性能参数—交易时延测试问题记录

说明

我的测试环境已上传到Github仓库，相关数据位置在下文也指出。

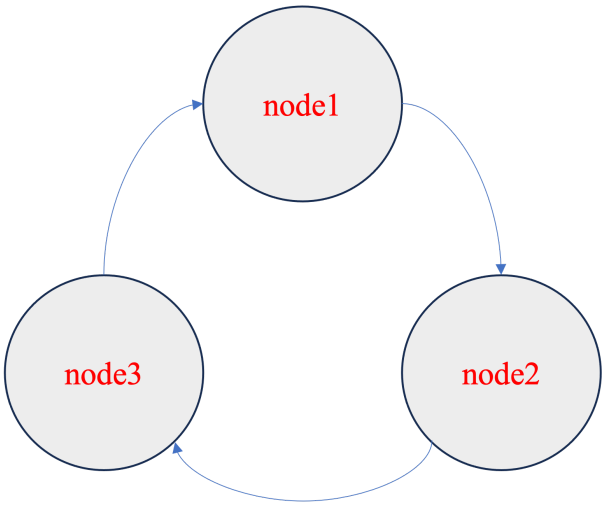
遇到的问题也通过**Bug**形式重点标出。

测试网络配置

使用以太坊客户端geth进行相关性能参数获取，搭建的网络使用三个节点，且均作为clique共识的授权节点记录在创世区块 `./docker_data/genesis.json` 的extradata字段中。

三个节点的数据存储路径为 `./docker_data/node1, ./docker_data/node2, ./docker_data/node3`

在启动区块链后，三个节点互相连接，在console中通过 `admin.peers` 可以找到对应的连接状态为 `static` 的节点。



按照clique共识协议，三个节点在开始挖矿后应该轮流充当打包区块的角色。

获取参数方法

测试时使用源码插入的方式，在源码中插入获取参数信息的函数并分片存入csv文件。测量方法是：

- 节点均开启挖矿 `miner.start()`
- 在 `./cmd/recorderfile` 文件下执行 `'make updateaccessconfig'` 指令开始编译，之后随着区块链系统的迭代，可以在 `./chainmaker.org/log` 下获取到参数文件。

交易时延测试

测量交易时延时，需要起始时间节点（交易提交时间）与结束时间节点（交易确认时间）。在源码插入时，交易提交时间是进入交易池的时间节点，在交易排队时延文件中记录`tx_queue_delay.csv`，以in标志区分。交易确认时间是在区块被验证完成后的时间点，区块的验证时间点在验证结束文件中记录`block_validation_efficiency_end.csv`。

首先我使用节点1的账户（后面简称为账户A）向节点2的账户（账户B）发送交易，该操作通过运行脚本`./docker_data/script.js`执行，经过较长时间（区块链挖矿工作进入稳定状态后）观察验证文件输出，可以看到输出的区块内交易数量为0。

此时查看记录块内交易吞吐量文件`tx_in_block_tps.csv`发现区块交易数量不为0，并且出现问题1:

- **Bug1: 区块号不是连续的，每隔3记录一次，并且出块间隔也不是15s，而是45s**

在console中测试，使用`eth.getBlock(blockNumber)`获取区块号对应区块的具体信息，发现链上区块两个有交易的区块之间隔着两个空块，也就是**三个节点的一个挖矿周期**。这时可以合理猜测，tps文件记录的数据可能只和节点1相关，而不是一个系统层面的信息。

进而分析为什么只有节点1打包交易而节点2、3没有，查看交易池信息`txpool.status`

```
> txpool.status
{
  pending: 69481,
  queued: 0
}
>
```

节点1是有等待被打包的交易的

```
> txpool.status
{
  pending: 0,
  queued: 0
}
>
```

另外两个没有

- **Bug2: 这是否说明，只有谁发交易，谁才能记录交易？这个问题也很奇怪**

那么继续从B到节点三的账户（C）发交易，继续查看交易池信息，节点三的pending曾短暂的有交易认领，查看验证文件输出不为0，但是一段时间后pending又变成了0，验证文件输出的区块内交易量又变成0。当C也向B发交易时，验证文件的输出才稳定的不为0，pending序列也非空了

- **Bug3: 区块验证文件的输出貌似和节点三的状态密切相关？也记录的不是一个系统层面的信息！**

后续进行数据分析，即通过数据库的merge操作等对文件进行分析

> 这里需要操作的原因是区块验证文件记录的tag为区块hash，而交易排队时延文件中则记录的是交易hash，需要将txhash与区块号匹配，进而与区块hash匹配，从而获得时间差。发现只有tps文件中有区块号与区块hash的对应记录。

经过操作后输出的文件是空的，这恰好也验证了上面的疑惑，似乎两个文件记录的数据是互斥的，不是系统层面而是节点层面的信息。

Tips：**在区块验证文件的时间流中，时间间隔很混乱，然而出块间隔是固定的且每个区块都应当经过验证过程，所以这里也很奇怪**