

上半节

GAS详解

(1) 通过对比web2讲解什么是Gas

<https://www.notion.so/Gas-12581438873b80ae86b7dc422060ec4a?pvs=4>

(2) 讲解Gas里面的的几个关键词

EIP1559, Gas Limit, Base Fee, Priority Gas fee

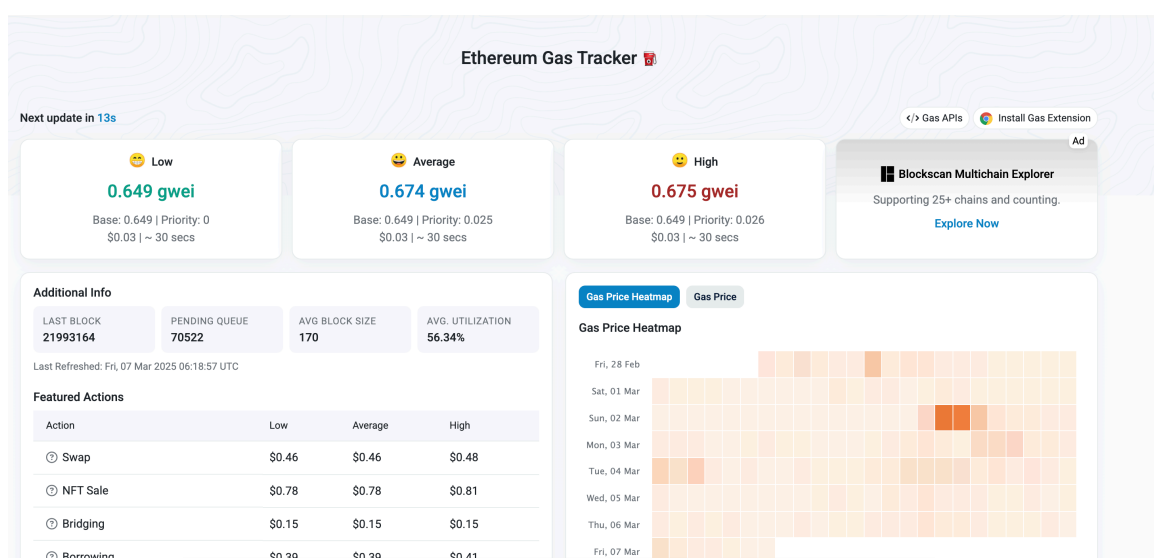
(3) 讲解如何设置Gas Limit

(4) 对比call, transfer, send, 解释什么是Gas forward

(4) 讲解如何预测Gas以及Gas Reort

1、初步感受下什么是Gas

(1) 通过GasTracker感受下gas <https://etherscan.io/gastracker>



从图中我们可以得到几个信息

(1) 不同的Gas费会有不同的交易速度，Gas费给的越高交易越快

(2) Gas是一直在变化的，大约14秒变一次

(3) 不同的操作所需要的Gas费不一样

(4) 以太坊会控制区块的使用率在50%左右

(2) 通过交易详情感受下GAS

```
https://etherscan.io/tx/0xd1a603fcfae63d1e204ac4f06a0608ad0758b926ce24eec60baf12c0475d38df
```

- Gas Limit & Usage by Txn:64,019 | 63,185 (98.7%)

解释：设置了GasLimit 64019，使用了98.7%

- Gas Fees:Base: 0.689125629 Gwei |Max: 0.897413793 Gwei |Max Priority: 0.001550476 Gwei

解释：设置基础服务费 0.689125629 Gwei，

最大优先手续费0.001550476 Gwei，

总共最大手续费 0.897413793 Gwei

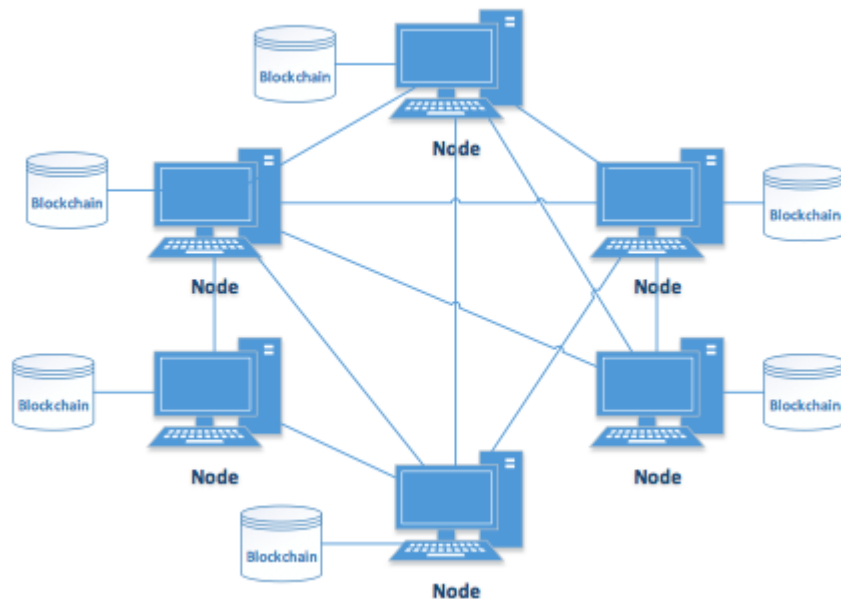
- Gas Price:0.690676105 Gwei (0.000000000690676105 ETH)

解释：BASE+PRIORITY

- Transaction Fee: 0.000043640369694425 ETH(\$0.09)

解释：Usage Gas * Gas Price = 63,185*0.690676105

2、Gas在Web3中意味着什么



🧑 思考

- (1) 在web2，因为会有中央服务器的存在，我们可以更新、重新、甚至剔除有问题的服务器，但是在去中心化的系统我们如何处理？
- (2) 如果有一段恶意代码写了一段死循环，会不会导致一台机器down机，又因为交易会广播，会不会导致其他机器也死机，使得整个网络瘫痪？
- (3) 以太坊是自发运行的系统，矿工如何为何愿意执行这些交易
- (4) 交易有时候高，有时候低，整个网络的繁忙程度如何衡量呢？

以太坊Gas费的目的

1. 避免垃圾信息的执行

因为Ethereum上每一笔交易都需要消耗gas 费，所以如果有恶意交易执行，会让交易发起方付出gas fee，因此可以一定程度上确保Ethereum的恶意或者垃圾信息。

2. 避免合约代码出现异常导致节点资源耗尽

例如当合约代码存在无限循环时，交易发起者的Gas limit会很快被耗尽，block的gas 也会很快被耗尽，计算就会被终止，有效避免了资源卡死。

3. 以太坊解决拜占廷问题的创新，激励矿工交易

4. 提供了一种衡量以太坊网络的方法

Ethereum gas and how to calculate

3、什么是GAS

一句话：

以太坊Gas是衡量以太坊网络上执行具体操作时所需要的计算工作量(the amount of computational effort)的单位(unit)。

需要注意的是:

无论交易是否成功或者失败，都会有gas fee被使用。（即使失败也会消耗一部分计算资源）

理解Gas包含两部分，

衡量计算资源的GAS USAGE

每个GAS的价格 GAS PRICE

而GAS FEE简单的理解就是，一个程序使用了多少的gas乘以每份gas需要多少的费用

$$\text{Total cost} = \text{gas usage} * \text{gas_price}$$

(1) GAS USAGE

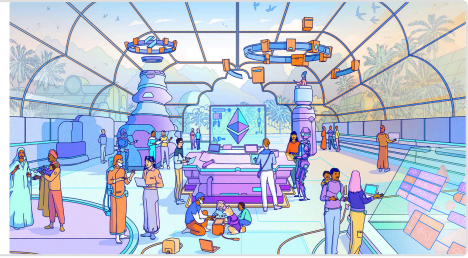
GAS usage 是以太坊上合约计算、交易转账所需要使用的资源量的衡量单位，一个代码越复杂，访问状态变量越多，他所需要的gas usage就越多，所以一段死循环或者一段恶意代码他可能会消耗很多gas，这样也就可以通过限制gas的使用量（gas limit），来减少恶意代码或者预期外的代码对用户资金以及计算资源的消耗。

- gas usage是由以太坊上面定义的各种字节操作定义的，

Opcodes for the EVM


A list of all available opcodes for the Ethereum virtual machine.

 <https://ethereum.org/en/developers/docs/evm/opcodes/>



- 我们通常所说的gas优化，也就是指通过优化程序逻辑，减少重复访问状态变量，来减少一段程序所需要的gas。

Solidity by Example

 <https://solidity-by-example.org/gas-golf/>

(2) GAS FEE (EIP1559)

GAS FEE又称为**GAS PRICE**，代表每一份**GAS**用户希望支付的金额

- EIP1559之前的GAS FEE

在EIP1559之前的GAS FEE完全是一个盲拍过程，用户随机给出自己的gas price，价高者的交易会被优先执行。这样有两个问题：

i. 下一个区块的gas price变化不定(volatile)

Block	Age	Txn	Uncles	Miner	Gas Used	Gas Limit	Avg.Gas Price
10236453	398 days 15 hrs ago	244	0	Ethermine	9,992,346 (99.92%)	10,000,000	42.63 Gwei
10236452	398 days 15 hrs ago	218	0	Ethermine	9,988,129 (99.98%)	9,990,249	334.68 Gwei
10236451	398 days 15 hrs ago	223	0	0x7d6dabd6b5c75291a3...	9,971,708 (99.91%)	9,980,504	40.47 Gwei

ii. 因为这个模式下gas price的选择完全是盲拍的，所以会导致用户的手续费会超额支付。

User A submits a transaction with a 1000 gwei gas price
All other users submit transactions with 10 gwei gas prices
A miner selects User A's transaction, and fills the rest of the block with 10 gwei transactions

===

A用户支付1000 gwei gas price，但实际上10gwei就够了

- **EIP1559之后的GAS FEE**

在EIP1559之后，一次交易所需要的Gas被分成两部分：基础费用+优先级费用

基础费用：由以太坊协议决定，是让交易被验证所需要的最低费用。需要注意的是，如果只支付Base fee，只表示交易被以太坊接受但是并不表示一定会有节点愿意验证这笔交易(pending)。因此为了确保你的交易能够被节点验证，还需要额外支付一笔gas fee 用于激励节点验证这笔交易（优先级费用）。

优先级费用：为了让交易更能被节点验证，我们需要支付一部分“优先级费用”以激励验证节点尽快验证这笔交易。费用大小取决于当前网络的使用情况，如果当前有很多交易等待被验证，那么优先级费用就会比较高，反之则可以少付一点。

因此，总的费用为：

$$TotalGasfee = unitsOfGasUsed * (baseFee + priorityFee)$$

在Etheruem.org中具了这么一个例子

Jordan has to pay Taylor 1 ETH. An ETH transfer requires 21,000 units of gas, and the base fee is 10 gwei. Jordan includes a tip of 2 gwei. The total fee would now be equal to $21,000 * (10 + 2) = 252,000 \text{ gwei}$ (0.000252 ETH).

在这笔交易被验证之后，BaseFee将会被Burn掉，PriorityFee将会给到Validator

4、进一步理解BASE FEE

基础费用是由以太坊协议决定的，为了让交易能够被验证所需要的最低费用。同时为了让这笔费用能够可被预测，在EIP1559之后，提出了Base Fee的计算模式。

BASE FEE的提出让交易手续费可以预测

Base fee的由前一个区块的大小和现在区块大小的差异来决定的，他因此反应了当前以太坊网络的拥挤程度。每一个区块的平均大小在1500万个gas，当交易过多时，区

块大小会增长到不超过3000万个gas，以太坊的机制会保证区块的平均gas在1500万，。

当目标区块的大小超过了一定阈值（1500万gas），基础费用将会在目标区块的base fee的基础上最多增加12.5%，相反的，如果目标区块的大小较小，那么基础费用将会在上一个区块的base fee基础上最多减少12.5%。

相较于eip1559之前的例子里的1000%的变化趋势，这样已经很可以预测了

- If the last block was exactly 50% full, the **Base Fee** will remain unchanged.
- If the last block was 100% full, the **Base Fee** will *increase* by the maximum 12.5% for the next block.
- If the last block was more than 50% full but less than 100% full, the **Base Fee** will *increase* by less than 12.5%.
- If the last block was 0% full – that is, empty – the **Base Fee** will *decrease* the maximum 12.5% for the next block.
- If the last block was more than 0% full but less than 50% full, the **Base Fee** will *decrease* by less than 12.5%.

实验🔪：观察block gas used的大小，与gas fee的变化

<https://etherscan.io/block/22005624>

<https://etherscan.io/block/22005625>

回答：可以看到当gas used少于50%时，下一个区块的base fee下跌，当gas used大于50%时，下一个区块的base fee上涨。

5、进一步理解PRIORITY FEE

如果不提供PriorityFee,那么矿工更愿意挖一个空的块,因为这样他在获得同样的挖块奖励的基础上所消耗的费用最低,所以为了激励矿工验证交易,交易发起方需要支付恰当数量的Priority fee (给少了交易不会被验证,给多了又浪费)。

思考🤔: 一笔交易如果priority过低,可能永远执行不被执行么?

回答: 可能,这就是为什么ethers tracker里pending queue 有6w多的原因,很多交易一直没有被执行。

实验🔪: 演示一下通过不同的手续费,交易执行的速度

⚠️ 注意:

(1) eip1559里删除了gasPrice,因为这个是之前的概念

(2) gasPrice被替换为两个参数:

maxPriorityFeePerGas: priority fee的最大值

maxFeePerGas: gas fee(base_fee+priority_fee)的最大值

(3) 第二部并不是必须的,eth节点会提供默认值

maxPriorityFeePerGas: eth_gasPrice - base_fee

maxFeePerGas: maxPriorityFeePerGas + 2 * base_fee

🌟 **eth_gasPrice**在EIP1559下,返回的是当前网络的设置的**base_fee+平均的priorityfee**

实验🔪: 如何在js里设置gas price

```
const { ethers } = require("hardhat");

async function execute() {
  const _contract = await ethers.getContractFactory("Demo");
  const demo = await _contract.deploy(1);
  await demo.waitForDeployment();
  let tx = await demo.inc();
  let receipt = await tx.wait();
}
```



```

console.log(recipt);
console.log(await demo.i());
let feedData = await ethers.provider.getFeeData();
tx = await demo.inc({
  maxPriorityFeePerGas: feedData.maxPriorityFeePerGas * 5n,
});
recipt = await tx.wait();
console.log(recipt);
console.log(await demo.i());
}
execute();

```

6、什么是GAS LIMIT

GAS LIMIT有两个含义：

- (1) 用户设置的一笔操作执行最大使用的GAS
- (2) 区块最大能够容纳的GAS

(1) 交易中的GAS LIMIT

如何设置Gas limit

- 在solidity call设置gas limit

```

// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;
import "hardhat/console.sol";

contract SetGasLimit{

    uint256 value;

    Reciver private r ;

    constructor(Reciver _r){
        r=_r;
    }
}

```

```

    }

    function invoke() external payable {

        r.bizMethod{gas:8000}();

    }
}

contract Reciver{
    receive() external payable {
        console.logUint(gasleft());
    }

    function bizMethod() external{

        (bool r,) = address(0).call{gas:5000}("");
        require(r);

    }
}

```

- 在ether.js中设置gasLimit

```

const hre = require("hardhat");

describe("GAS TEST", function () {
    let gasTest;
    this.beforeEach(async function () {
        const GasTest = await hre.ethers.getContractFactory("GasLimitTest");
        gasTest = await GasTest.deploy();
        gasTest.waitForDeployment();
    });
    it("execute", async function () {
        const options = { gasLimit: 21064 };
        try {
            await gasTest.execute(options);

```

```
    } catch (e) {  
        e.message.includes("Transaction ran out of gas");  
    }  
});  
});
```

💡 一笔以太坊交易最低要21000个gas

21000 gas is charged for any transaction as a "base fee". This covers the cost of an elliptic curve operation to recover the sender address from the signature as well as the disk and bandwidth space of storing the transaction.

For example, the default amount of gas limit for a standard ETH transaction consumes 21,000 gas, while the gas price chosen by the sender is 30 Gwei. Multiply these two figures together, we will get the actual cost of the transaction execution, amounting to 0.00063 ETH (equal to \$0.16821 at the time writing).

(2) BLOCK中的GAS_LIMIT

如果不设置gas limit，交易默认使用的gaslimit 为block.gasLimit

Under EIP-1559, there is also a maximum (or "hard cap") gas limit, set to twice the target, which is **30 million gas**. This means that a block can include transactions using up to 30 million gas. Since then Ethereum's block gas limit remained the same and, as of 2024, it is still at 30 million gas per block. Jan 24, 2024

5、如何进行Gas的预测

- 在Solidity中 使用GasLeft()

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
import "hardhat/console.sol";
contract EstimateGasCost{
    uint i;
    function estimateGasCost(uint256 data) external {
        uint256 gasStart = gasleft();
        // Execute the operation that consumes gas
        i++;
        i=i*data;
        // ...
        uint256 gasSpent = gasStart - gasleft();
        uint256 gasFee= gasSpent * tx.gasprice;
        console.log(gasFee);
    }
}
```

- 在ethersjs中使用 estimate gas

```
const hre = require("hardhat");

describe("GAS TEST", function () {
    let gasTest;
    this.beforeEach(async function () {
        const GasTest = await hre.ethers.getContractFactory("GasLimitTest");
        gasTest = await GasTest.deploy();
        gasTest.waitForDeployment();
    });
    it("estimate no params", async function () {
        console.log(await gasTest.execute.estimateGas());
    });
    it("estimate with params", async function () {
        console.log(await gasTest.setter.estimateGas(3));
    });
});
```

```
});  
});
```

6、使用Hardhat的GasReport功能

- hardhat gas report

第一步：安装依赖 `npm install --save-dev hardhat-gas-reporter`

第二步：添加`require("hardhat-gas-reporter")`到`hardhat.config`中

```
const hre = require("hardhat");  
  
describe("GasGolf", function () {  
  it("V1", async function () {  
    let GasGolf = await hre.ethers.getContractFactory("GasGolf");  
    gasGolf = await GasGolf.deploy();  
    await gasGolf.waitForDeployment();  
    await gasGolf.sumIfEvenAndLessThan99V1([1, 2, 3, 4, 5, 100]);  
    await gasGolf.sumIfEvenAndLessThan99V2([1, 2, 3, 4, 5, 100]);  
  });  
  // it("V2", async function () {  
  //   gasGolf.sumIfEvenAndLessThan99V1([1, 2, 3, 4, 5, 100]);  
  // });  
});
```