北京理工大學

本科生毕业设计（论文）

北京理工大学本科生毕业设计中期汇报

The Mid-term Summary Report of Undergraduate Graduation Project (Thesis) of Beijing Institute of Technology

学　　院：计算机学院

专　　业：计算机科学与技术

学生姓名：傅泽

学　　号：1120192062

指导教师：陆慧梅

2023 年 4 月 20 日

原创性声明

本人郑重声明：所呈交的毕业设计（论文），是本人在指导老师的指导下独立进行研究所取得的成果。除文中已经注明引用的内容外，本文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。

特此申明。

本人签名：　　日 期：　　年　月　日

关于使用授权的声明

本人完全了解北京理工大学有关保管、使用毕业设计（论文）的规定，其中包括：①学校有权保管、并向有关部门送交本毕业设计（论文）的原件与复印件；②学校可以采用影印、缩印或其它复制手段复制并保存本毕业设计（论文）；③学校可允许本毕业设计（论文）被查阅或借阅；④学校可以学术交流为目的，复制赠送和交换本毕业设计（论文）；⑤学校可以公布本毕业设计（论文）的全部或部分内容。

本人签名：　　日 期：　　年　月　日

指导老师签名：　　日 期：　　年　月　日

北京理工大学本科生毕业设计中期汇报

摘　要

本文为笔者对北京理工大学本科生毕业设计工作的中期总结汇报。在第一节中，笔者将简要回顾毕业设计题目的内容；第二节将讨论目前笔者已经完成的工作，介绍进行这些工作的目的、推进工作的方法、遇到的困难和解决方法；第三节中，笔者将反思截至目前工作面临的挑战和困难；最后一节将介绍笔者未来的工作目标，时间安排，以及针对第三节中提出的挑战的可能解决方案。

关键词：**北京理工大学**；**本科生**；**毕业设计（论文）**，**中期汇报**

The Mid-term Summary Report of Undergraduate Graduation Project (Thesis) of Beijing Institute of Technology

Abstract

This article is a mid-term summary report of the author’s undergraduate graduation project of Beijing Institute of Technology. In the first section, the author will briefly review the content of the graduation project topic; the second section will discuss the work that the author has completed so far, introduce the purpose of doing these work, the methods of advancing the work, the difficulties encountered and the corresponding solutions; in the third section, the author will reflect on the challenges and difficulties faced by the work; The last section will introduce the author’s future work goals, time schedule, and possible solutions to the challenges raised in the third section.

Key Words: BIT; Undergraduate; Graduation Project (Thesis); Mid-term Summary Report

目　录

摘　要 I
Abstract II
第 1 章　课题简介 1
第 2 章　当前进度 2
2.1　在传统单链区块链下进行复现工作 2
2.2　部分重构优化工作 6
2.2.1　对初始化并启动节点的脚本进行优化 6
2.2.2　对监控分支节点日志的脚本进行优化 6
2.3　外语文献翻译工作 9
第 3 章　现有问题 10
第 4 章　后续工作 11
附　录 12
附录 A　创世块配置文件 genesis.json 的内容 12
附录 B　初始化和启动节点的指令 12

第 1 章 课题简介

本毕业设计项目，旨在对实验室的已有工作——基于 Go-Ethereum 实现的树状区块链——进行性能测试，验证其基于 GeoHash 物理位置划分子链的设计，在诸如出租车调度等需要根据节点物理位置提供不同服务的应用场景中能达成相较传统单链区块链更高的效率；同时，对其进行优化，包括：简化配置、部署区块链的流程，重构已有代码提升可维护性和可拓展性等；最后，探讨将树状区块链的基于 Go 语言的 Go-Ethereum 实现，通过 Substrate 工具包移植到 Rust 编程语言上的可能性，并进行部分移植工作以佐证之。

第 2 章 当前进度

自开题以来，笔者已经完成了数项工作，推进了毕业设计项目的整体进度。现将其陈述如下：

2.1 在传统单链区块链下进行复现工作

树状区块链的设计思想为：将原本单链结构的区块链，以类似字典树的形式，转换为树状多链结构；而在其中作为键决定某个叶子节点应该属于哪个分支的，就是叶子节点的 GeoHash 值。因此，笔者特地选择了需要用到 GeoHash 值作为索引，以在不同地区提供不同服务的应用场景——基于 Dapp 的出租车调度系统，作为本项目的实验场景。

在实验室的以往工作中，已经存在在单链区块链上部署并使用该系统的记录。为详细比较该系统在两种不同的区块链上的性能表现差异，笔者首先借助虚拟机，在 Ubuntu 22.04 系统下完成了在单链区块链上部署并使用该调度系统的工作，基本复现了实验复现手册中记载的结果，并形成了实验日志以便日后查阅。下将结合笔者复现实验的步骤，进行简要介绍。

一个去中心化的区块链网络，由数台计算机共同组成，这些计算机就称之为“节点”。为使出租车调度系统正常运行，笔者搭建了由一个节点组成的区块链网络，供后续实验之用。

首先，需要构建这个节点。

1. 使用附录 A 中的配置文件，初始化节点；

2. 启动节点，并指定其 rpc 端口为 8545，以便外部程序与区块链互动；

3. 待节点启动之后，记录其 enode 信息。

一旦节点启动，终端中就会出现 JavaScript 控制台。此时，可以使用 Go-Ethereum 的官方文档中记录的各种指令，进行账户创建、解锁账户、启停挖矿、部署或调用合约等操作，与区块链进行交互。在此，笔者使用 personal.newAccount(”123456”) 新建了 8 个密钥均为 123456 的账户，一部分账号，将在出租车调度系统中承担司机与乘客的角色。

准备好区块链网络后，笔者继续完成了出租车调度系统的部署。该调度系统分为两部分，其中一部分是运行在区块链上的 DApp，以智能合约的形式存在；另一部分是运行在本地浏览器中的客户端，是一套由 Vue 2 编写的 Web GUI。本节中，笔者完成了智能合约的部署，并根据返回信息，修改了 Web GUI 中的部分参数，实现了系统的运行。

该系统一共需要部署两份合约。笔者将以其中一份 StoreMap.sol 为例，讲解合约部署的步骤。该合约为实验室已有工作的副本，存放于笔者实验的代码仓库中。使用 Remix IDE 在线开发环境对其进行编译后，可获得 ABI（形如 JavaScript 的列表）和字节码（代表一个十六进制数的字符串）。将获得的上述编译结果，复制到以下部署代码中：

```
abi = JSON.parse("压缩转义的ABI，可借助在线工具完成压缩转移步骤")
bytecode = "字符串形式的字节码"

StoreMapContract = web3.eth.contract(abi);
web3.eth.estimateGas({data: bytecode})
StoreMap = StoreMapContract.new({
    from: web3.eth.accounts[0],
    data: bytecode,
    gas: '3000000',
    position:"w2511111111111",
    txtime:277001
  },function (e, contract){
    console.log(e, contract);
    if(!e){
        if(!contract.address) {
            console.log("Contract transaction send: TransactionHash: " +
    contract.transactionHash + " waiting to be mined...");
        } else {
            console.log("Contract mined! Address: " + contract.address);
            console.log(contract);
        }
    }
});
```

代码 2.1: 合约部署代码

复制到节点的 JavaScript 控制台中，连续按压数次回车后，使用 miner.start(1) 开

始挖矿，注意输出，直到出现如下的合约地址：

```
null [object Object]
Contract mined! Address: 0xef00ade84bb560afe4b562bfd4a81300c17ac52f
[object Object]
```

代码 2.2: 挖矿输出合约地址

此时可以执行 miner.stop() 停止挖矿，并妥善保存好该合约的地址。

经过类似步骤，可以将另一份合约 StoreTraffic.sol 一同部署到区块链上，同样记录好合约地址。

合约部署结束后，按照实验室已有的手册进行操作，对客户端系统进行配置，将其中涉及到合约地址和账户公钥的代码更改为实际的合约地址和账户公钥，并上传地图文件，启动调度系统，即可观察到其运行效果。如下展示的分别是司机选择是否接单，和乘客到达目的地时系统的提示：

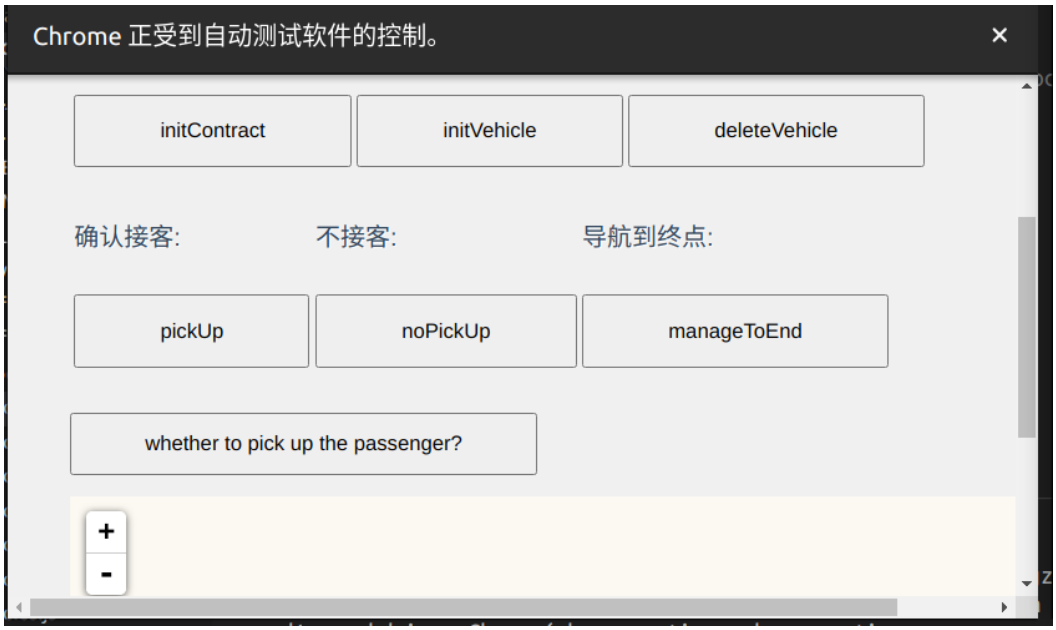

图 2-1 司机接客

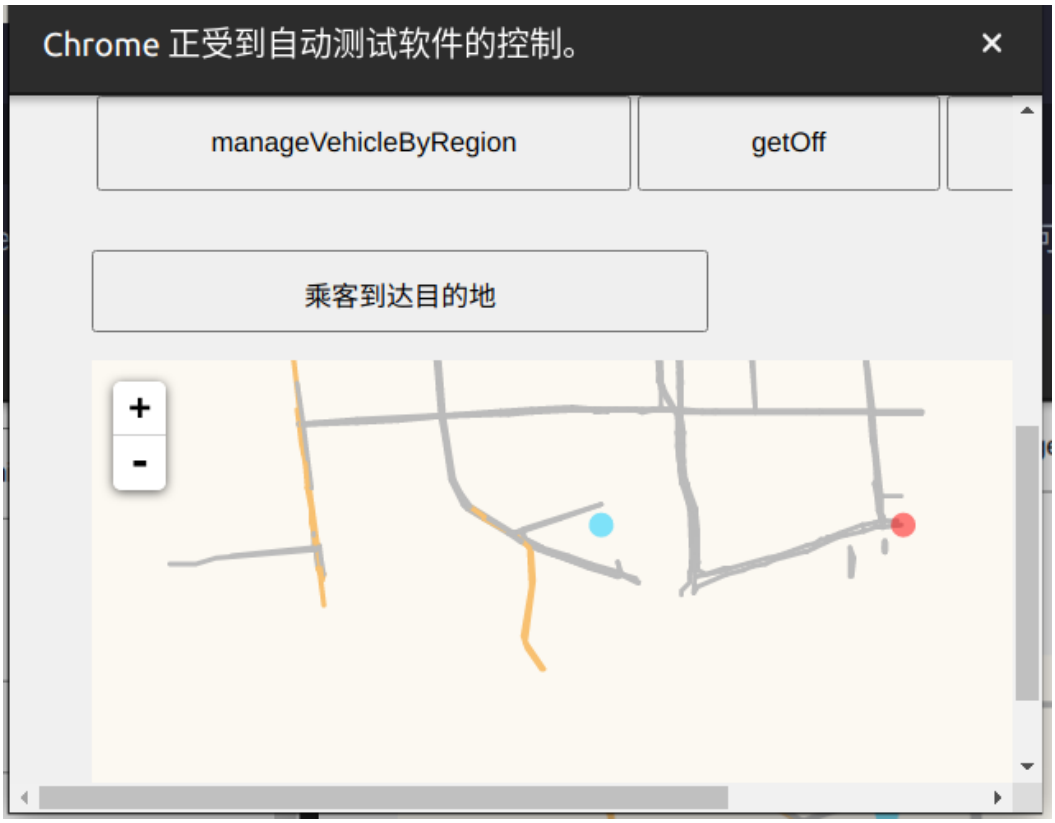
Chrome 正受到自动测试软件的控制。
manageVehicleByRegion
getOff
乘客到达目的地
+
-

图 2-2 乘客抵达

2.2 部分重构优化工作

为简化树状区块链的构建流程，实验室已有使用 Bash 脚本等工具替代人工手动输入代码，实现树状区块链初始化、添加对等节点等操作。然而，许多实验性质的脚本并不具备可复用性，异或是在使用中出现了意想不到的异常行为。为此，笔者对已有的脚本进行了一些重构和优化，并摘取其中几例加以说明。

2.2.1 对初始化并启动节点的脚本进行优化

若令一些初始化并启动节点的脚本再运行第二次，可能导致其直接崩溃，无法启动节点。根据其报错信息，笔者断定该问题如下：每次启动节点，脚本都将调用一个 JavaScript 预加载脚本，以设置分支区块。然而，该设置分支区块的过程并不能重复执行，并且脚本并未对是否已设置过分支区块进行检查。综上，笔者修改了预加载脚本的内容为：

```
// eth.setBranchBlock({from:eth.accounts[0],branchid:"w1",settime:10})  // 此行为原先的预加载脚本
if (eth.getBranchBlockByRegion("w1") === null) {
    // 若该分支区块未设置，则设置分支区块
    eth.setBranchBlock({ from: eth.accounts[0], branchid: "w1", settime: 10 })
} else {
    // 否则，执行对子叶子区块的访问操作以建立连接
    eth.getBranchBlockByRegion("w11")
    eth.getBranchBlockByRegion("w12")
}
```

代码 2.3: 修改后的预加载脚本

经过以上改进，脚本不再崩溃，并且能够确保每次运行均能正常启动节点。

2.2.2 对监控分支节点日志的脚本进行优化

在树状区块链运行时，分支节点会持续将输出信息到日志文件中，通过读取该日志文件即可得知分支节点接收到的交易双方、交易明细等信息，再利用这些信息进行一些操作。然而，随着分支节点运行，日志内容也会不断变多，因此，需要监控该日志文件的脚本每次读取均从新增添的行开始，而非每次都从第一行读取。实验室已有的脚本并不具备该功能，笔者经过研究，最终编写出如下代码，解决了这个问题：

```
"use strict";

const fs = require('fs');
const lineReader = require('line-reader');

const transferjs = require("./transfer_test2");

const filename = "../result/log_w1";
var trans_acc = "", trans_outchain = "", trans_acc_old = "",
  trans_outchain_old = "";

function main() {
    // 每次读之前都把日志文件清空
    fs.open(filename, "w", (_err, _fd) => {})

    let linePointer = -1

    fs.watchFile(
        filename,
        { persistent: true, interval: 1000 },
        (currentFileStatus, previousFileStatus) => {
            if (currentFileStatus.mtime > previousFileStatus.mtime) {
                let tempCounter = 0
                lineReader.eachLine(filename, (line, isLast) => {
                    if (tempCounter > linePointer) {
                        // 读取到新行，处理逻辑
                        console.log(line)

                        if (line === "--handler-TX_request--") {
                            console.log("Received handler-TX_request\n")
                        } else if (line.startsWith("***---from:")) {
                            line = line.slice(11, line.length - 6)
                            trans_acc = line.toString();
                            console.log("trans_account = " + trans_acc)
                        } else if (line.startsWith("***---outchain:")) {
                            line = line.slice(15, 29)
                            while (line[0] === 'a') {
                                line = line.slice(1,)
                            }
```

```
                            trans_outchain = line.toString()
                            console.log("outchain(target?) = " +
  trans_outchain)
                            if (trans_acc != trans_acc_old &&
  trans_outchain_old != trans_outchain) {
                                trans_acc_old = trans_acc;
                                trans_outchain_old = trans_outchain;
                                transferjs.get_outchain_info(trans_acc,
  trans_outchain)
                            }
                        }
                    }
                    linePointer = Math.max(tempCounter, linePointer)
                    tempCounter++
                })
                // console.log("linePointer = ", linePointer)
            }
        }
    )
}

main()

// 以下注释内容为实验室已有的日志监控脚本内容
// fs.watchFile(fn, { persistent: true, interval: 500 },
//     function (curr, prev) {
//         if (curr.mtime > prev.mtime) {
//             //文件内容有变化，那么通知相应的进程可以执行相关操作。例如读物
  文件写入数据库等
//             // console.log("1--counter1:",counter1)
//             // console.log("1--counter2:",counter2)
//             counter1 = counter2;
//             counter2 = 0;
//             read_file();
//         }
//     }
// )

// //读文件
```

```
// function read_file() {
//     lineReader.eachLine(fn, function (line, last) {
//         counter2++;
//         if (counter2 > counter1) {
//             // console.log("2--counter1:",counter1)
//             // console.log("2--counter2:",counter2)
//             if (line.toString() === '--handler-TX_request--') {
//                 console.log("--get--handler-TX_request--\n")
//             }
//             if (line.slice(0, 11).toString() === '***---from:') {
//                 line = line.slice(11, line.length - 6)
//                 trans_acc = line.toString();
//                 console.log("trans_acc:" + trans_acc)
//             }
//             if (line.slice(0, 15).toString() === '***---outchain:') {
//                 line = line.slice(15, 29)
//                 while (line.slice(0, 1).toString() === 'a') {
//                     line = line.slice(1,)
//                 }
//                 trans_outchain = line.toString()
//                 console.log("outchain:" + trans_outchain)
//                 if (trans_acc != trans_acc_old && trans_outchain_old !=
  trans_outchain) {
//                     trans_acc_old = trans_acc;
//                     trans_outchain_old = trans_outchain;
//                     transferjs.get_outchain_info(trans_acc, trans_outchain)
//                 }
//             }
//         }
//     });
// }
```

代码 2.4: 修改后的监控日志脚本

2.3 外语文献翻译工作

由于需要考察将基于 Go-Ethereum 实现的树状多链移植到 Substrate 上的可行性，笔者选择了翻译 Substrate 的英文官方文档，并完成了 5000 词的额定工作量。译文和对应的原文已上传至北京理工大学毕业设计管理系统。

第 3 章 现有问题

截至目前，笔者将进行工作历程中遇到的困难和挑战归结为以下几点：

1. 思维模式转变不及时，未能很好地从中心化的普通网络式思维切换到去中心化的区块链式思维

2. 对一些难度较大的工作具有畏难情绪，（例如阅读树状区块链的实现源代码），导致进度不及预期；

3. 时间安排不够妥当，对于独立的各项事务应当并行安排工作时间，以节省总时间；

4. 沟通不及时，和实验室中有工作内容相关的同学、前辈、老师沟通不够顺畅，导致获取、提供帮助，和同步进展不够及时；

第 4 章 后续工作

如上文所述，截至目前，笔者已经完成了出租车调度系统在普通的物理位置区块链上的复现工作、一部分代码的重构和优化工作以及外文文献翻译工作。为完成任务书中的全部任务，在后续工作中需要完成以下任务：

1. 使用两个子链和一个父链构建区块链网络，并于其上完成跨链转账实验，以获取转账的时间开销数据
2. 将出租车调度系统移植到树状区块链上，验证其可用性，并在单子链情况和一个父链四个子链的分区情况下进行出租车调度系统对比实验，对比二者在响应司乘请求的耗时、链上数据量和吞吐量的异同
3. 调研 Substrate 这一新兴的区块链工具组，了解利用它构建区块链，并于其上部署并调用智能合约的方法，并将出租车调度系统所用的部分合约移植到一个测试链上，验证调度系统移植的可行性

结合目前的工作状况和待完成的任务情况，笔者如此安排日后的工作：

表 4-1 统计表

预计完成日期	工作内容
4 月 10 日	完成双账号资产转移实验，并收集实验数据
4 月 13 日	完成更大规模的资产转移实验，并收集实验数据
4 月 19 日	在树状区块链的单子链上运行出租车调度系统，并收集数据；同时，进行 Substrate 调研
4 月 30 日	完成树状区块链上的单父链四子链的出租车调度实验，并收集数据；同时，完成调度系统的合约从 Solidity 翻译到 ink! 的工作
5 月 20 日	撰写毕业论文

附　录

附录 A　创世块配置文件 genesis.json 的内容

```
{
  "config": {
      "chainId": 7,
      "homesteadBlock": 0,
      "eip150Block": 0,
      "eip155Block": 0,
      "eip158Block": 0,
      "byzantiumBlock": 0,
      "constantinopleBlock": 0,
      "petersburgBlock": 0
  },
  "alloc": {},
  "coinbase": "0x0000000000000000000000000000000000000000",
  "difficulty": "0x20000",
  "extraData": "",
  "gasLimit": "0xffffffff",
  "nonce": "0x0000000000000042",
  "mixhash": "0
  x0000000000000000000000000000000000000000000000000000000000000000",
  "parentHash": "0
  x0000000000000000000000000000000000000000000000000000000000000000",
  "timestamp": "0x00"
}

```

代码 A.1: genesis.json 的内容

附录 B　初始化和启动节点的指令

```
geth1 --identity "MyEth" --rpc --rpcaddr 127.0.0.1  --rpcport "8545" --
  rpccorsdomain "*" --datadir gethdata --port "30303" --nodiscover --rpcapi "
  eth,net,personal,web3" --networkid 91036 init genesis.json
```

```
2
```

代码 B.2: 初始化节点的指令

```
1 geth1 --datadir ./gethdata --networkid 91036 --port 30303 --rpc --rpcaddr
    127.0.0.1 --rpcport 8545 --rpcapi 'personal,net,eth,web3,admin' --
    rpccorsdomain='*' --ws --wsaddr='localhost' --wsport 8546 --wsorigins='*' --
    wsapi 'personal,net,eth,web3,admin' --nodiscover --allow-insecure-unlock --
    dev.period 1 --syncmode='full' console
2
```

代码 B.3: 启动节点的指令