Constitutional

Abstract

This AIP proposes a number of improvements to Arbitrum chains, including the capability to leverage EIP 4844 to post batches of L2 transactions as Blobs on L1 Ethereum at a cheaper price. The proposal also includes support for most of the changes included in Ethereum's Dencun upgrade. The proposed ArbOS 20 "Atlas" upgrade will be ready for adoption by any Arbitrum Chain; this proposal concerns the Arbitrum One and Nova chains, as they are governed by the Arbitrum DAO. On a high level, an ArbOS upgrade can be seen as Arbitrum's equivalent of a hard fork - more can be read about the subject over in Arbitrum ArbOs upgrades.

Please note that ArbOS Version 20 "Atlas" is an upgrade that builds upon ArbOS version 11 which has been adopted by the ArbitrumDAO - this proposal increments the version number to 20 instead of 12 due to technical details that allow for better Orbit chain customizability.

Changes Included

1. Posting batches of transactions as blobs to L1 Ethereum (EIP-4844)

With ArbOS 20 "Atlas", any Arbitrum chain that settles on L1 Ethereum will be able to post transactions as either calldata or "blob-data" - the latter of which is to be introduced in EIP-4844 as part of Ethereum's Dencun upgrade.

Some notes on how this is implemented for Arbitrum:

- Applications on Arbitrum will not have to be modified or take any explicit action to get the benefits of using EIP-4844 (i.e. the whole chain opts-in with ArbOS 20 "Atlas").
- ArbOS 20 "Atlas" adds support for Arbitrum chains to send data in a blob storage format to data availability layers, like L1 Ethereum, that support the blob transaction type. This includes Arbitrum One and Nova. ArbOS 20 "Atlas" does not add support for Arbitrum chains to receive

data in a blob storage format. This means that any L3 Orbit chain settling to an L2 Arbitrum chain will post data to the underlying L2 Arbitrum chain as calldata. The L2 Arbitrum chain will then be able to post data to a L1 data availability layer like Ethereum using blobs.

- There currently aren't estimates on what the end-user gas savings of using blob data will be yet. This topic is something being actively worked on and monitored. Without mainnet data, the estimates for blob gas prices will not be accurate enough to reliably predict the cost reductions that users will experience - and even with mainnet data, the savings will vary by use case (i.e. no current way to predict the price impacts from all blob gas market participants yet). In general, however, the use of blobs will generally reduce the cost of using Arbitrum L2s.

This AIP includes a number of upgrades that enable Arbitrum chains to utilize EIP-4844:

- Updated Sequencer Inbox contract with new methods allowing the Sequencer to post Arbitrum transactions in blob data.
- Update to the Nitro fraud prover to support proving additional hashes, i.e., KZG and SHA256 preimages.
- Update to the core Nitro node software to handle parsing chain data from 4844 blobs.

Nitro core changes:

Current full code diff can be viewed via this link:Comparing consensus-v11...finalize-arbos-20 · OffchainLabs/nitro · GitHub
Alternatively, the code diffs are attached below:

nitro.diff

fraud-prover.diff

PRs:

- Support new preimage types in proving by PlasmaPower · Pull Request #1814 · OffchainLabs/nitro · GitHub
- Add WAVM KZG preimage proof by PlasmaPower · Pull Request #2064 · OffchainLabs/nitro · GitHub
- 4844 inbox reader by Tristan-Wilson · Pull Request #2092 · OffchainLabs/nitro · GitHub

- Enable Cancun fork in ArbOS 20 by PlasmaPower · Pull Request #285 · OffchainLabs/go-ethereum · GitHub
- Support and test ArbOS 20 by PlasmaPower · Pull Request #2108 · OffchainLabs/nitro · GitHub
- Cache storage keys by magicxyyz · Pull Request #1767 · OffchainLabs/nitro · GitHub
- add methods to configure brotli compression level in ArbOS 12 by ganeshvanahalli · Pull Request #1865 · OffchainLabs/nitro · GitHub
- Add missing L1 pricing getters to ArbGasInfo in ArbOS 20 by PlasmaPower · Pull Request #2109 · OffchainLabs/nitro · GitHub
- Add ArbOS precompile method to get scheduled upgrade by PlasmaPower · Pull Request #2112 · OffchainLabs/nitro · GitHub
- Prevent a 4844 header from being used inside Anytrust data by PlasmaPower · Pull Request #2115 · OffchainLabs/nitro · GitHub
- Finalize ArbOS 20 by PlasmaPower · Pull Request #2111 · OffchainLabs/nitro · GitHub

Nitro smart contract changes:

Current Full Diff: Comparing main...arbos-20-diff · OffchainLabs/nitro-contracts · GitHub

PRs:

- Add WAVM KZG preimage proof by PlasmaPower · Pull Request #104 · OffchainLabs/nitro-contracts · GitHub
- Include blob cost in extraGas field of batch posting reports by PlasmaPower · Pull Request #114 · OffchainLabs/nitro-contracts · GitHub
- Added postupgradeinit for the challenge manager by yahgwai · Pull Request #116 · OffchainLabs/nitro-contracts · GitHub
- Use 0x50 as the header byte for 4844 batches by PlasmaPower · Pull Request #118 · OffchainLabs/nitro-contracts · GitHub
- Blob data gas refunds by yahgwai · Pull Request #111 · OffchainLabs/nitro-contracts · GitHub
- Support zero basefee for gas estimation by PlasmaPower · Pull Request #119 · OffchainLabs/nitro-contracts · GitHub
- feat: add sha256 preimage support to osp by gzeoneth · Pull Request #90 · OffchainLabs/nitro-contracts · GitHub

2. Enable partial support for Dencun Execution Layer

Ethereum's Dencun upgrade is technically a combination of two upgrades: one to the Execution Layer and another to the Consensus Layer, named Cancun and Deneb, respectively. While the Consensus Layer upgrades are not applicable to Nitro (since Nitro is Arbitrum One and Arbitrum Nova's execution engine), some of the Execution Layer changes are, including EIP-1153 as described above. The full Cancun network upgrade specification for Ethereum can be foundhere.

Nitro's upgrade to ArbOS 20 "Atlas" adds support for three specific EIPs from Cancun, those being:

- Support the new transient storage EVM opcodes introduced inEIP-1153: TSTORE

and TLOAD

, offering a cheaper option than storage for data that's discarded at the end of a transaction.

- Support the new MCOPY

EVM opcode introduced in EIP-5656 for cheaper memory copying.

- Support the changes in functionality of theSELFDESTRUCT

EVM opcode, as outlined in EIP-6780.

Since Nitro uses Geth at the Core, the above 3 EIPs were implemented by merging in a Cancun-supported version of Geth upstream. More specifically, Geth version 1.13.1 was merged into Nitro's go-ethereum fork here:

github.com/OffchainLabs/go-ethereum

Merge geth 1.13.3

OffchainLabs:master

← OffchainLabs:merge-1.13.3

opened 07:48PM - 25 Jan 24 UTC

[

tsahee
](https://github.com/tsahee)

[+4624

-4836

](https://github.com/OffchainLabs/go-ethereum/pull/284/files)

On M2 mac, metrics/influxdb/ tests fail due to different floating point results

With the above merged, the below PR is used to enable the Cancun fork in ArbOS 20 “Atlas”.

github.com/OffchainLabs/go-ethereum

Enable Cancun fork in ArbOS 20

OffchainLabs:master

← OffchainLabs:cancun-arbos-20

opened 06:12PM - 28 Jan 24 UTC

[

PlasmaPower
](https://github.com/PlasmaPower)

[+9

-2

](https://github.com/OffchainLabs/go-ethereum/pull/285/files)

This also disables the BLOBBASEFEE opcode for Arbitrum chains

Dencun changes not

included in ArbOS 20 “Atlas”

Ethereum’s Dencun upgrade includes various proposals that impact both the execution layer and the consensus layer, as mentioned earlier. Among the execution layer changes, there are two EIPs that are explicitly not included in ArbOS 20. This section briefly covers the scope of the two excluded EIPs and a rationale for why they are not included in ArbOS 20.

- EIP-7516: BLOBBASEFEE

opcode

This EIP adds a new opcode to the EVM that returns the base fee of the current blob and is identical to the BASEFEE

opcode introduced in EIP-3198. This is applicable for Ethereum because it enables blob data users, such as rollup contracts or infrastructure, to programmatically account for the blob gas price.

This opcode is excluded from ArbOS 20 “Atlas” because Arbitrum chains are not currently planned to support receiving

blobs from other chains, like an L3 Orbit chain. Calls to the BLOBBASEFEE

opcode on Arbitrum One and Nova after the ArbOS 20 “Atlas” upgrade will revert.

- EIP-4788: Beacon block root in the EVM

This EIP commits a given block’s beacon chain root to the payload header for that particular block as a hash - essentially exposing beacon chain roots to the EVM. This is applicable for Ethereum because it improves the trust assumptions for a variety of applications that rely on data in the consensus layer. Examples of use cases that this would benefit include: staking pool, restaking constructions, smart contract bridges, and more.

This EIP is excluded from ArbOS 20 “Atlas” because Arbitrum’s execution engine (i.e. Nitro) does not have a consensus layer and relies on Ethereum consensus for data security and data finality. Calls to the beacon root opcode on Arbitrum One and Nova after the ArbOS 20 “Atlas” upgrade will revert.

Implementation

The canonical version of ArbOS 20 this proposal aims to adopt is implemented in the Arbitrum Nitro git commit hash cf2eadfcc1039eca9594c4f71477a50f550d7749

and can be viewed in Merge branch 'gate-get-scheduled-upgrade' into finalize-arbos-20 · OffchainLabs/nitro@cf2eadf · GitHub.

Upgrade Action smart contracts

The Action smart contracts used to execute the on-chain upgrade can be viewed inAIP 4844 + ArbOS 20 by yahgwai · Pull Request #244 · ArbitrumFoundation/governance · GitHub.

Verifying the ArbOS Code Difference

The current ArbOS version used on Arbitrum One and Arbitrum Nova is ArbOS 11, corresponding to the Arbitrum Nitro consensus-v11

git tag. You can verify this by running the previously mentioned steps to build the WASM module root on that git tag, which produces the WASM module root 0x8b104a2e80ac6165dc58b9048de12f301d70b02a0ab51396c22b4b4b802a16a4

, which is what the rollup contract’s wasmModuleRoot()

method returns for both Arbitrum One and Arbitrum Nova.

To audit the code difference from ArbOS 11 to ArbOS 20 “Atlas”, you could simply generate a full nitro diff with git diff consensus-v11 cf2eadfcc1039eca9594c4f71477a50f550d7749

(and also generate a diff of the go-ethereum submodule mentioned in that nitro diff). However, that includes a lot of code that isn’t part of the WASM module root. To filter down to just the replay binary which defines the state transition function, you can start by generating a list of files in the nitro and go-ethereum repositories included by the replay binary in either ArbOS 11 or ArbOS 20 “Atlas” with bash:

!/usr/bin/env bash

set -e mkdir -p ~/tmp # this script uses ~/tmp as scratch space and output

this script should be run in the nitro repository

git checkout cf2eadfcc1039eca9594c4f71477a50f550d7749 git submodule update --init --recursive make .make/solgen go list -f "{{.Deps}}" ./cmd/replay | tr -d '[]' | sed 's/ /\n/g' | grep 'github.com/offchainlabs/nitro/' | sed 's@github.com/offchainlabs/nitro/@@' | while read dir; do find "$dir" -type f -name '*.go' -maxdepth 1; done | grep -v '_test.go$' > ~/tmp/consensus-v20-nitro-files.txt go list -f "{{.Deps}}" ./cmd/replay | tr -d '[]' | sed 's/ /\n/g' | grep 'github.com/ethereum/go-ethereum/' | sed 's@github.com/ethereum/go-ethereum/@go-ethereum/@' | while read dir; do find "$dir" -type f -name '*.go' -maxdepth 1; done | grep -v '_test.go$' > ~/tmp/consensus-v20-geth-files.txt git checkout consensus-v11 git submodule update --init --recursive make .make/solgen go list -f "{{.Deps}}" ./cmd/replay | tr -d '[]' | sed 's/ /\n/g' | grep 'github.com/offchainlabs/nitro/' | sed 's@github.com/offchainlabs/nitro/@@' | while read dir; do find "$dir" -type f -name '*.go' -maxdepth 1; done | grep -v '_test.go$' > ~/tmp/consensus-v11-nitro-files.txt go list -f "{{.Deps}}" ./cmd/replay | tr -d '[]' | sed 's/ /\n/g' | grep 'github.com/ethereum/go-ethereum/' | sed 's@github.com/ethereum/go-ethereum/@go-ethereum/@' | while read dir; do find "$dir" -type f -name '*.go' -maxdepth 1; done | grep -v '_test.go$' > ~/tmp/consensus-v11-geth-files.txt sort -u ~/tmp/consensus-v11-nitro-files.txt ~/tmp/consensus-v20-nitro-files.txt > ~/tmp/replay-binary-nitro-dependencies.txt sort -u ~/tmp/consensus-v11-geth-files.txt ~/tmp/consensus-v20-geth-files.txt | sed 's@^[./]*go-ethereum/@@' > ~/tmp/replay-binary-geth-dependencies.txt

Now, ~/tmp/replay-binary-dependencies.txt

contains a list of dependencies of the replay binary that were present in ArbOS 11 or ArbOS 20 “Atlas”. To use that to generate a smaller diff of the nitro repository, you can run:

git diff consensus-v11 cf2eadfcc1039eca9594c4f71477a50f550d7749 -- cmd/replay $(cat ~/tmp/replay-binary-nitro-dependencies.txt)

The fraud prover is not part of the on-chain upgrade, but may be helpful in understanding the fraud proving smart contract changes and other components:

git diff consensus-v11 cf2eadfcc1039eca9594c4f71477a50f550d7749 -- arbitrator/prover arbitrator/wasm-libraries/ arbitrator/arbutil ':!/**Cargo.lock' ':!**/kzg-trusted-setup.json'

For the go-ethereum submodule, you can first find out what go-ethereum commit ArbOS 11 and 20 “Atlas” used:

$ git ls-tree consensus-v11 go-ethereum 160000 commit abe584818e104dd5b4fdb8f60381a14eede896de go-ethereum $ git ls-tree cf2eadfcc1039eca9594c4f71477a50f550d7749 go-ethereum 160000 commit 1acd9c64ac5804729475ef60aa578b4ec52fa0e6 go-ethereum

Those commit hashes are the go-ethereum commit hashes pinned by each nitro commit. Then, you can again use git diff

and the files generated by the earlier script to generate a diff limited to code used by the replay binary:

this should be run inside the go-ethereum submodule folder

git diff abe584818e104dd5b4fdb8f60381a14eede896de 1acd9c64ac5804729475ef60aa578b4ec52fa0e6 -- $(cat ~/tmp/replay-binary-geth-dependencies.txt)

This diff also includes the diff between upstream go-ethereum versions v1.11.6 and v1.13.3, as ArbOS 11 used the former and ArbOS 20 “Atlas” uses the latter. To filter out that difference, you can use this tool to find the intersection of two git diffs: Git diff intersection finder

We can use that to find the intersection of the diff of ArbOS 20 “Atlas”’s go-ethereum against ArbOS 11’s go-ethereum and the diff of ArbOS 20’s go-etheruem against upstream go-ethereum v1.13.3:

this should be run inside the go-ethereum submodule folder

git diff abe584818e104dd5b4fdb8f60381a14eede896de 1acd9c64ac5804729475ef60aa578b4ec52fa0e6 -- $(cat ~/tmp/replay-binary-geth-dependencies.txt) > ~/tmp/arbos-11-vs-20-geth.diff git diff v1.13.3 1acd9c64ac5804729475ef60aa578b4ec52fa0e6 -- $(cat ~/tmp/replay-binary-geth-dependencies.txt) > ~/tmp/arbos-20-vs-upstream-geth.diff diff-intersection.py ~/tmp/arbos-11-vs-20-geth.diff ~/tmp/arbos-20-vs-upstream-geth.diff --ignore-files 'core/blockchain.*go' arbitrum_types/txoptions.go 'rawdb/' 'rpc/**'*

The above command ignores files that are included by the replay binary but whose components are not used with these arguments: --ignore-files 'core/blockchain.*go' arbitrum_types/txoptions.go 'rawdb/' 'rpc/**'*

. To also review those diffs, you can remove those arguments.

Note that by default, diff-intersection.py

does a line based intersection. To instead do an intersection based on chunks in the diff, known as hunks in git terminology, you can add the --only-hunks

flag.

Next steps

The current proposal is currently undergoing security audits. As the audit is finalized, the proposal will be updated accordingly and the audit report will be made available to the public.

Even though the audit isn’t complete, the proposal is mature and ready to be discussed and considered for adoption by the DAO.