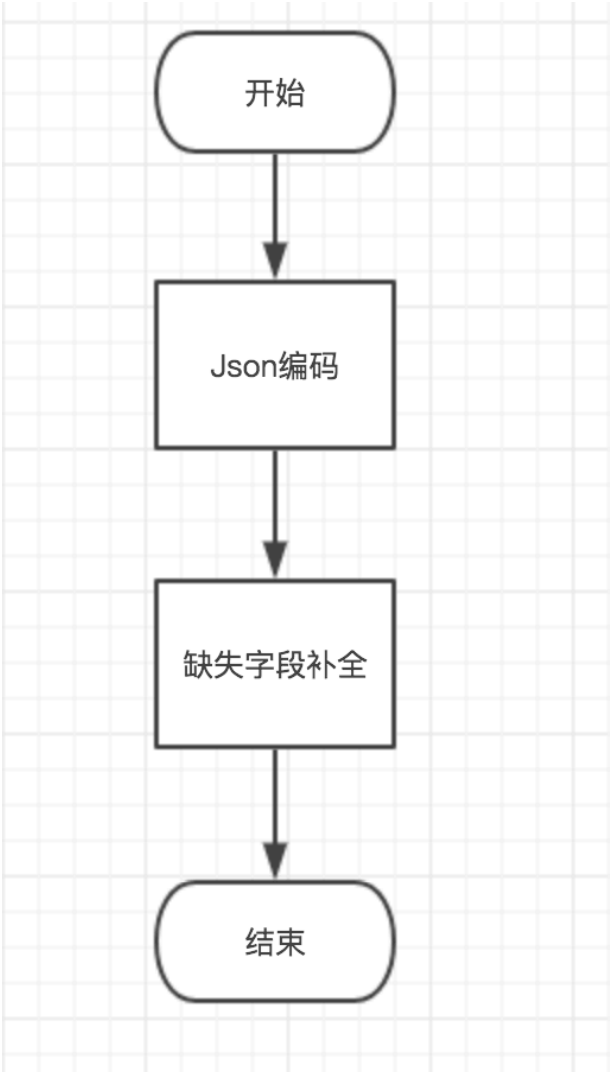

图 4.1.4.2 Json 预处理

4.1.4.3 数据存入 Hive 流程图

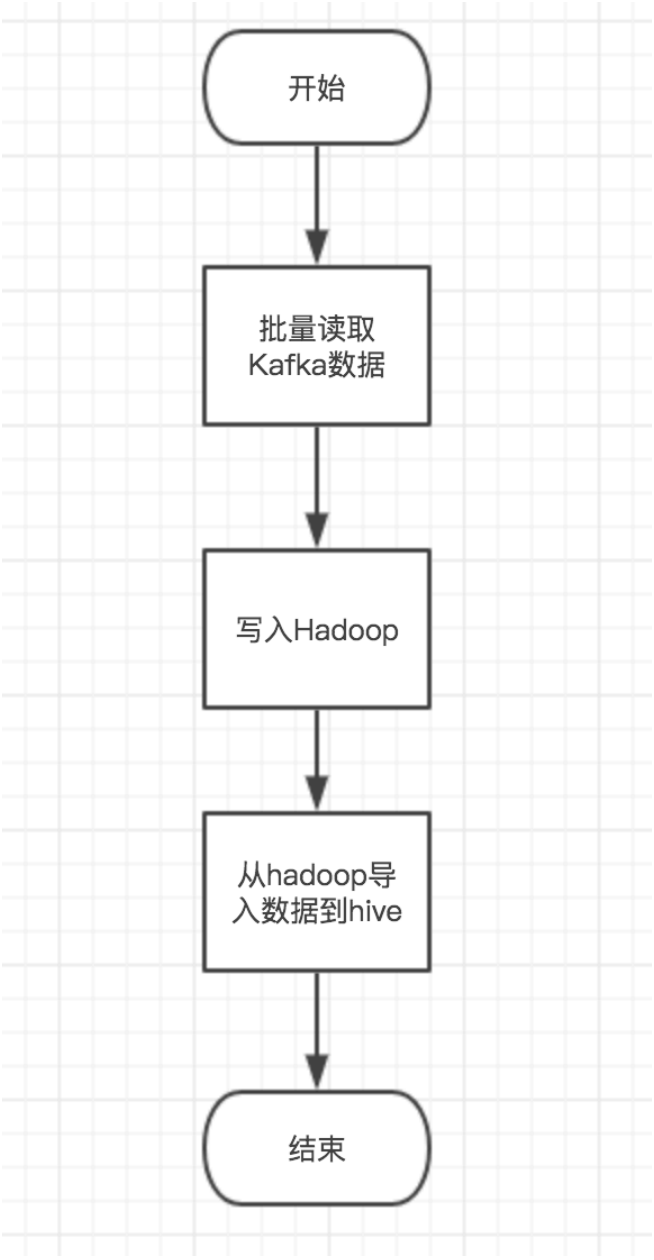

图 4.1.4.3 数据存入 hive

4.1.5 与本模块相关的代码表

后端代码：

表 4.1.5.1 后端代码

编号	文件名	作用
001	edu.cs.scu.controller. DataUploadController	探针数据上传接口
002	com.cs.scu.kafka.producer. KafkaProducers	Kafka 消息生产者，给分析服务器提供数据
003	com.cs.scu.kafka.producer.KafkaProducerForHive	Kafka 消息生产者，给 hive 提供数据
004	com.cs.scu.kafka.consumer.KafkaConsumerForHive	Kafka 消息消费者，从 kafka 消息队列中读出数据，并将数据存入 hive 中
005	com.cs.scu.hive.HiveService	Hive 连接配置
006	com.cs.scu.entity.Data	Json 处理实体类
007	com.cs.scu.entity.DataGroup	Json 处理实体类

4.1.6 应说明的问题与限制

探针上传数据非常快速，数据量也很大，注意性能优化和数据的完整性，不能丢失数据，数据须完整。

4.2 系统启始模块

4.2.1 模块编号与中文注释

M-01/*实现系统初始化配置*/

4.2.2 功能描述与性能描述

4.3.1.1 功能描述

表 4.3.1.1 系统启始模块功能描述表

编号	名称	描述	优先级
001	加载 spark 环境变量	在计算集群中，加载 SparkConf，用于下一步计算	高
002	加载 SQL 环境变量	在计算集群中，加载 SQLContext，方便下一步计算使用 SQL 语言处理数据	高
003	加载 Streaming 环境变量	在计算集群中，加载 StreamingContext，方便下一步计算使用 SQL 语言处理数据	高
004	初始化检查点及数据源	在计算集群中，初始化检查点目录以及数据源，方便下一步进行实时或离线计算	高
005	加载配置项	在计算集群中，加载配置项，方便下一步进行实时或离线计算进行阈值判断	高

4.3.1.2 性能描述

（1） 满足高并发
（2） 处理效率高

4.2.3 与本模块相关的代码表

表 4.2.3 系统初始化模块代码表

编号	文件名	作用
001	edu.cs.scu.scalautils.InitUtil	初始化环境工具

4.2.4 算法及处理流程

4.2.3.1 算法

直接逻辑处理，无算法

4.2.3.2 流程图

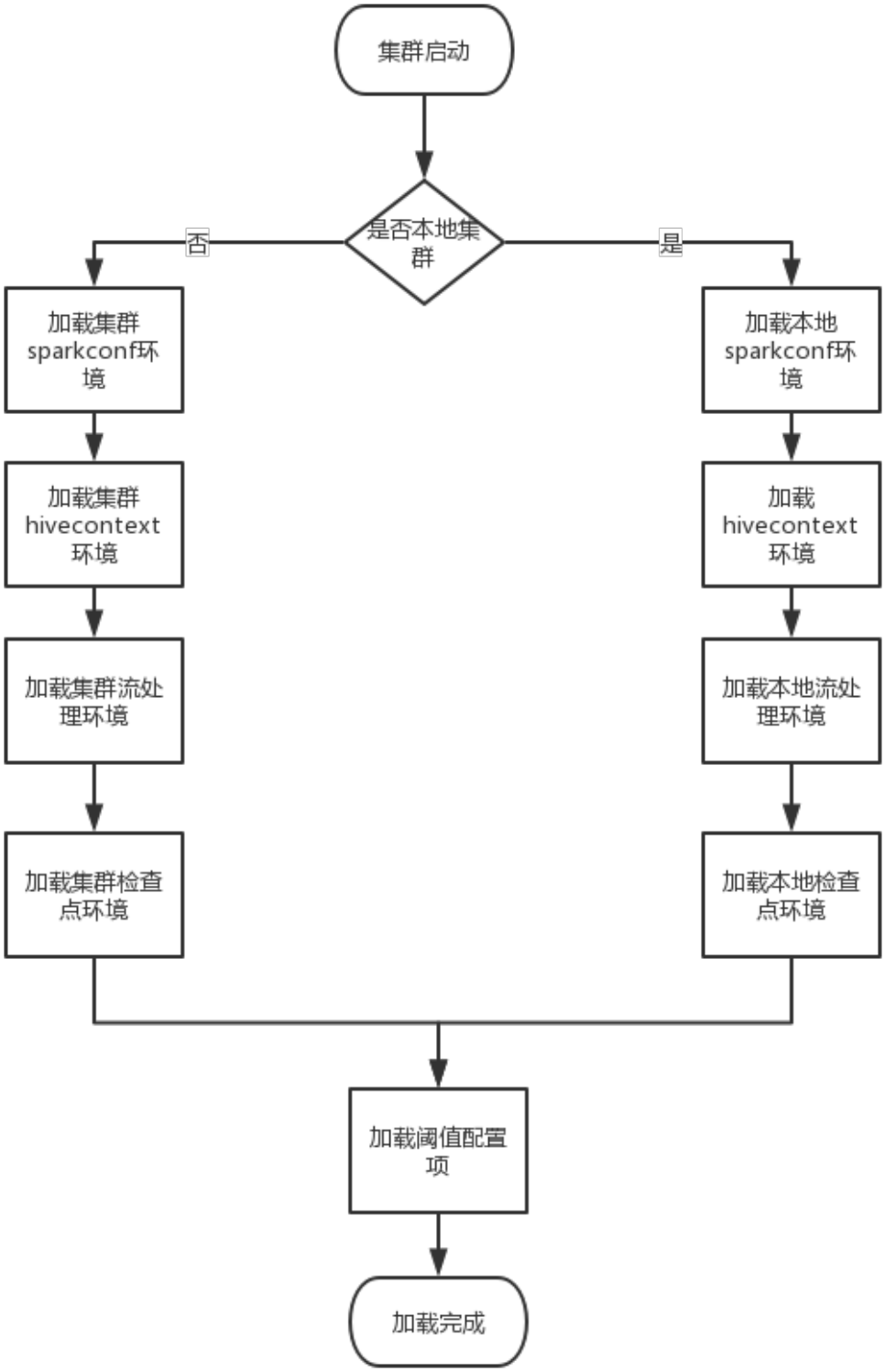

图 4.2.3.2 系统启始模块流程图

4.2.5 应说明的问题与限制

（1） 注意异常的捕获及处理
（2） 注意多线程的处理

4.3 读取配置项模块

4.3.1 模块编号与中文注释

M-02/*动态读取程序配置项*/

4.3.2 功能描述与性能描述

4.3.2.1 功能描述

表 4.3.2.1 读取配置项模块功能描述表

编号	名称	描述	优先级
001	获得配置属性	从配置文件中获取配置属性	高
002	得到 string 类型配置属性	将从配置文件中获取的配置属性转化为 string 类型	高
003	得到 int 类型配置属性	将从配置文件中获取的配置属性转化为 int 类型	高
004	得到 long 类型配置属性	将从配置文件中获取的配置属性转化为 long 类型	高
005	得到 boolean 类型配置属性	将从配置文件中获取的配置属性转化为 boolean 类型	高

4.3.2.2 性能描述

（1） 满足多线程访问
（2） 处理效率高

4.3 与本模块相关的代码表

表 4.3 读取配置模块代码表

编号	文件名	作用
001	edu.cs.scu.conf.MybatisSqlSession	获取 mybatis-config.xml 中相关配置项属性值
002	edu.cs.scu.conf.ConfigurationManager	获取 my.properties 中配置项属性值
003	edu.cs.scu.constants.DateConstants	与日期有关的常量
004	edu.cs.scu.constants.SparkConstants	与集群有关的常量
005	edu.cs.scu.constants.TableConstants	与表有关的常量
006	edu.cs.scu.constants.TimeConstants	与时间有关的常量
007	my.properties	配置项
008	db.properties	数据库配置项
009	mybatis-configuration.xml	Mybatis 配置项

4.3.4 算法及处理流程

4.3.4.1 算法

直接逻辑处理，无算法

4.3.4.2 流程图

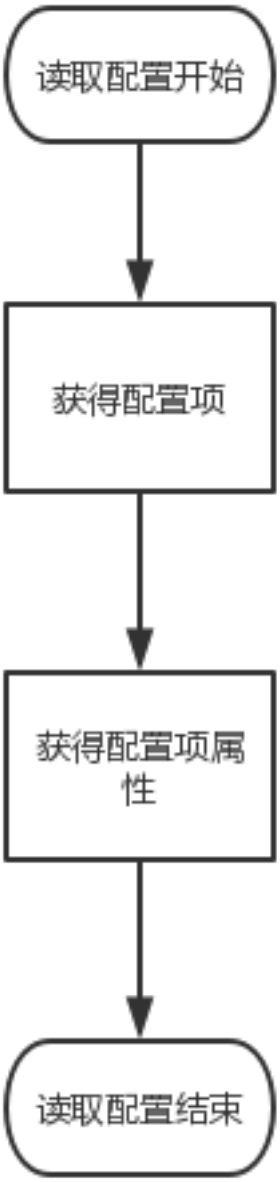

图 4.3.4.2 读取配置项模块流程图

4.3.5 应说明的问题与限制

注意多线程问题的处理

4.4实时处理模块

4.4.1 模块编号与中文注释

M-03/*实现数据实施处理*/

4.4.2 功能描述与性能描述

4.4.2.1 功能描述

表 4.4.2.1 实时处理模块功能描述表

编号	名称	描述	优先级
001	获取数据	从 Kafka 集群中读取实时数据	高
002	获得实时流量	在集群中分析实时数据，进行得到总流量	高
003	获得实时入店流量	在集群中分析实时数据，进行得到入店流量	高
004	获得实时入店率	在集群中分析实时数据，进行得到入店率	高
005	获得实时深访率	在集群中分析实时数据，结合阈值配置项,进行得到深访率	高
006	获得实时浅访率	在集群中分析实时数据，结合阈值配置项,进行得到浅访率	高
007	更新数据库	将分析的结果写入数据库，以便前端展示	高
008	运行日志	使用 log4j，对程序运行状态进行记录，	中

4.4.2.2 性能描述

表 4.4.2.2 实时处理模块性能表、

编号	名称	描述
001	处理效率	程序处理效率尽可能高，至少支持 1000 并发
002	分布式	程序运行在分布式系统上
003	异常处理	程序会对异常进行异常处理。

4.4.3 与本模块相关的代码表

表 4.4.3.1 实时处理模块代码表

编号	文件名	作用
001	edu.cs.scu.analyse.RealTimeAnalysis	获取 mybatis-config.xml 中相关配置项属性值
002	edu.cs.scu.scalautils.DataUtil	进行实施分析必要的数据工具，使用 Scala 编写
003	edu.cs.scu.javautils.DateUtil	进行实施分析必要的日期处理工具，使用 java 编写
004	edu.cs.scu.javautils.JsonUtil	进行实施分析必要的 json 处理工具，使用 java 编写
005	edu.cs.scu.javautils.MacAdressUtil	进行实施分析必要的 Mac 地址工具，使用 java 编写
006	edu.cs.scu.javautils.StringUtil	进行实施分析必要的字符串处理工具，使用 java 编写

4.4.4 算法及处理流程

4.4.4.1 算法

（1） 入店判断算法

在实时处理模块中需要判断一个用户是否入店，简单而言就是判断 Wi-Fi 探针捕获的用户手机数据中 range 字段与 rssi 字段。

程序中使用的是简单的判断算法，即：

若，$range < property.range \,\&\, rssi > property.rssi$

则，用户已入店

注意：$property$是用户阈值配置项

（2） 入店率计算

在实时处理模块中需要判断需计算入店率，即

$$入店率 = \frac{进店总人数}{总流量}$$

（3） 判断是否深度访问算法

在实时处理模块中需要判断一个用户入店后是否深度访问，简单来说就是判断用户的停留时常。

程序中使用简单的判断算法，即：

若，$visit_{time} > property.visit\_time$

则，用户已深度访问

注意：$property$是用户阈值配置项

（4） 深访率计算

在实时处理模块中需要判断需计算深访率，即

$$深访率 = \frac{深度访问人数}{进店总人数}$$

（5） 浅访率计算

在实时处理模块中需要判断需计算浅访率，即

$$浅访率 = 1 - \frac{深度访问人数}{进店总人数}$$

4.4.4.2 流程

在实时处理模块的流程图如下所示：

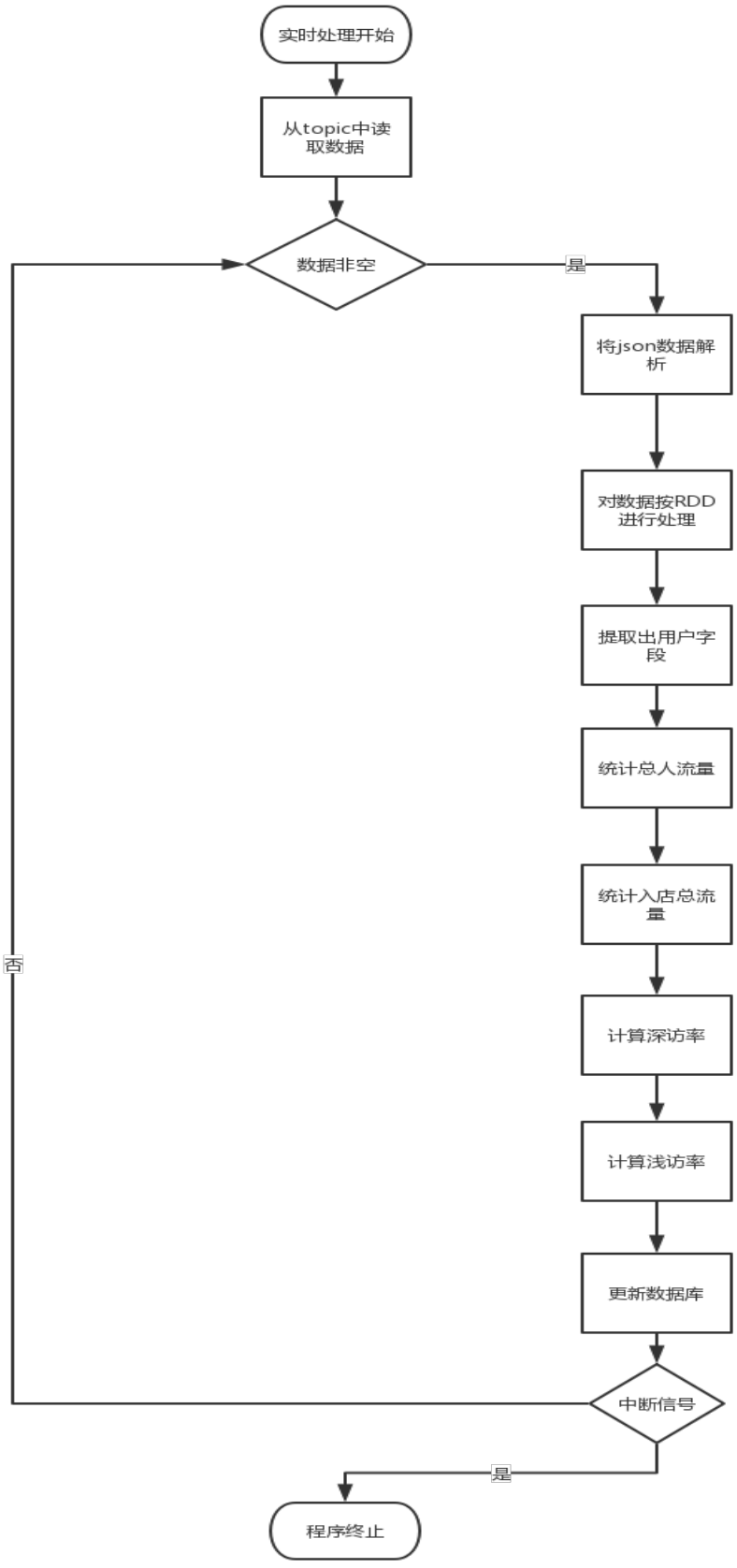

图 2.4.4.2 实时处理模块流程图

4.4.5 应说明的问题与限制

（1） 注意异常的捕获及处理
（2） 注意多线程的处理

4.5 离线处理模块

4.5.1 模块编号与中文注释

M-04/*离线处理模块*/

4.5.2 功能描述与性能描述

4.5.2.1 功能描述

表格 4.5.2.1 离线处理模块功能表

编号	名称	描述	优先级
001	获取来访周期	从 hive 中读取实时数据	高
002	获得环比数据	在 hive 中分析实时数据，进行得到总流量	高
003	运行日志	使用 log4j，对程序运行状态进行记录，	中

4.5.2.2 性能描述

（1） 满足多线程访问
（2） 处理效率高

4.5.3 与本模块相关的代码表

4.5.4 算法及处理流程

4.5.2.1 算法

直接逻辑处理，无算法

4.5.2.2 流程

图 4.5.2.2　离线分析模块流程图

4.5.5 应说明的问题与限制

（1） 注意异常的捕获及处理
（2） 注意多线程的处理

4.6 机器学习模块

4.6.1 模块编号与中文注释

M-05/*使用机器学习算法对数据行进处理*/

4.6.2 功能描述与性能描述

4.6.2.1 功能描述

表 4.6.2.1 机器学习模块功能描述表

编号	名称	描述	优先级
001	预测峰值流量	从 hive 中读取数据，进行分析	低
002	对顾客手机品牌进行类聚分析	在 hive 中读取数据，进行得到总流量	低
008	运行日志	使用 log4j，对程序运行状态进行记录，	中

4.6.2.2 性能描述

（1） 满足多线程访问
（2） 处理效率高

4.6.3 与本模块相关的代码表

4.6.4 算法及处理流程

4.6.4.1 算法

（1） 回归分析

回归分析是确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法。

$$Y = AX + B$$

其中Y,A,X,B均为矩阵

（2） 决策树

决策树(是在已知各种情况发生概率的基础上，通过构成决策树来求取净现值的期望值大于等于零的概率，评价项目风险，判断其可行性的决策分析方法，是直观运用概率分析的一种图解法。

其中的代表 C4.5 算法,它产生的分类规则易于理解，准确率较高。其缺点是：在构造树的过程中，需要对数据集进行多次的顺序扫描和排序，因而导致算法的低效。此外，C4.5 只适合于能够驻留于内存的数据集，当训练集大得无法在内存容纳时程序无法运行。

具体算法步骤如下；

（1） 创建节点 N

（2） 如果训练集为空，在返回节点 N 标记为 Failure

（3） 如果训练集中的所有记录都属于同一个类别,则以该类别标记节点 N

（4） 如果候选属性为空，则返回 N 作为叶节点，标记为训练集中最普通的类；

（5） for each 候选属性 attribute_list

（6） if 候选属性是连续的 then

（7） 对该属性进行离散化

（8） 选择候选属性 attribute_list 中具有最高信息增益率的属性 D

（9） 标记节点 N 为属性 D

（10） for each 属性 D 的一致值 d

（11） 由节点 N 长出一个条件为 D=d 的分支

（12） 设 s 是训练集中 D=d 的训练样本的集合

（13） if s 为空

（14） 加上一个树叶，标记为训练集中最普通的类

（15） else 加上一个有 C4.5（R - {D},C，s）返回的点

4.6.4.2 流程

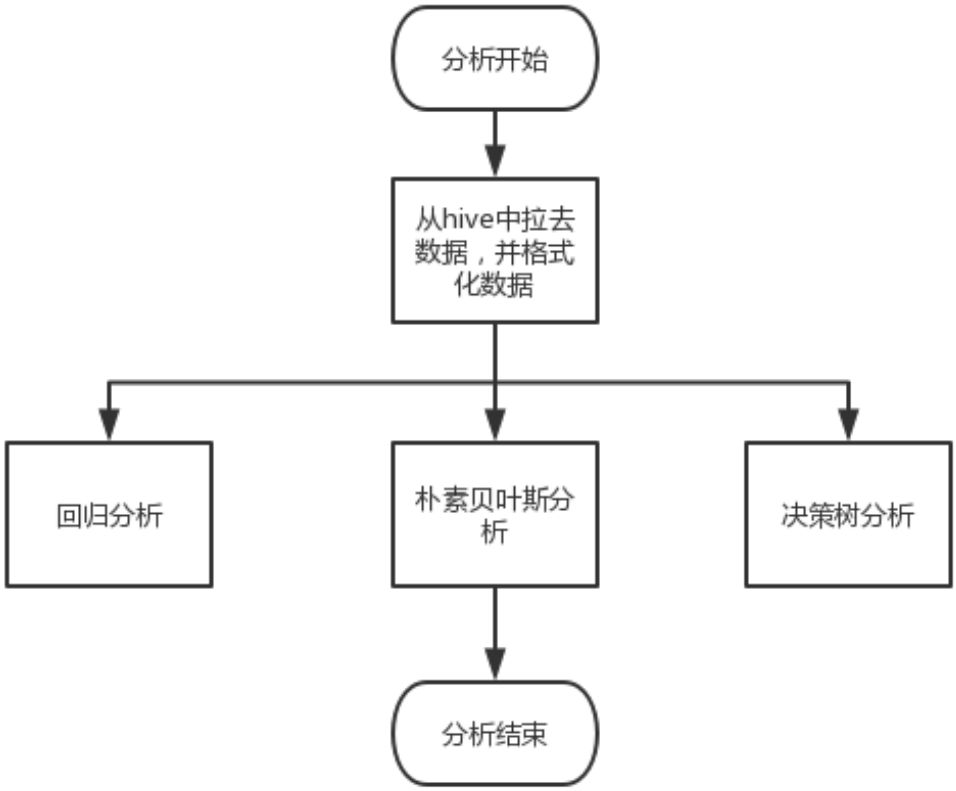

图 4.6.4.2 机器学习模块流程图

4.6.5 应说明的问题与限制

（1） 注意异常的捕获及处理
（2） 注意多线程的处理

4.7 数据库交互模块

4.7.1 模块编号与中文注释

M-06/*与数据库进行交互*/

4.7.2 功能描述与性能描述

4.7.2.1 功能描述

表 4.7.2.1 数据库交互模块功能描述

编号	名称	描述	优先级
001	支持批量插入	将数据批量插入进数据库	高
002	支持线程池	使用线程池	高
003	支持错误回滚	在数据插入错误时候，支持回滚操作	高
004	按需要查询数据库	能够按照需求，查询数据库中相关数据	高

4.7.2.2 性能描述

（1） 错误回滚
（2） 处理效率高
（3） 支持线程池

4.7.3 与本模块相关的代码表

表 4.7.3 数据库交互模块代码表

编号	文件名	作用
001	edu.cs.scu.dao.PropertyDao	阈值配置表接口类
002	edu.cs.scu.dao.ShopDao	商场表接口类
003	edu.cs.scu.dao.TaskDao	任务表接口类
004	edu.cs.scu.dao.UserDao	用户表接口类
005	edu.cs.scu.dao.UserVisitDao	用户访问表接口类
006	edu.cs.scu.dao.UserVisitTimeDao	用户访问时间表接口类
007	edu.cs.scu.dao.VendorMacDao	Mac 地址表接口类
008	edu.cs.scu.dao.impl.PropertyDaoImpl	阈值配置表实现类
009	edu.cs.scu.dao.impl.ShopDaoImpl	商场表实现类
010	edu.cs.scu.dao.impl.TaskDaoImpl	任务表实现类
011	edu.cs.scu.dao.impl.UserDaoImpl	用户表实现类
012	edu.cs.scu.dao.impl.UserVisitDaoImpl	用户访问表实现类

013	edu.cs.scu.dao.impl.UserVisitTimeDaoImpl	用户访问时间表实现类
014	edu.cs.scu.dao.impl.VendorMacDaoImpl	Mac 地址表实现类
015	edu.cs.scu.dao.factory.DaoFactory	数据库访问工厂类
016	edu/cs/scu/mapper/PropertyDao.xml	Mybatis 映射文件
017	edu/cs/scu/mapper/ShopDao.xml	Mybatis 映射文件
018	edu/cs/scu/mapper/TaskDao.xml	Mybatis 映射文件
019	edu/cs/scu/mapper/UserDao.xml	Mybatis 映射文件
020	edu/cs/scu/mapper/UserVisitDao.xml	Mybatis 映射文件
021	edu/cs/scu/mapper/UserVisitTimeDao.xml	Mybatis 映射文件
022	edu/cs/scu/mapper/VendorMacDao.xml	Mybatis 映射文件

4.7.4 算法及处理流程

4.7.4.1 算法

直接逻辑处理，无算法

4.7.4.2 流程图

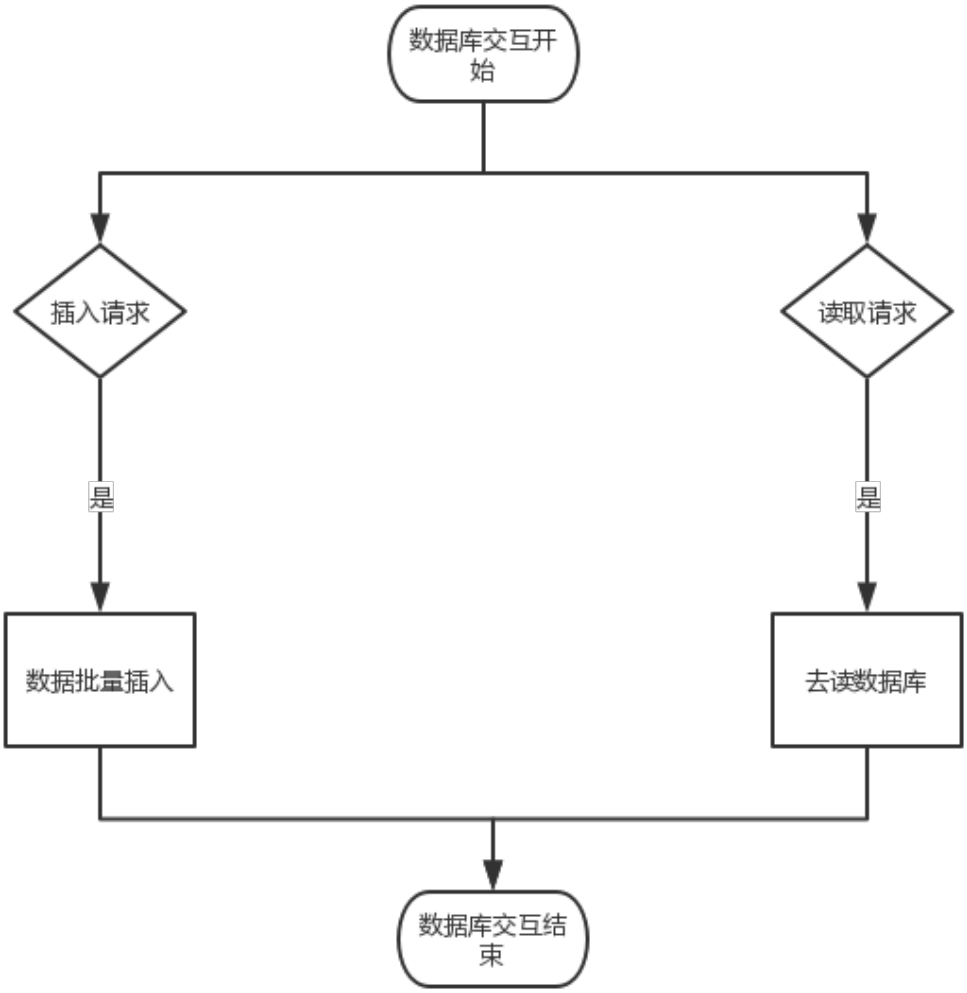

图 4.7.4.2 数据库交互模块流程图

4.7.5 应说明的问题与限制

（1） 注意异常的捕获及处理
（2） 注意多线程的处理

4.8 数据展示模块

该部分主要用来实时和非实时展示数据

4.8.1 模块编号与中文注释

W-01 /*实时展示流量信息*/

4.8.2 功能描述与性能描述

4.8.2.1 功能描述

表 4.8.2.1 数据展示模块功能列表

编号	名称	描述	优先级
001	建立 WebSocket 连接	检查浏览器是否支持 WebSocket，如果不支持就弹出警告，支持就与后端建立 WebSocket 连接	高
002	获取数据	后端定时从数据库里查询，获取到更新的数据后就推送给前端进行展示	高
003	初始化图表绘制	前端调用 Echarts 库初始化图表配置项	高
004	动态刷新数据	获取到数据之后就刷新图表	高

005	关闭 WebSocket 连接	当用户离开实时展示页面时 关闭 WebSocket 连接	高

4.8.2.2 性能描述

(1) 满足多线程访问

(2) 前端使用 immutableJS 优化性能

4.8.3 接口定义

4.8.3.1 后端接口

表 4.8.3.2 数据展示模块后端接口

名称	路径	注解
WebSocket 路径	/websocket	@ServerEndpoint

代码定义如下:

```
@ServerEndpoint(value="/websocket", encoders = {UserFlowEncoder.class})
```

其中 UserFlowEncoder.class 为自定义编码器，用于将 Java 对象编码为 Json 格式数据，方便传输。

4.8.3.2 前端接口

前端按照数据格式接收标准 Json 文件并进行解码。

4.8.3.3 数据格式

传输的 Json 格式数据如下：

表 4.8.3.3 数据展示模块数据格式

名称	字段
时间	Time
区域客流量	totalFlow
入店客流量	checkInFlow
入店率	checkInRatio
浅访率	shallowVisitRatio
深访率	deepVisitRatio
老顾客数量	oldVisitNumber
平均驻店时长	avgStayTime
平均来访周期	avgVisitCycle

4.8.4 算法流程图

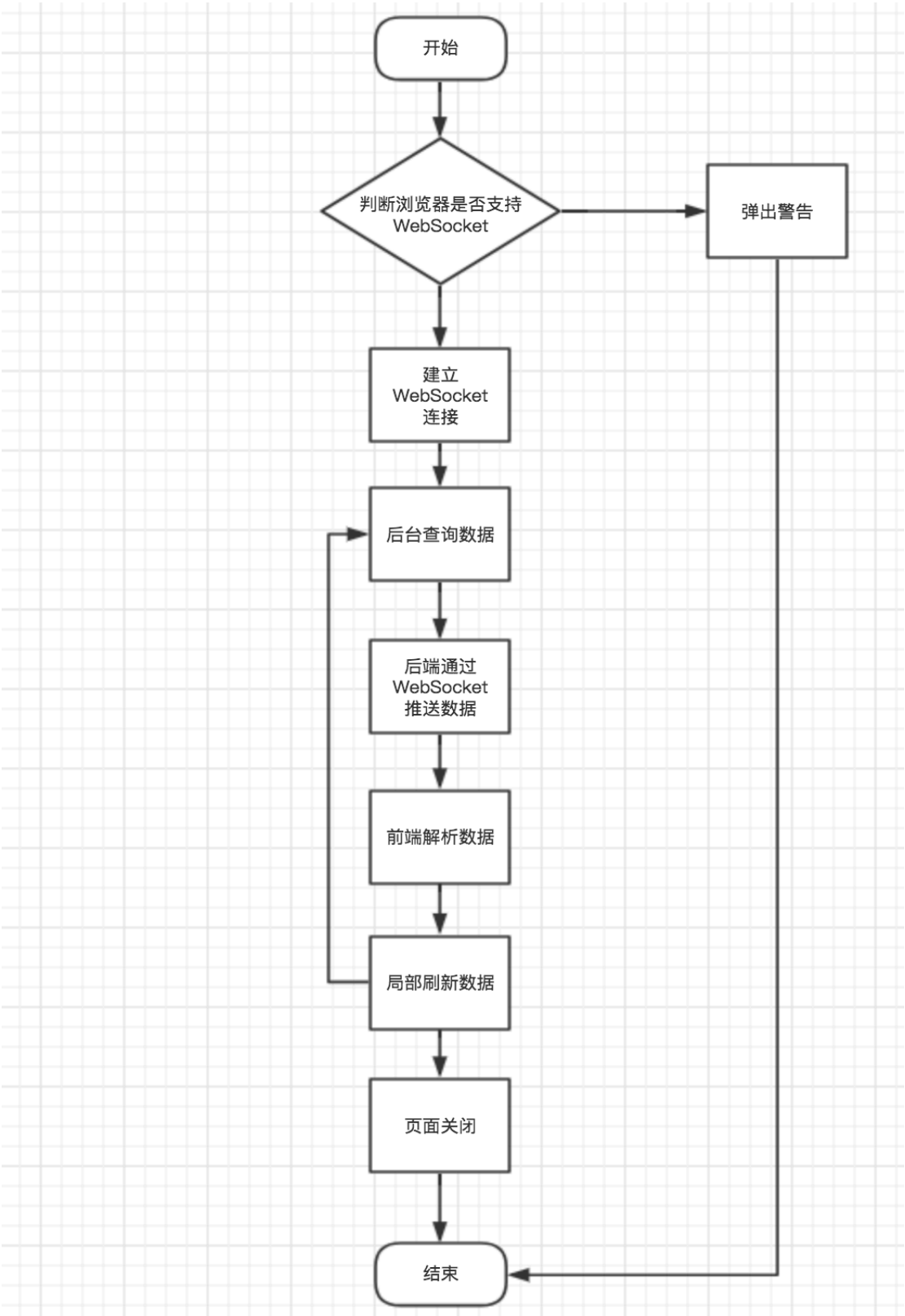

图 4.8.4 数据展示模块算法流程图

4.8.5 与本模块相关的代码表

后端代码：

表 4.7.5.1 数据展示模块代码表-1

编号	文件名	作用
001	edu.cs.scu.DBHelper.DBHelper	配置数据库属性，建立数据库连接
002	edu.cs.scu.DBHelper.DataDBManager	执行查询
003	edu.cs.scu.webSocket.handler.UserFlowEncoder	编码器，将 Java 对象编码为 Json 格式对象
004	edu.cs.scu.webSocket.hander.WebSocketEndPoint	WebSocket 服务器端口，负责建立 WebSocket 连接
005	edu.cs.scu.webSocket.Monitor	监控器，负责定时执行任务
006	edu.cs.scu.entity.UserVisitBean	负责管理游览信息的实体类

前端代码：

表 4.8.5.2 数据展示模块代码表-2

编号	文件名	作用
001	View/home/home-index.jsx	主界面，并负责建立和关闭 WebSocket 连接
002	View/home/echarts-manage.jsx	管理 echarts 配置项

003	View/home/style/home.less	管理 home-index.jsx 的样式
004	Component/mixin/Config.jsx	提供配置信息
005	Component/mixin/RenderData	React 与 React-redux 连接的桥梁

4.8.6 应说明的问题与限制

该页面 State 更新非常快速，数据量更新也很大，注意性能优化

4.9 商场管理模块

4.9.1 模块编号与中心注释

W-03 /*商场管理模块* /

4.9.2 功能描述与性能描述

4.9.2.1 功能描述

表 4.9.2.1 商场管理模块功能列表

编号	名称	描述	优先级
001	生成商场列表	对每个商场生产列表，列表包含商场ID\商场名称\商场地址\联系人\联系电话\平均入店率\累计入店量等数据	高
002	列表支持排序	管理员可以根据各种数据对列表进行排序	高
003	列表支持索引	管理员可以对列表进行索引	高
004	商场评分	该评分由系统生成，依据机器学习归类算法	高
005	商场评价	管理员对该商场进行评价	低
006	添加商场	管理员添加商场信息	高
007	查看商场详情	点击查看商场详情，可以查看商场的探针等信息	高
008	查看商场数据	对每个商场历史数据生成图表，方便查看	高
009	属性设置	管理员根据每个商场的情况设定属性，包括入店范围、信号范围、深访的时间判断	高

4.9.2.2 性能描述

(1) 满足多线程访问

(2) 前端使用 immutableJS 优化性能

(3) 控制机器学习算法的训练时间

4.9.3 接口描述

4.9.3.1 添加商场

后端接口：

表 4.9.3.1.1 商场管理模块接口-1

名称	路径	注解
添加商场	/addShop.action	@RequestMapping

数据格式如下：

表 4.9.3.1.2 商场管理模块数据格式

名称	字段
用户名	UserName
商场地址	Shop_addr
商场名称	Shop_name
商场管理员	Shaop_manager
商场联系方式	Shop_telephone
商场描述	Shop_describe

返回数据：

表 4.9.3.1.2 商场管理模块后端返回数据格式

名称	内容	类型

添加商场成功	Success	String

4.9.3.2 查询商场列表

后端接口

表 4.9.3.2.1 商场管理模块数据接口

名称	路径	注解
添加商场	/queryShopInfo.action	@RequestMapping

数据格式

表 4.9.3.2.1 商场管理模块请求数据格式

名称	字段
用户名	UserName

返回数据格式如下：

表 4.9.3.2.3 商场管理模块返回数据格式

名称	字段
用户名	UserName
商场地址	Shop_addr
商场名称	Shop_name
商场管理员	Shaop_manager
商场联系方式	Shop_telephone
商场描述	Shop_describe
累计入店量	totalChecInFlow

平均入店率	AvgCheckInFlow
平均深访率	Avg_deepAccessRatio

4.9.3.3 修改商场信息

后端接口：

表 4.9.3.3.1 商场管理模块后端接口

名称	路径	注解
添加商场	/updateShop.action	@RequestMapping

数据格式如下：

表 4.9.3.3.2 商场管理模块数据格式

名称	字段
用户名	UserName
商场地址	Shop_addr
商场名称	Shop_name
商场管理员	Shaop_manager
商场联系方式	Shop_telephone
商场描述	Shop_describe

返回数据：

表 4.9.3.3.3 商场管理模块数据格式

名称	内容	类型

修改商场信息成功	Success	String

4.9.3.4 查看商场详情

后端接口

表 4.9.3.4.1 商场管理模块后端接口

名称	路径	注解
查看商场详情	/queryDetail.action	@RequestMapping

数据格式：

表 4.9.3.4.1 商场管理模块数据格式

名称	字段
用户名	UserName
商场 ID	Shop_id

返回数据格式

表 4.9.3.4.2 商场管理模块数据格式

名称	字段
用户名	UserName
商场地址	Shop_addr
商场名称	Shop_name
商场管理员	Shaop_manager
商场联系方式	Shop_telephone
商场描述	Shop_describe

累计入店量	totalChecInFlow
探针 ID	Probe_ID
平均入店率	AvgCheckInFlow
平均深访率	Avg_deepAccessRatio
探针状态	

4.9.3.5 属性设置

根据商场需要设置每个商场、每个探针的属性，包括判断入店低距离范围、判断为深度访问的时间长短。

后端接口

表 4.9.3.5.1 商场管理模块接口

名称	路径	注解
属性设置	/setProperty.action	@RequestMapping

数据格式

表 4.9.3.5.2 商场管理模块数据格式

名称	字段
用户名	UserName
商场 ID	Shop_id
探针 ID	Probe_id
入店范围	CheckInRange
深访时间	DeepAcessTime

信号强度	RSSIRange
新老周期	VisitCycle

返回数据格式

表 4.9.3.5.3　商场管理模块返回数据格式

名称	内容	类型
属性设置成功	Success	String

4.9.3.6 属性查询

用户查询每个商场、探针的属性设置

后端接口

表 4.9.3.6.1　商场管理模块接口

名称	路径	注解
属性设置	/queryProperty.action	@RequestMapping

数据格式：

表 4.9.3.6.2　商场管理模块接口

名称	字段
探针 ID	Probe_id

商场 ID	Shop_id

返回数据格式

表 4.9.3.6.3　商场管理模块接口

名称	字段
用户名	UserName
商场 ID	Shop_id
探针 ID	Probe_id
入店范围	CheckInRange
深访时间	DeepAcessTime
信号强度	RSSIRange
新老周期	VisitCycle

4.9.4 算法流程图

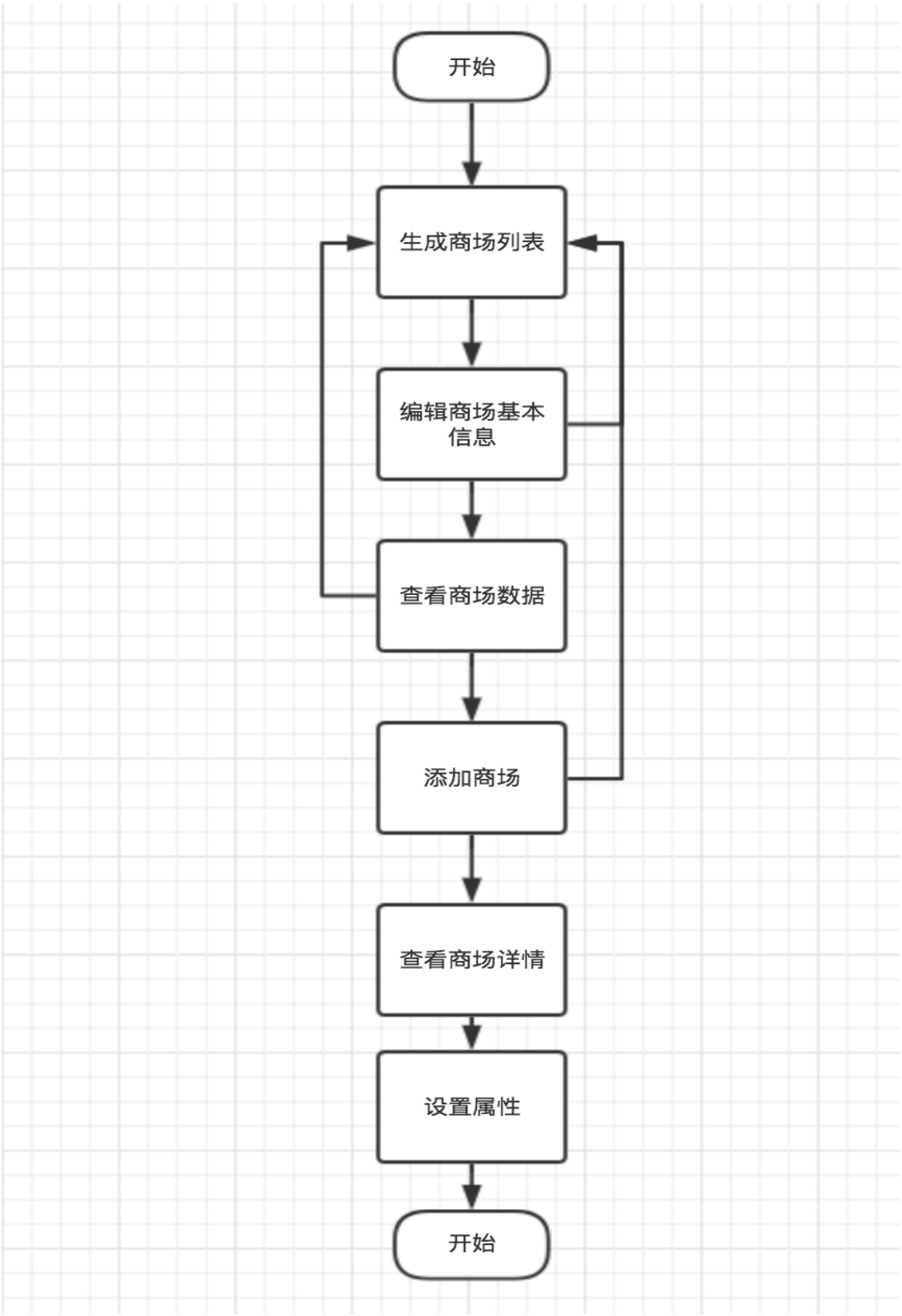

图 4.9.4 商场管理模块流程图

4.9.5 与本模块相关的代码

后端：

表 4.9.5.1 商场管理模块代码列表

编号	文件名	作用
001	edu.cs.scu.controller.ShopController	接收关于商场的请求参数
002	edu.cs.scu.controller.ProbeController	接收关于探针的请求参数
003	Edu.cs.scu.entity.ProbeInfo	管理探针信息的实体类
004	Edu.cs.scu.entityShopInfo	管理商场信息的实体类
005	Edu.cs.scu.service.ShopService	商场服务的接口
006	Edu.cs.scu.service.ShopServiceImpl	商场服务的实现类
007	Edu.cs.scu.propertyService	属性设置分服务接口
008	Edu.cs.scu.propertyServiceImpl	属性设置服务的实现类
009	Edu.cs.scu.propertyMapper	属性设置的 Mapper
010	Edu.cs.scu.ShopMapper	商场管理的 Mapper
011	Edu.cs.scu.propertyMapper.xml	Mybatis 配置文件
012	Edu.cs.scu.ShopMapper.xml	Mybatis 配置文件

前端：

表 4.9.5.2 商场管理模块代码列表

编号	文件名	作用
001	Component/mixin/Auth.jsx	前端登陆拦截器
002	Component/mixin/Config.jsx	前端配置管理组件

003	Redux/action/index.jsx	执行 Fetch 的 action
004	Redux/reducer/index.jsx	Redux 的 Reducer
005	Redux/store/store.jsx	管理全局状态的 store
006	Router/route.jsx	执行路由切换
008	Style/common.less	全局通用 style
009	View/shop_manager.jsx	商场管理组件
010	View/monitorSetting.jsx	属性设置组建
011	Component/bcrumb/bcrumb.jsx	全局面包屑
012	Component/mixin/EditablCell.jsx	实现可编辑表格的组件
013	Component/mixin/ShopName.jsx	添加商场的隐藏表格

4.10 历史数据查询模块

4.10.1 模块编号与中心注释

W-05 /*历史数据查询模块*/

4. 10.2 功能描述与性能描述

4.7.2.1 功能描述

表 4.10.5.3 历史数据查询模块功能列表

编号	名称	描述	优先级
001	用户活跃年历	根据历史数据，生成一年来用户活跃年历图	高
002	用户活跃月历	根据历史数据，生成一个月来用户活跃年历图	高
003	用户活跃日历	生成一天中每个小时的流量情况	高
004	年份选择	用户选择希望展示的数据的年份	高
005	月历选择	用户选择希望展示的数据的月份	高
006	日期选择	用户选择希望展示数据的日期	高
007	历史用户	展示探针探测到的所有用户一览表	高

4.10.2.2 性能描述

(1) 满足多线程访问

(2) 前端使用 immutableJS 优化性能

4.10.3 接口描述

4.10.3.1.用户活跃年历

后端接口：

表 4.7.3.1.1 历史数据查询模块接口列表

名称	路径	注解
用户活跃年历	/userActivityYearController.action	@RequestMapping

数据格式

表 4.10.3.1.2 历史数据查询模块数据格式

名称	字段
用户名	UserName

返回数据格式：

表 4.10.3.1.3 历史数据查询模块返回数据格式

名称	字段
日期	Date
活跃量	ActivityDegree

4.10.3.2 用户活跃月历

后端接口：

表 4.10.3.2.1 历史数据查询模块数据格式

名称	路径	注解

用户活跃年历	/userActivityMonthController.action	@RequestMapping

数据格式

表 4.10.3.2.2 历史数据查询模块数据格式

名称	字段
用户名	UserName

返回数据格式：

表 4.10.3.2.3 历史数据查询模块数据格式

名称	字段
日期	Date
活跃量	ActivityDegree

4.10.3.3 用户活跃日历

后端接口：

表 4.10.3.3.1 历史数据查询模块接口列表

名称	路径	注解
用户活跃年历	/userActivityMonthController.action	@RequestMapping

数据格式

表 4.10.3.3.2 历史数据查询模块请求数据格式

名称	字段
用户名	UserName

返回数据格式：

名称	字段
日期	Date
活跃量	ActivityDegree

4.10.4 算法流程图

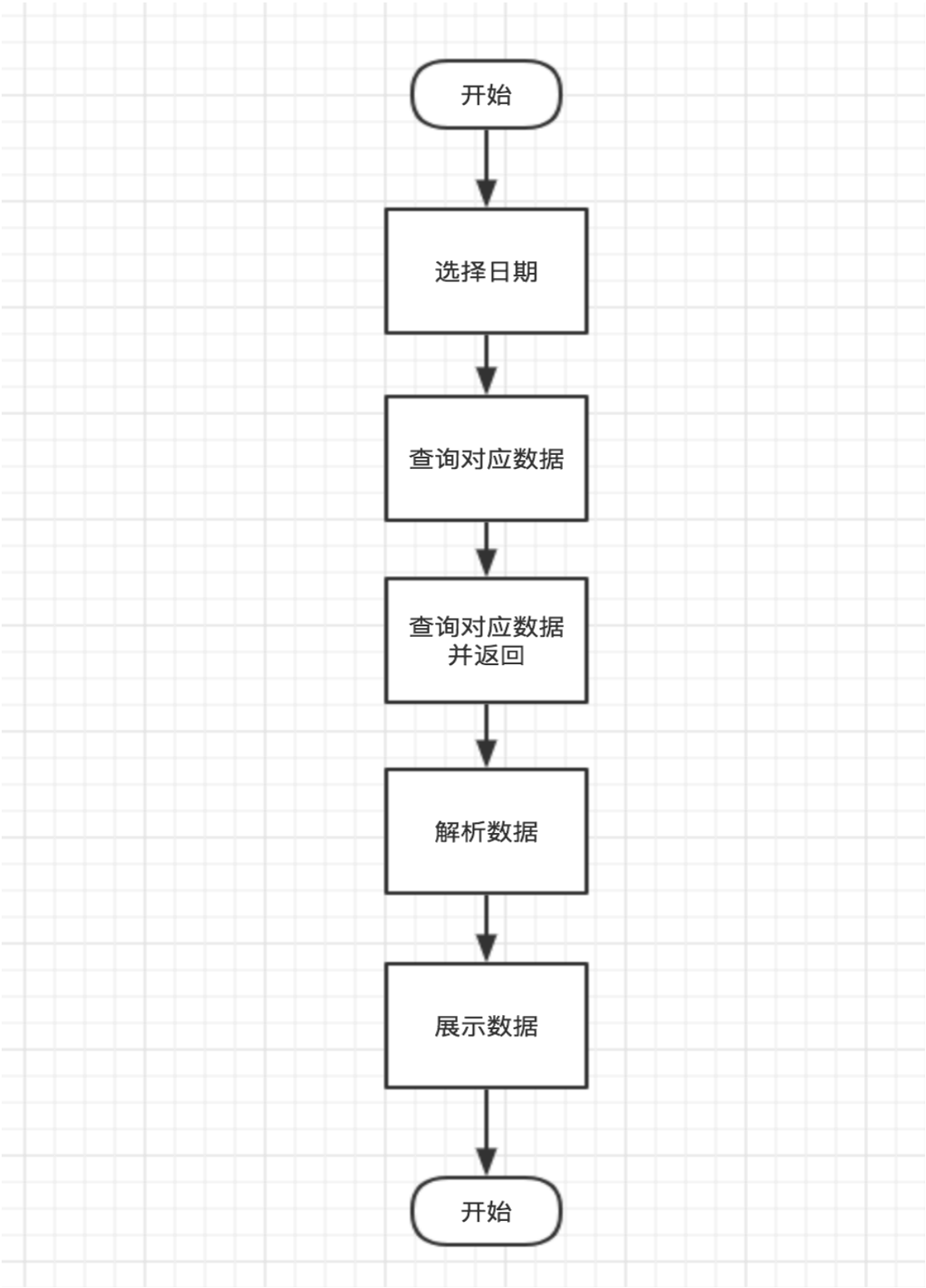

图 4.10.5 历史数据查询模块流程图

4.10.5 与本模块相关的代码

后端：

表 4.10.5.1 历史数据查询模块数据格式

编号	文件名	作用
001	edu.cs.scu.controller.QuryHistoryDataController	接收关于历史数据查询的请求参数
002	edu.cs.scu.service.QueryHistoryDataService	查询历史数据的接口
003	edu.cs.scu.service.impl.QueryHistoryDataServiceImpl	实现历史数据的类
004	edu.cs.scu.entity.HistoryData	管理历史数据的实体类
005	edu.cs.scu.mapper.QueryHistoryDataMapper	管理历史数据查询的 Mapper
006	edu.cs.scu.mapper.QueryHistoryDataMapper.xml	Mybatis 配置文件

前端：

表 4.10.5.2 历史数据查询模块数据格式

编号	文件名	作用
001	Component/mixin/Auth.jsx	前端登陆拦截器
002	Component/mixin/Config.jsx	前端配置管理组件
003	Redux/action/index.jsx	执行 Fetch 的 action
004	Redux/reducer/index.jsx	Redux 的 Reducer

005	Redux/store/store.jsx	管理全局状态的 store
006	Router/route.jsx	执行路由切换
007	Style/common.less	全局通用 style
008	View/allDataView.jsx	界面展示的 Component
009	View/ProbeUserData.jsx	探针探测到的所有用户的数据列表

4.10.6 应说明的问题与限制

使用 ImmutableJS 做性能优化

4.11 管理员用户模块

4.11.1 模块编号与中心注释

W-02 /*管理员用户模块* /

4.11.2 功能描述与性能描述

4.11.2.1 功能描述

表 4.11.2.1 管理用户模块功能列表

编号	名称	描述	优先级
001	用户登陆	管理员账号必须为手机号	高
002	用户注册	用户进行注册	高

003	输入验证	对于用户登陆时输入的内容都进行验证，比如输入的账户不是手机号时及时提醒	高
004	短信验证	登陆时给手机号发送短信验证码，验证通过才能登陆	高
005	修改密码	用户可以修改自己密码，需要手机验证	低
006	登陆验证	登陆失败时给出提示，异步刷新，登陆成功则进入管理页面	高
007	Session 验证	定时检查用户 Session，当 Session 失效时访问任何页面都重定向到登陆界面	高
008	登陆拦截	未登陆的用户访问任何页面都重定向到登陆界面	高

4.11.2.2 性能描述

无

4.11.3 接口定义

4.11.3.1 后端接口

表 4.11.3.1 管理用户模块后端接口

名称	路径	注解
用户登陆 Controller	/UserLogin.action	@RequestMapping

4.11.3.2 前端接口

使用 Fetch 发送数据，接收标准 Json 文件并进行解析

4.11.3.3 数据格式

前端发向后端的数据：

表 4.11.3.3.1 管理用户模块数据格式-1

名称	字段
账户名	Username
密码	Password
验证码	veriyfyCode

后端发送给前端数据

表 4.11.3.3.2 管理用户模块数据格式-1

名称	字段
账户名	Username
昵称	Nickname

4.11.4 算法流程图

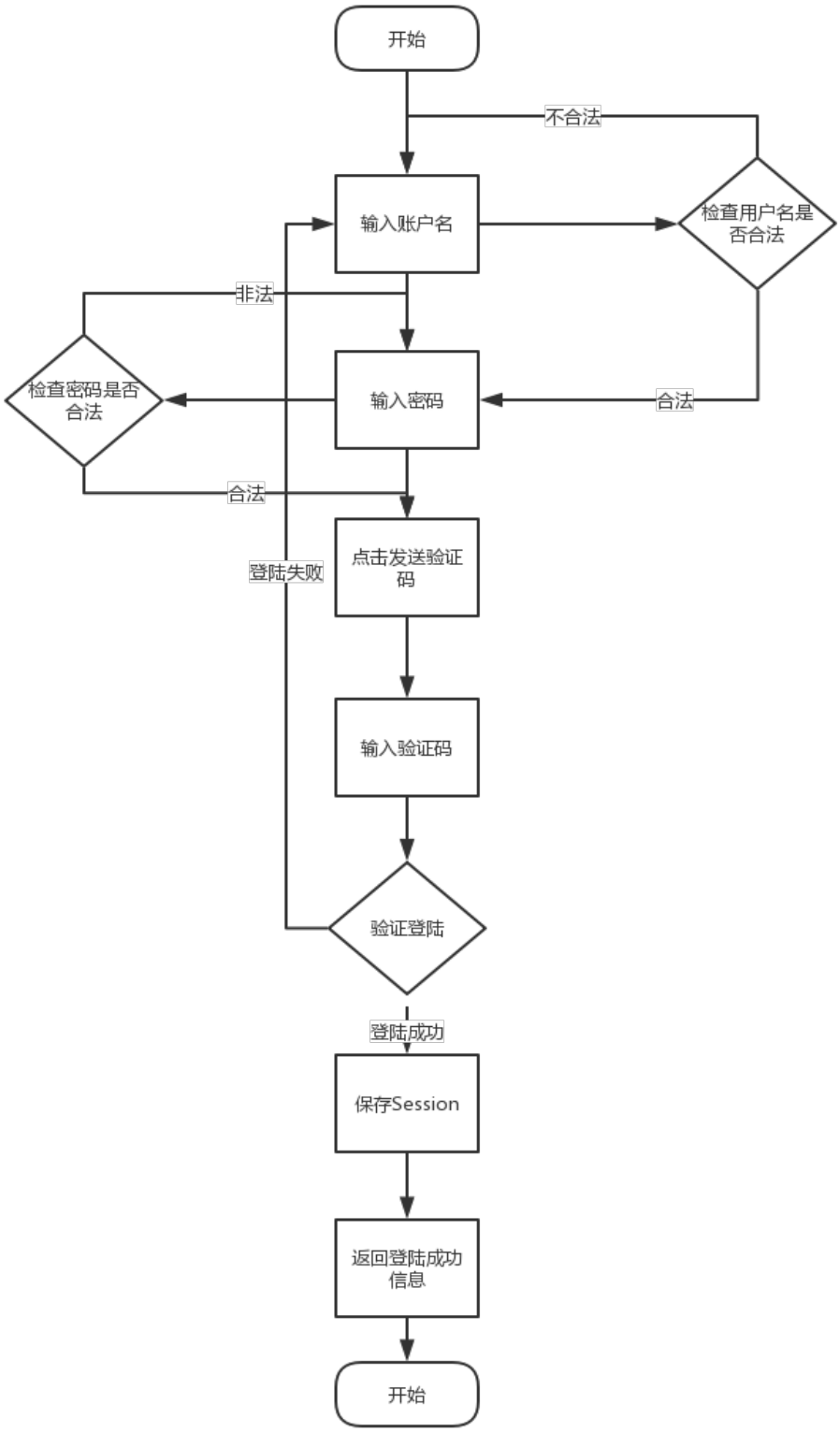

图 4.8.4 管理用户模块算法流程图

4.11.5 与本模块相关的代码表

后端：

表 4.11.5 管理用户模块代码表-1

编号	文件名	作用
001	edu.cs.scu.controller.LoginController	接收用户登陆参数
002	edu.cs.scu.entity.User	管理用户属性的实体类
003	edu.cs.scu.mapper.UserMapper	管理用户查询的 Mybatis 接口
004	edu.cs.scu.mapper.UserMapper.xml	执行 SQL 查询的 mybatis 配置文件
005	edu.cs.scu.ervice.LoginService	管理登陆的服务接口
006	edu.cs.scu.service.impl.LoginServiceImpl	实现登陆的服务接口

前端：

表 4.11.5 管理用户模块代码表-2

编号	文件名	作用
001	Component/mixin/Auth.jsx	前端登陆拦截器
002	Component/mixin/Config.jsx	前端配置管理组件
003	Redux/action/index.jsx	执行 Fetch 的 action
004	Redux/reducer/index.jsx	Redux 的 Reducer
005	Redux/store/store.jsx	管理全局状态的 store
006	Router/route.jsx	执行路由切换
008	Style/common.less	全局通用 style
009	View/login/login.jsx	登陆界面
010	View/login/style/login.less	登陆界面样式文件

011	Component/bcrumb/bcrumb.jsx	全局面包屑

4.11.6 应说明的问题与限制

4.11.6.1 网络传输加密

网络传输中使用 SSL 加密协议，SSL 是 Netscape 公司设计的主要用于 WEB 的安全传输的协议，它可以保证数据在传输过程中不被窃取和监听

4.11.6.2 登陆密码使用 Sha1 加密存储

SHA-1 是一种数据加密算法，该算法的思想是接收一段明文，然后以一种不可逆的方式将它转换成一段（通常更小）密文，也可以简单的理解为取一串输入码（称为预映射或信息），并把它们转化为长度较短、位数固定的输出序列即散列值（也称为信息摘要或信息认证代码）的过程。

使用 SHA-1 加密可以确保密码不被泄漏。