Nitro uses general-purpose compression to reduce cost. Here's a quick summary of how we do that and some research issues it raises.

As a rollup, Nitro puts transactions' calldata on the L1 chain as Ethereum calldata. This is typically done in large batches, which are put together by the Sequencer. A batch might be 100 kilobytes or more.

It's fine to compress these batches, in order to reduce the cost of posting them on L1, as long as the L2 code can use a deterministic algorithm to decompress and recover the original transaction data. (After decompressing, the L2 code will check users' signatures on the decompressed transactions, to ensure the integrity of the transactions.)

In the original Arbitrum software stack, we used a home-brew compression scheme to compress individual transactions. This helped, but we knew we could do better.

In Nitro, we use brotli, a well-known general purpose compression method, to compress each batch. Because batches are fairly large, this can make batches much smaller. We looked at some other general-purpose compressors, such as zlib, but chose brotli because it looks to get the most effective compression of the algorithms that are practical for a rollup.

One of the interesting research questions that arises is how much to charge each user tx for its calldata. If we compress a batch of many transactions, and achieve a factor of (say) 3 reduction in size, do we charge each transaction for 1/3 of its uncompressed calldata size? Or do we somehow try to figure out which transactions deserve more credit for the compressibility of the batch, and give them a better deal? And what does it even mean for a transaction to "deserve credit" for making other transactions more compressible? (This can be defined theoretically but the obvious formal definitions will be too expensive to evaluate trustlessly in practice.)