Hi everyone, i’ve been meaning to ask you about a small research project i’ve been working on for a few weeks now.

Specifically batch calling of smart contract view and property methods in the scope of fast data retrieval.

Basically i have a contract with a method that accepts an arbitrary length binary input value that then runs each request as a staticcall() and stores the returned value in a buffer ( to be parsed on return by the caller ).

Input example:

first 2 bytes - how many calls are we making

32 bytes - is return value dynamic length ( boolean )

32 bytes - address depends on result of call ID X

32 bytes - address

32 bytes - method signature

- Note nothing is set in stone as i’m still working on standardising the input protocol.

Return example:

An arbitrary length bytes memory buffer.

Reasoning:

Using an implementation of this code proposal, we could in theory just serve a JS payload from swarm ( or centralised server ) that then does 1 RPC call ( per 15k properties ) to a node to retrieve everything they need.

History:

The main project i work on, has to load a collection of entities that can result in a rather large request pool ( ~ 350 per entity ).

1 - We started with the usual web3 contract requests over HTTPS and WS, but even loading 1 ( 350 properties ) will take seconds which is something unacceptable.

2 - When that failed we moved to implementing web3 batch calls up to a limit of 1000, which works fine, if you have like 5 seconds to load the necessary data for 10 entities ( 3500 requests ), yet most nodes will automatically reject your calls if you go over the 5k in 5 minutes limit.

** So we’re wasting a lot of resources, making a ton of unnecessary calls, and the node itself is loading the EVM for each and every CALL, and we have to make our user wait for 60 seconds to process all of these ( even on WS / IPC ). Again unacceptable.

3 - In contract batch calling idea was born

From my tests (https://rinkeby.etherscan.io/address/0x703f4fe20351de6228833e5d2a25323193ab96fe#code ) we should be able to load at least 15000 properties in one call ( more or less depending on the final input protocol ) and stay way below the 50m gas Geth default.

I’m asking here, because if this does get used we are going to see a rather large increase of resource usage on public nodes like Infura ( personal nodes should be just fine handling this, and the app should probably ask about the limit and try to fill it if possible ).

Is someone else working on this ?

Thank you