GM everyone, it is @madlabman. Recently, I attempted to adoptEIP-4788 to one of the projects I'm currently working on. I found out that there's a problem with the time during which the CL block roots are available for use on EL. In this post, I would like to propose a solution that will help to overcome this issue and improve EIP-4788 capabilities for DeFi protocols like Lido.

TLDR:

Introduce a contract with persistent storage of block roots pulled from the EIP-4788 buffer. In addition, the storage should provide the ability to back-fill any missing root using a proof. Eventually, the storage will allow querying any required root for a 3rd party contract.

Problem statement

EIP-4788 made consensus layer block roots available on the execution layer. It has a variety of use cases, especially for liquid staking protocols, such as Lido. But the current state of things makes this EIP less applicable than one can imagine. The problem lies in the way EIP is implemented to store the roots mentioned above. It uses a buffer with a limited capacity, meaning only a subset of recent-enough roots will be available. While there is more than a 27-hour window at the moment, and it covers most of the use cases, under certain conditions, it could not be enough. For example, if a staking protocol allows its operators to report withdrawals of validators by themselves, there is no guarantee that the operators will do it in time. Even for the protocol-driven permissioned oracles, there is some place for accidents resulting in missing of an opportunity to deliver a report within the time frame.

That's why it's crucial to have a spare mechanism for using the proofs of CL structures, even for blocks whose roots have already been overwritten in the buffer.

Proposed solution

To extend an availability of block roots on EL the separate long-term storage of roots is proposed. The main difference of this storage is that it accumulates not all the roots created, but only the selected ones, hence called "checkpoints". Implemented as a standalone contract, the storage will accept a call to a permissionless method, which will query the EIP roots buffer and store the latest one as a "checkpoint".

These "checkpoints" play a crucial role in the storage. They are not only a trustless source of a root not presented in the buffer anymore, but a way to add additional "checkpoints". Anyone can bring a proof of a block's root somewhere close to one of the "checkpoints" known to the storage and back-fill the missing root. Pulling roots from the buffer frequently enough makes it cost-efficient to back-fill any historical root at the storage. Ultimately, it leads to full coverage of CL block roots on the execution layer in a dense manner.

While providing long proofs to the end user contracts to prove any old data is still possible, using single storage for this purpose makes UX much smoother. Additionally, wide storage adoption across protocols leads to deeper coverage and reduced costs of using proofs.

I'm seeking comments and suggestions from the community regarding the proposed design. You can delve into the technical details of the proposed solution here.