We recently (March 4) updated the algorithm that sets the L2 gas price for computation. Here's a quick summary of what we changed. We're interested in any suggestions of feedback on this part of the design.

In this post I'll describe how it worked before the upgrade. The next post in the thread will describe what we changed and why.

The L2 compute price adjusts automatically via a mechanism inspired by Ethereum's EIP-1559 basefee setting algorithm. The price depends on gas usage compared to a "speed limit" parameter. When usage is above the speed limit over a period of time, the L2 compute price automatically adjusts upward; when usage is below the speed limit, it adjusts downward. There is a floor below which the L2 compute price can't go.

Before the change, we used a "gas pool" based pricing system. Think of the gas pool as a "gas tank" that can hold up to a certain amount of gas. Compute gas (not including gas charged for storage) used by the system gets subtracted from the gas pool; and every time one second elapses a fixed amount of gas is added to the pool (unless it's full). The rate of adding gas to the pool is the chain's "speed limit" which you can think of as a gas limit per second that is enforced on average. If the gas pool gets completely empty, the chain rejects all transactions until the pool gets more gas.

In that pre-change system, each time one second passed, if the gas pool fullness was a fraction f

(between 0 and 1), the price would be multiplied by the factor (1 + (1-2f)/120)

. This would boost the price by 0.8% if the pool was empty, cut it by 0.8% if the pool was full, and leave it unchanged if the pool was half full.

The logic of this scheme is that we wanted to keep gas usage at or below the speed limit on average. And we wanted to allow bursts of traffic that used more than the speed limit, but only for a limited time. If there was a sustained period of usage above the speed limit, first the price would escalate temporarily until usage dipped back below the speed limit. But if the price escalation wasn't enough to stop the overuse, eventually the gas pool would be completely empty causing a "circuit breaker" to trip and the system would reject all transactions for a period of seconds before re-opening.

Currently the speed limit is 120,000 ArbGas of computation per second, and the gas pool size is 200,000,000 ArbGas. As an example, suppose usage jumps up to 200,000 ArbGas per second (80,000 above the speed limit). Then the gas pool would shrink by 80,000 per second. If this continues for 21 minutes, the gas pool will be 99,200,000 – less than half full – so the L2 compute gas price will start escalating, very slowly at first, and then more rapidly if the pool continues to shrink – but never increasing by more than 0.8% per second. If the usage doesn't decrease after another 20 minutes, despite the price escalation, the circuit breaker will trip and the system will reject perhaps 30 seconds of transactions before re-opening. But this is very unlikely because the price will have had plenty of time to adjust, and users to respond by reducing their traffic.

This system has worked quite well, but we saw ways to improve it. See the next post for details on that.