|  |
| --- |
| **准备工作**  1、三个账户，eth.account[0]为默认账户,挖矿所得的奖励都会进入到这个账户  > eth.getBalance(eth.accounts[0])736031150000000000000> eth.getBalance(eth.accounts[1])500050000000000000> eth.getBalance(eth.accounts[2])500050000000000000  **普通交易所需的gas**  > eth.estimateGas({from:eth.accounts[1], to: eth.accounts[2], value:50000000000000})21001> eth.gasPrice20000000000  如上，显示这笔account[1] => account[2] 的交易需要21001 gas, 当前的gasPrice为 20000000000，下面来**验证一下**  给账户1解锁，发送这笔交易，并开启挖矿打包  > eth.sendTransaction({from:eth.accounts[1], to: eth.accounts[2], value:50000000000000})  I0318 00:24:21.360815 internal/ethapi/api.go:1143] Tx(0x33b58084a35e99245b9c931204a0d161b9d00f9fae5ffb307aff29f200e5cd30) to: 0x49fbd70ca9f90972806c375a111d08950d203f96"0x33b58084a35e99245b9c931204a0d161b9d00f9fae5ffb307aff29f200e5cd30"  待交易被打包后  > eth.getBalance(eth.accounts[1])499580000000000000> eth.getBalance(eth.accounts[2])500100000000000000  由 cost = gas \* gasPrice , （ 账户1减少的资产 - 账户2增加的资产）/ gasPrice = 消耗的gas，即以下公式应该是成立的  (500050000000000000 - 499580000000000000) - (500100000000000000 - 500050000000000000) = 21001 \* 20000000000  **然而** 细心的同学应该会发现，这个公式并不能成功，21001 这个数字怎么看怎么别扭，如果减1 这个公式就成功了！！！继续  查看这笔交易的明细  > eth.getTransactionReceipt("0x33b58084a35e99245b9c931204a0d161b9d00f9fae5ffb307aff29f200e5cd30"){  blockHash: "0x8e411163367bc42a70[ECC](http://8btc.com/article-138-1.html" \t "http://8btc.com/_blank)230d05dd2038afe0dccfab29c8a718a57bdbea0b2fa",  blockNumber: 134,  contractAddress: null,  cumulativeGasUsed: 21000,  from: "0x27c649b7c4f66cfaedb99d6b38527db4deda6f41",  gasUsed: 21000,  logs: [],  logsBloom: "0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",  root: "0x2008f134f3328e48d4d05919666a5924767b00b286cf1ff27b7956654d5b6482",  to: "0x49fbd70ca9f90972806c375a111d08950d203f96",  transactionHash: "0x33b58084a35e99245b9c931204a0d161b9d00f9fae5ffb307aff29f200e5cd30",  transactionIndex: 0}  gasUsed: 21000 这就对上了，那么为什么eth.estimateGas() 计算的结果要多1 呢？这是有原因滴，如果计算出来的值和gasUsed相等，那这笔交易有可能是失败的，但是如果 gasUsed 小于计算出来的值，那么可以判断这笔交易成功了  **调用合约方法所需要的gas**  普通的转账交易所需要的gas是固定的21000，但是调用合约方法所需要的gas并不一定，总结来说占用的资源（计算量、内存等）越多，那么所需要的gas也就越多。先准备一个最简单的合约  pragma solidity ^0.4.8;  contract Test {  uint public num;    function setNum(uint newNum) {  num = newNum;  }}  部署到私有链，这个过程就不再演示，最后合约实例testInstance。再看看eth.accounts[1] 和 eth.accounts[2] 的资产信息，方面后面计算  > eth.getBalance(eth.accounts[1])499580000000000000> eth.getBalance(eth.accounts[2])500100000000000000  计算调用合约方法setNum() 所需要的gas  > testInstance.setNum.estimateGas(4, {from: eth.accounts[1]})41645  开始调用  > testInstance.setNum.sendTransaction(4, {from: eth.accounts[1]})  I0318 07:21:31.344279 internal/ethapi/api.go:1143] Tx(0x3fad05f17f7904e08dcb9257ad28f85f29bd54c4729784fa39a9df88e3fcffab) to: 0x03a4fb357f8c38694ab536d09003076033442f9e"0x3fad05f17f7904e08dcb9257ad28f85f29bd54c4729784fa39a9df88e3fcffab"  开启挖矿，让这笔交易被打包之后，再来查看下gasUsed 跟上面计算出来的数字是否吻合  > eth.getTransactionReceipt('0x3fad05f17f7904e08dcb9257ad28f85f29bd54c4729784fa39a9df88e3fcffab'){  blockHash: "0x494f5f6fc0c156f105ffe3e4e1aa886c60f916a5998d44a03916b3f2cc733b8a",  blockNumber: 139,  contractAddress: null,  cumulativeGasUsed: 41644,  from: "0x27c649b7c4f66cfaedb99d6b38527db4deda6f41",  gasUsed: 41644,  logs: [],  logsBloom: "0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",  root: "0x857063e074cc3195ee2f3962438f3f6c31a759cfae461448e8726a5fa069d1ae",  to: "0x03a4fb357f8c38694ab536d09003076033442f9e",  transactionHash: "0x3fad05f17f7904e08dcb9257ad28f85f29bd54c4729784fa39a9df88e3fcffab",  transactionIndex: 0}  可以看到gasUsed:41644 比计算出来的少1，原因上面已经讲过了，这里不赘述！ |