Has there ever been any thought or consideration given to block persistent storage for a contract?

I’m doing some state channel/sharding research and thinking and I’m running across a few situations where I’m limited in what I can do because multiple transactions operating on my contract in the same block can only be reconciled by writing a storage variable.

Example: Calculating a replacement Merkle root.

Transaction 1: Comes with a value and a proof. The transaction operates on the value and adds a new leaf to the current branch of the merkle tree and swaps out the required values up the branch resulting in a new root.

Transaction 2: Wants to do the same thing on a different branch but can’t because the root has changed.

Current Solution: Keep track of each replacement hash from transaction 1 in a mapping(block -> mapping(source hash -> replacement hash). Unfortunately, this is very expensive as it will be at least 20,000 gas for each hash that has been replaced. If this is a long proof it could get very expensive (not to mention it is eliminating the core reason I was trying to build this contract -> use less gas).

Desired Solution: I only need these values during the current block. It would be nice to have a much cheaper 0x46 BLOAD and 0x47 BSTORE that could store this mapping during the processing of the block so that my contract can swap out hashes for any additional transactions included in the block.

This storage would need to only be accessible from my contract in order to keep other contracts from trying to manipulate it.