PDX Utopia

合约之间关系记录图谱

版本 1.0.0

目 录

[1 问题说明 1](#_Toc1666545840)

[2 解决办法 2](#_Toc6725059)

## 

**1 问题说明**

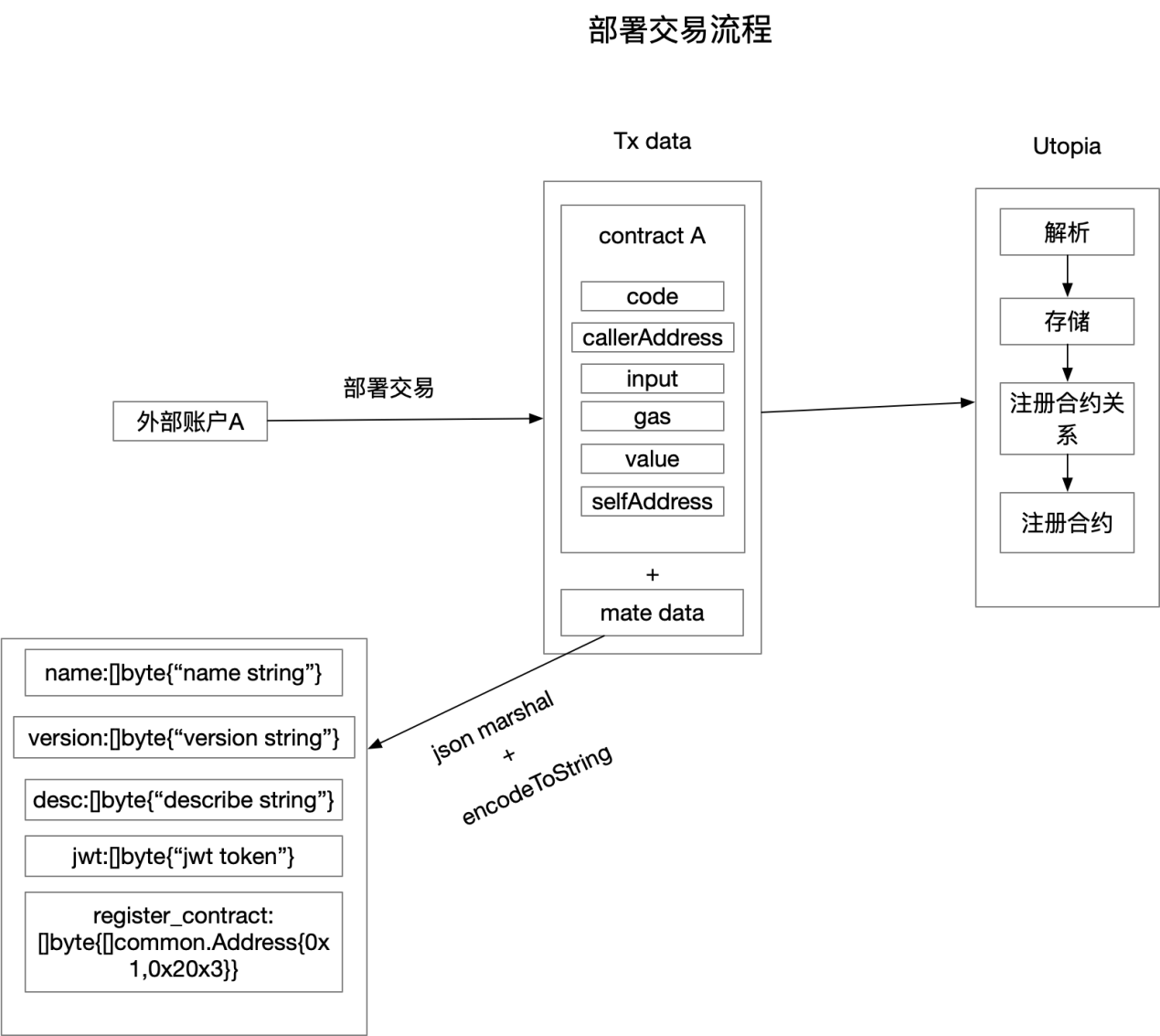
对于之前版本的utopia中分桶的逻辑，分桶逻辑会根据不同的from和to对交易进行分桶而后并行执行以提高执行效率，根对于此种分桶方法，如外部账户A通过发送交易Tx1的方式调用合约账户B，而智能合约账户B中调用了合约账户C。同时，外部账户A’通过发送交易Tx2的方式调用合约账户B’，而智能合约账户B’中调用了合约账户C。在分桶的时候，Tx1,Tx2会被分到不同的桶中并行执行，而交易的结果最终会出现不确定性。从而造成坏块。

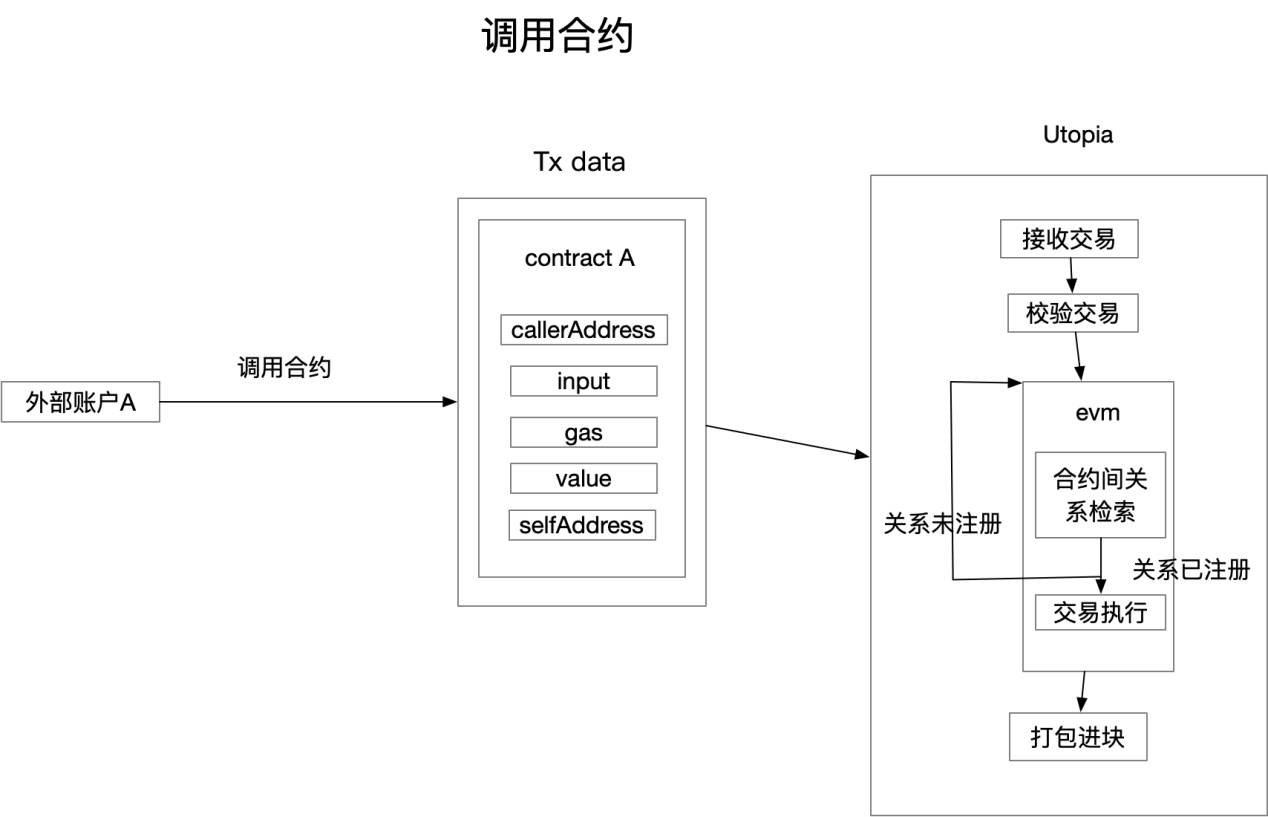
**2 解决办法**

增加合约之间关系的注册与检测，本方案分成两部分来实现，第一部分，交易分桶的检索。第二部分，检索交易执行时如果出现合约之间的互相调用，则校验关系，注册允许互相调用的可以正常调用，未注册的则调用失败。其中，合约之间的注册关系，可以通过在发送部署合约交易的时候附带mate data将合约之间允许调用关系附带进来，协议栈在收到部署交易的时候会对payload进行解析，当出现附带信息的时候，对附带信息进行记录。其中附带信息填充方式如表1。

对于第一部分的详细描述：以前逻辑分桶方法为根据交易的from以及to进行分桶，相同的from以及相同的to都在同一个桶中执行。现在，在检索数据合约地址的to的时候将与所有与to有关的地址都放入同一桶中。

对于第二部分的详细描述：合约在执行的时候进行检索，当出现合约调用合约的时候，对合约关系进行检索，如果合约之间的调用关系已注册那么，允许其继续调用。如果合约之间的关系没有注册，则调用失败。

整个流程图如下：



代码实现：

|  |
| --- |
| //需要注册的合约之间的调用关系  address := common.HexToAddress("0x1a255c4D7632EaF3785639152Cf7F1Cc80Eb1795")  register := []common.Address{address}  registerData, err := rlp.EncodeToBytes(register)  if err != nil {  println(err)  }  //meta 信息  type Meta map[string][]byte  meta := Meta{"name": []byte("tonysu"), "version": []byte("1.2"), "desc": []byte("tony's contract"), "jwt": []byte("eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCJ9.eyJhayI6IjAzOTBkNWQxMDQ4MjMzMDRlYjQ0Mjc2NTQ1Y2U0YjNiYmVkYmEyODE3MTYyOGExMjYyYjBmZjBiNThiNTllM2QyZiIsImwiOjYwMCwibiI6ImVlZmZmZWZyZXJld2RmZmRzZjJycmZkc21mbGpsanJyIiwiciI6ImQiLCJzIjoxMjM0NSwic2siOiIwMjU5NWQ1NTM2OTczMDVjNzY3MGRmZDkyNjI4ZTVmZjY4MDgwMzM1MjY1ZWRmODA0YWVhNGU2ZThkZjUxMTI0NjQifQ.V61YuRL1HidgdgbKHRtxvk3x8KJsy7QUqAwjjxy4HGNfdukdWM15obTDuE8YbtOI3aCfc2UcaUgwvlwRB\_wB\_Q"), "register\_contract": registerData}  metaByte, \_ := json.Marshal(meta)  metaStr := hex.EncodeToString(metaByte)  //utopia 转成common.Hash 附带了meta data的标志  flagNo0x := "26cad4db1a82fab9e41bf5c0fbb4937a27b93786a4f27d3b9704805f698d3e65"  //calldata 不能带有0x前缀  datas := ""  //payload 拼接  dataStr := flagNo0x + "\_" + datas + "\_" + metaStr  //payload转换  data := []byte(dataStr)  //构建交易  tx := types.NewContractCreation(nonce, amount, gasLimit, gasPrice, data)  signer := types.NewEIP155Signer(big.NewInt(ChainID))  signedTx, \_ := types.SignTx(tx, signer, privKey)  if txHash, err := client.SendRawTransaction(context.TODO(), signedTx); err != nil {  fmt.Println("send raw transaction err:", err.Error())  } |