Nov 30, 2022 4 min read

9 Biggest Gas Guzzlers in Solidity on Ethereum

Updated: Jan 9, 2023

solidity and ethereum gas cost

Improving smart contract gas fees

This article lists 9 of the most expensive Ethereum operations. If you are looking to increase the gas efficiency of your solidity smart contract, it's best to keep an eye out for the biggest costs so that you know where to find savings.

Biggest gas operations in Ethereum

1. Creating a smart contract: 32,000 gas

Anyone who has created a smart contract on Ethereum knows how expensive it is. Unsurprisingly, most expensive Ethereum transaction is to create a new contract, clocking in at 32k gas. Not only that, for each byte you deploy, it's an additional 200 gas. So if your contract is 20kb in size, it will cost over a whopping 4 million gas!

2. Normal ethereum transaction: 21,000

You can't do anything about this cost, but it tells you that you should minimize the number of transactions to save the user gas. For example, you can batch transactions with Multicall or skip the approval step in an ERC20 transfer with ERC20 Permit.

3. Initializing a storage variable: 20,000 gas

When you store a non-zero value in a storage variable for the first time, you pay 20,000 gas (actually more, but let's not get too technical for now). This is one of the most expensive every day operations smart contracts do, so limiting the amount of new data you store to the blockchain will make your contracts more efficient. Non-zero doesn't mean this only applies to integers. It also applies to addresses, boolean values, bytes… anything.
There is a sneaky cost of arrays because of this: when you push the first item onto the array, you also increase a storage variable that tracks the length, and you also pay the cost of storing the item. So the first push onto an array will generally be expensive. Try to avoid emptying and adding the first item to an array too many times.
When a variable has never been set, Ethereum treats it like it doesn't exist at all, so there is no storage cost until it is used.

4. Transfer of value: 9,000 gas

If a wallet transfers Ether, there is no extra cost. It's included in the 21,000. But if a smart contract sends Ether, each transfer costs 9,000 gas. Not quite as bad as storing a new variable, but can add up fast if you do multiple payments in one transaction.

5. Accessing a storage value 2,100 or 100

Remember how we said above it actually costs more than 20,000 gas to initialize a variable to a non-zero value? That's because every write operation to storage also must pay an access operation.
If you are just reading a storage value, it will cost 2,100 or 100 gas (explained momentarily). If you are writing, you must pay the write cost and the access cost. The first time you access a storage variable in a transaction, you must pay 2,100 gas. If you access that same variable within the same transaction, the cost will be 100 for the access. Consider a code where you are minting tokens up to a certain limit. You read the supply variable from storage to check you haven't exceeded the limit and pay 2,100 gas.

Then, you write to the storage variable to increment it if you haven't exceeded the limit. The second time you access it for increment, you pay 100 gas (plus the write cost).
In general, it's wasteful to pay that 100 gas. You should ***cache*** the storage variable before checking the limit, and increment that cache. Then write to storage **once** when you are done with it.

6. Changing a storage value 2,900 + (100 or 2,100)

Changing a storage variable from zero to non-zero costs at least 20,000 gas (plus 100 or 2,100 depending on whether the access was warm or cold). The reason this particular operation is expensive is because the Ethereum blockchain must create a new index for this variable. However, if the index already exists, then the cost is a lot less. In this case, the cost is 2,900.

7. Logging data (375 gas) 8 for each byte, 375 for each topic

Logging data (or emitting events) causes data to be stored on the blockchain permanently. Therefore, the user must pay the cost. However, this kind of storage is cheaper because logs cannot be accessed from a smart contract, so the indexing does not need to be accessible from the EVM. Of course, the more data you log, the more you pay. Logs can have data that is indexable or not (enabling someone using the geth client to quickly query a range of values). This kind of index is naturally more expensive.

8. Keccak256 (30 gas)

The Keccack256 hash actually costs more than 30 gas generally. If you hash a long string, it will naturally cost more than hashing an integer or an address. The keccak256 requires data to be stored in memory, and that has its own cost (which is usually negligible, so it is not in this list). However, the base cost is 30 gas; this makes sense, since it is a much more complicated operation than other mathematical operations (arithmetic operations cost between 3 and 5 gas).

9. Blockhash (20 gas)

The blockhash is typically used as a source of pseudo-randomness. You should be **very careful** trying to generate random numbers on the blockchain. The blockchain is deterministic, any method to generate random numbers fully on chain is necessarily predictable, at least to a certain degree. That said, if you chose to access the blockhash, it will cost you 20 gas. Why is this more expensive than other global variables msg.sender or block.number? The blockhash variable allows you to look back as much as 256 blocks, so that's more information the node needs to keep track of.

Want to become a gas optimization expert? Apply to our solidity bootcamp.