文章编号: 1000-5641(2018)05-0144-10

基于区块链的仓单管理系统

齐学成[1,2], 朱燕超[1,2], 邵奇峰[1,2], 张 召[1,2], 金澈清[1,2]

(1. 华东师范大学 数据科学与工程学院, 上海 200062;
2. 华东师范大学—欧冶产业互联网大数据与区块链实验室, 上海 200062)

摘要: 在当前的电子仓单业务中, 仓单真实性需要第三方机构背书. 但机构失信导致重复质押事件时常发生, 这给国家造成了巨大损失; 而且数据采用集中管理方式, 不公开, 商品溯源困难. 为了解决这两个问题, 利用区块链系统高度透明、去中心化、去信任化、不可篡改的特点, 设计实现了基于区块链的仓单管理系统, 确保了标的仓单的准确性和真实性. 在此基础上, 在区块链系统上构建了倒排索引, 提高了查询效率, 且支持复杂查询; 同时, 实现了基于表述性状态传递(Representational State Transfer, REST) 的微服务架构, 为多方接入提供了灵活接口, 也为企业已有系统的集成及Web端、移动端的实现提供了支持.

关键词: 区块链; 电子仓单; 超级账本
中图分类号: TP932 **文献标志码**: A **DOI:** 10.3969/j.issn.1000-5641.2018.05.012

A warehouse receipt management system based on blockchain technology

QI Xue-cheng[1,2], ZHU Yan-chao[1,2], SHAO Qi-feng[1,2], ZHANG Zhao[1,2], JIN Che-qing[1,2]

(1. *School of Data Science and Engineering, East China Normal University, Shanghai* 200062*, China;*
2. *ECNU-Ouyeel Joint Laboratory on Big Data and Blockchain for Industrial Internet, Shanghai* 200062*, China*)

Abstract: Currently, the authenticity of warehouse receipts requires endorsement by a third party in the warehouse receipt e-business. However, fraud cases where receipts are repeatedly pledged often occur, causing huge losses to the country. The data, moreover, is often centrally managed and unavailable to the public, making it difficult to trace records for goods. To solve these problems, this paper designs and implements a warehouse receipt management system, based on blockchain technology, which offers a high degree of transparency, decentralization, trust, and an unchangeable historical record. The system can ensure the accuracy and authenticity of the warehouse receipt. The paper builds an

收稿日期: 2018-07-04
基金项目: 国家863计划项目(2015AA015307); 国家自然科学基金(61402180, 61432006, 61672232, 61370101, 61532021, U1501252, U1401256)
第一作者: 齐学成, 男, 博士研究生, 研究方向为区块链. E-mail: qixuecheng@gmail.com.
通信作者: 张 召, 女, 教授, 博士生导师, 研究方向为区块链系统研发、海量数据管理与分析. E-mail: zhzhang@dase.ecnu.edu.cn.

inverted index structure on the blockchain system to improve the efficiency of queries and also support complex queries. The paper proposes a REST(Representational State Transfer)-based service architecture, which provides a flexible, multiple access interface, making it easy to integrate with existing systems and provide support for the development of web and mobile applications.

Keywords: blockchain technology; warehouse receipt; Hyperledger

0 引　　言

随着互联网数字经济的快速发展, 资产数字化越来越受到关注. 仓储物作为实物资产的重要组成, 往往依托电子仓单交易实现资产数字化. 仓单是保管人收到仓储物后给存货人开付的凭证, 除作为提取仓储物的凭证外, 还可以通过信任背书, 转让仓单项下货物的所有权. 但是在当前的仓单业务中, 往往需要第三方机构背书, 这种方式通常会增加交易的成本, 降低效率; 且采用传统的集中式数据存储方式, 数据安全难以保证. 同时, 行业内存在背书机构信任不足、货权交易频繁带来的溯源困难等诸多行业痛点.

区块链的出现, 为上述问题的解决带来了可能. 2008 年, 署名为中本聪的学者在一个密码学讨论组上发布了一篇名为《比特币: 一种点对点式的电子现金系统》的论文, 文中首次提出了区块链的概念, 并以此为基础设计了一种不需要第三方中介机构(如银行、支付平台等)参与的自组织、去中心化、去信任化的电子现金系统[1]. 2013 年, Buterin 发表了以太坊白皮书[2], 支持可编程的智能合约, 区块链进入 2.0 时代. 2014 年, R3 区块链联盟成立, 致力于制定银行业的区块链行业标准与协议[3]. 2015 年 Linux Foundation 发起了 Hyperledger Project[4], 目的是要共同建立并维系一个跨产业的、开放的、分布式账本技术平台. 区块链是一种按照时间顺序将数据区块以顺序相连的方式组合成的一种链式数据结构, 并以密码学方式保证的不可篡改和不可伪造的分布式账本[5]. 区块链通常分为数据层、共识层、合约层和网络层. 数据层采用块链式数据结构来存储和验证数据; 共识层采用如 POW(Proof of Work)、POS(Proof of Stake)、PBFT(Practical Byzantine Foult Tolerance) 等共识算法来保证节点数据的一致性; 合约层采用自动化脚本构成智能合约来控制交易过程; 网络层采用采用 P2P(Peer to Peer) 网络来组织分散的节点, 并利用密码学的方式保证数据传输和访问的安全. 总地来说, 区块链本质上是一种基于 P2P 网络的去中心化、去信任化、不可篡改的分布式数据库.

本文利用区块链高度透明、去核心、集体维护、去信任、不可更改和伪造的特点, 设计实现了基于区块链的仓单管理系统, 有效地解决了传统仓单业务中数据集中式管理方式带来的问题, 同时保证了数据的可追溯和不可篡改. 考虑到区块链中状态数据使用的是 Key-Value 存储方式, 本文在系统中构建了倒排索引, 以提高非主键值的查询效率, 且支持复杂查询. 同时, 本文系统实现了基于 REST 的微服务架构, 为多方接入提供了灵活接口, 也为企业已有系统的集成及 Web 端、移动端的实现提供了友好支持.

1 区块链技术

区块链是一种按照时间顺序将数据区块以顺序相连的方式组合成的一种链式数据结构, 如图 1 所示. 由图 1 可知, 区块由区块头和交易日志两部分构成, 其中区块头中会记录前一个区块(父区块)的哈希值, 区块与区块间通过哈希链连接起来, 这样把每个区块连接到各自

父区块的哈希值序列就是一条一直可以追溯到第一个区块(创世区块)的链条. 可以看出, 区块链上任何一个区块的更改, 都会影响到该区块的所有后续区块, 当一个区块后链有很多区块后, 这样的重新计算需要耗费巨大的计算量, 也就保证了区块链数据的不可篡改性. 节点间依靠 P2P 网络连接, 并运用共识协议保证不同节点间数据一致. 例如比特币系统, 采用 POW 算法, 主要原理是让全网节点解密哈希难题, 最先算出该问题的节点作为“优胜者”, 可以获得一定数量的比特币作为奖赏, 同时获得优先记账权, 将网络上发生的交易优先记录到本地的区块上; 之后, 再把区块信息同步到所有其他全节点上; 其他节点在收到“优胜者”节点发出的区块信息并且验证通过后, 就直接进行下一次“挖矿”, 这也就保证了每个全节点上的数据一致性.

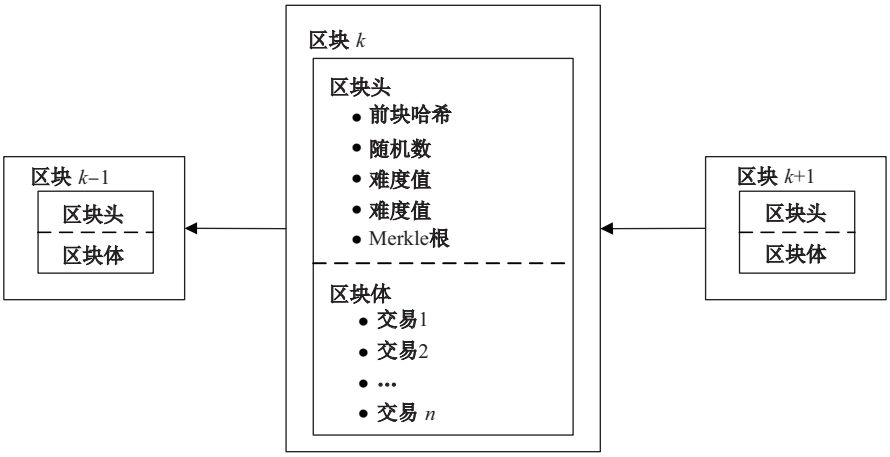

图 1 区块结构

Fig. 1 Blocks structure

2 系统设计

2.1 区块链开发方案的选择

通过前期广泛的调研, 并结合仓单业务的需求, 本文主要了解了以下 3 种开源的区块链平台.

超级账本 Hyperledger: Linux 基金会 2015 年发起的区块链开源项目, 提供分布式账本平台, 支持各种增值解决方案的开发, 以智能合约为中心, 商用情形考虑比较周到. Hyperledger 适合联盟链、私有链的构建, 平台设计了独立的 CA(Certificate Authority) 节点, 支持完善的权限管理机制.

以太坊 Ethereum: 开源的区块链底层平台, 支持可编程的智能合约. 开发人员通过编写符合业务需求的智能合约就能够在以太坊平台上开发和发布分布式应用. 以太坊自身实现了一套图灵完备的脚本语言 EVM (Ethereum Virtual Machinecode) 语言, 用户使用高级语言编程, 再通过编译器转换成 EVM 语言运行在 EVM 虚拟机上. 以太坊还提供了丰富的 API(Application Programming Interface) 和接口, 且在算法、智能合约和账本扩展性有较大改善.

瑞波 Ripple[6]: 世界上第一个开放的支付网络, 瞄准细分应用场景, 银行间的直接转账、外汇兑换和跨境支付结算. 目前未考虑智能合约、隐私和监管支持.

目前区块链按照节点组织方式分为公有链、联盟链、私有链. 考虑到仓单业务场景中,

参与成员多、业务较复杂、货物价值大等特点, 比较符合联盟链的组织方式, 即只有获得授权的成员方可加入网络. 且信息选择性公开. 同时, 仓单管理系统还需要较高效的系统性能、良好的安全保证和严格的权限管理. 在上述开源平台中, 以太坊开发的应用大多属于公有链, 任意用户都可自由加入网络. Ripple 目前只用于支付应用, 但是不支持智能合约, 系统不具有良好的拓展性. Hyperledger 平台以智能合约为中心, 设计了独立的 CA 节点, 能够进行严格的权限管理. 所以, 针对上述区块链平台的特点和优势, 结合仓单业务场景和需求, 本文选用Hyperledger平台进行区块链系统开发.

2.2　设计目标

在仓单业务中, 往往存在仓库、平台、监管、物流、银行等多方参与, 业务逻辑比较复杂. 当不同成员需要进行数据共享时, 传统解决方案一般是在各方间搭建数据访问中间件来进行数据互换, 这就带来了“一账多记”的问题, 即相同的记录分别被记在多个参与方的账本上. 这种方式很容易造成数据的不一致性, 引起争端, 并使得多方数据共享在传统的解决方案中难以实现, 数据追溯也变为不可能. 但是在仓单业务中, 高效、安全多方接入是必须, 同时仓单作为保管人收到仓储物后给存货人开付的凭证, 需要频繁地在金融交易中进行流通, 可追溯性会变得尤为重要. 因此, 考虑到上述问题和需求, 本文提出以下设计目标.

(1) 系统对多方接入提供友好支持, 各方共同维护同一本公共账本.

(2) 采用严格的权限管理机制, 通过非对称加密方式验证成员身份, 且分配其相应权限.

(3) 提供良好的性能, 及时响应请求, 满足仓单业务的性能需求.

(4) 提供丰富的 REST API 接口, 为多方接入提供灵活接口, 支持企业已有系统的集成及 Web 端、移动端的接入.

2.3　业务流程

业务流程如图 2 所示, 货主可以根据需求提出仓单的申请, 但必须在货主仓储物资产价值内. 仓库审核仓单申请所填的仓储资产情况, 如实批准或拒绝该申请. 监管公司进行商品质检和仓储资产确认, 批准或拒绝该申请. 平台对仓单进行登记, 并根据仓库和监管的审核意见, 决定批准或拒绝该申请. 仓库、监管、平台分别审核, 只要有一方审核失败, 则此次申请不通过并反馈货主; 若三方都审核通过, 则生成仓单.

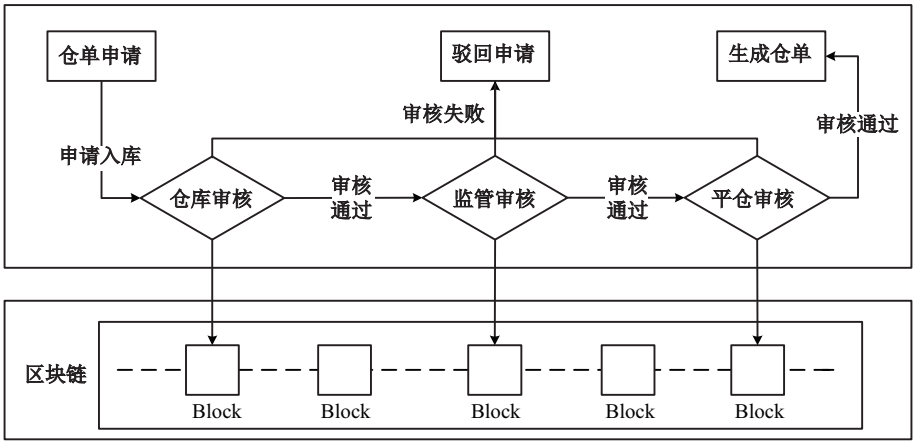

图 2　仓单业务流程

Fig. 2　Warehouse receipt work flow

2.4 系统架构

整个系统的设计采用了 3 层应用架构, 如图 3 所示, 分别为应用层、服务层、数据层. 应用层提供用户的操作功能页面; 服务层负责应用层和数据层的交互; 数据层则是提供存储的区块链系统.

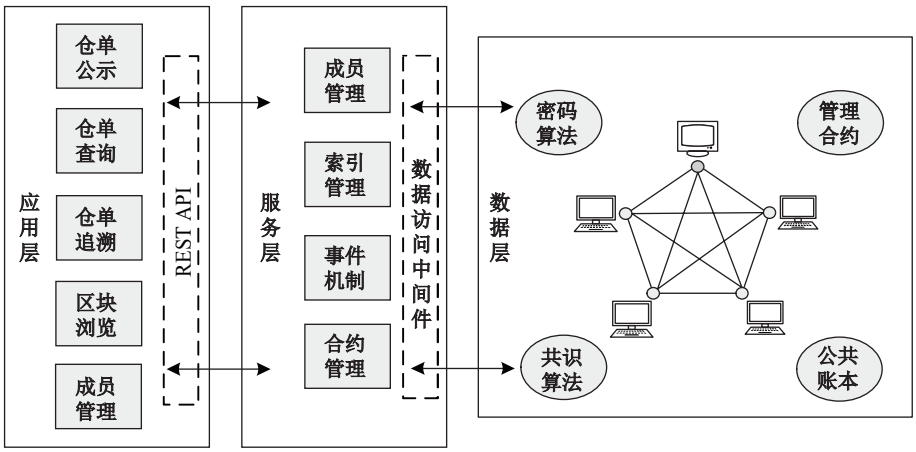

图 3 系统架构

Fig. 3 System architecture

2.4.1 数据层

数据层是由区块链提供的分布式存储, 采用开源项目 Hyperledger Fabric 进行开发. 区块链网络是由 4 个 Peer 节点和 1 个Membersrvc 节点构成的 P2P 网络. Membersrvc 是权限管理节点, 用来验证区块链网络中Peer节点的身份合法性, 所有 Peer 节点都要向 Membersrvc 节点注册. Peer 节点分为 Validating Peer 和 Non-validating Peer: Validating Peer 负责达成共识, 验证、执行交易并维护总账; Non-validating Peer 只会验证交易, 并不执行, 节点间通过 PBFT 算法保证数据一致性. 数据层的主要作用是存储仓单业务中的审计数据, 并且确保数据不可篡改、仓单历史可追溯.

2.4.2 服务层

服务层作为数据层和应用层的中间层, 向上为应用层提供丰富的 REST API, 向下将应用层的逻辑操作转换成对数据层区块链网络的操作. 服务层还有成员管理、索引机制、事件机制、合约管理等功能. 成员管理主要功能是对成员进行授权和验证; 索引机制是维护系统中的索引; 事件机制是响应系统中发生的事件; 合约管理是执行合约的部署、更新和销毁.

2.4.3 应用层

应用层为多角色的 Web 系统, 角色分为货主、仓库、监管、平台. Web 系统选用 Spring-side 框架, 并对其进行扩展和定制. 应用层加入了严格的权限管理机制, 结合 Bootstrap 作为前端 UI 框架, 提高了系统的交互性. 主要功能如下.

(1) 权限管理: 运用 Apache-shrio 框架对用户进行身份验证和授权.

(2) 仓单申请: 货主可以根据需求提出仓单的申请, 但必须在货主仓储物资产数量内.

(3) 仓库审核: 仓库审核仓单申请所填的仓储资产情况, 批准或拒绝该申请.

(4) 监管审核: 监管公司进行商品质检和仓储资产确认, 批准或拒绝该申请.

(5) 平台审核: 平台对仓单进行登记, 并根据仓库和监管的审核意见, 决定批准或拒绝发

布仓单.

3　关键技术

3.1　仓单业务到区块链的映射

在传统的仓单业务系统中, 应用层业务操作映射到数据层就是对数据库中相应表的操作. 而在区块链系统中, 情况会变得复杂. 区块链上的数据分为交易数据和状态数据. 交易数据记录的是交易的参与双方及交易详细内容; 状态数据是指交易发生后, 交易双方的账户状态数据. 它们的存储方式也不同, 状态数据保存在 Key-Value 数据库中, 而交易数据则会保存在区块中, 区块再按照时间戳顺序连接. 新的区块生成后, 会通过共识算法来达成区块链系统的数据一致性, 一旦达成共识就不可逆转. 在 Hyperledger Fabric 中, 智能合约是区块链和状态数据交互的唯一渠道, 可视作一段部署在区块链上可自动运行的程序. 智能合约会按照事先的约定, 接收和储存价值, 也可以进行价值转移, 由代码强制执行, 完全自动且无法干预. 在Hyperledger Fabric中, Chaincode(链码)即为“智能合约”, 部署在Hyperledger Fabric 的 Validating peer 上. Chaincode 采用 Go 或 Java 语言编写, 并通过 gRPC 协议与相应的 Peer 节点进行交互. 当 Chaincode 被部署时, 会执行 Init 函数, 进行仓单数据库的初始化; 货主执行仓单申请操作时, Chaincode 会执行仓单创建, 在验证货主身份合法后, 会为货主生成一份仓单申请表; 审核人员在审核仓单给出审核意见时, Chaincode 会执行审核操作, 在验证审核人员身份合法后, 记录仓单审核结果.

3.2　倒排索引构建与复合查询

Hyperledger Fabric 中, 数据分为状态数据和区块数据. 其中, 底层采用的是 Key-Value 数据库 LevelDB[7]存储状态数据. LevelDB 的特点是主键查询速度快, 但是对于非主键数据的查询速度较慢, 而且不能很好地支持较为复杂的查询. 而在仓单业务中, 针对仓单内容的非主键查询和如模糊查询等复杂查询是非常必要的. 基于上述问题, 本文基于 CouchDB[8]构建了倒排索引, 即一种面向单词的索引机制, 来加快非主键值的查询, 同时支持例如模糊查询等复杂查询. 倒排索引是实现“单词-文档矩阵”的一种具体存储形式, 通过倒排索引, 可以根据单词快速获取包含这个单词的文档列表. 倒排索引主要由两个部分组成: “单词词典”和“倒排文件”, 如图4所示.

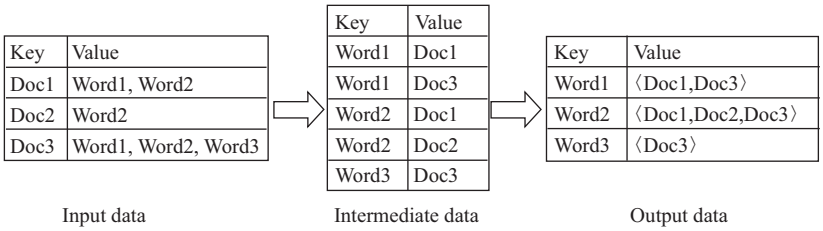

图 4　倒排索引结构

Fig. 4　Inverted index structure

在区块链系统中, 一旦交易经过网络共识后被记录到区块中, 因为区块是不可以被篡改的, 也就意味着共识后的数据不可更改. 这样的特性, 避免了表数据更新带来的索引更新的问题, 简化了倒排索引的维护. 该功能实现分为 Listener 模块和 Persistence 模块. Listener 模块有两阶段的工作: 第一阶段会实时监听区块链节点, 监测是否有新的完成共识的数据, 如监测到就进入第二阶段; 进入第二阶段后, Listener 模块就将数据立即传输给 Persistence

模块进行数据持久化. Persistence 模块是建立倒排索引的关键部分, 实现的主要思路是在 CouchDB 中维护一张 Value-Key 索引表, 在接收到 Listener 模块发来的数据后, 数据会被按照Value-Key 的结构倒插入到索引表中, 使得 Value 变成了新的“Key”. 在做非主键查询时, 在 Value-Key 索引表中查询新的“Key”值, 找到原先的 Key 值, 再用原先的 Key 值查询得到记录, 这样极大地提升了查询效率. 在此基础上, Persistence 模块还提供了复合查询, 支持模糊查询和范围查询, 搜索条件可以是仓单审核状态、仓单号、保管单位、仓库地址、存货人等信息.

4 原型系统演示

4.1 环境配置

服务器: 5 节点的集群; 含 2 TB 存储; 千兆以太网; Ubuntu 14.04 操作系统.

编程语言: Go(区块链)和 Java(网站).

编程环境: JDK 1.8; Spring-side; MyEclipse; Gogland.

4.2 功能演示

(1) 仓单查询模块

如图 5 所示, 仓单查询支持根据仓单号的精确查询, 还支持根据仓单审核状态、监管审核状态、平台审核状态的条件查询和根据关键词的模糊查询. 例如, 某货主在登录后进入查询页面, 其想查询仓单号为“W20170523162648000”, 保存在上海动产测试仓库的货物仓单的审核状态. 该货主不仅可以输入仓单号进行精确查询, 而且可以输入仓库地址、存货人等关键词进行模糊查询. 模糊查询也支持部分单词查询, 例如输入“2017”进行查询, 即会返回该货主所有 2017 年申请的仓单记录.

图 5 仓单查询页面

Fig. 5 Warehouse receipt inquiry page

(2) 仓单审核模块

如图 6 所示, 用户提出的仓单申请, 加入到仓库的待审仓单列表; 待仓库审核完之后再转交给监管审核; 监管审核完之后, 再转交给平台审核; 平台审核完成后, 仓单审核流程结束. 如果货主的仓单申请成功, 上海动产测试仓库管理员的待审仓单中就会增加此条记录. 管理员在审核仓库内货物信息真实性后, 决定是否予以通过. 如果通过, 则审核人、审核时间等信息就会打包记录到区块中上链, 一旦上链就不可篡改, 然后进入接下来的审核步骤.

(3) 审核完成界面

当仓单审核完成后, 页面有“审核完成”字符提示, 同时会有仓库审核、监管审核、平台

审核后的 3 条记录, 每步审核都会记录在区块链中, 形成可追溯的“审核链”, 且可以点击查看详细信息. 如图 7 所示, 该仓单已是审核完成状态, 在履历标签下分别是仓库、监管、平台的审核记录, 可以点击详情查看审核详细信息, 追溯每步仓单审核记录.

图 6　仓单审核页面

Fig. 6　Warehouse receipt audit page

图 7　审核完成界面

Fig. 7　Audit completion page

(4) 查看区块详情

区块中记录了审核的详细信息, 包括交易数据、时间戳、数字签名等. 如图 8 所示, 从该区块详细信息中可以得知, 此区块生成时间是Unix时间“1496027306”, 仓单号为“W20170529110654000”的仓单在“2017-05-29 11:07:35”创建, 包含的货物是材质为 H234 的螺纹钢, 重量为 1 吨(1 000 kg), 数量为 1 件, 目前审核状态是“平台已审”, 即已审核成功.

5　性能测试

硬件环境: 4 台 Peer 节点服务器; 1 台 Membersrc 节点服务器; 主节点提供 REST 服务. 每台服务器硬件配置如表 1 所示.

表 1　服务器硬件配置

Tab. 1　Server hardware configuration

类别	配置
CPU	8 核 2.4 GHz
内存	32 GB DDR3
硬盘	顺序读取: 177 MB/s 顺序写: 80 MB/s
网卡	1 000 MB/s

开发环境: Atom; JDK 1.8.

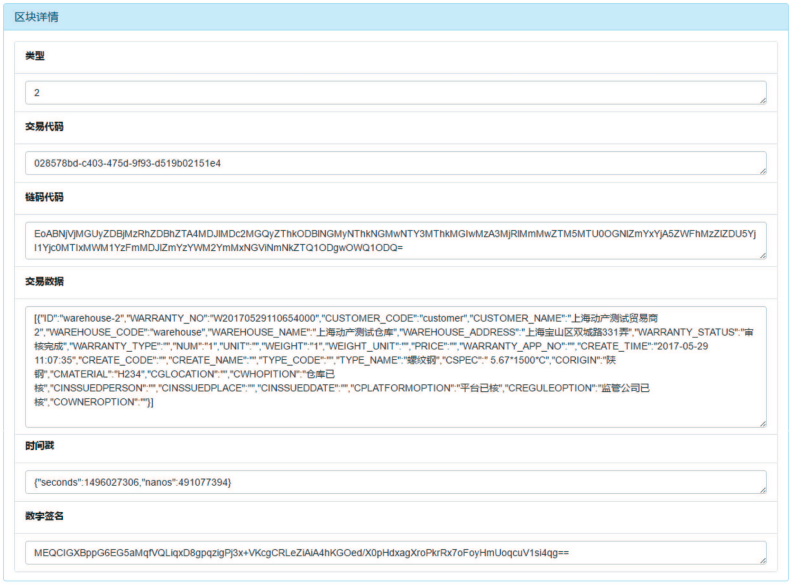

图 8 审核详情页面

Fig. 8 Audit details page

测试程序采用多线程编程, 每个线程模拟 1 个 Client(客户端) 与区块链系统交互, 不断在系统中增加记录. 共进行 6 组测试, Client 每组依次递增 5 个. 记录下请求次数、响应时间等实验数据, 计算不同负载下系统的吞吐量, 同时监控服务器 CPU、内存、磁盘 I/O、网络等资源使用状况.

实验结果如图 9 所示, 横坐标是 Client 数量, 纵坐标是 TPS(Transaction Per Second, 吞吐量). Client 数量在 10 到 25 之间, 服务器未达到硬件瓶颈, TPS 稳定提升; 当 Client 数量在 25 之后继续提升, TPS 趋于稳定, 但到达 30 时并有略微下降, 此时服务器磁盘 I/O 达到峰值, 成为系统瓶颈. 通过实验结果, 可以得出本系统 TPS 约为 240, 因为仓单业务用户群体小、数量少, 基本能够满足仓单业务场景需求. 但是考虑到并发数大的应用场景, 本文实现的系统性能还有很大的提升空间.

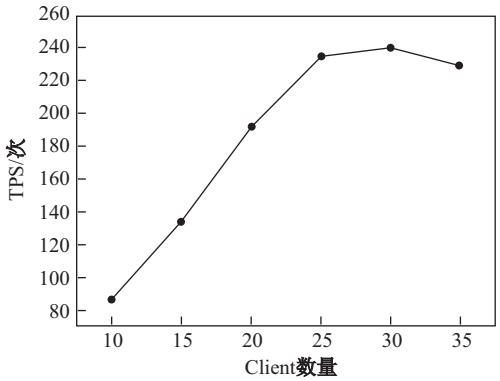

图 9 系统吞吐量测试结果

Fig. 9 System throughput results

6 总结与展望

本文基于区块链设计实现了一个仓单管理系统, 将仓单业务中审计数据记录在区块中, 保

证了数据可追溯、不可篡改. 在此基础上, 本文在区块链系统上构建了倒排索引, 提高了查询效率, 且支持复杂查询. 同时, 实现了基于 REST 的微服务架构, 支持多方接入. 由于研究时间和现有知识水平的限制, 本文的研究工作还不够完善, 需要进一步探索和研究, 作为下一步的研究内容, 将要进行的工作主要有以下几点.

(1) 目前系统吞吐量(TPS)较低, 主要原因是节点共识耗时太长, 下一步工作考虑优化PBFT共识算法.

(2) 支持当新节点加入区块链系统或者节点故障重启后数据快速同步.

(3) 区块链中数据为全备份, 这会占用大量的存储和网络带宽. 因此下一步工作将考虑在数据安全、有效的情况下让区块链系统支持分片(Sharding).

[参 考 文 献]

[1] NAKAMOTO S. Bitcoin: A peer-to-peer electronic cash system [R/OL]. [2018-06-11] https://bitcoin.org/ bitcoin.pdf.

[2] ETHEREUM FOUNDATION. Ethereum [EB/OL]. [2018-6-11]. https://www.ethereum.org.

[3] R3CEV. R3 [EB/OL]. [2018-06-11]. https://www.r3.com.

[4] LINUX FOUNDATION. Hyperledger [EB/OL]. [2018-06-11]. https://www.hyperledger.org.

[5] 袁勇, 王飞跃. 区块链技术发展现状与展望[J]. 自动化学报, 2016, 42(4): 481-494.

[6] RIPPLE. Ripple [EB/OL]. [2018-06-11]. https://ripple.com.

[7] GOOGLE. LevelDB [EB/OL]. [2018-06-11]. http://leveldb.org.

[8] APACHE. CouchDB [EB/OL]. [2018-06-11]. http://couchdb.org.

(责任编辑: 李　艺)

(上接第 134 页)

[30] YAO B, CHEN Z, GAO X, et al. Flexible aggregate nearest neighbor queries in road networks[C]//International Conference on Data Engineering. New York: IEEE, 2018: 1-12.

[31] MA J, YAO B, GAO X, et al. Top-k Critical Vertices Query on Shortest Path[J]. IEEE Transactions on Knowledge and Data Engineering, 2018, 99(1): 1-13.

[32] XIE D, LI G, YAO B, et al. Practical private shortest path computation based on oblivious storage[C]//International Conference on Data Engineering. New York: IEEE, 2016: 361-372.

[33] YAO B, XIAO X, LI F, et al. Dynamic monitoring of optimal locations in road network databases[J]. The International Journal on Very Large Data Bases, 2014, 23(5): 697-720.

[34] XIAO X, YAO B, LI F. Optimal location queries in road network databases[C]//International Conference on Data Engineering. New York: IEEE, 2011: 804-815.

(责任编辑: 张　晶)