Gas and ink in Stylus

Gas and ink are the pricing primitives that are used to determine the cost of handling specific opcodes and host I/Os on Stylus. For an overview of specific opcode and host I/O costs, seeOpcode and host I/O pricing .

ALPHA RELEASE, PUBLIC PREVIEW DOCS Stylus is currently tagged as analpha release. The code has not been audited, and shouldnot be used in production scenarios . This documentation is currently inpublic preview .

To provide feedback, click theRequest an update button at the top of this documentjoin the Arbitrum Discord , or reach out to our team directly by completingthis form .

Stylus gas costs

Stylus introduces new pricing models for WASM programs. Intended for high-compute applications, Stylus makes the following more affordable:

- Compute, which is generally10-100x
- cheaper depending on the program. This is primarily due to the efficiency of the WASM runtime relative to the EVM, and the quality of the code produced by Rust, C, and C++ compilers. Another factor that matters is the quality of the code itself. For example, highly optimized and audited C libraries that implement a particular cryptographic operation are usually deployable without modification and perform exceptionally well. The fee reduction may be smaller for highly optimized Solidity that makes heavy use of native precompiles vs. an unoptimized Stylus equivalent that doesn't do the same.
- Memory, which is100-500x
- cheaper due to Stylus's novel exponential pricing mechanism intended to address Vitalik's concerns with the EVM's per-call,quadratic memory pricing policy
- . For the first time ever, high-memory applications are possible on an EVM-equivalent chain.
- Storage, for which the Rust SDK promotes better access patterns and type choices. Note that while the underlyingSLOAD
- andSSTORE
- operations cost as they do in the EVM, the Rust SDK implements an optimal caching policy that minimizes their use. Exact savings depends on the program.
- VM affordances, including common operations likekeccak
- and reentrancy detection. No longer is it expensive to make safety the default.

There are, however, minor overheads to using Stylus that may matter to your application:

- The first time a WASM is deployed, it must beactivated
- . This is generally a few million gas, though to avoid testnet DoS, we've set it to a fixed 14 million. Note that you do not have to activate future copies of the same program. For example, the same NFT template can be deployed many times without paying this cost more than once. We will soon make the fees paid depend on the program, so that the gas used is based on the complexity of the WASM instead of this very conservative, worst-case estimate.
- Calling a Stylus program costs 128-2048 gas. We're working with Wasmer to improve setup costs, but there will likely always be some amount of gas one pays to jump into WASM execution. This means that if a contract does next to nothing, it may be cheaper in Solidity. However if a contract starts doing interesting work, the dynamic fees will quickly make up for this fixed-cost overhead.

Though conservative bounds have been chosen for testnet, all of this is subject to change as pricing models mature and further optimizations are made. Since gas numbers will vary across updates, it may make more sense to clock the time it takes to perform an operation rather than going solely by the numbers reported in receipts.

Ink and gas

Because WASM opcodes are orders of magnitude faster than their EVM counterparts, almost every operation that Stylus does costs less than1 gas . “Fractional gas” isn’t an EVM concept, so the Stylus VM introduces a new unit of payment known as ink that’s orders of magnitude smaller.

1 gas =

10 , 000 ink

Intuition

To build intuition for why this is the case, consider theADD instruction.

In the EVM

1. Pay for gas, requiring multiple look-ups of an in-memory table
2. Consider tracing, even if disabled
3. Pop two items of the simulated stack
4. Add them together
5. Push the result

In the Stylus VM

1. Execute a single x86 or ARMADD
2. instruction

Note that unlike the EVM, which charges for gas before running each opcode, the Stylus VM strategically charges for many opcodes all at once. This cuts fees considerably, since the VM only rarely needs to execute gas charging logic. Additionally, gas charging happensinside the program , removing the need for an in-memory table.

The ink price

The ink price, which measures the amount of ink a single EVM gas buys, is configurable by the chain owner. By default, the exchange rate is1:10000 , but this may be adjusted as the EVM and Stylus VM improve over time.

For example, if the Stylus VM becomes 2x faster, instead of cutting the nominal cost of each operation, the ink price may instead be halved, allowing 1 EVM gas to buy twice as much ink. This provides an elegant mechanism for smoothly repricing resources between the two VMs as each makes independent progress.

User experience

It is important to note that users never need to worry about this notion of ink. Receipts will always be measured in gas, with the exchange rate applied automatically under the hood as the VMs pass execution back and forth.

However, developers optimizing contracts may choose to measure performance in ink to pin down the exact cost of executing various routines. Theink_left function exposes this value, and various methods throughout the Rust SDK optionally accept ink amounts too.

See also

- Opcode and host I/O pricing reference
- : Detailed costs per opcode and host I/OEdit this page Last updatedonMar 22, 2024 Previous Stylus testnet information Next Opcode and host I/O pricing