My goal here is to capture the broader concepts at play surrounding the Oil & Karma proposals, Vitalik's counter proposal and the related meta transaction concept which is a specific use case of the more generic concept of making untrusted calls in the EVM.

At a broad level this entire problem is centered around the two opposing problems of:

1. Needing to reprice opcodes as the EVM evolves
2. The breakage that can occur when opcodes are repriced
3. https://aragon.org/blog/istanbul-hard-fork-impact
4. https://aragon.org/blog/istanbul-hard-fork-impact

We want

to reprice things for changes like accounting for witness sizes, but we cannot

reprice things without potentially breaking existing code that depends on the current prices.

Untrusted EVM calls

We'll define the concept of an "Untrusted Call" as being any of the *CALL*

style opcodes in the EVM for which both caller

and callee

do not trust each other.

Since caller

and callee

only differ by a single letter, I'll use the terminology parent

and child

for easier differentiation

From the parent

perspective, this typically takes the form of accounting after

the child call has occurred. One example of this is measuring how much gas the child consumed. Because of this, the parent's call frame needs to be insulated from a revert in the child propagating to the parent call frame. In the current EVM, this is done by specifying the gas allotment for the child such that a revert/out-of-gas in the child does not cause the parent to revert as well.

From the child

perspective, this typically takes the form of ensuring that the parent cannot cause the child to fail in unexpected ways by either providing insufficient gas, or by making the call when the call stack is already near the maximum depth.

In both perspectives, parent and child accomplish these requirements by observing the remaining gas via the GAS

opcode. In order to do this accounting, one must typically make an estimate or measure how much gas the parent call needs after the child has finished, and these estimates or measurements are anchored to the existing gas prices.

It is worth enumerating the use cases for untrusted EVM calls.

1. Sending transactions for another party
2. Such as maintaining anonymity while pulling funds out of a mixer
3. Such as the Ethereum Alarm Clock
4. Such as maintaining anonymity while pulling funds out of a mixer
5. Such as the Ethereum Alarm Clock
6. Paying for transactions in other currencies

7. Such as paying gas fees in DAI
8. Such as paying gas fees in DAI

The ideal case would be a solution that preserves the existing mechanisms, but it is worth pointing out that it might

be ok to break the existing mechanism if a new

mechanism was added to allow the existing functionality to be implemented using the new mechanism.

Why Oil/Karma probably don't fix this.

One proposed solution is to introduce a new unit of accounting separate from gas that is not observable. This would in theory preserve the existing gas schedule and introduce a new accounting mechanism that could be changed without the burden of backwards incompatibility since it is not observable.

It is my opinion that this thinking is flawed.

If any opcode ever costs more oil

than it does in gas

, then the child will be able to cause a reversion that propagates through the parent call frame. This would break meta transactions.

Why UNGAS probably doesn't fix this

Core Paper

Forward-compatible EVM | Core Paper

Design of a forward-compatible EVM that will not break contracts when applying feature upgrades.

Another proposed solution is to take the more extreme approach and make gas un-observable. The idea is to make gas un-observable and thus, remove the ability to write code that depends on the gas schedule.

It is my opinion that this approach is also flawed.

1. Just because gas is no longer observable, does not mean that people would not write code that depends on the underlying gas schedule.
2. It fundamentally breaks meta-transaction style functionality which would be highly contentious.

Having our

cake

and eating it too

https://en.wikipedia.org/wiki/You_can't_have_your_cake_and_eat_it

We want …

1. Preservation of existing use cases for gas accounting for untrusted EVM calls
2. Ability to make changes to the gas schedule without breaking things.

Make the gas schedule itself observable

One approach that comes to mind is to look at the mechanisms available for smart contracts to do gas accounting. Currently, they can observe the remaining gas using the GAS

opcode, but any code dealing with gas accounting must be written against the current gas schedule. We could potentially address the problem by making the gas schedule observable as well. In theory

this would allow for gas requirements to be computed at runtime which would allow for contract code to observe gas in a manner that was robust against changes to the gas schedule.

It is unlikely that we can make it possible to observe all gas costs since things like expanding the memory are priced dependent on the current execution state.

Ungas & a new EVM mechanism

Ungas could be viable, but it would be a major backwards incompatible change that by itself would remove a popular use case. I believe that we would need to couple Ungas with a new mechanism upon which the existing use cases for gas observation could be built. Account abstraction looks like a promising route.