**1. Solidity**

Solidity adalah bahasa pemrograman lanjutan berdasarkan kontrak pintar. Ini memiliki sintaks yang mirip dengan JavaScript. Ini mendukung jenis statis, integrasi, pustaka, dan jenis yang ditentukan pengguna komposit. Ini dapat dikompilasi ke dalam rakitan EVM dan oleh karena itu dapat dieksekusi pada semua node Ethereum. Ada bahasa pemrograman kontrak pintar lainnya: Serpent, Vyper, dan LLL. Tidak diragukan lagi, Solidity adalah bahasa pemrograman terpanas dan paling populer untuk kontrak pintar. EVM adalah sandbox runtime. Jadi semua kontrak pintar yang bertumpu pada Ethereum dipisahkan dari lingkungan sekitarnya. Akibatnya, smart contract pada EVM tidak dapat mengakses jaringan, sistem file, atau proses lain di Ethereum. Solidity adalah bahasa pengecekan tipe statis.

Kompilernya dapat memeriksa:

• Semua fungsi harus ditentukan

• Objek tidak boleh null

• Operator tidak valid

**2. Smart contract**

Seperti namanya, kontrak pintar adalah kontrak otomatis. Faktanya, smart contract adalah program komputer yang berjalan secara otomatis ketika kondisi tertentu terpenuhi. Kontak pintar adalah seperangkat instruksi yang dikembangkan oleh Solidity (bisa jadi bahasa pemrograman lain, tetapi buku ini hanya menggunakan Solidity). Bahasa pemrograman Solidity didasarkan pada IFTTT (yang merupakan logika IF-THIS-THEN-THAT: jalankan kode jika beberapa kondisi terpenuhi). Karena kontrak pintar dijalankan di EVM, ia tidak dapat mengakses jaringan, sistem file, atau proses lain yang berjalan di EVM. Smart contract dapat mengakses data eksternal melalui Oracle jika perlu. Umumnya, smart contract dapat diimplementasikan berdasarkan dua jenis sistem:

1. Mesin Virtual (VM): Ethereum

2. Buruh Pelabuhan: Fabric

Semua konten berikut termasuk diskusi, kode, dan diagram didasarkan pada kerangka kerja kontrak pintar di Ethereum.

**3. GAS**

GAS digunakan untuk mengukur berapa banyak langkah yang diperlukan transaksi pada EVM. Sangat mudah: transaksi Anda rumit, yang berarti membutuhkan lebih banyak sumber daya komputasi (seperti waktu dan memori CPU), Anda harus membayar lebih banyak GAS. Semua opcode pada EVM akan dikenakan biaya GAS dan tidak mungkin berubah di masa depan. Metrik terkecil GAS adalah wei, dan 1 eth = 10^18 wei = 10^9 gwei.

1. Harga Gas: Harga Satuan Gas

2. Gas Limit: Batas atas Gas yang pengguna bersedia membayar

Gas Limit x Harga Gas = biaya maksimum yang bersedia dibayar pengguna

Dalam satu transaksi, jika biaya maksimum yang Anda tentukan tidak habis, maka dana yang tersisa akan dikembalikan ke akun Anda. Setelah Gas habis, transaksi akan berhenti di mana pun itu berada dan pengecualian kehabisan gas akan dilemparkan. Semua perubahan status yang dibuat oleh transaksi akan dikembalikan.

Namun, setelah dana digunakan, meskipun transaksi gagal, pengguna tidak akan mendapatkan pengembalian dana karena dana tersebut dihargai kepada penambang. Pengguna perlu membayar biaya untuk menggunakan sumber daya komputasi dan biaya yang terdiri dari tiga komponen:

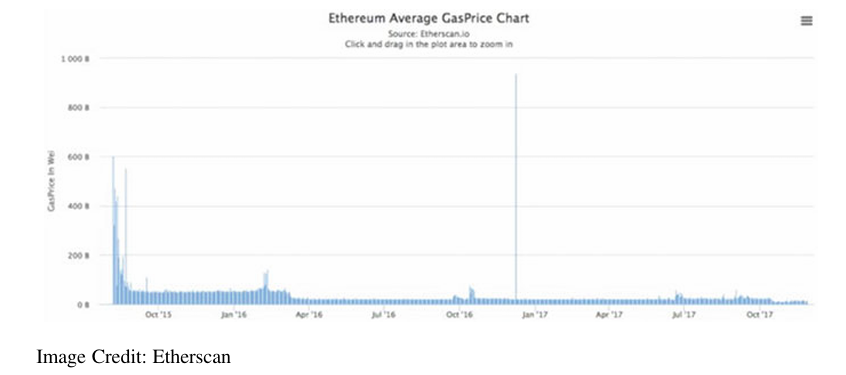
1. Biaya komputasi

2. Biaya transaksi (pembuatan kontrak atau panggilan pesan)

3. Biaya penyimpanan (memori, penyimpanan data akun/kontrak)

Jika kontrak Anda menyimpan data ke database, semua node di Ethereum akan melakukan hal yang sama, yang sangat mahal. Itulah mengapa Ethereum akan mengenakan biaya untuk penyimpanan dan mendorong penggunaan penyimpanan yang lebih sedikit. Jika op adalah untuk menghapus item penyimpanan, Ethereum akan melakukan ini secara gratis, atau bahkan mendapatkan pengembalian dana.

Berikut adalah grafik tentang harga GAS rata-rata—https://etherscan.io/chart/gasprice



**3.1 Kenapa GAS?**

Mengapa kita membutuhkan GAS? Secara umum, ada tiga alasan utama: keuangan, teori, dan komputasi.

Dari sudut pandang keuangan, tujuannya adalah untuk memberi insentif kepada penambang untuk melakukan transaksi dan kontrak pintar dengan menggunakan waktu dan sumber daya mereka sendiri. Banyak operasi kompleks membutuhkan lebih banyak sumber daya komputasi; artinya, mereka perlu membayar lebih banyak GAS. Jika pengguna ingin transaksi mereka diprioritaskan, dia dapat mengirimkan transaksi dengan harga GAS yang lebih tinggi. Dengan cara ini, transaksi dapat diproses lebih cepat oleh penambang yang diberi insentif oleh biaya transaksi yang lebih tinggi. Sebagai kompensasi untuk sumber daya komputasi yang diinvestasikan oleh penambang, GAS menjadi lebih penting setelah konsensus bermigrasi ke Proof of Stake (POS). Di era POS, penambang tidak lagi dihargai dengan penambangan blok dan pengepakan transaksi, lebih penting bagi penambang untuk memproses transaksi dan dibayar untuk mengeluarkan sumber daya di blockchain.

Tujuan teoritis adalah untuk menyelaraskan insentif peserta di jaringan. Sebagian besar teori blockchain membahas bagaimana mengurangi aktor berbahaya atau jahat di lingkungan yang tidak dapat dipercaya. GAS sebagian mengatasi masalah ini: Penambang diberi insentif untuk bekerja di jaringan dan pengguna tidak diberi insentif untuk bertindak buruk atau menulis kode berbahaya karena mereka menempatkan eter mereka sendiri (dalam bentuk gas) dalam risiko.

Dari sudut pandang komputasi, alasan komputasi di balik GAS kembali ke aspek lama dan mendasar dari teori komputasi—Masalah Penghentian. Masalah Penghentian adalah masalah menentukan apakah program arbitrer akan berhenti berjalan atau apakah akan berjalan selamanya hanya dengan melihat deskripsi dan nilai input. Pada tahun 1936, Alan Turing memutuskan bahwa tidak mungkin bagi mesin apa pun untuk memecahkan Masalah Penghentian. Di EVM, ini berarti penambang tidak pernah dapat mulai memproses transaksi dan tahu 100% bahwa transaksi tidak akan berlangsung selamanya. Dengan GAS—khususnya, batas GAS—jumlah gas yang terbatas selalu melekat pada transaksi. Bahkan jika penambang mulai memproses transaksi yang dikodekan untuk berlanjut tanpa batas waktu—baik dari bug atau serangan pada jaringan—gas pada akhirnya akan

**3.2 Komponen GAS**

GAS dibagi menjadi tiga komponen: biaya GAS, harga GAS, dan batas GAS. Di Ethereum, rumus perhitungan biaya transaksi sangat sederhana:

|  |
| --- |
| Biaya Transaksi (Biaya Tx) = Biaya GAS Aktual \* Harga GAS |

Misalnya, jika transaksi membutuhkan 50 GAS untuk menyelesaikannya, dengan asumsi bahwa Harga GAS yang ditetapkan oleh pengguna adalah 2 Gwei, maka total biaya transaksi adalah 50 \* 2 1/4100 Gwei.

**3.2.1 Biaya GAS**

GAS Cost menunjukkan bahwa setiap opcode membutuhkan berapa banyak GAS, yang telah ditentukan sebelumnya dalam kertas kuning Ethereum. Misalnya, opcode "tambahan" membutuhkan tiga GAS, kurang memperhatikan fluktuasi harga eter. GAS untuk setiap opcode tidak akan diubah. Itulah mengapa GASdigunakan untuk memperkirakan biaya transaksi, bukan eter. Jika eter digunakan untuk biaya GAS, harganya mungkin sangat bervariasi.

**3.2.2 Harga GAS**

GAS Price menunjukkan bahwa satuan GAS sama dengan berapa banyak eter. Umumnya, Gwei digunakan sebagai satuan perhitungan. Satu Gwei sama dengan 1 miliar Wei, dan Gwei adalah 10 ^ 9 eter tepatnya. Artinya, 1 Gwei = 0,000000001 ETH. 1 Wei adalah satuan metrik terkecil untuk eter dan tidak dapat dibagi. Jika Anda menetapkan harga GAS ke 20 Gwei, itu berarti Anda akan membayar 0,00000002 eter untuk setiap langkah. Ternyata, semakin tinggi Harga GAS, semakin banyak yang akan Anda bayar. Ada beberapa situs web seperti ethgasstation.info yang memberikan harga rata-rata GAS. Terkadang, pengguna mungkin bersedia membayar harga yang lebih tinggi untuk membuat transaksi mereka mendapatkan prioritas dalam dipilih dan dieksekusi oleh penambang. Ini karena: GAS yang ditentukan dalam transaksi akan dikirim ke penambang dan penambang akan mengurutkan semua transaksi di pool lokal mereka berdasarkan harga GAS. Transaksi dengan harga GAS yang lebih tinggi akan memiliki peluang lebih besar daripada yang memiliki harga GAS lebih rendah.

**3.3.3 Limit GAS**

GAS Limit adalah batas atas penggunaan GAS untuk transaksi tertentu. Itu adalah langkah maksimum yang diperlukan untuk mengeksekusi transaksi Anda. Batas GAS akan lebih dari yang sebenarnya digunakan. Karena kompleksitas transaksi bervariasi, GAS yang sebenarnya digunakan hanya diketahui setelah transaksi selesai. Jadi sebelum Anda mengirimkan transaksi, Anda perlu menetapkan batas atas penggunaan GAS. Jika batas GAS ditetapkan terlalu rendah, penambang akan mencoba menyelesaikan transaksi hingga GAS habis. Penambang akan mendapatkan hadiah ketika GAS habis karena penambang telah menghabiskan waktu dan daya untuk transaksi pengguna. Dan di blockchain, transaksi akan diatur ke false. Mekanisme GAS diatur untuk melindungi pengguna dan penambang: mereka tidak akan kehilangan dana atau daya karena kode buggy dan serangan berbahaya.

**4. Ether (ETH)**

Ether adalah token yang dikeluarkan di Ethereum. ETH adalah simbol pendek dari eter. Dan ETH adalah mata uang kripto yang dapat diperdagangkan. Di Ethereum, Ether terutama digunakan untuk membayar biaya transaksi. Biaya transaksi sama dengan biaya GAS dikalikan dengan harga GAS, dan dibayarkan dalam ETH. Pengguna dapat menetapkan harga GAS, tetapi perlu diingat bahwa jika harga GAS terlalu rendah, ada kemungkinan tidak ada penambang yang mau mengemas transaksi.

**5. Account**

Setiap akun memiliki alamatnya sendiri. Di ruang alamat, ada dua jenis akun: satu adalah akun milik eksternal (EOA) yang dikendalikan oleh kunci publik/pribadi. Biasanya, hanya orang yang dapat memegang akun tersebut yang digunakan untuk menyimpan ETH; yang lainnya adalah akun kontrak yang dikendalikan oleh kode. Kedua jenis ini memiliki beberapa perbedaan, tetapi yang paling penting adalah hanya EOA yang dapat memulai transaksi.

Kedua jenis akun dapat mengirim dan menerima eter. Oleh karena itu, keduanya memiliki bidang keseimbangan untuk melacaknya. Namun, akun kontrak juga dapat menyimpan data. Oleh karena itu, mereka memiliki bidang penyimpanan, dan bidang kode yang berisi instruksi mesin tentang cara memanipulasi data yang disimpan. Dengan kata lain, akun kontrak adalah kontrak pintar yang hidup di blockchain.

**contract** account = balance + code + storage

**external** account = balance + empty + empty

**6. Transaction**

Transaksi adalah pesan yang dikirim dari akun ke akun lain. Kami dapat mentransfer ETH antar akun melalui pengiriman transaksi ke EOA. Jika akun target adalah akun kontrak, pengiriman transaksi akan memicu eksekusi kodenya. Karena setiap transaksi akan dieksekusi di semua node di Ethereum, eksekusi kode atau transaksi akan dicatat oleh blockchain.

Dalam bab ini, kami memperkenalkan semua konsep dan istilah yang diperlukan sebagai dasar untuk memahami bahasa pemrograman Solidity setelahnya. Kita akan mulai mengeksplorasi detail Solidity di bab-bab berikutnya.