EXPOSING THE METRICS OF ICP BLOCKCHAIN CANISTERS

Troiak K.Y., Boltenkov V.A.
Odesa National Polytechnic University, Odesa

Predicting and evaluating performance of a system is a vital element of software development. When it comes to smart-contracts, such measurements decide if the contract will be sustainable or should be scrapped, as such they are of utmost importance. Smart-contracts on ICP are called canisters and require payment in their currency cycles for executing function calls, using instructions, inputting and storing data. Main language of their development is Motoko and it has libraries: canbench, bench as well as a tool ICP Pricing Calculator for measuring cycles usage for simple scenarios, such as operations that fit inside a single function call or storing an arbitrary amount of data for a certain time period. ExperimentalInternetComputer library provides a function performanceCounter, which allows to measure instruction usage inside a single function without accounting for costs of calling a function and inputting data.

Available tools can provide a quick estimation of instruction usage, however, some things are left insufficiently studied: amount of instructions for receiving function calls, total amount of memory used by the canister, instruction usage of garbage collection. Also, it is not possible to precisely estimate performance of tasks which require multiple function calls under different states of a canister.

The objective of this work is to improve the ability to measure performance of ICP canisters in complex scenarios.

An undocumented Motoko library Prim.mo was uncovered, which exposed metrics of run time system (rts) of canisters [1]. Each metric starts with rts_ and provides low-level stats of a canister by means of a function call which exposes it. They include stats of the canister that are updated after each function call. These metrics were analysed, their principles of work were discovered experimentally and part of them was aggregated into a testing suite that would allow to perform benchmarking in complex scenarios. A way was found to expose aforementioned metrics in canisters and obtain their values by means of calling a query function. The main principle of measurement is capturing states of metrics before and after the target function call and comparing them to detect changes.

A library motoko-benchmarking-ts was developed and published together with detailed instructions for usage [2]. It has made it possible to perform more complex benchmarking, such as profiling multi-call operations, measuring allocation and deallocation of memory or garbage collection efficiency.

References:

1. Dfinity Foundation. Prim.mo [Electronic resource]. – Available at: https://github.com/dfinity/motoko/blob/master/src/prelude/prim.mo (date of access: 19.04.2025).

2. Troiak K. Motoko-benchmarking-ts [Electronic resource]. – Available at: https://www.npmjs.com/package/motoko-benchmarking-ts (date of access: 19.04.2025).