Gas Schedule Proposal

Overview

Based on the current research, we propose a new gas cost schedule for the Ethereum Virtual Machine (EVM). The current gas schedule is based on the original yellow paper and has rarely been updated since the launch of the Ethereum mainnet. The current gas schedule has several issues that have been identified in the research. The proposed gas schedules aim to address these issues and provide a more accurate representation of the computational cost of EVM operations.

While the measurements in the Stage IV report are based on solid research, the proposed gas cost schedule is more subjective. This is why we propose two different gas schedules: a conservative one and a radical one. Each has pros and cons.

Conservative Gas Schedule Proposal

The idea behind the conservative gas schedule is to limit changes only to the most mispriced elements. By doing so, we aim to minimize the impact on the existing ecosystem while still improving security. This should also be easier to implement, as it requires fewer changes to the existing codebases in EVM clients.

The opcodes proposed for change are those that have been identified to be mispriced during the research.

Opcode	Name	Current Gas	Proposed Gas
08	MULMOD *	8	14
0A	EXP	10 + 50 * exponent_byte_size	10 + 20 * exponent_byte_size
20	KECCAK256 *	30 + 6 * minimum_word_size + memory_expansion_cost	50 + 30 * minimum_word_size + memory_expansion_cost
30	ADDRESS *	2	5
33	CALLER *	2	5
37	CALLDATACOPY *	3 + 3 * minimum_word_size + memory_expansion_cost	5 + 3 * minimum_word_size + memory_expansion_cost
39	CODECOPY *	3 + 3 * minimum_word_size + memory_expansion_cost	5 + 3 * minimum_word_size + memory_expansion_cost
3E	RETURNDATACOPY *	3 + 3 * minimum_word_size + memory_expansion_cost	5 + 3 * minimum_word_size + memory_expansion_cost
52	MSTORE *	3 + memory_expansion_cost	5 + memory_expansion_cost
53	MSTORE8 *	3 + memory_expansion_cost	5 + memory_expansion_cost
60 - 7F	PUSHx	3	2
80 - 8F	DUPx	3	2
90 - 9F	SWAPx	3	2

Opcode	Name	Current Gas	Proposed Gas
56	JUMP	8	3
57	JUMPI	10	5
5C	TLOAD	100	20
5D	TSTORE	100	50
A0	LOG0	375 + 375 * topic_count + 8 * data_size + memory_expansion_cost	35 + 35 * topic_count + 8 * data_size + memory_expansion_cost
A1	LOG1	375 + 375 * topic_count + 8 * data_size + memory_expansion_cost	35 + 35 * topic_count + 8 * data_size + memory_expansion_cost
A2	LOG2	375 + 375 * topic_count + 8 * data_size + memory_expansion_cost	35 + 35 * topic_count + 8 * data_size + memory_expansion_cost
A3	LOG3	375 + 375 * topic_count + 8 * data_size + memory_expansion_cost	35 + 35 * topic_count + 8 * data_size + memory_expansion_cost
A4	LOG4	375 + 375 * topic_count + 8 * data_size + memory_expansion_cost	35 + 35 * topic_count + 8 * data_size + memory_expansion_cost
F1	CALL *	address_access_cost: 100 (warm) / 2600 (cold)	address_access_cost: 150 (warm) / 2600 (cold)
F4	DELEGATECALL *	address_access_cost: 100 (warm) / 2600 (cold)	address_access_cost: 150 (warm) / 2600 (cold)
FA	STATICCALL *	address_access_cost: 100 (warm) / 2600 (cold)	address_access_cost: 150 (warm) / 2600 (cold)
F3	RETURN *	0 + memory_expansion_cost	5 + memory_expansion_cost
FD	REVERT *	0 + memory_expansion_cost	8 + memory_expansion_cost

Precompile	Name	Current Gas	Proposed Gas
01	ECRECOVER *	3000	12000
06	ECADD *	150	3000
07	ECMUL *	6000	10000
0A	POINTEVAL *	50000	300000

> The * indicates increased gas cost for the given opcode or precompile. This should be implemented with caution as it might break backward compatibility.

The current cost for the `memory_expansion_cost` is calculated as `quadratical_cost + 3 * memory_word_count`. We propose to change it to `quadratical_cost + 2 * memory_word_count` as the results indicate that memory expansion costs are quite low.

Radical Gas Schedule Proposal

The idea behind the radical gas schedule proposal is a complete overhaul of the current gas schedule. Rather than just addressing the most mispriced opcodes, we propose altering all opcodes to better reflect the computational cost of the operations.

Let's run through the consequences of the radical gas schedule proposal. The cheapest operations are priced at 1. Then we gradually adjust the gas cost for more complex operations. As a result, most arithmetic and basic opcodes will be much cheaper, i.e., valued at 1 rather than 3 or 5. Then all other operations will be adjusted accordingly, usually by lowering the gas cost. This matches some of the sentiments in the community.

> Client Implementation notes:
>
> Such a radical change to the gas schedule would require a fully configurable schedule in EVM clients. This would allow clients to easily switch between different gas schedules, but also different chains.

In this scenario, the storage cost remains at the same level as this reflects the network cost of storing data. Thus the radical gas schedule proposal increases the disparity between the cost of storage and computation. This is a good thing as it makes it more expensive to store data than to compute it. The memory expansion cost is lowered but still keeps its characteristic of being quadratic, thus improving network security.

Pros:

- The gas cost reflects the computational cost of the operations
- The larger gap between the cost of storage and computation promotes more efficient use of the network
- Configurable gas schedules allow for simpler updates in the future
- Configurable gas schedules can better match L2 chain requirements

Cons:

- EVM Clients need to implement configurable gas schedules
- The radical changes may lead to unforeseen issues or challenges

The radical gas schedule was derived by rescaling the calculated gas costs from the research. The rescale factor is the key to achieving the desired effect. For this purpose, we took an average of the basic arithmetic opcodes. In this proposal, the rescale factor is `1/4.6 = 0.217391304`.

The tables below contain the additional Rescaled Fractional column. This shows the actual gas cost of the opcode after rescaling. It could be useful for further discussion on the proposed Fractional Gas Costs schedule.

Opcode	Name	Current Gas	Rescaled Fractional	Proposed Gas
01	ADD	3	0.5	1
02	MUL	5	1.1	1
03	SUB	3	0.6	1
04	DIV	5	0.9	1
05	SDIV	5	1.4	1
06	MOD	5	1.0	1
07	SMOD	5	1.5	1

Opcode	Name	Current Gas	Rescaled Fractional	Proposed Gas
08	ADDMOD	8	1.8	2
09	MULMOD	8	3.0	3
0A	EXP	10 + 50 * exponent_byte_size		2 + 4 * exponent_byte_size
0B	SIGNEXTEND	5	1.1	1
10	LT	3	0.5	1
11	GT	3	0.6	1
12	SLT	3	0.7	1
13	SGT	3	0.7	1
14	EQ	3	0.6	1
15	ISZERO	3	0.4	1
16	AND	3	0.5	1
17	OR	3	0.6	1
18	XOR	3	0.6	1
19	NOT	3	0.4	1
1A	BYTE	3	0.6	1
1B	SHL	3	1.2	1
1C	SHR	3	0.9	1
1D	SAR	3	1.4	1
20	KECCAK256	30 + 6 * data_word_size + memory_expansion_cost		10 + 6 * data_word_size + memory_expansion_cost
30	ADDRESS	2	1.1	1
32	ORIGIN	2	0.5	1
33	CALLER	2	1.0	1
34	CALLVALUE	2	0.4	1
35	CALLDATALOAD	3	0.7	1
36	CALLDATASIZE	2	0.4	1
37	CALLDATACOPY	3 + 3 * data_word_size + memory_expansion_cost		1 + 1 * data_word_size + memory_expansion_cost
38	CODESIZE	2	0.5	1
39	CODECOPY	3 + 3 * data_word_size + memory_expansion_cost		1 + 1 * data_word_size + memory_expansion_cost

Opcode	Name	Current Gas	Rescaled Fractional	Proposed Gas
3A	GASPRICE	2	0.4	1
3B	EXTCODESIZE	address_access_cost		address_access_cost
3C	EXTCODECOPY	0 + 3 * data_word_size + memory_expansion_cost + address_access_cost		0 + 1 * data_word_size + memory_expansion_cost + address_access_cost
3D	RETURNDATASIZE	2	0.5	1
3E	RETURNDATACOPY	3 + 3 * data_word_size + memory_expansion_cost		1 + 1 * data_word_size + memory_expansion_cost
3F	EXTCODEHASH	address_access_cost		address_access_cost
41	COINBASE	2	0.6	1
42	TIMESTAMP	2	0.5	1
43	NUMBER	2	0.5	1
45	GASLIMIT	2	0.4	1
46	CHAINID	2	0.5	1
47	SELFBALANCE	5	1.3	1
50	POP	2	0.4	1
51	MLOAD	3	1.0	1
52	MSTORE	3 + memory_expansion_cost		1 + memory_expansion_cost
53	MSTORE8	3 + memory_expansion_cost		1 + memory_expansion_cost
56	JUMP	8	0.7	1
57	JUMPI	10	1.1	1
58	PC	2	0.4	1
59	MSIZE	2	0.4	1
5A	GAS	2	0.4	1
5C	TLOAD	100	4.1	4
5D	TSTORE	100	10.0	10
5B	JUMPDEST	1	0.3	1
5E	MCOPY	3 + 3 * data_word_size + memory_expansion_cost		1 + 1 * data_word_size + memory_expansion_cost
5F	PUSH0	2	0.4	1
60 - 7F	PUSHx	3	0.5	1

Opcode	Name	Current Gas	Rescaled Fractional	Proposed Gas
80 - 8F	DUPx	3	0.3	1
90 - 9F	SWAPx	3	0.5	1
A0	LOG0	375 + 375 * topic_count + 8 * data_size + memory_expansion_cost		7 + 7 * topic_count + 8 * data_size + memory_expansion_cost
A1	LOG1	375 + 375 * topic_count + 8 * data_size + memory_expansion_cost		7 + 7 * topic_count + 8 * data_size + memory_expansion_cost
A2	LOG2	375 + 375 * topic_count + 8 * data_size + memory_expansion_cost		7 + 7 * topic_count + 8 * data_size + memory_expansion_cost
A3	LOG3	375 + 375 * topic_count + 8 * data_size + memory_expansion_cost		7 + 7 * topic_count + 8 * data_size + memory_expansion_cost
A4	LOG4	375 + 375 * topic_count + 8 * data_size + memory_expansion_cost		7 + 7 * topic_count + 8 * data_size + memory_expansion_cost
F0	CREATE	32000 + 2 * data_word_size + memory_expansion_cost + deployment_code_execution_cost + 200 * deployed_code_size		32000 + 1 * data_word_size + memory_expansion_cost + deployment_code_execution_cost + 40 * deployed_code_size
F5	CREATE2	32000 + 2 * data_word_size + 6 * data_word_size + memory_expansion_cost + deployment_code_execution_cost + 200 * deployed_code_size		32000 + 1 * data_word_size + 1 * data_word_size + memory_expansion_cost + deployment_code_execution_cost + 40 * deployed_code_size
F1	CALL	0 + memory_expansion_cost + code_execution_cost + address_access_cost + positive_value_cost + value_to_empty_account_cost		0 + memory_expansion_cost + code_execution_cost + address_access_cost + positive_value_cost + value_to_empty_account_cost
FA	STATICCALL	0 + memory_expansion_cost + code_execution_cost + address_access_cost		0 + memory_expansion_cost + code_execution_cost + address_access_cost
F4	DELEGATECALL	0 + memory_expansion_cost + code_execution_cost + address_access_cost		0 + memory_expansion_cost + code_execution_cost + address_access_cost
F3	RETURN [1]	0 + memory_expansion_cost		0 + memory_expansion_cost

Opcode	Name	Current Gas	Rescaled Fractional	Proposed Gas
FD	REVERT [2]	0 + memory_expansion_cost		0 + memory_expansion_cost

[1] The calculated gas costs for RETURN is `1 + memory_expansion_cost`. To avoid price increase this is kept at the current level and needs to be subsidized by the network.

[2] The calculated gas costs for REVERT is `2 + memory_expansion_cost`. To avoid price increase this is kept at the current level and needs to be subsidized by the network.

Precompile	Name	Current Gas	Rescaled Fractional	Proposed Gas
01	ECRECOVER [3]	3000	3246.7	3000
02	SHA2-256	60 + 12 * data_word_size		10 + 4 * data_word_size
03	RIPEMD-160	600 + 120 * data_word_size		60 + 40 * data_word_size
04	IDENTITY	15 + 3 * data_word_size		15 + 3 * data_word_size
05	MODEXP	0 + max(200, complexity_cost)		0 + max(70, complexity_cost)
06	ECADD [3]	150	694.6	150
07	ECMUL	6000	2677.7	2700
08	ECPAIRING	45000 + 34000 * sets_count		8000 + 7000 * sets_count
09	BLAKE2F	0 + 1 * rounds_count		0 + 1 * rounds_count
0A	POINTEVAL	50000	21242.8	21000

[3] The proposed gas is subsidized by the network to keep the cost at the current level.

Also, the following elements of the dynamic gas cost have been adjusted:

Element	Current	Proposed	Notes
memory_expansion_cost	(memory_size_word ** 2) / 512 + (3 * memory_size_word)	(memory_size_word ** 2) / 512	This means that the first 22 words of memory are free. Then the cost grows quadratically.
address_access_cost	100 (warm) | 2600 (cold)	5 (warm) | 2600 (cold)	
MODEXP complexity_cost	multiplication_complexity * calculate_iteration_count / 3	multiplication_complexity * calculate_iteration_count / 30	This is optional, due to the relatively low popularity of the MODEXP precompile.