On-chain gated computing by patching the control flow of arbitrary smart contracts

The problem

Imagine the problem space layer-2 solutions have to face.

For example: to make it possible to run arbitrary smart contracts on layer-2 and allowing

the computation and outcome of these contracts to be verifiable on the root-chain;

ideally in a permissionless way and without any sort of ‘special’ authorities or governance protocols.

Solutions we have right now for this are systems like solEVM-enfocer or any truebit like verification game,

but they come with it’s own problems, the biggest one being the time it takes to decide on disputes

-

intending to resolve computation in a on-chain verification game protocol.

This cripples layer-2 solutions because it becomes easy to block/delay finalization of side-chain Blocks on

the root-chain Plasma-style bridge contract. The only reason for this is the big complexity and therefore time

it takes to resolve a dispute.

Theory for a potentially better system

If we think about executing contracts that are easily exceeding the root-chain block gas limit,

one possible solution is to write the contracts in a resumable

way - to gain the ability to execute

a programm in a chunkable way. But this is a complex task and does not scale in practice for every contract.

Let’s assume a general chunkable/gated computing model:

- Stop the execution at any point in (the EVM run-)time,
- Checkpoint the execution environment (stack, memory, etc).
- Save it somewhere and resume the execution at a later time.

Like the suspend-to-ram or deep sleep mode of your favourite computing machine.

Analogy:

The flow of a river that we can suspend or resume depending on a condition.

- Gated

Layer-2 Example

A potential layer-2 solution could use that in the following way:

- Allowing users to submit the solution/outcome of executing a block.

We assume good faith and provide incentives that submitting the correct solution (a solution no-one challenges because it is correct)

provides the solver or potential challengers with positive or negative financial outcomes.

- Having the ability that any honest user can challenge a solution by invoking the execution of that block

on the root-chain in one or more transactions until the execution is resolved/done.

TL;DR

We assume at least one honest user always challenges incorrect solutions buy executing the compution in one or more transactions and

gaining the bond from the solver after the execution was executed on-chain.

How - Patching arbitray smart contracts to own the control flow

With the above in mind, the layer-2 root-chain Bridge

-contract that manages the state of the side-chain

can analize and patch arbitray smart contracts to ‘hijack’ the control flow of the program and therefore

provide a controlled and verifiable way to execute a transaction

in chunks by deploying a patched version

of a smart contract with checkpoints and functionality like intercepting calls to other contracts inside that

arbitrary contract to provide custom functionality or state checks.

That checkpointing function

or let’s name it a retpoline

- a term VM & Kernel developers are propably aware of.

In a nutshell, we patch the bytecode by adding and/or changing control flow instructions like JUMP

and therefore gaining the ability for code execution inside a (potentially untrusted) EVM context.

In this case, the retpoline is Bridge-controlled code that watches for gas usage to know when to stop execution and

overriding instructions like CALL

to provide chain-specific functionality.

Now, gate your feedback