EIP 2929: Gas cost increases for state access opcodes

Return to Ethereum ERCs

: **Note**: The following is an excerpt from the official Ethereum site. It is provided here as a convenience and is not authoritative. Refer to the original document as the authoritative reference.

Table 1: Data sheet for Gas cost increases for state access opcodes

Title	Gas cost increases for state access opcodes
EIP	2929
Author	• Vitalik Buterin (@vbuterin), • Martin Swende (@holiman)
Status	final
Created	2020-09-01
Description	https://github.com/ethereum/EIPs/blob/master/EIPS/eip-2929.md
Specification	https://github.com/ethereum/EIPs/blob/master/EIPS/eip-2929.md#Specification
Category	Core

Simple Summary

Increases gas cost for **SLOAD**, ***CALL**, **BALANCE**, **EXT***, and **SELFDESTRUCT** when used for the first time in a transaction.

Abstract

Increase the gas cost of **SLOAD (0x54)** to 2100, and the ***CALL** opcode family (**0xf1**, **f2**, **f4**, **fA**), **BALANCE 0x31** and the **EXT*** opcode family (**0x3b**, **0x3c**, **0x3f**) to 2600. Exempts (i) precompiles, and (ii) addresses and storage slots that have already been accessed in the same transaction, which get a decreased gas cost. Additionally reforms SSTORE metering and **SELFDESTRUCT** to ensure “de-facto storage loads” inherent in those opcodes are priced correctly.

Motivation

Generally, the main function of gas costs of opcodes is to be an estimate of the time needed to process that opcode, the goal being for the gas limit to correspond to a limit on the time needed to process a block. However, storage-accessing opcodes (**SLOAD**, as well as the ***CALL**, **BALANCE** and **EXT*** opcodes) have historically been underpriced. In the 2016 Shanghai DoS attacks, once the most serious client bugs

were fixed, one of the more durably successful strategies used by the attacker was to simply send transactions that access or call a large number of accounts.

Gas costs were increased to mitigate this, but recent numbers suggest they were not increased enough. Quoting https://arxiv.org/pdf/1909.07220.pdf:

> *Although by itself, this issue might seem benign,* ***EXTCODESIZE*** *forces the client to search the contract ondisk, resulting in IO heavy transactions. While replaying the Ethereum history on our hardware, the malicious transactions took around 20 to 80 seconds to execute, compared to a few milliseconds for the average transactions*

This proposed EIP increases the costs of these opcodes by a factor of ~3, reducing the worst-case processing time to ~7-27 seconds. Improvements in database layout that involve redesigning the client to read storage directly instead of hopping through the Merkle tree would decrease this further, though these technologies may take a long time to fully roll out, and even with such technologies the IO overhead of accessing storage would remain substantial.

A secondary benefit of this EIP is that it also performs most of the work needed to make stateless witness sizes in Ethereum acceptable. Assuming a switch to binary tries, the theoretical maximum witness size not including code size (hence “most of the work” and not “all”) would decrease from **`(12500000 gas limit) / (700 gas per BALANCE) * (800 witness bytes per BALANCE) ~= 14.3M bytes`** to **`12500000 / 2600 * 800 ~= 3.85M bytes`**. Pricing for code access could be changed when code merklization is implemented.

In the further future, there are similar benefits in the case of SNARK/STARK witnesses. Recent numbers from Starkware suggest that they are able to prove 10000 Rescue hashes per second on a consumer desktop; assuming 25 hashes per Merkle branch, and a block full of state accesses, at present this would imply a witness would take **`12500000 / 700 * 25 / 10000 ~= 44.64 seconds`** to generate, but after this EIP that would reduce to **`12500000 / 2500 * 25 / 10000 ~= 12.5 seconds`**, meaning that a single desktop computer would be able to generate witnesses on time under any conditions. Future gains in STARK proving could be spent on either

> (i) using a more expensive but robust hash function or
> (ii) reducing proving times further, reducing the delay, and hence improving the user experience of stateless clients that rely on such witnesses.