拉撒路UTP

V0.1

目录

[1 历史 4](#_Toc38578269)

[2 需求 5](#_Toc38578270)

# 1 历史

本节记录本文档的修改历史

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 日期 | 作者 | 内容 |
| v0.1 | 2020年4月23日 | 时空大神 | 初版本 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 2 系统结构

拉撒路系统的系统架构描述

|  |  |  |
| --- | --- | --- |
| 编号 | 要件 | 内容 |
| 1 | 服务器 | 一个完整的系统需要（1+2n）+1 服务器：  一台作为 nexus-node节点，1+2n 台作为parallel-node节点。 |
| 2 | 节点启动顺序 | 先启动nexus-node节点再启动parallel-node节点 |
| 3 | nexus-node节点的确定 | 启动节点的时候根据配置文件确定。  Nexus\_rotation\_interval时间周期后Nexus和parallel-node之间的切换， 从parallel-node候选池中中根据reputation值随机选出新的nexus-node |
| 4 | 节点部署 | nexus-node节点负责打包、签名、处理、广播交易。  parallel-node节点负责对交易投票验证 |

# 3 系统状态

系统不同状态的描述

|  |  |
| --- | --- |
| 系统空闲 | 系统空闲时 nexus-node节点和 parallel-node节点处于监听 UDP 端口状态。nexus-node节点监 听是否有新的交易请求，parallel-node监听 nexus-node节点是否广播发送交易验证信息。  nexus-node节点向 parallel-node节点发送心跳包检测节点是否在线，将不在线的节点信 息从 nexus-node节点维护的在线节点表中移除。 |
| 系统运行 | 发送交易/验证交易 |

# 4 系统运行状态描述

详细描述系统运行状态的不同方式

## 4.1 交易处理

|  |  |
| --- | --- |
| 编号 | 用途 |
| 1 | nexus-node创建交易信息，用交易发起方账户的私钥对交易进行签名，  将交易信息序列化后通过 UDP socket 向 nexus-node发送数据。 |
| 2 | nexus-node节点通过UDP端口接收到交易信息 |
| 3 | 对接收到交易进行签名验证 |
| 4 | *对签名验证通过后的交易进行预处理。* |
| 5 | *交易预处理完毕后对交易结果验证，检测相关账户的余额是否正确。* |
| 6 | *对预处理成功后的交易的签名进行双重哈希运算。* |
| 7 | 将带有交易签名哈希的交易信息通过UDP socket广播到除了nexus-node节点之外的所有在线节点 |
| 8 | parallel-node节点通过UDP端口接收从nexus-node节点发送过来的交易信息并进行处理 |
| 9 | 更新parallel-node账本的状态，使得和nexus-node账本状态一致 |
| 10 | parallel-node节点对交易信息进行投票并将投票信息发送到nexus-node节点。 |
| 11 | nexus-node节点通过UDP socket接收到parallel-node投票信息，检测账本状态和nexus-nod节点账本状态一致的parallel-node节点数量是否超过总的在线parallel-node节点数量的2/3，如果超过2/3则nexus-nod节点将预处理交易的结果最终写入nexus-node节点上的永久性账本文件上。 |
| 12 | nexus-node节点将最终确认的交易信息同步到parallel-node节点。 |
| 13 | parallel-node*节点从*nexus-node*节点拉取账本数据。* |

## 4.2 TPS测速

|  |  |
| --- | --- |
| 编号 | 用途 |
| 1 | 生成相关配置文件 nexus-id.json、parallel-id.json 用于创建 Nexus/核心、  Parallel/平行节点。 |
| 2 | 开启TbkGiver服务向所有测试节点发送token |
| 3 | 在一台服务器上启动Nexus/核心节点，Nexus/核心节点启动了Fetch Stage、Sigverify Stage、Banking Stage、Write Stage服务用于接收、发送、处理交易请求 |
| 4 | 两台服务器用作Parallel/平行节点，Parallel/平行节点启动Fetch Stage、Replicate Stage服务对nexus-nod节点发送过来的packet进行处理 |
| 5 | 创建50，000个测试账户用于生成转账交易，给这些测试账户空投测试token用于发送交易。 |
| 6 | 向Nexus/核心节点账户地址空投50，000个token用于向测试账户进行转账交易 |
| 7 | 检查各个账户余额是否充足 |
| 8 | 余额充足则在Nexus/核心节点和测试账户间来回进行转账交易 |
| 9 | 启用多线程向Nexus/核心节点发送交易 |
| 10 | 检测交易签名确认交易是否处理成功，计算交易发送到确认成功的时间（如果确认时间超过3 mins则丢弃交易） |
| 11 | 检测账户的余额是否正确 |
| 12 | 每间隔1秒钟计算一次tps |
| 13 | 测试结束，统计计算转账交易测试期间内的最大tps |

# 5 测试策略

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 目标 | 过程 | 预期 |
| 1 | 生成keypair文件 | 创建以下文件：  nexus-id.json  parallel-id.json  client-id.json  用于存放节点seed  并用于创建seed-id.json | 文件生成 |
| 2 | 生成节点配置文件 | 创建nexus.json文件存放nexus-id、address信息 | 文件生成 |
| 创建parrallet.json文件存放parallel-id、address信息 | 文件生成 |
| 3 | 启动TbkGiver服务 | 运行TbkGiver服务用于向用户账户空投代币 | 启动成功 |
| 4 | 启动Nexus/核心节点 | 根据nexus.json 和ledger配置文件启动CompleteNode全节点，写日志到log文件  启动CompleteNode全节点  检测余额是否充足 余额不足则进行空投  检测CompleteNode全节点轮转状态 | 启动成功 |
| 5 | 启动Proposer/提案节点 | 找到Nexus/核心节点并从Nexus/核心节点同步账本数据  启动CompleteNode全节点  检测余额是否充足 余额不足则进行空投  检测CompleteNode全节点轮转状态 | 启动成功 |
| 6 | 启动BenchMarker/测速节点 | 进行bench-tps测试：  1. 获取client id , 网络地址，测试周期等信息 2. 选择Nexus/核心节点 3. 找出网络上所有的节点 4. 创建随机数， 根据随机数生成keypair 5. 空投50\_000代币到Nexus/核心节点， 空投1个代币到barrier地址 6. 多线程transfer交易 7. 查看节点余额是否不足 8. 余额充足则创建交易在节点和测试carrier地址之间进行50\_000次转账 9. 多线程对交易进行发送、验证 10. 统计计算tps | 启动成功 |