**A Comprehensive Guide To Gas And Gas Price In Solidity**

[[Gabriel Delight](https://medium.com/@gabrieldelight08?source=post_page-----bfb9c00970af--------------------------------)](https://medium.com/@gabrieldelight08?source=post_page-----bfb9c00970af--------------------------------)

[[StackAnatomy](https://medium.com/stackanatomy?source=post_page-----bfb9c00970af--------------------------------)](https://medium.com/stackanatomy?source=post_page-----bfb9c00970af--------------------------------)

[Gabriel Delight](https://medium.com/@gabrieldelight08?source=post_page-----bfb9c00970af--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F471488744e3b&operation=register&redirect=https%3A%2F%2Fmedium.com%2Fstackanatomy%2Fa-comprehensive-guide-to-gas-and-gas-price-in-solidity-bfb9c00970af&user=Gabriel+Delight&userId=471488744e3b&source=post_page-471488744e3b----bfb9c00970af---------------------post_header-----------)

Published in

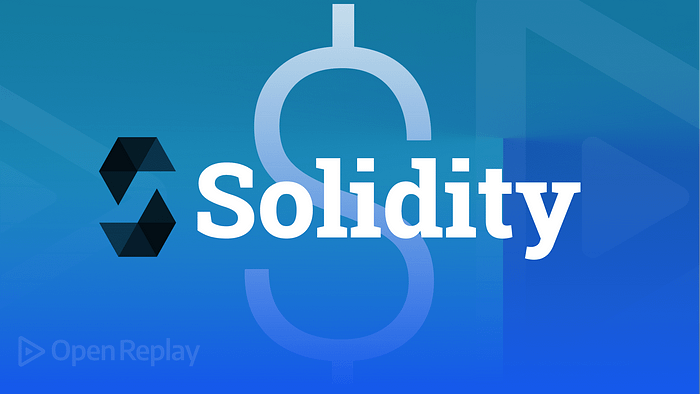
[StackAnatomy](https://medium.com/stackanatomy?source=post_page-----bfb9c00970af--------------------------------)

·

11 min read

·

Feb 15, 2023



The execution cost of a function or transaction on the [Ethereum network](https://ethereum.org/en/) is referred to as “gas” in the context of Solidity. The amount of [Ether](https://docs.soliditylang.org/en/v0.8.17/units-and-global-variables.html?highlight=Ether+#ether-units) that a user is prepared to spend on each unit of gas is known as the “gas price.”

The cost of gas significantly affects how quickly the network will complete a transaction. Because miners are more inclined to give larger reward transactions priority, transactions with higher gas prices will often be processed more rapidly than those with lower gas prices. It’s important to consider that the cost of gas is not constant and is subject to change depending on supply and demand. The cost of gas may be greater during periods of high demand and lower during periods of low demand. When submitting a transaction, users can indicate the gas price they are willing to pay.

Gas in Solidity is a defense against attacks that would cause the Ethereum network to run out of resources. Because Ethereum runs [smart contracts](https://docs.soliditylang.org/en/v0.8.17/introduction-to-smart-contracts.html?highlight=Ethereum+network) on a [decentralized platform](https://aws.amazon.com/blockchain/decentralization-in-blockchain/), it is feasible for malicious people to try to exhaust all of the resources to interfere with the network. To prevent this from happening, the Ethereum network uses a system called “gas” to limit the amount of work that can be done in a single transaction or function execution.

Each operation in Solidity has a gas cost associated with it, and the total gas cost for a given transaction or function execution is the sum of the gas costs of all the operations it includes. The maximum quantity of gas the user is willing to spend for the operation is specified when they submit a transaction or call a function. The operation will be canceled and any changes performed will be undone if the gas runs out before it is finished. This keeps resource exhaustion threats at bay and guarantees the stability and security of the Ethereum network.

**Why does Gas reduce the amount of Computation?**

The amount of processing that a transaction necessitates is not directly correlated with the gas price in Solidity. Instead, the price of gas reflects the value that the transaction’s sender is willing to exchange for each unit of gas. The maximum quantity of gas you are prepared to pay for the calculation must be specified when sending a transaction on the Ethereum network, as well as the gas price, which is the amount of Ether (the network’s native coin) you are willing to pay for each unit of gas.

**Your chance of having your transaction appear in the following block mined on the Ethereum blockchain increases with the amount of gas you are ready to pay.** This is because miners who are in charge of validating and adding transactions to the blockchain are more likely to give higher-gas-priced transactions priority because they will generate a larger profit from processing them. The amount of computation necessary for a transaction does not, however, directly decrease with an increase in the price of gas. Instead of making your transaction more appealing to miners, it just increases the likelihood that it will be executed promptly. The actions you are doing can be made simpler, or you can use more effective methods to lower the amount of processing needed for a transaction.

**What happens if a transaction runs out of Gas?**

On the Ethereum network, if a transaction runs out of gas while it is being processed, it will be aborted, and all the modifications it made to the blockchain will be undone. The gas used to send the transaction will not be reimbursed to the sender.

The Ethereum network has a system in place to make sure that transactions can’t consume computing power indefinitely. The Ethereum network can guarantee that all transactions can be processed quickly and effectively without any one transaction monopolizing the available resources by limiting the amount of gas that each transaction is permitted to use.

To ensure that you set an adequate gas limit for your transactions while creating a smart contract in Solidity, it is crucial to carefully examine how much gas your contract will consume. Your transactions could run out of gas and fail if you don’t allow enough gas, which could be problematic for your contract or the users of your contract.

**What is a Gas Price Unit?**

The amount of Ether (the Ethereum network’s native coin) that the sender of a transaction is willing to pay out for each unit of gas is known as the gas price in the [Ethereum blockchain](https://docs.soliditylang.org/en/v0.8.17/index.html?highlight=e+Ethereum+blockchain#getting-started). The maximum quantity of gas you are ready to spend for the calculation must be specified when sending a transaction on the Ethereum network, as well as the gas price, or the amount of ether you are willing to pay for each unit of gas.

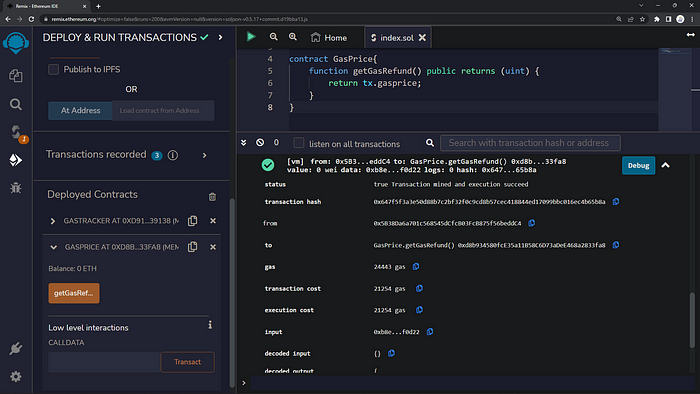
The smallest unit of Ether, known as “wei,” is used to describe the cost of gas. One Ether has 1,000,000,000,000,000,000 wei. Wei is the most common unit used to specify the price of gas, however, it is also possible to use wei or ether (1 ether equals 1,000,000,000,000,000,000 wei).

**How to get Gas Price in Solidity using Remix**

The “gasPrice” keyword in Solidity can be used to retrieve the most recent gas price for a transaction. The number of wei that the sender of the transaction is willing to pay for each unit of gas is returned by the “gasprice” keyword as a uint value. Here is an example of how you can use the “gasprice” keyword in Solidity:

pragma solidity ^0.5.3;  
  
contract GasPrice {  
 function getGasRefund() public returns (uint) {  
 return tx.gasprice;  
 }  
}

Output:



To test this contract using Remix, you can do the following:

1. Open Remix in your web browser ([https://remix.ethereum.org](https://remix.ethereum.org/)).
2. Click on the “Compile” tab to compile the contract.
3. Click on the “Run” tab to open the Remix runtime environment.
4. In the “Deployed Contracts” section, click on the “Create” button to deploy a new instance of the contract.
5. In the “Instance” section, click on the “gasprice” button to call the “gasprice” function.

The “gasprice” function will return the current gas price for the transaction as a uint value (please make sure you click the getGasRefund button). You can then use this value to determine the amount of Ether that the person sending the transaction is willing to pay for each unit of gas.

Please note that the gas price returned by the “gasprice” keyword may vary depending on the current state of the Ethereum network. The gas price is determined by the market demand for computational resources on the network, and it may fluctuate over time.

**How to reduce Gas Price in a transaction**

We’ll talk about how to reduce gas prices in a transaction in this part. There are a few techniques to lower the gas cost for an Ethereum transaction.

* Optimize the procedures being used: A transaction will require more gas to execute the more complicated the operations being done. Reduce the amount of gas required for the transaction by simplifying the procedures being carried out, which will lower the cost of gas.
* Often use effective algorithms: Some algorithms are more effective than others for some tasks. You can lower the amount of gas required for a transaction and, as a result, lower the cost of gas by using more efficient algorithms.
* Use a lower gas price: When you send a transaction, you can specify the gas price you are willing to pay for the computation. By specifying a lower gas price, you can reduce the cost of the transaction. However, be aware that if the gas price is too low, your transaction may not be processed promptly, as miners are likelier to prioritize transactions with higher gas prices.
* Use a less expensive blockchain: Due to variations in their architectures and the demand for computational resources, some blockchain networks have lower gas prices than others. If possible, you can lower the cost of your transactions by using a less expensive blockchain network.

**What is the Gas limit?**

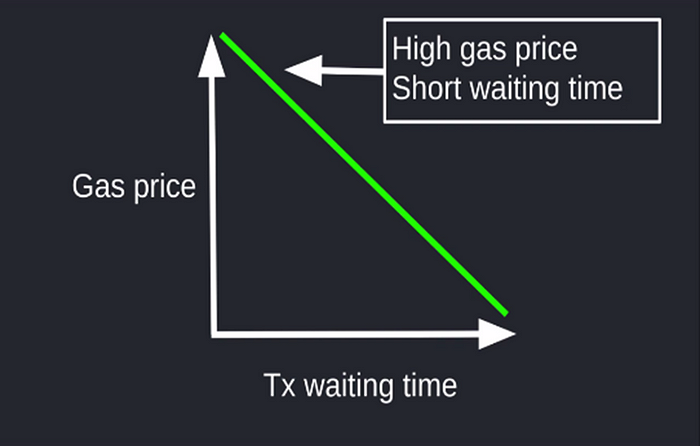
The maximum quantity of gas a transaction can use on the Ethereum blockchain is known as the gas limit. On the Ethereum network, you must mention the gas price or the amount of ether you are willing to pay for each unit of gas, as well as the maximum amount of gas you are willing to pay for the calculation when sending a transaction.

The gas limit is a crucial safety precaution that aids in preventing transactions from using an excessive portion of the Ethereum network’s computing capacity. A transaction will be canceled, and all modifications made to the blockchain will be undone if it uses more gas than allotted.

In Solidity, you can specify the gas limit for a transaction using the “gas” keyword. For example:

function myFunction() public {  
 // Set the gas limit to 100,000  
 uint gasLimit = 100000;  
 // Send a transaction with a gas limit of 100,000  
 address(this).transfer(msg.value);  
}

The gas limit must be carefully considered while creating your transactions or contracts. And you should always consider “the higher Gas limit set, the more Ether you spend”.



You should also note that setting the gas limit too high can lead to additional costs, while setting it too low can lead to rejected transactions. You should try to fix the gas limit as precisely as possible based on the difficulty of the activities being done and the expected gas consumption of the transaction.

**What is Block Gas Limit**

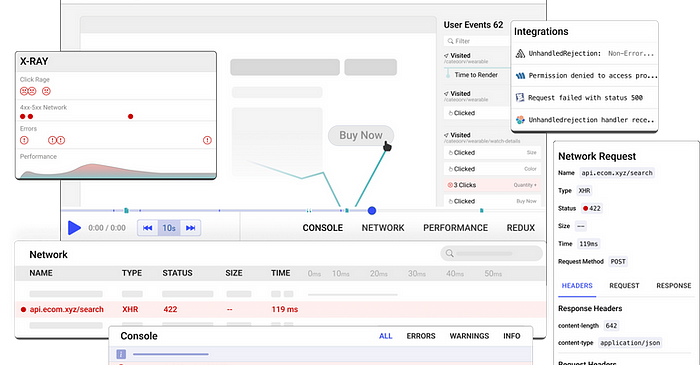
The maximum quantity of gas that all transactions in a block on the Ethereum blockchain are permitted to use is known as the block gas limit. The Ethereum network’s block gas limit is a safety precaution that works to stop blocks from using too many computing resources. Some transactions will be rejected and excluded from the block if the total gas consumption of all the transactions in the block is more than the block gas limit.

The Ethereum network establishes the block gas limit, and it is modified over time to ensure the network can continue processing transactions effectively without running out of computing power. The current demand for computing resources on the network may have an impact on the block gas limit, which is not a constant number.

When creating your transactions or contracts, it’s crucial to consider the block gas limit because if it’s set too high, it could result in extra expenses. Based on the difficulty of the activities being done and the anticipated gas consumption of the transaction, you should try to fix the gas limit as precisely as you can.

**Session Replay for Developers**

*Uncover frustrations, understand bugs and fix slowdowns like never before with*[***OpenReplay***](https://github.com/openreplay/openreplay)*— an open-source session replay suite for developers. It can be****self-hosted****in minutes, giving you complete control over your customer data*



*Happy debugging!*[*Try using OpenReplay today.*](https://openreplay.com/)

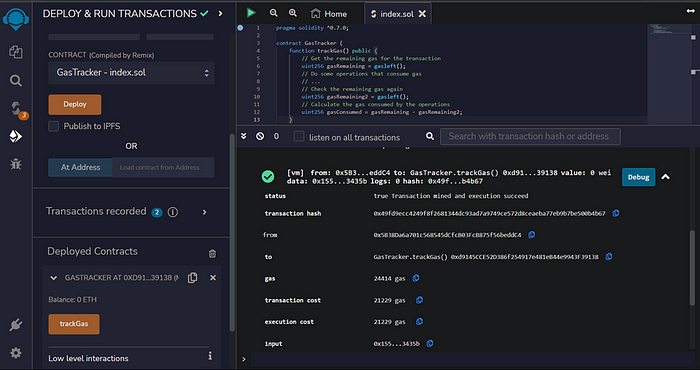
**How to calculate Gas in Solidity**

In Solidity, you can use the “gasleft()” function to get the remaining amount of gas for the current transaction. The “gasleft()” function returns a uint256 value, the number of units of gas remaining for the transaction. You can use this value to track the gas consumption of your contract and to ensure that you do not exceed the gas limit for the transaction.

Here is an example of how you can use the “gasleft()” function in Solidity:

pragma solidity ^0.7.0;  
  
contract GasTracker {  
 function trackGas() public {  
 // Get the remaining gas for the transaction  
 uint256 gasRemaining = gasleft();  
 // Do some operations that consume gas  
 // ...  
 // Check the remaining gas again  
 uint256 gasRemaining2 = gasleft();  
 // Calculate the gas consumed by the operations  
 uint256 gasConsumed = gasRemaining - gasRemaining2;  
 }  
}

Output:



In this example, the “trackGas()” function gets the remaining gas for the transaction, performs some operations that consume gas, and then gets the remaining gas again. The difference between the initial gas remaining and the final gas remaining is the amount of gas that was consumed by the operations.

You can use this technique to track the gas consumption of your contract and to ensure that you do not exceed the gas limit for the transaction. You can also use this information to optimize your contract code by identifying areas where you can reduce the gas needed to execute the contract.

**Basic Solidity-Based Gas Optimization Process**

Here are some general steps that you can follow to optimize the gas consumption of your Solidity-based contracts:

* the data structures and methods utilized in your contract: For completing particular tasks, some algorithms and data structures are more effective than others. You can lower the gas required to execute your contract by selecting more efficient algorithms and data structures.
* Avoid using loops: ***Please, this is one of the parts you should consider the most***. Loops can be very gas-intensive, especially if they are used to perform complex operations. If possible, you should try to avoid using loops in your contract.
* Use the “inline” keyword: When calling a function from another contract, you can use the “inline” keyword to specify that the called function should be inlined into the calling function rather than being executed as a separate call. This can help reduce your contract’s gas consumption, as it eliminates the need for an additional call.
* Use “view” or “pure” functions: Functions that are marked as “view” or “pure” do not modify the state of the contract and therefore do not consume any gas. If you have functions that do not need to modify the state of the contract, you should mark them as “view” or “pure” to reduce the gas consumption of your contract. Please [click here](https://docs.soliditylang.org/en/v0.8.17/types.html?highlight=function#function-types) to learn more about functions in solidity.

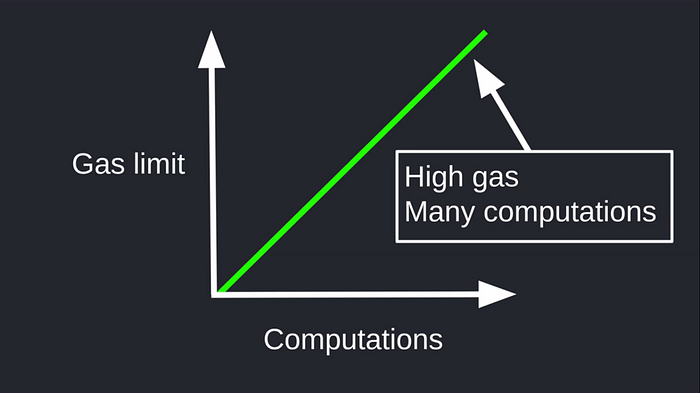
**Gas Statistics and basic principles**

Every operation and computation in the [Ethereum virtual machine](https://docs.soliditylang.org/en/v0.8.17/introduction-to-smart-contracts.html?highlight=Ethereum+network#index-6) (EVM) has a gas cost attached to it. A computation’s resource usage (such as CPU and memory usage) directly relates to how much gas the computation uses. The total cost of a contract or transaction is determined by adding the gas costs of all the operations carried out.

The goal of the gas system is to limit the number of resources a transaction can use and prevent transactions from executing indefinitely. This ensures that the Ethereum network can process transactions predictably and efficiently.

Here is a fundamental principle to remember while dealing with the Gas Price Limit in [Smart Contracts](https://docs.soliditylang.org/en/v0.8.17/introduction-to-smart-contracts.html?highlight=Ethereum+network):

The higher the Gas limit, the higher the computation (, The faster the transaction):



You can see from the figure above that the high Gas Limit set in the [smart contract](https://docs.soliditylang.org/en/v0.8.17/introduction-to-smart-contracts.html?highlight=Ethereum+network) will increase transaction speed.

It is essential to carefully consider the gas costs of your contract and its operations. If the gas cost of a function is too high, it may be difficult or even impossible for users to execute the function. On the other hand, if the gas cost is too low, the function may be vulnerable to being exploited by attackers.

**Conclusion**

I hope you found reading this post enjoyable. This article included topics like gas, gas prices in solidity, and the role gas plays in transactions. We also discussed a few aspects of calculating gas pricing, and it is my earnest goal that this post has helped us comprehend the significance of gas and gas prices while creating smart contracts. After reading this article, I will encourage you to study more about using gas and writing some practical code for resolving Gas problems by consulting the [official documentation](https://docs.soliditylang.org/en/v0.8.17/introduction-to-smart-contracts.html?highlight=gas).



*Originally published at*[*https://blog.openreplay.com*](https://blog.openreplay.com/introducing-the-phonon-css-framework/)*.*

[Blockchain](https://medium.com/tag/blockchain?source=post_page-----bfb9c00970af---------------blockchain-----------------)

[Gas](https://medium.com/tag/gas?source=post_page-----bfb9c00970af---------------gas-----------------)