

**本科生毕业设计(论文)**

# 中期报告

**树状区块链性能测试及基于Rust的重写**

**Tree-Like Blockchain Performance Tests and Improvements**

|  |  |
| --- | --- |
| 学院： | 计算机学院 |
| 专业： | 计算机科学与技术 |
| 班级： | 07111905 |
| 学生姓名： | 傅泽 |
| 学号： | 1120192062 |
| 指导教师： | 陆慧梅 |

|  |
| --- |
| **一、毕业设计（论文）主要研究内容、进展情况及取得成果**  **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。 2北京理工大学本科生毕业设计（论文） 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： 1 abi = JSON.parse("压 缩 转 义 的ABI， 可 借 助 在 线 工 具 完 成 压 缩 转 移 步 骤") 2 bytecode = "字 符 串 形 式 的 字 节 码" 3 4 StoreMapContract = web3.eth.contract(abi); 5 web3.eth.estimateGas({data: bytecode}) 6 StoreMap = StoreMapContract.new({ 7 from: web3.eth.accounts[0], 8 data: bytecode , 9 gas: '3000000', 10 position:"w2511111111111", 11 txtime:277001 12 },function (e, contract){ 13 console.log(e, contract); 14 if(!e){ 15 if(!contract.address) { 16 console.log("Contract transaction send: TransactionHash: " + contract.transactionHash + " waiting to be mined..."); 17 } else { 18 console.log("Contract mined! Address: " + contract.address); 19 console.log(contract); 20 } 21 } 22 }); 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 3北京理工大学本科生毕业设计（论文） 始挖矿，注意输出，直到出现如下的合约地址： 1 null [object Object] 2 Contract mined! Address: 0xef00ade84bb560afe4b562bfd4a81300c17ac52f 3 [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 4北京理工大学本科生毕业设计（论文） 图 2-2 乘客抵达 5北京理工大学本科生毕业设计（论文） **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： 1 // eth.setBranchBlock({from:eth.accounts[0],branchid:"w1",settime:10}) // 此 行 为 原 先 的 预 加 载 脚 本 2 if (eth.getBranchBlockByRegion("w1") === null) { 3 // 若 该 分 支 区 块 未 设 置， 则 设 置 分 支 区 块 4 eth.setBranchBlock({ from: eth.accounts[0], branchid: "w1", settime: 10 }) 5 } else { 6 // 否 则， 执 行 对 子 叶 子 区 块 的 访 问 操 作 以 建 立 连 接 7 eth.getBranchBlockByRegion("w11") 8 eth.getBranchBlockByRegion("w12") 9 } 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： 6北京理工大学本科生毕业设计（论文） 1 "use strict"; 2 3 const fs = require('fs'); 4 const lineReader = require('line-reader '); 5 6 const transferjs = require("./transfer\_test2"); 7 8 const filename = "../result/log\_w1"; 9 var trans\_acc = "", trans\_outchain = "", trans\_acc\_old = "", trans\_outchain\_old = ""; 10 11 function main() { 12 // 每 次 读 之 前 都 把 日 志 文 件 清 空 13 fs.open(filename , "w", (\_err, \_fd) => {}) 14 15 let linePointer = -1 16 17 fs.watchFile( 18 filename , 19 { persistent: true, interval: 1000 }, 20 (currentFileStatus , previousFileStatus) => { 21 if (currentFileStatus.mtime > previousFileStatus.mtime) { 22 let tempCounter = 0 23 lineReader.eachLine(filename , (line, isLast) => { 24 if (tempCounter > linePointer) { 25 // 读 取 到 新 行， 处 理 逻 辑 26 console.log(line) 27 28 if (line === "--handler -TX\_request --") { 29 console.log("Received handler -TX\_request\n") 30 } else if (line.startsWith("\*\*\*---from:")) { 31 line = line.slice(11, line.length - 6) 32 trans\_acc = line.toString(); 33 console.log("trans\_account = " + trans\_acc) 34 } else if (line.startsWith("\*\*\*---outchain:")) { 35 line = line.slice(15, 29) 36 while (line[0] === 'a') { 37 line = line.slice(1,) 38 } 7北京理工大学本科生毕业设计（论文） 39 trans\_outchain = line.toString() 40 console.log("outchain(target?) = " + trans\_outchain) 41 if (trans\_acc != trans\_acc\_old && trans\_outchain\_old != trans\_outchain) { 42 trans\_acc\_old = trans\_acc; 43 trans\_outchain\_old = trans\_outchain; 44 transferjs.get\_outchain\_info(trans\_acc , trans\_outchain) 45 } 46 } 47 } 48 linePointer = Math.max(tempCounter , linePointer) 49 tempCounter++ 50 }) 51 // console.log("linePointer = ", linePointer) 52 } 53 } 54 ) 55 } 56 57 main() 58 59 // 以 下 注 释 内 容 为 实 验 室 已 有 的 日 志 监 控 脚 本 内 容 60 // fs.watchFile(fn, { persistent: true, interval: 500 }, 61 // function (curr, prev) { 62 // if (curr.mtime > prev.mtime) { 63 // //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 64 // // console.log("1--counter1:",counter1) 65 // // console.log("1--counter2:",counter2) 66 // counter1 = counter2; 67 // counter2 = 0; 68 // read\_file(); 69 // } 70 // } 71 // ) 72 73 // //读 文 件 8北京理工大学本科生毕业设计（论文） 74 // function read\_file() { 75 // lineReader.eachLine(fn, function (line, last) { 76 // counter2++; 77 // if (counter2 > counter1) { 78 // // console.log("2--counter1:",counter1) 79 // // console.log("2--counter2:",counter2) 80 // if (line.toString() === '--handler -TX\_request --') { 81 // console.log("--get--handler -TX\_request --\n") 82 // } 83 // if (line.slice(0, 11).toString() === '\*\*\*---from:') { 84 // line = line.slice(11, line.length - 6) 85 // trans\_acc = line.toString(); 86 // console.log("trans\_acc:" + trans\_acc) 87 // } 88 // if (line.slice(0, 15).toString() === '\*\*\*---outchain:') { 89 // line = line.slice(15, 29) 90 // while (line.slice(0, 1).toString() === 'a') { 91 // line = line.slice(1,) 92 // } 93 // trans\_outchain = line.toString() 94 // console.log("outchain:" + trans\_outchain) 95 // if (trans\_acc != trans\_acc\_old && trans\_outchain\_old != trans\_outchain) { 96 // trans\_acc\_old = trans\_acc; 97 // trans\_outchain\_old = trans\_outchain; 98 // transferjs.get\_outchain\_info(trans\_acc , trans\_outchain) 99 // } 100 // } 101 // } 102 // }); 103 // } 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。 |
| **二、存在的问题和拟解决方案**  **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； |
| **三、下一步研究任务与进度安排**  **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 |
| **四、指导教师意见**  傅泽同学按照毕设的工作安排有序地推进论文工作。后续希望更加有效地完成剩余任务。  **签字：**  2023年4月22日 |
| **成绩： ，占比：0.00%** |
| **五、中期审核负责人意见**  签字：  年 月 日 |