 单位代码 **10006**

学号 **18373**

分类号 **TP3**

****

毕业设计(论文)

遥感图像处理任务可视化编排系统

的设计与实现

|  |  |
| --- | --- |
| 学院名称 | 软件学院 |
| 专业名称 | 软件工程 |
| 学生姓名 | 张潇菡 |
| 指导教师 | 沃天宇 |

**2022**年**6**月

北京航空航天大学

**本科生毕业设计（论文）任务书**

Ⅰ、毕业设计（论文）题目：

遥感图像处理任务可视化编排系统的设计与实现

Ⅱ、毕业设计（论文）使用的原始资料（数据）及设计技术要求：

采用Vue、Django前后端技术构建交互平台，算子任务基于镜像实验，依赖Docker构建自动部署调度平台，通过已有的镜像进行部署构建。借助Socket构建消息文件传输平台，实现可靠的消息传输

Ⅲ、毕业设计（论文）工作内容：

基于可视化技术，采用DAG对遥感图像处理相关算法的依赖关系进行编排建模，实现后台自定义调度部署，提高遥感图像任务云上部署和调度的效率。主要内容可划分为可视化任务编排的设计与实现、云平台自定义调度与部署的设计与实现

Ⅳ、主要参考资料：

齐舸. 异构云平台下多DAG 工作流的节能调度策略研究[D].西安电子科技大学,2020.DOI:10.27389/d.cnki.gxadu.2020.000777.

常志明. 遥感图像并行算法的研究及其网格服务的实现[D].国防科学技术大学,2004.

Carl Boettiger. An introduction to Docker for reproducible research[J]. ACM SIGOPS Operating Systems Review,2015,49(1):

高畅. 大数据可视化编排与展示系统的设计与实现[D].北京邮电大学,2018.

黄科. 基于微服务的虚拟机自动化编排系统的设计与实现[D].电子科技大学,2019.

蔡凌峰,魏祥麟,邢长友,邹霞,张国敏.故障场景下的边缘计算DAG 任务重调度方法[J].计算机科学,2021,48(10):334-342.

金子威. 基于K8S的Docker分布式容器自动化运维系统的设计与实现[D].中南民族大学,2018.

龚正. Kubernetes 权威指南：从 Docker 到 Kubernetes 实践全接触：纪念版[M]. 北京：北京电子工业出版社，6-27

软 件 学院 软件工程 专业 182113 班

学生 张潇菡

毕业设计（论文）时间： 2021 年12月24日至 2022 年5月27日

答辩时间： 2022 年 6 月 日

成 绩：

指导教师：

兼职教师或答疑教师（并指出所负责部分）：

21 系（教研室） 主任（签字）：

注：任务书应该附在已完成的毕业设计（论文）的首页。

**本人声明**

我声明，本论文及其研究工作是由本人在导师指导下独立完成的，在完成论文时所利用的一切资料均已在参考文献中列出。

作者：

签字：

时间：2022 年 6 月 日

遥感图像处理任务可视化编排系统的设计与实现

学生：张潇菡

指导教师：沃天宇

摘要

云计算的快速发展在一定程度上为数据量爆炸增长的遥感图像处理提供了有效的解决方案，然而遥感图像处理流程编排和调度面临诸多问题，主要有计算机专业知识要求高，运维难度陡增，采用硬编码方式，开发效率低等问题。针对传统的通过编写脚本文件进行任务编排存在专业知识要求高、工作量冗余的问题，由此提出可视化云编排理念，将业务和技术解耦，并实现遥感图像任务云上自动部署和构建，提高遥感图像任务编排效率和云平台资源利用率。

为实现上述目标，本文结合C/S模式事务能力处理强的优点和B/S模式下网络易扩性和分布式的特点，构建C/S模式和B/S模式交叉使用的三层混合部署结构。B/S架构面向用户，使用SVG画布实现了可视化编排和可视化调度，简化用户的编排操作。C/S架构下的服务端负责云平台的维护。其通过监听数据库中数据的变化创建服务，将服务调度到多个客户端上并发运行，利用云的强算力，加快遥感任务处理性能。

本课题完成了遥感图像处理任务可视化编排系统的设计和实现，经过全面的测试，验证了系统具有较好的安全性、鲁棒性和可扩展性，可以高效、简单的完成遥感图像处理过程，基本达到了预期的效果。

关键词： 遥感图像，云平台，可视化，混合架构

英文标题

Author : Xiaohan Zhang

Tutor : Tianyu Wo

Abstract

The rapid development of cloud computing has provided an effective solution for remote sensing image processing with the explosive growth of data volume to a certain extent. However, there are many problems in the process layout and scheduling of remote sensing image processing, mainly including high requirements for computer professional knowledge, steep increase in operation and maintenance difficulty, the adoption of hard coding and low development efficiency. Based on the concept of visual cloud choreography, user operation and platform maintenance are decoupled to realize automatic deployment and construction of remote sensing image task on cloud, which can greatly improve the efficiency of remote sensing image task processing.

In order to achieve the above objectives, this paper combines the advantages of strong transaction capability of C/S mode and the characteristics of easy expansion and distribution of network under B/S mode to construct a three-layer hybrid deployment structure with the cross-use of C/S mode and B/S mode. B/S architecture is user-oriented and uses SVG canvas to realize visual choreography and scheduling, simplifying user's choreography operations. The server in the C/S architecture is responsible for cloud platform maintenance. It creates services by monitoring the changes of data in the database, schedules the services to multiple clients for concurrent operation, and uses the low ductility of edge nodes on the cloud to speed up the performance of remote sensing task processing.

This topic has completed the design and implementation of remote sensing image processing task visual arrangement system, through comprehensive testing, verified that the system has good security, robustness and expansibility, can efficiently and simply complete the remote sensing image processing process, basically achieved the desired effect.

Key words: remote sensing imagery, cloud platform, visualization, hybrid architecture

目录

[1 绪论 1](#_Toc29651)

[1.1 课题背景与意义 1](#_Toc912)

[1.1.1 课题背景 1](#_Toc18560)

[1.1.2 课题意义 3](#_Toc16999)

[1.2 国内外研究现状 3](#_Toc19190)

[1.2.1 传统任务编排系统 3](#_Toc7470)

[1.2.2 通用任务编排系统的研究 5](#_Toc14026)

[1.2.3 特定任务编排系统的研究 6](#_Toc32092)

[1.3 研究目标与内容 6](#_Toc25316)

[1.3.1 研究目标 6](#_Toc27716)

[1.3.2 研究内容 6](#_Toc12591)

[1.4 论文组织结构 7](#_Toc21115)

[2 需求分析 8](#_Toc9435)

[2.1 总体需求分析 8](#_Toc14242)

[2.2 功能需求 9](#_Toc7050)

[2.2.1 用户管理模块 9](#_Toc11589)

[2.2.2 客户端管理模块 9](#_Toc18729)

[2.2.3 可视化遥感图像处理模块 10](#_Toc9710)

[2.3 非功能需求 11](#_Toc21311)

[2.3.1 鲁棒性需求 11](#_Toc16799)

[2.3.2 易用性需求 11](#_Toc4082)

[2.3.3 可扩展性需求 11](#_Toc9041)

[2.3.4 安全性需求 11](#_Toc19999)

[2.3.5 性能需求 12](#_Toc16166)

[3 总体设计 13](#_Toc30510)

[3.1 系统部署架构设计 13](#_Toc20336)

[3.2 系统技术架构设计 14](#_Toc8332)

[3.3 系统功能模块设计 15](#_Toc20162)

[3.3.1 用户管理模块的功能 16](#_Toc4593)

[3.3.2 客户端管理模块的功能 16](#_Toc1950)

[3.3.3 可视化遥感图像处理模块的功能 16](#_Toc11997)

[3.4 数据库设计 17](#_Toc3710)

[3.4.1 用户管理模块的数据库设计 17](#_Toc25839)

[3.4.2 客户端管理模块的数据库设计 18](#_Toc24911)

[3.4.3 可视化遥感图像处理模块的数据库设计 18](#_Toc3588)

[4 详细设计与实现 25](#_Toc32663)

[4.1 部署架构的设计与实现 25](#_Toc27711)

[4.1.1 B/S模式架构搭建 25](#_Toc10448)

[4.1.2 C/S模式架构搭建 26](#_Toc22381)

[4.2 功能模块的设计与实现 29](#_Toc6866)

[4.2.1 用户管理模块 29](#_Toc11871)

[4.2.2 客户端管理模块 30](#_Toc14390)

[4.2.3 可视化遥感图像处理模块 31](#_Toc29738)

[5 测试和评估 41](#_Toc32531)

[5.1 测试目标 41](#_Toc121)

[5.2 测试环境 41](#_Toc14997)

[5.3 功能性测试 41](#_Toc9038)

[5.3.1 用户管理测试 41](#_Toc28204)

[5.3.2 客户端管理测试 42](#_Toc24548)

[5.3.3 可视化遥感图像处理测试 43](#_Toc12146)

[5.4 非功能性测试 45](#_Toc16278)

[结论与展望 47](#_Toc3059)

[致谢 49](#_Toc1262)

[参考文献 50](#_Toc17018)

# 绪论

## 课题背景与意义

### 课题背景

随着遥感探测设备、遥感平台、数据通信等相关技术的发展，现代遥感技术已经进入到一个动态、快速、多手段处理观测数据的新阶段,采集到的遥感数据量、信息量急剧增长[1]。通过遥感手段获得的对地观测数据每天以TB级的速度飞速增长，遥感影像数据也朝着高分辨率、多波段的海量数据方向发展[2]。数据量的迅速增长、遥感应用的拓展对海量遥感图像数据的处理效率提出了极大的挑战，也对图像处理工作提出更高的要求：即缺乏专业能力的人员借助极少外力帮助，可以快速、高效完成复杂的图像处理过程[11]。

传统方法已经无法进行有效分析和处理海量的遥感图像数据，随着云计算的迅猛发展，利用云来处理遥感图像数据已经成为趋势。AI Earth[3]提供遥感、气象等多源对地观测数据的云计算分析服务，用数据感知地球世界。SenseEarth[4]实现了遥感影像可视化云解译平台，以其高精度、高效率的强大算力，弥补了传统卫星影像重访率低且解译技术时效性低的问题。云计算可以把网络上的服务资源虚拟化，维护动态、可伸缩的虚拟化资源计算服务资源，通过互联网提供按需共享的服务。管理人员通过遥感图像处理系统云平台的维护，整合遥感信息和技术资源。用户不必负责整个服务资源的调度、管理和维护，即可通过网络，不受时间和地点的约束，获取一站式的空间信息云服务，最大限度的使用虚拟资源池，处理大规模计算问题[5]。

云计算的落地离不开虚拟化技术和自动化的运维平台。目前的虚拟化技术，主要分为以KVM(Kernel Virtual Machine)[6]和Xen[7]为代表的硬件辅助虚拟化和以Docker和rocket[8]为代表的操作系统级别的虚拟化[10]。VM(Virtual machines)模拟物理服务器，包括存储、网络和操作系统，比物理服务器更可移植、更快，但仍然是完整的服务器系统。容器将密度和优化提升到一个新的水平，其仅仅虚拟化运行应用程序的核心。容器直接运行在内核上，更轻量级，启动更快，管理也更流畅，在一个物理服务器上可以构建更多的容器[9]，在存储优化、CPU使用率、运维效率等方面带来的收益均比虚拟机带来的收益大得多[10]。

卫星遥感应用处理过程可以分为遥感数据的接收与分发、数据处理与信息提取两个重要阶段[11]。遥感数据的数据处理与信息提取过程可以初步分为预处理、信息增强、信息提取和融合等多个步骤，每个步骤又可以进一步细分，比如预处理过程可以细分为图像裁剪、几何校正和物理校正，如图 1.1所示。遥感图像数据处理与信息提取过程就是可以针对不同场景，通过描述依赖关系将处理步骤灵活编排。把遥感图像处理过程分成彼此独立的功能区间，封装为容器，对处理过程的编排解析为对容器的编排，对过程的调用解析为对容器的调用。利用云的强算力和边缘的低延性的特性,将单个遥感任务拆分,调度到多个边缘节点的容器上,实现任务内和任务间的并行处理,提高任务处理效率。目前受欢迎的容器调度平台以K8S、Mesos、Swarm为代表，为开发者提供了新颖、便捷的软件集成开发、部署、运维。然而，传统的容器调度依赖于脚本的编写，对计算机专业的知识储备要求较高。Docker原生的Swarm[12]依赖于Docker指令，K8S[13]对容器编排是编写YAML文件，Mesos[14]依托于C++、Java和Python语言编写的框架文件，需要查阅相应的API文档。遥感算法专业人员不善于使用这类平台。同时因海量遥感图像处理的数据庞大以及图像处理任务复杂多变的特点，遥感算法专业人员在编码过程中产生了大量的时间和人工成本。因此，开发一套简单易用的遥感图像可视化编排系统，具有较大的工程意义。

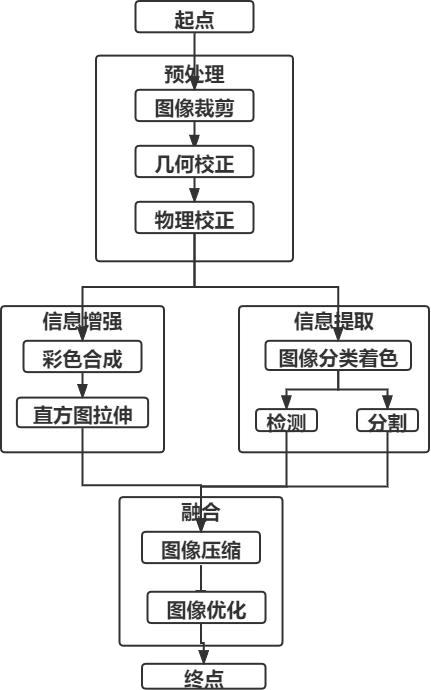


图 1.1 遥感图像数据处理与信息提取过程

### 课题意义

本文旨在利用可视化技术和云计算技术,基于可视化云编排理念，设计实现遥感图像处理任务可视化编排系统，通过模块化、拖拽式方式编排，根据应用场景快速构建处理流程，简化遥感图像任务编排流程，实现遥感图像处理算法的自动云上部署和调度，促进处理流程的快速落地，提高运维效率，解决因海量遥感图像处理步骤复杂、数据内容复杂而难以进行高效率 批量化处理的问题，为遥感图像专业人员提供有效处理遥感图像的手段，具有较大的实践意义。

## 国内外研究现状

### 传统任务编排系统

任务编排是指以任务为单位按照自定义的方式进行编排，描述任务执行过程。可将任务编排工作流建模为有向无环图(Database Availability Group,DAG)，该模型讲任务收集在一起，利用依赖项和关系说明运行逻辑。存在依赖关系的任务之间存在执行先后关系,不存在依赖关系的任务之间不存在约束关系,可以同时调度在不同的节点上同时运行,提高执行效率和任务的等待时间。如图 1.2所示,任务A和任务C之间不存在约束关系,可以同时运行。任务A和任务B之间存在依赖关系,需要先执行任务A再执行任务B。DAG本身并不关心任务内部发生的事，它只关心如何执行他们，包括运行的顺序重试的次数和执行是否超时。

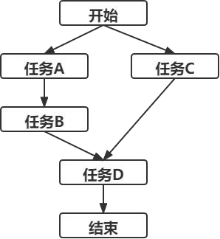


图 1.2 DAG工作流

近年来用于编排任务和数据工作流的各种通用工具快速出现。Apache-Airflow[15]是一个以编程方式创作、调度和监视工作流的平台，支持Mesos等资源调度,实现了非常完备的调度结果的可视化功能，其将工作流定义为代码，带来可维护性高、可测试性和协作性好优点的同时，导致对用户在计算机专业知识储备方面的要求极高。Luigi[16]因其简单但可伸缩和可扩展的特点，使得可以容易的构建复杂的任务依赖关系图，适用于长期运行的流式批处理任务，但其缺点在于需要编写更多的自定义代码以实现任务。Argo[17]是一个开源容器原生工作流引擎，具有轻量级和功能全面的优点,其用于在 Kubernetes 上编排并行作业,将多步骤工作流建模为一系列任务，或使用DAG捕获任务之间的依赖关系，支持高效地编排和运行计算密集型作业，但任务的编排依赖于编写类似于YAML文件定义的[领域专用语言(Domain Specified](https://www.infoq.cn/article/dsl-discussion) Language,DSL)代码, 功能实现全面,但应用比较底层,不适用于用户。这些任务的维护对计算机专业知识的要求较高，不利于遥感图像专业人士的使用。同时因海量遥感图像处理的数据庞大以及图像处理任务复杂多变的特点，处理在编码过程中产生了大量的时间和人工成本。综上所述，需要一个基于可视化技术的任务编排系统，来填补这种需求：无专业背景的人员在很少的人工干预下，可以快速、高效的编排遥感图像任务。

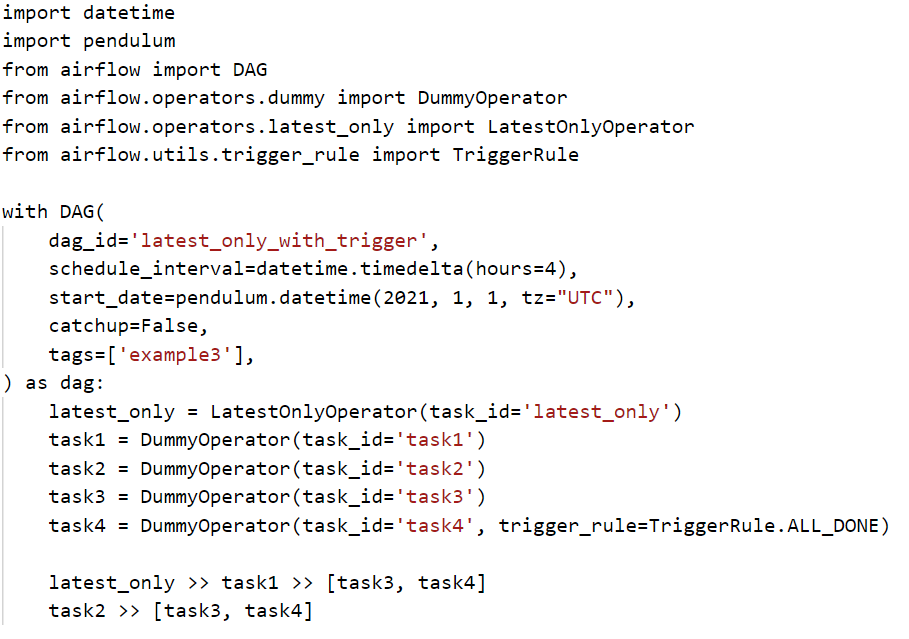


图 1.3 Apache-Airflow示例代码和工作流的可视化

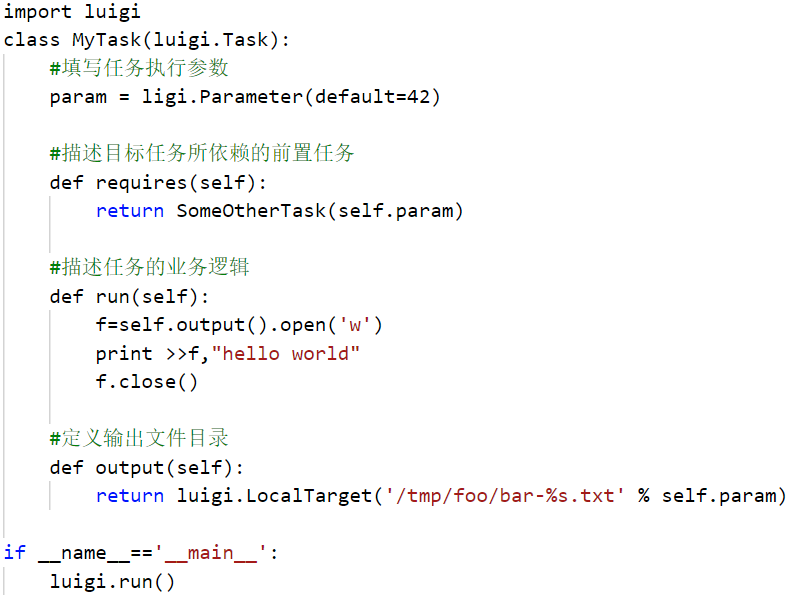


图 1.4 Luigi示例代码

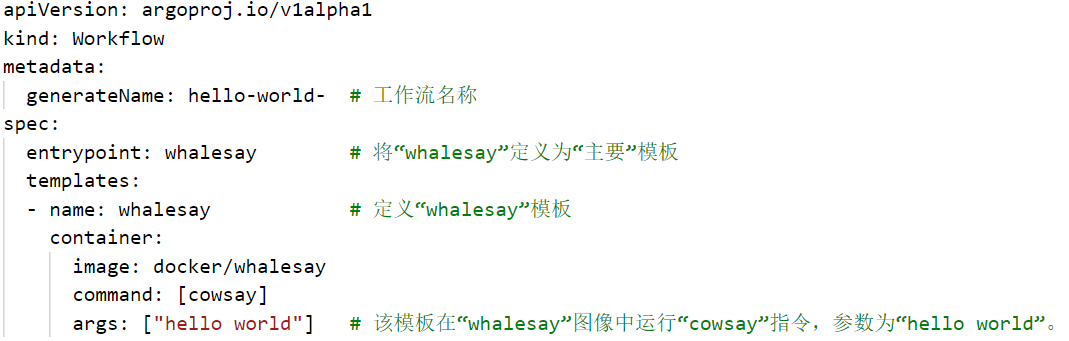


图 1.5 Argo一个简单规范的示例

### 通用任务编排系统的研究

任务编排系统具体来说是指应用中切分为单独的组件并制作为镜像，通过配置文件或者拖拽控件实现组件编排，以及通过虚拟机管理工具实现管理、部署和启动的系统[18]。针对部署速度慢以及适用场景单一的问题，文献[18]提出了基于Docker的混合云应用编排方案，提出了批量调度算法等性能优化算法，实现了良好的部署速度、分布均衡性和可靠性，但在监控完善性和模板管理上仍然存在不足。针对多约束的组合优化问题，文献[19]提出了约束规划模型，提出了综合考虑资源受限、任务约束、时间约束的任务调度方法。文献[20]实现了基于微服务的虚拟机自动化编排系统，可以实现自动执行和自动部署，但编排模板受限，灵活度不高，同时任务调度策略并不是最优解。文献[21]实现了大数据编排与展示的可视化系统,存在多平台适配不好、可视化模型渲染速率较低、可视化编排不够灵活的问题。文献[22]实现了面向APP应用的云编排系统,实现了业务和技术的有效解耦，促使应用的快速构建。

以上通用任务编排系统的研究，提出的性能优化算法等为本文在实现编排、调度方面提供了技术基础。另外，虽然在任务编排系统上已经有了上述相关研究，也有部分系统支持可视化任务监控。但以上研究实现大多是基于Python、YAML等语言进行任务编排，目前很少有针对任务编排可视化方面的研究。因此需要一个可视化的任务编排系统来填补这种需求。

### 特定任务编排系统的研究

目前大多数研究者倾向于研究面向任务编排系统的研究，而对特定领域的任务编排系统的研究较少。文献[23]采用观测目标数最大化原则进行编排，提出了相控阵雷达的观测任务编排算法，发挥雷达的最大性能。文献[24]实现了能够分布式检测的自动数采系统，在分中心上编排采集和上报任务，并维护一张任务表，进行任务的管理和调度。而遥感图像处理涉及图像切割等特定类型的任务，对任务的编排造成影响；并且任务的调度根据处理需求，涉及到反射率、温度等定量参数，需要用户的个性化设置。因此需要一个面向遥感图像的任务编排系统来填补这种需求。

## 研究目标与内容

### 研究目标

本文的主要目标是采用可视化技术，面向遥感图像处理搭建任务系统，实现可视化编排和工作流的监控。基于云平台自动管理资源，达到提高遥感图像任务云上部署和调度的目标。提供便捷的可视化操作界面。

### 研究内容

本文从遥感图像处理任务编排和调度系统的实现方面进行研究，主要内容如下：

1. 构建便捷的可视化操作界面，实现算子的管理、任务的可视化编排和工作流的监控。
2. 系统对任务DAG建模进行解析，形成JSON数据，易于机器的解析和生成。
3. 为了方便信息管理，设计MySQL数据库存储任务表和任务调度表，以状态为标志，对正常服务进行执行和故障服务进行重执行。
4. 边缘节点和核心节点交互采用C/S架构，本课题实现在Linux系统上通过Socket方式与边缘节点进行通信。

## 论文组织结构

本文研究了遥感图像处理任务可视化编排系统的设计与实现，全文的组织结构共包括六个部分，具体情况如下：

第一章，绪论。本章首先介绍了遥感图像处理任务可视化编排系统的课题背景和意义，调研了任务编排系统的相关技术研究现状，最后介绍了文本的主要研究内容和文章的组织结构。

第二章，需求分析。通过业务流程图介绍系统总体需求，通过用例图和用例说明文档阐述了系统的功能性需求，最后对系统的非功能需求做了补充说明。

第三章，总体设计。通过系统技术架构图介绍说明了该系统的技术架构设计，通过系统部署架构图介绍说明了该系统的部署架构设计，通过功能模块图介绍开发的功能模块，最后以类图和数据库表的形式介绍了数据设计。

第四章，系统设计与实现。通过流程图、状态图、顺序图、前端页面样式图对系统各个组成模块的设计与实现进行了详细的阐述，并对其中的关键技术模块的工作过程进行了简要叙述。

第五章，测试。对需求中设计的功能和非功能需求设计测试用例，并进行了测试验证，确保项目正常运行。

总结和展望。对本项目开发所作的工作进行总结，并提出系统中有待改进、可以优化的部分以及可能的改进设想。

# 需求分析

## 总体需求分析

本课题设计实现的遥感图像处理任务可视化编排系统旨在实现三个需求：

1. 利用可视化技术提高提高遥感图像任务编排效率。
2. 依托Docker平台的自动维护和自动调度，简化遥感图像任务云上部署和操作的流程。
3. 综合考虑图像分割等特殊处理任务，构建适用于遥感图像的可视化编排系统。

遥感图像处理任务可视化编排系统的业务流程图如图 2.1所示。

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

图 2.1 任务流程图

本系统有用户和管理员两种角色。用户在可视化界面管理数据信息。管理员负责云平台的维护和管理。

用户在可视化界面进行算子信息、任务信息、应用信息的管理。可以进行增删改查处理。针对任务信息，利用可视化拖拽进行任务信息编排；针对应用信息，可以实时查看应用执行情况，获取应用执行的中间结果和最终结果。

管理员负责服务端和客户端的部署管理。管理员启动新的客户端，关闭旧的客户端，并在平台上管理客户端信息。新的客户端会和服务端建立消息连接。服务端和多个客户端进行信息的交互。管理员管理客户端信息，将服务端视为核心节点，客户端视为边缘节点。核心节点负责维护Docker平台上边缘节点和算子部署的一致性，并利用边缘节点的高效性，高效处理应用信息。

## 功能需求

本系统的参与者包括用户和管理员。该项目的功能需求可以划分为用户管理模块，面向管理员的客户端管理模块，面向用户的可视化遥感图像处理模块。

### 用户管理模块

用户管理模块的用例如图 2.2所示。本系统有用户、管理员两种角色。所有用户和管理员都可以进行登录账号、修改密码和退出登录三种基础操作。用户可以查看个人信息。因为系统不是对外开放的，所以需要管理员手动创建用户。此外，管理员拥有查看用户列表、修改用户资料和删除用户的特殊权限。

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

图 2.2 用户管理模块用例图

### 客户端管理模块

客户端管理模块的用例如图 2.3所示。该模块的参与者为管理员。管理员可以管理客户端信息。管理员必须预先在目标IP上构建客户端，和服务端成功建立连接，随后管理员在平台上填写客户端信息，新建客户端，平台检验连接有效性，如无有效连接，会新建失败。管理员可以在平台上管理客户端的镜像存储地址、图像存储地址、管理员账号密码等信息，用于后续算子的构建和调度。

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

图 2.3 客户端管理用例图

### 可视化遥感图像处理模块

可视化遥感图像处理模块的用例如图 2.4所示。该模块的参与者包括时间和用户，主要可以分为算子管理、应用管理、任务管理三个部分。

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

图 2.4 可视化遥感图像处理模块

算子管理的用例包括查看算子列表、新建算子、删除算子、编辑算子、解析算子和部署算子。用户可以通过在平台上上传算子镜像，预设算子构建信息和算子调度信息，在平台上新建算子，对算子实行增删改查。平台每隔一段时间会查看一次数据库，从数据库中获取用户新上传的算子，生成算子构建服务，并执行服务，完成算子的自动构建。平台需要维护算子构建的一致性，针对构建失败和客户端新建的情况，采取相应的操作，保证所有算子在所有平台上完成构建。

应用管理的用例包括查看应用列表、新建应用、删除应用和编辑应用。用户可以在应用编辑界面通过拖拽、连线和信息填写完成应用的可视化编排，生成有效的DAG建模，同时实现编排结果的重显示和重编排，优化用户的使用满意度。用户在保存DAG建模时，平台需要根据一定的逻辑校验，并将错误建模提示返回给用户；针对图像分割等特殊应用，应用生成独特的校验逻辑。平台对DAG建模数据完成校验和解析，生成拓扑排序，便于后续的使用。

任务管理的用例包括查看任务列表、新建任务、删除任务、查看任务、解析任务和调度子任务。用户可以针对应用创建任务，平台获取应用的解析数据，生成服务，完成遥感图像任务在云平台上的自动调度，并针对调度失败完成重调度，避免因网络异常等因素造成任务执行的崩溃。在任务查看界面，用户可以查看任务在边缘节点上的分配情况和当前调度情况，并在任务调度结束后获取调度结果。

## 非功能需求

### 鲁棒性需求

系统的鲁棒性主要体现在对系统内部和用户行为的各种错误进行全面的考虑和处理，比如管理员新建用户时填写的账号名重复，用户编辑任务时DAG建模不符合规范（比如存在环或者算子入度与实际输入数量不匹配），服务端进行调度任务时发生网络异常等情况导致调度失败等等，当这些错误发生时，系统应当作出一定的处理和反馈，保证系统正常运行不崩溃，并给用户反馈当前的错误信息，提示其正确操作。

### 易用性需求

可用性是指系统的易用性，以及它提供的操作是否易于用户理解。本系统面向管理员和用户，要保证系统整体逻辑清晰，前端字体清晰、间距适当，且界面不太拥挤。同时，每个工作入口均应明确，应将按钮放置在清晰可见的位置。采用图表展示、拖拽绘图等可视化方法，让用户快捷获取所需要的信息，进行操作。

### 可扩展性需求

该系统将在以后的阶段中经历长期迭代过程。应注意的是，整个项目目录结构清晰，文件名清晰易懂。注释代码方便以后的扩展或重建。随着甲方需求的变化，可以向系统中添加新的功能模块，以确保整个系统具有高度的可扩展性。

### 安全性需求

系统的接口对请求人进行权限验证，根据token对用户身份鉴权能提升系统的安全性，使得私密数据不被泄露、重要数据不被恶意修改。此外，数据库需要对用户密码加密，在一定程度提升数据的安全性。

### 性能需求

对于用户和管理员而言，页面上每个操作之后的加载时间不应太长。页面加载时间受计算机性能的影响越快越好。后端数据库的有效操作以及表示层的前端与后端之间的直接有效通信，对于快速及时地进行前端响应也至关重要。针对算子的部署和任务的调度涉及到文件的传送和脚本执行，需要较长的执行时间。该系统的服务端采用多线程调度，用户无需等待实际的执行，先将数据提交，系统先返回后处理。对于服务的执行涉及到文件的传递，需要尽快和无误的传送文件。任务的调度会优先分配在性能较好的客户端上，并实现任务内和任务间的并行调度，让多个客户端同时运转，处理信息，降低任务调度所需要的时间，提高性能。

# 总体设计

## 系统部署架构设计

系统部署架构图如图 3.1所示。本系统需要具备用户需要的数据管理和任务编排的功能，又需要具备分布式容器的构建、管理和容器的编排和调度功能。为此，系统使用C/S模式和B/S模式交叉使用的三层混合部署结构，集齐C/S模式事务能力处理强的优点和B/S模式下网络易扩性和分布式的特点，实现用户模式下的开放性和通用性以及云平台的封闭性、专用性和动态伸缩性[25]。

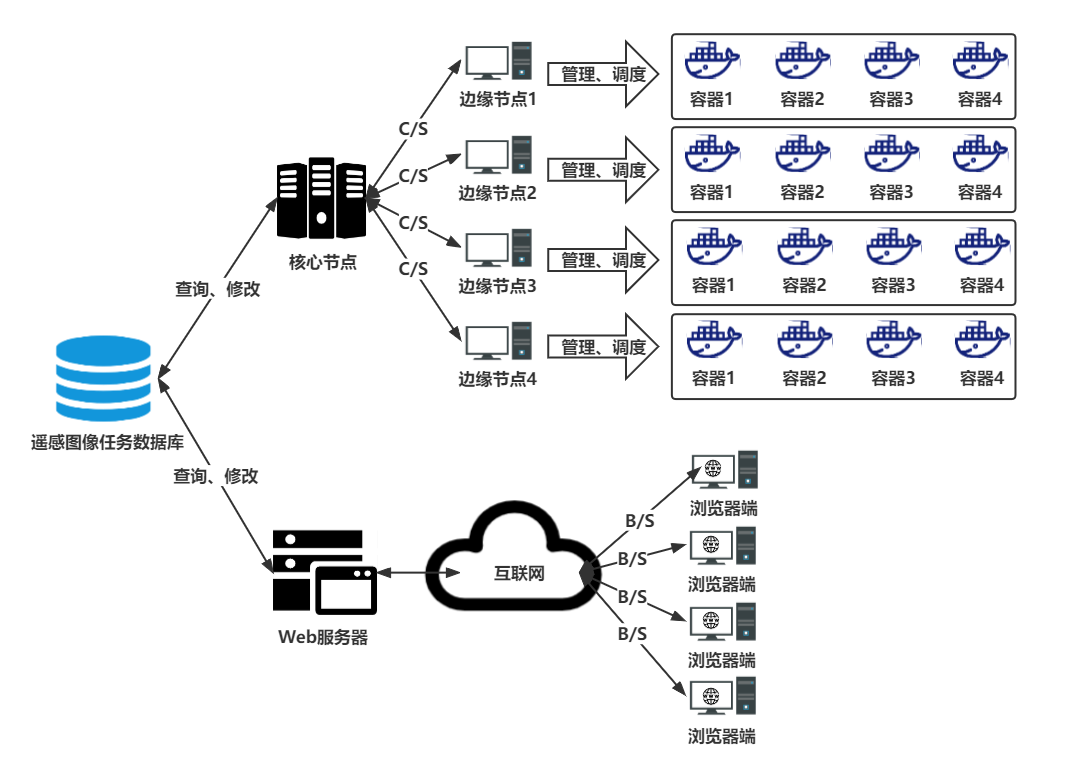


图 3.1 系统部署架构图

B/S结构面向外部用户，此架构方便用户进行访问，无需安装客户端，只需要输入一个网址即可进行查询，能比较方便地为用户提供服务，实现用户需要的数据管理的功能。其中B（Browser）端为浏览器端，负责发送请求，对返回数据接收和并可视化显示；S（Server）端为Web服务器端，负责处理请求，解析数据，对数据库进行查询和修改操作，返回结果。其中Web服务器端提供整个系统所需的计算能力，包括系统运行环境、系统业务逻辑和系统的数据源；浏览器端实现用户交互，提供可视化界面，隐藏底层逻辑，便于用户使用并采用可视化技术显示数据。

C/S结构面向管理员，整合服务资源，是整个任务编排系统的控制中心，维护Docker容器平台，负责管理云上容器的管理和调度。其中S（Server）端为核心节点，通过监听遥感图像任务数据库中算子和任务的变化，针对算子的部署和服务的创建处理底层逻辑，将任务调度映射为容器调度，给服务分配客户机。C（Client）端为边缘节点，负责和服务端通信，获取服务，完成实际的容器构建和调度的过程。

## 系统技术架构设计

系统技术架构设计如图 3.2所示。

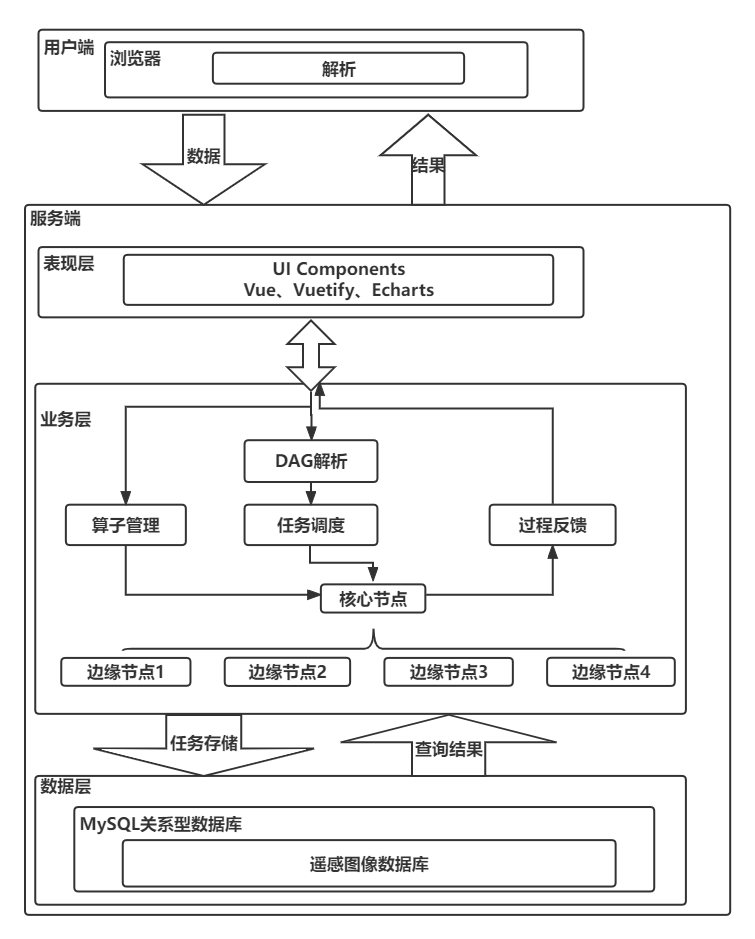


图 3.2 系统技术架构图

服务端的分层设计采用的是Spring经典的MVC层次结构，将系统分为数据层（Models）、表现层（Views）和业务层（Controller）。系统运行环境为java JDK 1.8.0\_211，采用主流的springboot框架作为服务器开发框架，并集成mybatis完成数据库的连接和访问操作。

表现层采用Vue框架，采用Vuetify样式库保证整个网站风格和样式的统一性。另外为了使得工作流监控、任务统计信息的展示更加直观，采用Echarts中实现数据的可视化。

业务层的处理主要由基于Django框架的后端和基于Python语言编写的核心节点控制。业务的处理流程为：用户与浏览器交互，完成算子、应用等信息管理操作，进行DAG建模，并向后端发出请求，业务层接收表现层传来的请求，完成信息管理并解析DAG数据，存入数据库。核心节点监控到数据库中数据的变化生成服务，将服务调度到边缘节点上运行，并将调度结果返回给表现层。

数据层包含MySQL关系型数据库，存放用户管理的基本信息和服务调度的过程信息。

## 系统功能模块设计

系统的功能模块如图 3.3所示。根据需求分析，开发时将系统分为用户管理、客户端管理、可视化遥感图像处理三个功能模块，其中可视化遥感图像处理模块包括算子管理、应用管理、任务管理三个子模块。不同的功能模块在开发时对应着不同的微服务。

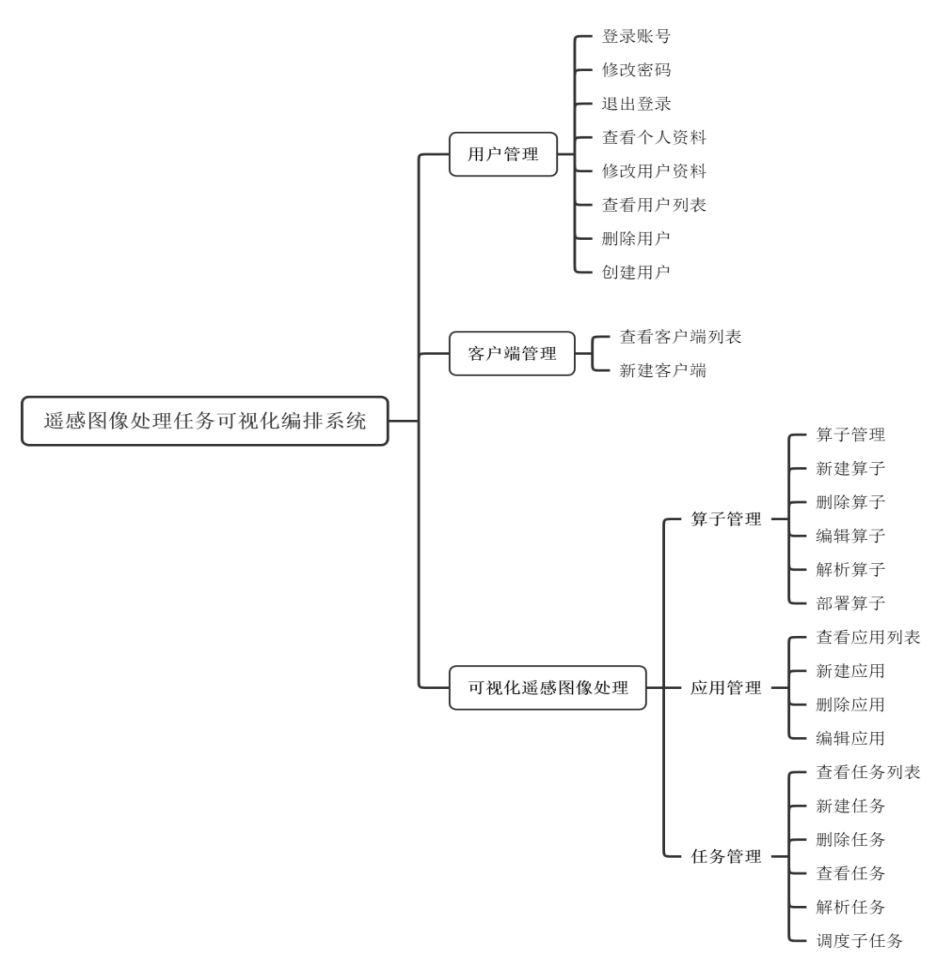


图 3.3 功能模块图

### 用户管理模块的功能

提供基本的登录注册功能。针对系统中的界面和功能，增加鉴权验证：用户直接通过浏览器的路由访问界面时，前端会判断用户的登录状态以及是否有访问的权限，针对无权限的访问，会进行系统提示，并跳转至相应的页面；针对后端所有接口，在进行相关的操作前会判断调用者的权限，如果存在超出权限的调度操作，则不再进行后续的操作，直接返回权限错误的错误码。同时该模块提供面向管理员的用户管理功能，管理员具有创建用户和删除用户的特殊权限，并将创建的账号通过邮件告知对方。

### 客户端管理模块的功能

管理员在客户端管理模块上可以管理客户端列表，维护客户端信息。系统所需要的客户端信息包括管理员账号密码、本地镜像存放目录、本地图片存放目录等，分别运用于算子构建和算子调度的过程中。管理员在平台上新建客户端前，需要先在目标IP上部署客户端，和服务端建立有效的连接。随后，管理员在平台上新建客户端，输入目标IP、端口，并输入客户端信息。最后，系统通过判断目标IP客户端的连接有效性，在连接无效的情况下后续服务器在镜像构建和子任务调度时会获取数据库中的客户端信息，维护客户端上的文件存放。

### 可视化遥感图像处理模块的功能

可视化遥感图像处理模块可以分为算子管理、应用管理和任务管理。

在算子管理模块，用户可以上传算子，由平台实现算子的自动构建，并负责维护算子的Docker边缘节点上的部署一致性。考虑到遥感图像任务的处理涉及到更多自定义参数，支持用户对算子的类型和算子信息支持更多的自定义形式。用户在上传算子的时候，支持对算子参数、运行代码、输入路径、输出路径的自定义。考虑到算子分割等特殊任务，支持算子的运行有多个输入和输出，通过设置参数类型，分别为输入、输出和无来识别。

在应用管理模块，实现应用的可视化编排，简化编排流程。将算子视为单位，通过拖拽和连线进行可视化编排，补全算子参数信息。平台需要完成对编排数据的检验和解析。同时考虑到分割图像和合并图像的特殊遥感处理任务，设计更加灵活的可视化编排。

在任务管理模块，实现遥感任务的自动调度和对工作流的监控。平台需要对新建任务数据进行解析，由核心节点控制，让任务调度到多个边缘节点上运行，实现任务内和任务间的并发，并针对调度失败的情启动重调度。用户可以查看工作流，在任务查看界面查看任务的分配情况的当前执行步骤。

## 数据库设计

数据库类图如图 3.4所示。

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

图 3.4 数据库类图

### 用户管理模块的数据库设计

user表记录用户和管理员的信息，包括用户名、密码、账号备注字段，利用is\_admin字段区分用户和管理员，如表 3.1所示。

表 3.1 user表

| 字段名 | 类型 | 描述 | 备注 |
| --- | --- | --- | --- |
| ID | int | 用户标识 | 主键 |
| username | varchar | 用户名 |  |
| password | varchar | 密码 |  |
| description | varchar | 账号描述 |  |
| is\_admin | boolean | 是否为管理员 | True表示为管理员，False表示为用户 |
| email | varchar | 邮箱 |  |

### 客户端管理模块的数据库设计

client表记录的信息，包括客户端IP、端口、系统账号名、图像存储路径等字段，如表 3.2所示。

表 3.2 client表

| 字段名 | 类型 | 描述 | 备注 |
| --- | --- | --- | --- |
| ID | int | 客户端标志 | 主键 |
| IP | varchar | 客户端所在IP |  |
| port | int | 客户端所在端口 |  |
| operator\_dir | varchar | 镜像存储目录 |  |
| img\_dir | varchar | 图像存储目录 |  |
| password | varchar | 密码 |  |
| account | varchar | 账号名 |  |

### 可视化遥感图像处理模块的数据库设计

算子管理模块包含operator表、operator\_parameter表和distribute\_wait表。

operator表记录算子的信息，包括镜像存储目录、输入文件路径、输出文件路径、容器执行代码、镜像文件名称等字段，其中user\_ID字段是引用user表ID字段的外键，如表 3.3所示。

operator\_parameter表记录算子参数的信息，包括名称、类型等字段，其中operator\_ID字段是引用operator表ID字段的外键，如表 3.4所示。

distribute\_wait表记录算子构建的信息，包括目标算子、状态等字段，其中operator\_ID字段是引用operator表ID字段的外键，client\_ID字段是引用client表ID字段的外键，如表 3.5所示。

表之间的关系如下：

1. 算子和用户是多对一的关系。一个算子只有一位创建者，但一个用户可能创建不同算子，其后同理，不再赘述。
2. 算子参数和算子是多对一的关系。一个算子参数只能属于一个算子，但一个算子可能拥有多个算子参数。
3. 算子构建信息和算子是一对多的关系。 一个算子构建信息只针对一个算子，但一个算子可能有多个算子构建信息。
4. 算子构建信息和客户端和一对多的关系。一个算子构建信息只针对一个客户端，但一个客户端可能有多个算子构建信息。

表 3.3 operator表

| 字段名 | 类型 | 描述 | 备注 |
| --- | --- | --- | --- |
| ID | int | 算子标识 | 主键 |
| user\_ID | int | 用户 | 外键，指向user(ID) |
| file | varchar | 镜像存储目录 |  |
| filename | varchar | 镜像tar包名称 |  |
| input\_dir | varchar | 输入文件路径 |  |
| output\_dir | varchar | 输出文件路径 |  |
| code | varchar | 容器的执行脚本 |  |
| description | int | 算子描述 |  |

表 3.4 operator\_parameter表

| 字段名 | 类型 | 描述 | 备注 |
| --- | --- | --- | --- |
| ID | int | 算子参数标志 | 主键 |
| operator\_ID | int | 算子 | 外键，指向operator(ID) |
| name | varchar | 参数名称 |  |
| parameter\_type | varchar | 参数类型 | 分为input、output、none |
| sequence | int | 参数顺序 |  |
| description | int | 参数描述 |  |

表 3.5 distribute\_wait表

| 字段名 | 类型 | 描述 | 备注 |
| --- | --- | --- | --- |
| ID | int | 构建标志 | 主键 |
| operator\_ID | int | 算子标志 | 外键，指向operator  (ID) |
| client\_ID | int | 外键 | 外键，指向client(ID) |
| status | varchar | 构建状态 |  |
| infos | varchar | 算子构建相关参数 |  |
| log | varchar | 构建日记 |  |

应用管理模块包含app表、DAG\_node表、node\_parameter表和DAG\_line表。

app表记录应用的信息，包括名称字段，其中user\_ID字段是引用user表ID字段的外键，如表 3.6所示。

DAG\_node表记录DAG图节点的信息，包括可视化坐标x、可视化坐标y、独特标识符等字段，其中operator\_ID字段是引用operator表ID字段的外键，app\_ID字段是引用app表ID字段的外键，如表 3.7所示。

node\_parameter表记录DAG图节点参数的信息，包括参数值等字段，其中node\_ID字段是引用DAG\_node表ID字段的外键，operator\_parameter\_ID字段是引用operator\_parameter表ID字段的外键，如表 3.8所示。

DAG\_line表记录DAG图线段的信息，包括输入节点、输出节点等字段，其中from\_node\_ID、to\_node\_ID字段是引用DAG\_node表ID字段的外键，如表 3.9所示。

表之间的关系如下：

1. DAG图节点和应用是多对一的关系。一个DAG图节点只属于一个应用，但一个应用可能含有多个DAG图节点。
2. DAG图节点和算子是多对一的关系。一个DAG图节点只应用一个算子，但一个算子可能对应多个DAG图节点。
3. DAG图节点参数和DAG图节点是多对一的关系。一个DAG图节点参数只属于一个DAG图节点参数，一个DAG图节点可能含有多个DAG图节点。
4. DAG图节点参数和算子参数是多对一的关系。一个DAG图节点参数只对应一个算子参数，一个算子参数可能产生多个DAG图节点参数。
5. DAG图线段和输入节点是一对多的关系。一个DAG图线段只含有一个输入节点，一个输入节点可能存在多个DAG图线段，其后同理，不再赘述。

表 3.6 app表

| 字段名 | 类型 | 描述 | 备注 |
| --- | --- | --- | --- |
| ID | int | 应用标志 | 主键 |
| user\_ID | int | 用户 | 外键，指向user(ID) |
| name | varchar | 应用名称 |  |
| DAG\_info | varchar | DAG数据解析 |  |
| description | varchar | 应用描述 |  |

表 3.7 DAG\_node表

| 字段名 | 类型 | 描述 | 备注 |
| --- | --- | --- | --- |
| ID | int | 节点标志 | 主键 |
| operator\_ID | int | 算子 | 外键，指向operator(ID) |
| app\_ID | int | 应用 | 外键，指向app(ID) |
| pos\_x | int | 可视化坐标x |  |
| pos\_y | int | 可视化坐标y |  |
| identity | int | 特殊标识符 |  |

表 3.8 node\_parameter表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 描述 | 备注 |
| ID | int | 节点参数标志 | 主键 |
| node\_ID | int | 节点 | 外键，指向node(ID) |
| operator\_parameter\_ID | int | 算子参数 | 外键，指向operator\_parameter(ID) |
| value | varchar | 参数值 |  |
| export | boolean | 是否导出 | 当对应operator\_parameter的parameter\_type为output时有用。 |

表 3.9 DAG\_line表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 描述 | 备注 |
| ID | int | 线段标志 | 主键 |
| from\_node\_ID | int | 外键 | 外键，指向DAG\_node(ID) |
| to\_node\_ID | int | 外键 | 外键，指向DAG\_node(ID) |

任务管理模块包含task表、task\_node\_status表、task\_output表和task\_node\_wait表。

task表记录任务总体的信息，包括名称、状态等字段，其中app\_ID字段是引用app表ID字段的外键，user\_ID字段是引用user表ID字段的外键，如表 3.10所示。

task\_node\_status表记录子任务运行情况的信息，包括状态等字段，其中task\_ID字段是引用task表ID字段的外键，node\_ID字段是引用DAG\_node表ID字段的外键,如表 3.11所示。

task\_output表记录任务导出文件的信息，包括名称、存储地址等字段，其中task\_ID字段是引用task表ID字段的外键，如表 3.12所示。

task\_node\_wait表记录子任务调度情况的信息，包括状态等字段，其中client\_ID字段是引用client表ID字段的外键，task\_node\_status\_ID字段是引用task\_node\_status表ID字段的外键,如表 3.13所示。

表之间的关系如下：

1. 任务和应用是多对一的关系。一个任务只来自一个应用，但一个应用可能产生多个任务，其后同理，不再赘述。
2. 子任务运行信息和任务是多对一的关系。一个子任务运行信息只属于一个应用，但一个任务可能含有多个子任务运行信息。
3. 子任务运行信息和DAG图节点是多对一的关系。一个子任务运行信息只对应一个DAG图节点，一个DAG图节点可以产生多个子任务运行信息。
4. 子任务调度信息和客户端是多对一的关系。一个子任务调度信息只运行在一个客户端上，一个客户端上可以调度多个子任务调度信息
5. 子任务调度信息和子任务运行信息是一对一的关系。一个子任务调度信息只应用一个子任务运行信息，一个子任务运行信息只产生一个子任务调度信息

表 3.10 task表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 描述 | 备注 |
| ID | int | 任务标志 | 主键 |
| app\_ID | int | 外键 | 外键，指向app(ID) |
| user\_ID | int | 外键 | 外键，指向user(ID) |
| name | varchar | 名称 |  |
| input | varchar | 目标图像存储地址 |  |
| status | varchar | 任务状态 | 分为Start、Run、Pause、End |
| description | int | 参数描述 |  |

表 3.11 task\_node\_status表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 描述 | 备注 |
| ID | int | 子任务运行标志 | 主键 |
| task\_ID | int | 外键 | 外键，指向task(ID) |
| node\_ID | int | 外键 | 外键，指向DAG\_node(ID) |
| status | varchar | 子任务运行状态 | 分为Start、Run、Pause、End |

表 3.12 task\_output表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 描述 | 备注 |
| ID | int | 导出文件标志 | 主键 |
| task\_ID | int | 外键 | 外键，指向task(ID) |
| name | varchar | 文件名称 |  |
| file | varchar | 文件存储地址 |  |

表 3.13 task\_node\_wait表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 描述 | 备注 |
| ID | int | 子任务调度标志 | 主键 |
| status | varchar | 子任务调度状态 |  |
| rear\_wait\_ids | varchar | 子任务后置节点调度标志的拼接 |  |
| infos | varchar | 子任务调度相关参数 |  |
| client\_ID | int | 外键 | 外键，指向client(ID) |
| task\_node\_status\_ID | int | 外键 | 外键，指向task\_node\_status(ID) |

# 详细设计与实现

## 部署架构的设计与实现

### B/S模式架构搭建

B/S架构下的服务端采用基于Python语言的Django框架，在项目中配置mysqlclient，创建Model来操作MySQL数据库中的十三张表。服务端为浏览器的页面提供接口，维护系统的逻辑，将用户和管理员提交的数据存储到遥感图像数据库中。

B/S架构下的浏览器采用Vue框架进行网页编写，用Vuetify框架美化样式。根据需求分析，开发实现了9个前端页面：登录页、客户端管理页、算子管理页、应用管理页、应用编辑页、任务管理页、任务详情页、自定义错误处理页404、自定义错误处理页500。

应用编辑页采用原生SVG画布实现遥感图像任务的可视化编排，其通过Vue框架的component将算子封装为组件，将算子信息和位置信息定义为组件的属性，定义连线规则来表示拓扑组件之间的关联关系，并以SVG图像的方式展示在页面中。该编排方式提取算子资源的基本要素，屏蔽底层具体实现细节，使用户专注与应用编排的整体框架，而非模板语法的学习，实现业务和技术的解绑。可视化编排的操作流程如图 4.1所示。任务管理页和任务详情页使用Apache ECharts实现遥感图像任务调度的可视化查看和任务数据统计。

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

图 4.1 可视化编排流程图

### C/S模式架构搭建

C/S架构下的服务端和客户端基于Python进行脚本编写。服务器作为核心节点，对遥感图像数据库进行数据监控，监听数据库中operator表、client表和task表的数据变化生成服务，存储到distribute\_wait表和task\_node\_wait表中。服务端每间隔一段时间会从distribute\_wait表和task\_node\_wait表中查询服务，和客户端进行消息和文件的传递，进行信息通信。客户端作为边缘节点，接收核心节点的消息并进行实际处理，执行服务并给核心节点反馈执行情况。服务器会不断更新数据库中服务的执行情况。

#### 消息传递

服务器和客户端通过Socket通信，进行消息的传递。消息包括服务信息和心跳消息。心跳消息的作用是检查服务端和客户端之间的连接是否有效。服务器会通过监听数据库中算子信息和任务信息、客户端信息的变化，生成服务。服务包括以下两种：

1. 镜像构建。服务器需要维护所有边缘节点上镜像的一致性。当有新的客户端接入系统时，服务器会从数据库中获取所有的算子信息，生成所有算子到新客户端上构建的任务，存入算子构建表；当有新的算子上传到系统上时，服务器会从数据库中获取所有的客户端列表，生成新算子到所有客户端上构建的任务，存入算子构建表。
2. 子任务调度。服务器需要为新建的任务下和故障子任务分配客户端。当用户新建任务时，服务器会采用任务调度算法，优先采用不忙碌的、性能好的客户端，将每个子任务分配到指定客户端上，并尽量实现任务内和任务间的并行；当发现有子任务因客户端故障，无法继续运行时，服务器会分配一个新的客户端，重新生成一个子任务调度服务。服务器将子任务调度服务存到子任务调度表中。

服务端从数据库的算子构建表和子任务调度表中获取服务，利用JSON数据类型，将消息类型存储在cmd关键字中，将消息其余信息存在其余关键字中，并将JSON数据进行编码，转为二进制数据。客户端接收二进制数据并解码，还原为JSON数据，客户端利用cmd关键字进行判断，转而进行不同的操作。消息类型一共有以下两种类型：

1. load\_container类型为构建算子的命令，将本地镜像tar包进行加载，实际调度命令如图 4.2所示，需要参数信息如表 4.1所示。



图 4.2 算子构建命令

表 4.1 算子参数参数

|  |  |
| --- | --- |
| 关键字 | 含义 |
| file\_location | 镜像本地存储地址 |

1. exec\_task类型为调度子任务的命令，将指定镜像运行，获取需要的结果，实际调度命令如图 4.3所示，需要参数信息如表 4.2所示。该命令通过构建数据卷的方式，实现容器内部和主机之间文件的传递和交互。

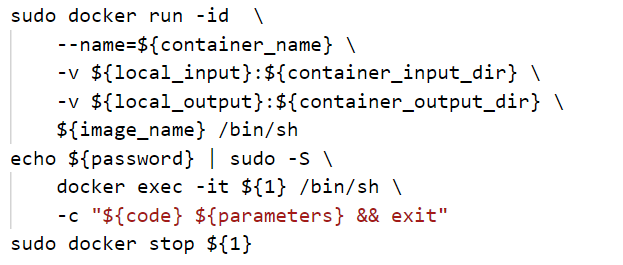


图 4.3 子任务调度命令

表 4.2 子任务调度参数

|  |  |
| --- | --- |
| 关键字 | 含义 |
| image\_name | 镜像本地存储地址 |
| container\_name | 容器名称 |
| local\_input | 本地输入文件目录 |
| local\_output | 本地输出文件目录 |
| container\_input\_dir | 容器输入文件目录 |
| container\_output\_dir | 容器输出文件目录 |
| code | 容器运行代码 |
| parameters | 容器运行参数 |
| password | 管理员密码 |

#### 文件传递

为了保证文件传递的稳定性，服务器和客户端通过建立TCP连接，进行文件的传递。因遥感图像处理的特殊性，服务端和客户端之间要传递图像数据；服务器需要把镜像tar包发送给客户端。基于这两种需求的需要，服务端和客户端均会创建一个文件接收服务器，监听本地端口，接收文件。当客户端需要发送文件时，会创建一个线程，运行文件发送客户端，通过已知的目标机的IP和端口，和目标机上的文件接收服务器建立TCP连接，发送文件。发送步骤如图 4.4所示。初始化消息包含四个信息：文件类型、本地文件大小、本地文件路径。文件类型包括镜像和图片两种类型。

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

图 4.4 发送文件

## 功能模块的设计与实现

### 用户管理模块

用户管理模块需要对数据库中的用户进行增删改查操作。

在用户登录时，会对用户的账号权限进行判断，如果是管理员，登录后会跳转到面向管理员的客户端管理界面；如果是普通用户，会跳转到面向用户的应用管理页面。在页面的访问过程中，均会对用户的权限进行重新判断。使用者通过浏览器的路由直接跳转，进入不存在界面时，会跳转到自定义404页面；使用者的登录状态会在浏览器上保存7天，如果登录状态失效，进入系统页面时，会重定向到登录界面；使用者在登录状态进入超越权限、禁止查看的界面时，系统会提示账号权限不够，并注销登录状态，重定向到登录页面。

管理员具有特殊权限，可以新建、修改和删除用户信息。后端的接口首先会进行用户鉴权鉴权，如果调用者为普通用户，则后端返回权限错误的错误码。管理员新建用户，输入账号名、密码、账号备注和邮箱，后端在数据库中查询账号名是否存在于数据库，如果已存在，则返回账号已存在的错误码。接着，后端会判断传过来的参数是否合理，比如密码符合密码格式规范，邮箱符合邮箱格式规范，如果错误，则返回密码格式错误、邮箱格式错误的错误码。经过这一系列的判断之后，后端会新建用户，存到数据库中，并将新建的用户信息，包括账号名、密码和账号备注，发送到指定邮箱中，通知用户。

### 客户端管理模块

客户端管理模块需要对B/S架构下的客户端信息进行管理，提供可视化界面对客户端的信息进行增删改查。管理员需要在新的服务器上搭建客户端，打开服务，和服务端建立连接。服务端维护一个客户端连接数据。客户端可以主动断开连接，服务端会在客户端连接数据中删除该客户端对象；服务端会和所有连接着的客户端不断发出心跳消息，如果在一定时间内收不到回复，则认为该连接失效，在客户端连接数据中标注。后端每天定时和服务端进行一次通信，交换客户端连接数组，检查数据库中保存的客户端对象是否仍然和服务端建立有效连接，并在客户端管理页面显示每个客户端的连接状态。

管理员新建客户端的页面效果图如图 4.5所示，顺序图如图 4.6所示。用户在客户端管理界面点击新建客户端按钮，前端会显示弹框，需要用户输入客户端定位信息，即IP和端口，和客户端后续使用需要的信息，包括镜像存放地址、管理员账号密码灯信息。用户点击新建按钮，前端会发送请求，把新建客户端的信息传给后端，后端首先会和服务端进行通信，服务端在在客户端连接数组中检查，查看该客户端是否已经和服务端建立连接，如果不在，后端则会返回客户端建立连接失败的错误码，要求用户重新填写数据。经过连接验证后，后端才会将客户端信息存储在client表中。

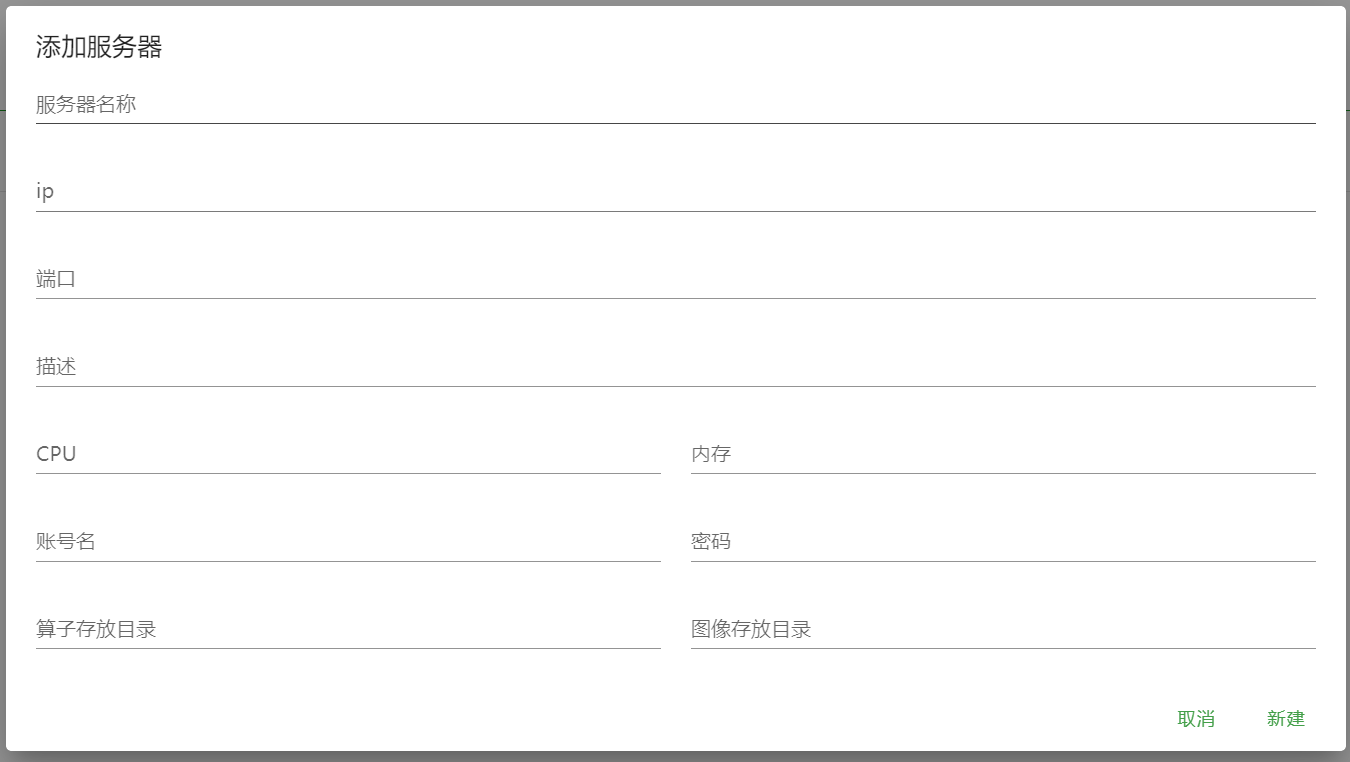


图 4.5 新建客户端效果图

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

图 4.6 新建客户端顺序图

### 可视化遥感图像处理模块

#### 算子管理模块

算子管理模块中的功能和操作涉及到对数据库中的operator表、operator\_parameter表和distribute\_wait表进行操作，包括用户对平台上的算子进行增删改查的管理和服务器维护算子在客户端上的一致性部署两个功能。

用户通过上侧导航栏的“算子管理”按钮可进入算子管理的页面，该页面会获取平台管理的所有算子信息，并进行分页显示。针对每个算子信息条目，可以点击“详情”按钮，平台会显示弹窗，显示算子的名称、运行代码、算子参数的详细信息，并可以对这些信息进行编辑修改;点击删除按钮，平台会删除该算子。在算子列表的上方，点击“新建”按钮，平台显示新建按钮的弹窗，新建算子弹窗效果图如图 4.7所示。用户在弹窗上输入信息，并上传打包好的镜像。“运行脚本”指镜像内初始文件的运行命令。“输入文件路径”和“输出文件路径”涉及到后续构建镜像的数据卷，实现容器内和主机间文件的交互。“参数”指镜像内初始文件的运行参数，会按照输入的顺序将所有参数值进行拼接，用空格分隔，作为运行脚本的参数。参数类型分为输入文件、输出文件和空，输入文件和输出文件的值为文件的绝对路径。在算子调度时，根据参数类型进行判断参数是否为文件，如果是文件，则需要相应的操作完成客户端和服务端文件的交互。用户点击“新建”按钮，前端会先进行信息校验，如果参数为空，则会提示用户将算子信息补充完整。校验成功后，前端会向后端发起新建算子的请求，后端将算子信息存储到operator表和operator\_parameter表中。



图 4.7 新建算子效果图

B/S下的服务器保证每个客户端都部署有系统上所有的镜像，保证部署的一致性。当有新的算子上传，服务端监听遥感图像数据库中算子数据的变化，收到消息，准备将目标镜像部署到平台所有客户机上，获取客户端列表，针对每个客户端生成一个算子构建服务，存储到distribute\_wait表；当有新的客户端加入平台时，服务端监听遥感图像数据库中客户端数据的变化，收到消息，准备将所有镜像部署到目标客户机上，获取算子列表，针对每个生成一个算子构建服务，存储到distribute\_wait表。服务器查询数据库的distribute\_wait表，获得待部署的镜像信息，会和目标客户端进行消息的交互，完成tar包的分发、镜像的加载，实现算子自动构建。可以通过命令行工具查看构建过程和构建成功的结果，新建算子的服务端构建输出如图 4.8所示，添加客户端的服务端构建输出如图 4.9所示。算子构建服务执行流程图如图 4.10所示。

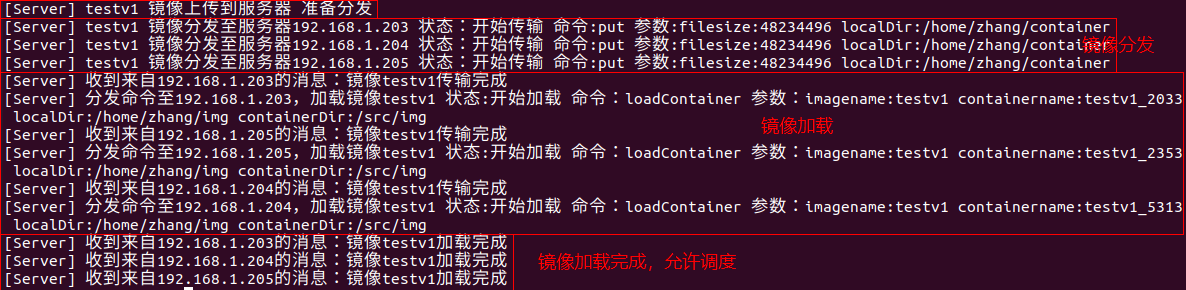


图 4.8 新建testv1镜像，所有客户端均收到消息进行构建



图 4.9 添加客户端192.168.1.205，客户端对平台上的所有镜像进行构建

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

图 4.10 执行算子构建服务流程图

算子构建服务的状态转移过程如图 4.11所示。

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

图 4.11 算子构建服务状态图

算子构建构成一共有5个正常状态和3个异常状态。正常状态包括start、file\_loading、file\_loading、msg\_running和end,是算子构建正常过程的描述。异常状态包括file\_failed、msg\_failed和failed,描述算子构建过程中因网络或者硬件等因素造成的文件传递失败和构建失败，服务器针对异常失败的情况会采取重构建的手段，防止因系统异常导致无法继续正常运行。如果失败次数过多，服务器会拒绝重构建，将构建错误的日志反馈给管理员，管理员根据报错信息检查客户端的网络环境和硬件环境，在检查确认无误后会重启构建任务。重构建也保证了客户端平台上算子的一致性部署。

#### 应用管理模块

应用管理模块中的功能和操作涉及到对数据库中的app表、DAG\_node表、node\_parameter表和DAG\_line表进行操作，基本操作有新建应用、修改应用、查看应用列表和删除应用，包括可视化编排和DAG建模数据解析两个核心功能。

可视化编排页面效果如图 4.12所示。页面首先会调用后端，获取用户上传的所有算子列表，显示在页面的左侧。用户点击算子列表中的一个算子条目，捕获鼠标按下事件onmousedown，前端会新建一个绝对定位的算子对象，该对象包含算子相关信息以及含有一个独特标识符。页面监听鼠标移动事件onmousemmove，使得该算子对象随着鼠标的拖拽而移动。页面详情的中间是一个SVG画布，鼠标拖拽算子对象到SVG画布上，画布监听到鼠标松开事件onmouseup，认为算子进入画布，在画布内生成一个相似的新对象，即DAG图的节点。画布会监听鼠标的算子鼠标点击、松开和移动事件，通过拖拽，完成节点和节点之间的连线，描述算子之间的前置后置关系，便利、快捷的进行任务编排。在连线过程中，需要对DAG图进行校验，平台对算子的参数进行解析，一个输入文件对应一个入度，一个输出文件对应一个出度，计算出每个算子的入度和出度后，节点的入度不能大于算子的入度，节点的出度不能大于算子的出度。用户进行错误的连线时，前端会出现提示，并取消错误连线。用户点击SVG画布中的每个算子，前端会获取算子的详细信息，显示在页面右侧，用户需要对各个信息进行填写。

平台将DAG图解析为由nodes和lines组成的JSON数据，JSON数据样例如图 4.13所示。nodes由多个node组成,一个节点对应一个node，保存有算子信息、节点位置、独特标识符数据。lines由多个line组成，一条连线对应一个line，包含前置节点信息、起始位置、后置节点信息和结束位置。每个node会有独特标识符，每个line利用node的独特标识符识别前置节点和后置节点。WEB服务器将JSON数据和数据库中的DAG\_node表、node\_parameter表和DAG\_line表进行映射，完成对DAG图数据的存储和读取。实现DAG图的保存和重新读取。

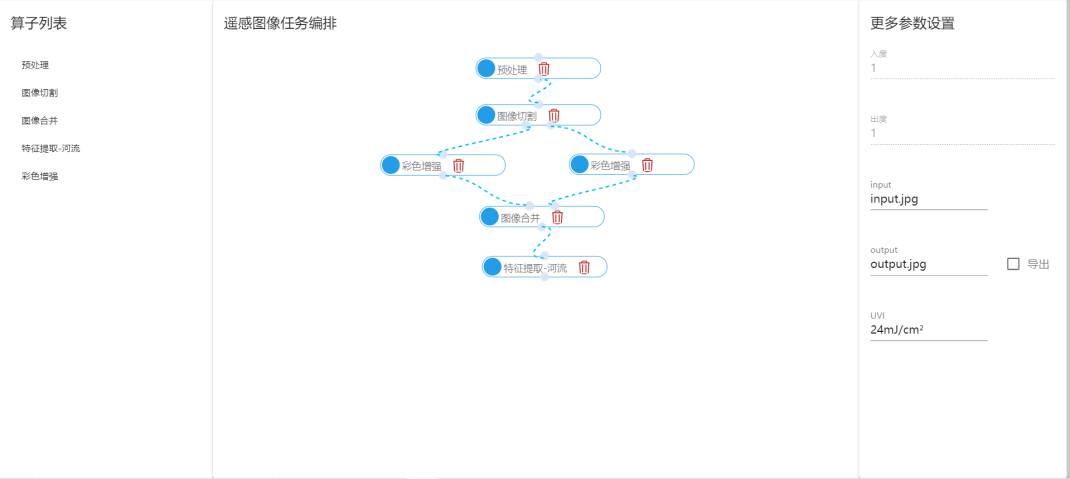


图 4.12 可视化任务编排



图 4.13 JSON数据样例展示

用户完成任务编排后，点击“保存”按钮，前端首先会完成DAG图的校验和节点参数的校验。其中DAG图的校验逻辑包括不能成环、所有节点的入度等于相应算子的入度、所有节点的出度等于相应算子的出度，节点参数的校验逻辑为不能为空。符合校验逻辑后，前端会向后端发起保存应用的请求，并根据应用填写的条件字段添加请求参数。后端在数据库中添加、修改应用的记录。利用广度优先遍历算法对DAG图数据进行解析，将DAG节点串行化，并获取每个DAG节点的后置节点，以拓扑排序的顺序，将DAG图解析数据存储到app表的DAG\_info中。后端返回保存成功的信息，前端跳转到应用管理的界面。

#### 任务管理模块

任务管理模块中的功能和操作涉及到对数据库中的task表、task\_node\_status表、task\_output表和task\_node\_wait表进行操作，基本操作有新建任务、查看任务、查看任务列表，包括任务调度和可视化任务调度两个核心功能。

用户通过上侧导航栏进入任务管理界面，前端会调用后端接口，获取任务信息列表，在此页面显示。用户选择任务列表上方的“新建”按钮，前端会显示弹窗，要求用户输入任务名称，选择目标应用，上传目标遥感图像。所有信息输入完成后，用户点击弹窗中的“新建”按钮，前端会向后端发起新建任务的请求，并根据任务填写的条件字段添加请求参数。后端在数据库中添加任务的记录。服务端监听数据库中任务信息的变化，当发现由新任务创建时，会获取新任务的目标应用的DAG\_info字段，进行解析，将DAG\_info字段中的每一条数据视为一个子任务，存储到task\_node\_status表中，并存储子任务运行状态。服务端获取客户端列表，为子任务分配客户端，结合客户端信息和子任务运行信息生成子任务调度信息，存储到task\_node\_wait表中。在task\_node\_wait表中，存储任务可执行状态的进一步划分。服务端每隔5秒会读取task\_node\_wait表，获取所有子任务调度信息，利用任务调度状态，获取可执行子任务调度信息。C/S模式下的服务端可和多个客户端进行通信，多个客户端可并发运行任务节点，实现任务内的并发和任务间的并发。子任务调度服务的顺序图如图 4.14所示。

本系统通过实现任务内子任务间的并发和任务间的并发，加快任务调度速度。如果子任务和子任务之间存在前置和后置的关系，则认为两个子任务之间存在直接依赖关系；前置子任务的前置子任务和后置子任务的后置子任务之间存在间接依赖关系。存在直接或者间接依赖关系的子任务之间存在调度时间上的先后关系；不存在直接或者间接依赖关系的子任务之间不存在调度时间上的先后关系，可以通过分配不同的客户端实现子任务间的并发，降低任务的总体执行时间。通过实现将任务和任务分配在不同的客户端上并发执行，可以降低系统的总体执行时间。

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

图 4.14 子任务调度服务顺序图

本系统通过设计节点的运行状态实现子任务的乱序调度和重调度。如图 4.15所示，节点的运行状态，分别为不可执行、可执行、正在执行、执行结束，其中可执行状态又可以进一步细分为3类。每个节点唯有在其所有前置节点变更为执行结束的状态时才可从不可执行状态变更为可执行状态。

一个节点运行结束后，会更新task\_node\_status表和task\_node\_wait表中相应节点的执行状态为执行结束，并利用task\_node\_wait表中后置节点信息，减少其后置节点的执行条件，更新遥感图像容器数据库中可执行节点的信息。服务端获取的可执行节点列表，一定具备了可执行条件，不需要判断他是否属于特定任务，不需要按照固定的顺序调度。服务端仅仅靠筛选子任务节点所在的客户端，使尽量多的客户端同时调度算子。一个节点运行过程中，会不断更新task\_node\_wait表中的调度状态，服务端可通过对可执行节点进行调度时，通过task\_node\_wait表中的调度状态判断当前执行到哪一步，进而调度下一步，而无需从头调度。同理，当客户端发生故障返回错误信息或者长期不返回时，服务端视为此子任务调度失败，通过重调度和重新分配客户端的方法使子任务恢复调度成功的状态，不影响其他节点的执行。当调度失败次数过多，服务端判定此为不可修复的故障，返回错误码，需要管理员检查调度环境。

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

图 4.15 子任务调度服务状态图

节点的执行分为3个步骤：输入文件的发送、容器执行脚本的发送、输出文件的回传，服务端步骤详情如下：

1. 输入文件的发送：服务端从task\_node\_wait表中获知哪些调度参数为输入文件，将输入文件从服务端传递到客户端的指定目录，传递完成，即可视为容器运行的条件已经完成。
2. 容器执行脚本的发送：服务端发送容器执行相关参数，并利用数据卷的对应，实现容器虚拟环境下和主机目录之间的文件传递。客户端接收到容器执行的命令和参数后，会异步执行相应的容器。
3. 输出文件的回传：在执行完成后，客户端先后向服务端发送执行完成的消息并回传输出文件。服务端接收输出文件并将文件复制、重命名至该节点的后置节点所需的输入文件相对应的目录。

图 4.16展示了预处理节点执行过程中服务端的输出。

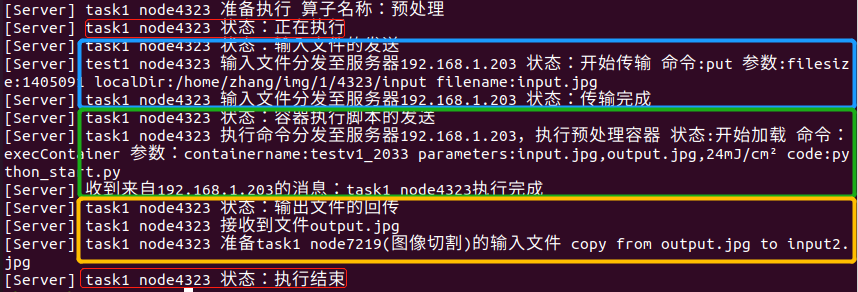


图 4.16 预处理节点执行过程下，服务端的输出

客户端会接收文件、处理命令，转去执行对应的脚本。

在任务查看界面主要是利用SVG画布对任务调度结果的可视化展示，包括子任务调度状态、子任务分配客户端的信息展示。如图 4.17所示。

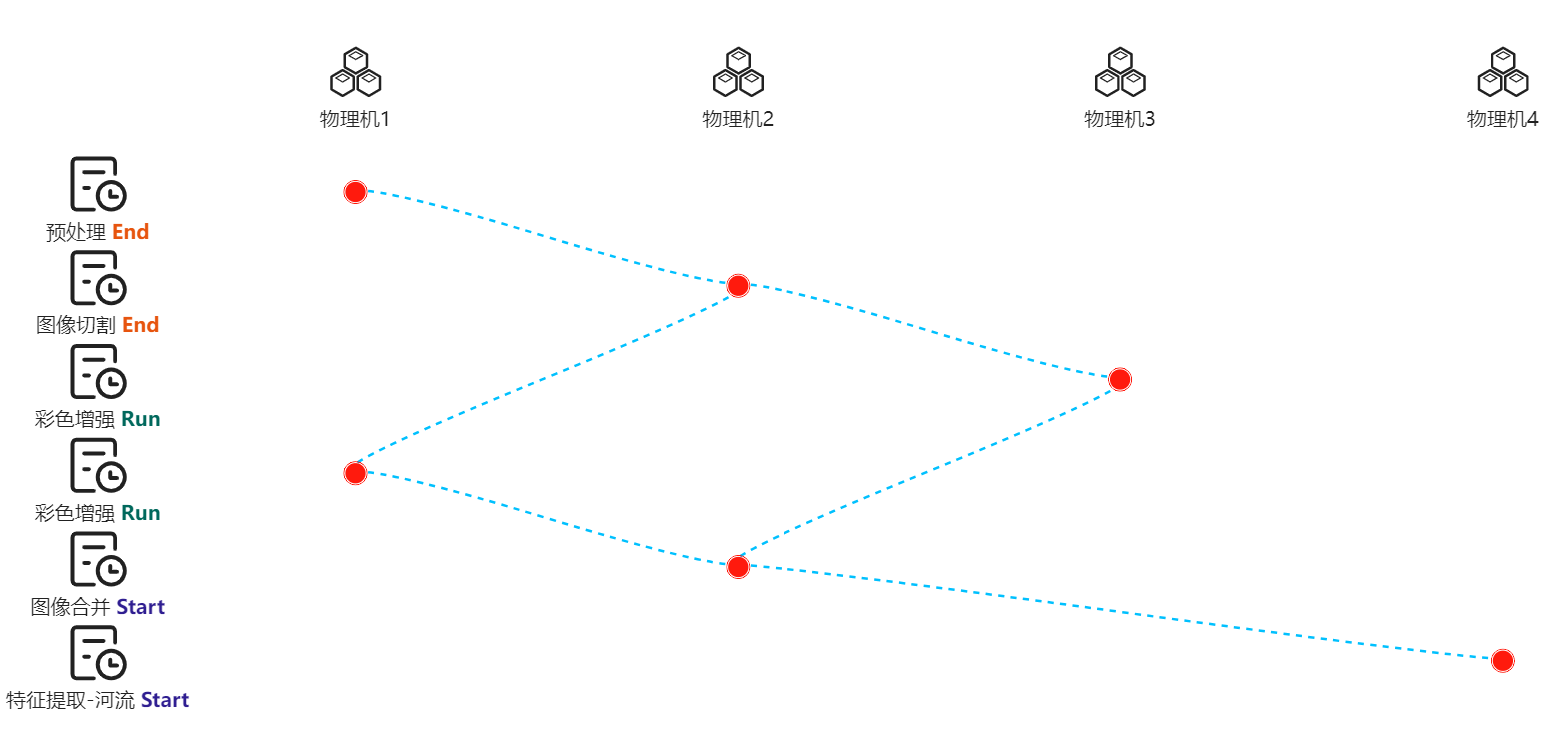


图 4.17 可视化任务调度

# 测试和评估

## 测试目标

为确保系统上线后正常运行，本章将分别测试在本文中实现的模块。本章的主要任务是从用户的角度测试系统。对所有模块功能进行完整的测试，以确保正确的数据处理。对所有功能进行全面的业务逻辑测试，以确保不会出现任何未处理的情况。

## 测试环境

当前测试环境为远程服务器-本地浏览器测试，即数据库、系统代码均部署在远程服务器上，通过本地浏览器进行公网访问。其中远程服务器的系统测试环境为：

CPU：1核

内存：2GB

外存：40GB

操作系统：Linux

数据库：MYSQL 8.0

本地浏览器的测试环境为：

CPU：Intel(R)Core(TM) i5-6300HQ CPU @2.30GHz

内存：8GB

外存：1TB

浏览器：Google Chrome

## 功能性测试

下面将以测试用例表格的形式，对模块中的每个功能点进行测试并给出结果。

### 用户管理测试

表 5.1 用户管理测试样例

| 编号 | 功能描述 | 测试步骤和期望目标 | 测试结果 |
| --- | --- | --- | --- |
| 1.1 | 登录账号 | 用户输入用户名和密码后，点击登录按钮后，用户可以进入系统。 | 达到期望目标 |
| 1.2 | 查看个人资料 | 用户点击查看个人资料按钮后，查看到自己的用户名、邮箱、手机号等个人信息页面。 | 达到期望目标 |
| 1.3 | 修改密码 | 用户点击修改密码按钮，输入两次修改后的密码，点击确认，系统修改用户密码。 | 达到期望目标 |
| 1.4 | 退出登录 | 用户点击退出登录按钮，用户退出系统，重新返回登录页面。 | 达到期望目标 |
| 1.5 | 修改用户资料 | 管理员点击修改按钮，输入修改后的用户信息，点击确认后可以看到修改后的用户详情信息。 | 达到期望目标 |
| 1.6 | 查看用户列表 | 管理员通过导航栏的“用户管理”进入用户管理界面，可以查看到用户列表信息 | 达到期望目标 |
| 1.7 | 删除用户 | 管理员选择要删除的用户条目，然后单击“删除”按钮，管理员将看到已删除的用户不再在用户列表中。 | 达到期望目标 |
| 1.8 | 创建用户 | 管理员点击新建用户按钮，输入用户的账号名、姓名、邮箱和账号备注信息，点击确认成功注册用户。 | 达到期望目标 |

### 客户端管理测试

表 5.2 客户端管理测试样例

| 编号 | 功能描述 | 测试步骤和期望目标 | 测试结果 |
| --- | --- | --- | --- |
| 2.1 | 新建客户端 | 管理员点击新建客户端按钮，输入IP、端口等信息，点击确认新建用户。 | 达到期望目标 |
| 2.2 | 查看客户端列表 | 管理员通过导航栏的“客户端管理”进入客户端管理界面，可以查看到客户端列表信息 | 达到期望目标 |

### 可视化遥感图像处理测试

#### 算子管理测试

表 5.3 算子管理测试样例

| 编号 | 功能描述 | 测试步骤和期望目标 | 测试结果 |
| --- | --- | --- | --- |
| 3.1 | 查看算子列表 | 用户通过导航栏的“算子管理”进入算子管理界面，可以查看到算子列表信息 | 达到期望目标 |
| 3.2 | 新建算子 | 用户点击新建算子按钮，输入算子名称、运行代码、参数等信息，点击确认新建算子。 | 达到期望目标 |
| 3.3 | 删除算子 | 用户选择要删除的算子条目，然后单击“删除”按钮，用户将看到已删除的算子不再在算子列表中。 | 达到期望目标 |
| 3.4 | 编辑算子 | 用户点击修改按钮，输入修改后算子名称、运行代码、参数等信息，点击确认后，可以看到修改后算子详细信息页面。 | 达到期望目标 |
| 3.5 | 解析算子 | 服务器监听到算子信息和客户端信息变化，生成算子构建服务保存到数据库 | 达到期望目标 |
| 3.6 | 部署算子 | 服务端获取算子构建服务，和客户端进行信息通信，完成算子构建 | 达到期望目标 |

#### 应用管理测试

表 5.4 应用管理测试样例

| 编号 | 功能描述 | 测试步骤和期望目标 | 测试结果 |
| --- | --- | --- | --- |
| 4.1 | 查看应用列表 | 用户通过导航栏的“应用管理”进入应用管理界面，可以查看到应用列表信息 | 达到期望目标 |
| 4.2 | 新建应用 | 用户点击新建应用按钮，拖拽算子进行可视化编排，进行连线，输入算子参数信息，点击确认新建应用。 | 达到期望目标 |
| 4.3 | 删除应用 | 用户选择要删除的应用条目，然后单击“删除”按钮，用户将看到已删除的应用不再在应用列表中。 | 达到期望目标 |
| 4.4 | 编辑应用 | 用户点击编辑按钮，可以看到已经保存好的应用信息，对编排信息和参数进行进行重新填写，点击保存后，应用保存并跳转到应用管理页面。 | 达到期望目标 |

#### 任务管理测试

表 5.5 任务管理测试用例

| 编号 | 功能描述 | 测试步骤和期望目标 | 测试结果 |
| --- | --- | --- | --- |
| 5.1 | 查看任务列表 | 用户通过导航栏的“任务管理”进入应用管理界面，可以查看到任务列表信息 | 达到期望目标 |
| 5.2 | 新建任务 | 用户点击新建任务按钮，输入目标应用、任务名称，并上传目标图像，点击确认新建任务。 | 达到期望目标 |
| 5.3 | 删除任务 | 用户选择要删除的任务条目，然后单击“删除”按钮，用户将看到已删除的任务不再在任务列表中。 | 达到期望目标 |
| 5.4 | 查看任务 | 用户点击任务按钮，可以进入任务详情界面，实时查看子任务调度情况和当前执行情况 | 达到期望目标 |
| 5.5 | 解析任务 | 服务端监听到到数据库中新建任务的消息，解析任务，分配客户端，生成子任务调度服务存储到数据库中 | 达到期望目标 |
| 5.6 | 调度子任务 | 服务端获取子任务调度服务，和客户端进行交互，实现任务内和任务间的并发 | 达到期望目标 |

## 非功能性测试

系统的非功能性需求主要体现在系统的鲁棒性、易用性、可扩展性、安全性和性能方面。

系统的鲁棒性体现在对用户误操作的提示的和对系统内部错误的处理。管理员新建用户时填写的账号名重复，系统会提示该账号名已存在。当用户输入错误的账号名或密码登录时，系统会提示账号名不存在或密码错误。当用户输入错误的访问路由时，系统会跳转到404错误提示页，告诉用户当前访问路由不存在。用户在进行任务可视化编排过程中，容易出现不符合DAG建模规范的地方，比如成环或者不符合算子连接要求，系统会对用户的错误操作进行提示直到修改正确，才能提交应用。每个应用中的算子参数信息如果为空，会导致算子调度失败，所以当用户提交空白算子参数值时，系统也会进行提示。针对算子构建和子任务调度的过程，会收到网络和硬件等因素的干扰导致执行服务失败，本平台通过设计服务执行过程中的多状态转移，实现重构建和重调度，保证系统的正常运行。并针对无法修复的调度和构建问题，系统会返回错误码给管理员进行提示。

系统的易用性体现在构造逻辑清晰的界面和提供可视化图表分析数据。系统整体逻辑清晰，前端字体清晰、间距适当，且界面不太拥挤。本系统面向管理员和用户，由管理人员负责遥感图像处理系统云平台信息的维护，由服务端实现容器的自动调度，简化管理员和用户的操作。采用图表展示信息汇总、拖拽绘图等可视化方法，让用户快捷获取所需要的信息，进行操作。

系统的可扩展性体现在对新建客户端的兼容性。当前系统连接有5个客户端，管理员想要创建新的客户端连接，只需要获取客户端代码，修改内部的IP、端口等配置信息，在任意一台Linux系统上运行即可。客户端通过Socket通过和服务端建立连接，服务端也会自动维护客户端连接数据。对于新建的客户端，平台提供了客户端管理界面可以对客户端的信息进行后续的新建和修改。

系统的安全性体现在用户和管理员身份的鉴别和数据接口的保护。用户和管理员在未登录状态时，是无法使用系统的任何功能的，强行输入路由访问内部页面时，系统会判断当前用户的状态，如为未登录状态会被重定向到登录页，这样就能抵御一定程度的安全攻击，防止系统的各个功能被滥用，保护系统的数据接口。针对每个接口的调用，后端都会进行鉴权判断，用户如果直接调用接口进行新建客户端等操作时，后端会认定为无权限，返回无权限的错误码。

系统的性能体现在页面的响应较快、较稳定的文件传输和较优的任务并发调度。普通查询请求的响应时间均在1s以内，满足了用户使用时的基本性能需求。普通遥感图像的传递一般在2s以内，较大的镜像，通常在500MB左右，也能在2分钟内传输完成。服务端采用多线程，在传送文件的过程中，也可以和客户端进行消息通信，并尽量将任务同时分配到不同的客户端上，保持让客户端忙碌，处理信息，让服务端同时和多个客户端进行通信，降低任务调度所需要的时间，提高性能。

# 结论与展望

总结

随着遥感图像数据量的迅速增加，传统的遥感图像处理系统已经无法应用，存在工作量大且冗余、专业知识要求高、管理效能低等诸多弊病。随着云计算的快速发展，利用云强算力的特点，构建云平台可以很好的解决传统任务处理慢的特点。因传统的任务编排对计算机专业知识要求较高，云平台的运维效率比较低，提出了可视化云编排的改改，，简化编排流程，提高任务云上部署和调度的效率。

基于这种现状，本文提出了一个遥感图像处理任务可视化编排系统。由管理员负责遥感图像云平台的维护，利用边缘节点的高效性，实现任务云上部署和调度的高效性。用户在平台上可以进行面向个人的算子管理、应用管理和任务管理。在算子管理中，用户可以自定义算子信息，实现个性化设计，并由平台实现自动构建。在应用管理中，平台实现了可视化编排技术，用户可以轻松的进行遥感图像任务DAG建模，极大的简化了任务编排的流程，由系统完成DAG建模解析。在任务管理中，平台实现了自动调度和故障重调度，通过任务内和任务间的并发，极大加快任务的执行速度，并利用可视化图标让用户可以查看任务的执行过程。

本论文首先进行了需求分析，然后设计了系统架构，根据需求划分了开发的功能模块，并进行了数据库设计。在对模块进行划分后，按照模块来进行详细的设计和开发工作，最后对系统的每一个功能进行了测试。

目前系统实现了用户管理、客户端管理、可视化遥感任务处理三大模块，前后端对接成功，构建了B/S和C/S混合架构体系，基本实现了预期的效果。

本人工作内容

1. 分析需求，对后端所需模块进行划分，并进行数据库和接口设计；
2. 搭建前端框架，实现前端部分页面；
3. 实现后端大部分接口，完成用户管理、客户端管理、可视化遥感任务处理三大大模块；
4. 编写测试用例，对系统各功能点进行测试；
5. 撰写毕业论文，并准备答辩。

展望

对照本文的研究目标，当前的遥感图像处理任务可视化编排系统已经基本满足前文描述的系统功能需求和非功能需求，但是在一些方面还是有许多改进的空间：

**调度算法的优化**

本系统在进行任务调度时，会将子任务分配到不同的客户端上进行并发运行，当前调度速率还能接受，但是之后生成更多子任务和加入更多客户端时服务端调度响应时间会线性增加，因此需要考虑通过把客户端的处理任务、子任务的复杂性等因素加入调度算法中，尽量充分利用服务端和客户端的处理性能。

**镜像的管理**

用户可能会上传几GB的较大镜像，如果用户同时上传较多镜像，这会给服务端造成较大的发送文件的压力。后续考虑实现统一的文件管理系统，实现服务端和客户端之间的文件共享。

# 致谢

这个项目是我大学以来第一次全栈式开发。在开发过程中，不管是前端页面的精雕细琢，后端功能的苦苦思索，还是服务器配置部署的谨慎尝试，都凝结了我太多的心血，如今总算是圆满结束了。随着项目的完工，四年的大学生活也接近尾声。

首先，我要感谢沃天宇老师，作为我的导师，他认真负责、严谨求实。沃老师在我进行毕业设计的过程中为我提供了很多帮助，也为我提供了学习环境良好的实验室工位和实验所需的各种硬件服务器，在过去的半年里，在系统的整体设计方面，沃老师从全局的角度，给我指导，帮助我理清系统的业务流程。在论文撰写方面，沃老师也凭借着自己多年丰富的经验给出我最中肯、实用的建议。

其次要感谢实验室学长迂博和马博的帮助，开题后，我对项目的开发方向存在错误理解，对项目的落地技术更是一头雾水，还好有实验室的学长一步一步引导我，为项目的开发奠定了坚实的基础，并在论文撰写方面细心的指引，提出修改意见。

其次，我要感谢我大学四年学习生涯所遇到的所有老师。感谢你们勤勤恳恳地奋斗在教育一线，感谢你们笃学求真的治学态度，感谢你们对教育事业的责任心。你们的工作让我的知识得到扩充，你们的品质让我的精神世界更加丰富。

同时，我要感谢我大学四年所遇到所有好友和导员。感谢你们能够与我探讨学习遇到的难题，感谢你们对生活的帮助，感谢你们对我人生规划的建议。大学四年生活的每一步，都感谢你们的陪伴，是你们的帮助才能让我看得更高、攀登得更远。

最后，我要特别感谢我的父母，感谢你们这二十多年来对我的养育和教导，感谢你们点点滴滴的悉心照料，感谢你们在我学习路上的默默陪伴与支持。

参考文献

1. 常志明. 遥感图像并行算法的研究及其网格服务的实现[D].国防科学技术大学,2004.
2. 郭延辉. 高光谱遥感图像分类技术研究[D].陕西师范大学,2020.DOI:10.27292/d.cnki.gsxfu.2020.000010.
3. Aliyun. AI Earth[EB/OL]. <https://engine-aiearth.aliyun.com/#/,> 2022-04-05.
4. SenseTime. [SenseEarth[EB/OL]](https://rs.sensetime.com/" \l "tec-s-title). <https://rs.sensetime.com/#tec-s-title,> 2022-04-05.
5. 李乔,郑啸.云计算研究现状综述[J].计算机科学,2011,38(04):32-37.
6. Kivity A , Kamay Y , Laor D , et al. kvm: the Linux virtual machine monitor[C]// Linux Symposium. 2007.
7. Al I P . Xen 3.0 and the Art of Virtualization[J]. proceedings of the ottawa linux symposium, 2005.
8. Brian Goff. [CHANGELOG --rocket](https://rocket.readthedocs.io/en/latest/CHANGELOG/)[EB/OL]. <https://rocket.readthedocs.io/en/latest/CHANGELOG/>, 2022-04-01.
9. Steve Buchanan,Janaka Rangama,Ned Bellavance. Introducing Azure Kubernetes Service[M].Apress, Berkeley, CA.
10. 金子威. 基于K8S的Docker分布式容器自动化运维系统的设计与实现[D].中南民族大学,2018.
11. 赵忠明,高连如,陈东,岳安志,陈静波,刘东升,杨健,孟瑜.卫星遥感及图像处理平台发展[J].中国图象图形学报,2019,24(12):2098-2110.
12. Deepak Vohra. Docker Management Design Patterns[M]. Apress, Berkeley, CA.
13. Steve Buchanan, Janaka Rangama, Ned Bellavance. Introducing Azure Kubernetes Service[M]. Apress, Berkeley, CA.
14. The Apache Software Foundation. [Apache Mesos - Documentation Home](https://mesos.apache.org/documentation/latest/)[EB/OL]. <https://mesos.apache.org/documentation/latest/,> 2022-04-05.
15. The Apache Software Foundation. Apache Airflow Documentation[EB/OL]. <https://airflow.apache.org/docs/apache-airflow/stable/index.html>, 2022-04-05.
16. The Luigi Authors. Luigi Documentation[EB/OL]. https://luigi.readthedocs.io, 2022-04-05.
17. Argo Project Authors. [Argo Workflows](https://argoproj.github.io/workflows/)[EB/OL]. <https://argoproj.github.io/workflows/,> 2022-04-05.
18. 孙庚泽.基于Docker的混合云应用编排方案研究[D].西安电子科技大学,2015.
19. 开彩红,肖瑶,方青.基于人工蜂群算法的中继卫星任务调度研究[J].电子与信息学报,2015,37(10):2466-2474.
20. 黄科.基于微服务的虚拟机自动化编排系统的设计与实现[D].电子科技大学,2019.
21. 高畅.大数据可视化编排与展示系统的设计与实现[D].北京邮电大学,2018.
22. 郭爱春,李洪全,丁浩,商和龙.基于可视化云编排技术的智能终端业务应用[J].电工技术,2021(22):161-163.DOI:10.19768/j.cnki.dgjs.2021.22.054.
23. 姚婧,宋锐,王壮,胡卫东.空间探测相控阵雷达系统的观测任务编排仿真[J].计算机仿真,2006(10):65-68+71.
24. 刘伟权. 野外WSN的监测系统任务编排及调度的实现[D].西安工业大学,2015.
25. 廖志英,董安邦.基于C/S和B/S混合结构的管理信息系统运行模式[J].计算机工程与应用,2002(02):184-185+249.