Aave V3.3 Origin Gas Audit Report

Prepared by Cyfrin

Version 1.0

"Public Goods" audit initiated by Cyfrin not by request of Aave

Lead Auditor

Dacian

May 9, 2025

Contents

1 About Cyfrin

Cyfrin is a Web3 security company dedicated to bringing industry-leading protection and education to our partners and their projects. Our goal is to create a safe, reliable, and transparent environment for everyone in Web3 and DeFi. Learn more about us at cyfrin.io.

2 Disclaimer

The Cyfrin team makes every effort to find as many vulnerabilities in the code as possible in the given time but holds no responsibility for the findings in this document. A security audit by the team does not endorse the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the solidity implementation of the contracts.

3 Risk Classification

	Impact: High	**Impact: Medium**	**Impact: Low**
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

4 Protocol Summary

Aave is a decentralised non-custodial liquidity protocol where users can participate as suppliers or borrowers. Suppliers provide liquidity to the market while earning interest, and borrowers can access liquidity by providing collateral that exceeds the borrowed amount.

5 Audit Scope

The scope focused on finding gas optimizations in Aave's V3.3 liquidation code, but also extended to other parts of the codebase as well.

6 Executive Summary

Over the course of 4 days, the Cyfrin team conducted a gas audit on the Aave V3.3 Origin smart contracts provided by Aave. In this period, a total of 26 gas optimizations were found.

Aave's existing code already uses a number of gas optimization techniques so attempting to further optimize was an interesting challenge. We were able to achieve further optimizations by carefully using a combination of techniques such as:

- caching storage to remove multiple identical storage reads (G-1, G-2, G-3, G-4, G-5, G-17, G-18, G-26)
- using named return variables to remove local variables and for memory returns (G-7, G-10, G-12, G-13)
- for cached in-memory `userConfig`, pass it by memory reference to child functions which may modify it, then write it to storage once at the end of the parent function (G-6, G-14, G-15, G-16)
- optimize away "context" structs used to avoid "stack too deep" errors but which were not actually needed (G-8, G-23, G-24)

- where "context" structs were needed, move as many variables from them as possible into the functions themselves while still avoiding "stack too deep" errors (G-9, G-11)
- read then increment counters in the same statement (G-19, G-21)
- return quickly without performing unnecessary storage reads (G-20)
- don't copy entire structs from storage to memory when only a small number of struct fields are actually required (G-22, G-25)
- refactor modifiers into internal functions where this allows for more optimized parent functions (G-25)

Each modification was made as a separate commit to our working repository including both code and gas snapshot changes; every finding has the linked commit(s) in its recommendations. Additionally all unit tests continued to pass and we also ran Aave's existing Echidna-based invariant fuzzing suite which failed to break any invariants after all our changes had been made.

Measuring against Aave's existing gas snapshots we achieved a total gas reduction of -59,732 with the highest class of total gas reductions being for Pool Operations at -34,294 and the second-highest of -10,172 in the Treasury Collector. Since each function can have multiple snapshot gas measurements, it is more useful to look at the average gas reduction per function:

- borrow -1412
- flashLoan/flashLoanSimple -1123
- liquidationCall -3337
- repay -460
- supply -455
- withdraw -873
- cancelStream -2918
- withdrawFromStream -2086

The full gas snapshot changes follow:

`AToken.transfer.json` total gas reduction -1779:

```
full amount; receiver: ->enableCollateral: 144881 -> 144747 (-134)
full amount; sender: ->disableCollateral;: 103318 -> 103144 (-174)
full amount; sender: ->disableCollateral; receiver: ->enableCollateral: 145060 -> 144890 (-170)
full amount; sender: ->disableCollateral; receiver: dirty, ->enableCollateral: 133158 -> 132817 (-341)
full amount; sender: collateralDisabled: 103139 -> 103001 (-138)
partial amount; sender: collateralDisabled;: 103139 -> 103001 (-138)
partial amount; sender: collateralDisabled; receiver: ->enableCollateral: 144881 -> 144747 (-134)
partial amount; sender: collateralEnabled;: 103347 -> 103070 (-277)
partial amount; sender: collateralEnabled; receiver: ->enableCollateral: 145089 -> 144816 (-273)
```

`Pool.Getters.json` total gas reduction -5113:

```
getReserveData: 32157 -> 31733 (-424)
getUserAccountData: supplies: 0, borrows: 0: 22641 -> 22633 (-8)
getUserAccountData: supplies: 0, borrows: 0 with eMode enabled: 22641 -> 22633 (-8)
getUserAccountData: supplies: 1, borrows: 0: 58553 -> 57843 (-710)
getUserAccountData: supplies: 1, borrows: 0 with eMode enabled: 61360 -> 60592 (-768)
getUserAccountData: supplies: 2, borrows: 0: 87575 -> 86730 (-845)
getUserAccountData: supplies: 2, borrows: 0 with eMode enabled: 90767 -> 90229 (-538)
getUserAccountData: supplies: 2, borrows: 1: 118931 -> 117872 (-1059)
getUserAccountData: supplies: 2, borrows: 1 with eMode enabled: 122123 -> 121370 (-753)
```

`Pool.Operations.json` total gas reduction -34,294:

```
borrow: first borrow->borrowingEnabled: 256480 -> 255188 (-1292)
borrow: recurrent borrow: 249018 -> 247485 (-1533)
flashLoan: flash loan for one asset: 197361 -> 197030 (-331)
flashLoan: flash loan for one asset and borrow: 279057 -> 277770 (-1287)
flashLoan: flash loan for two assets: 325455 -> 324788 (-667)
flashLoan: flash loan for two assets and borrow: 484439 -> 481435 (-3004)
flashLoanSimple: simple flash loan: 170603 -> 170274 (-329)
liquidationCall: deficit on liquidated asset: 392365 -> 389059 (-3306)
liquidationCall: deficit on liquidated asset + other asset: 491921 -> 487272 (-4649)
liquidationCall: full liquidation: 392365 -> 389059 (-3306)
liquidationCall: full liquidation and receive ATokens: 368722 -> 365368 (-3354)
liquidationCall: partial liquidation: 383166 -> 380485 (-2681)
liquidationCall: partial liquidation and receive ATokens: 359520 -> 356793 (-2727)
repay: full repay: 176521 -> 176087 (-434)
repay: full repay with ATokens: 173922 -> 173432 (-490)
repay: partial repay: 189949 -> 189518 (-431)
repay: partial repay with ATokens: 185129 -> 184642 (-487)
supply: collateralDisabled: 146755 -> 146354 (-401)
supply: collateralEnabled: 146755 -> 146354 (-401)
supply: first supply->collateralEnabled: 176366 -> 175803 (-563)
withdraw: full withdraw: 165226 -> 164659 (-567)
withdraw: partial withdraw: 181916 -> 181430 (-486)
withdraw: partial withdraw with active borrows: 239471 -> 237903 (-1568)
```

`Pool.Setters.json` total gas reduction -2579:

```
setUserEMode: enter eMode, 1 borrow, 1 supply: 140836 -> 139849 (-987)
setUserEMode: leave eMode, 1 borrow, 1 supply: 112635 -> 111707 (-928)
setUserUseReserveAsCollateral: disableCollateral, 1 supply: 93456 -> 93114 (-342)
setUserUseReserveAsCollateral: enableCollateral, 1 supply: 105167 -> 104845 (-322)
```

`ProtocolDataProvider.json` total gas reduction -858:

```
getATokenTotalSupply: 35491 -> 35486 (-5)
getInterestRateStrategyAddress: 40256 -> 39832 (-424)
getTotalDebt: 35491 -> 35486 (-5)
getUserReserveData: 72607 -> 72183 (-424)
```

`StataTokenV2.json` total gas reduction -1724:

```
claimRewards: 359669 -> 359882 (+213)
deposit: 280955 -> 280135 (-820)
depositATokens: 219311 -> 219151 (-160)
redeem: 205837 -> 205356 (-481)
redeemAToken: 152633 -> 152370 (-263)
```

`WrappedTokenGatewayV3` total gas reduction -3212:

```
borrowETH: 250413 -> 249121 (-1292)
depositETH: 222614 -> 222223 (-609)
repayETH: 192937 -> 192503 (-566)
withdrawETH: 259299 -> 258554 (-745)
```

The final two did not have existing gas snapshots so we added them to the existing tests and comparing after our optimizations:

`RewardsController.json` total gas reduction -1 (net almost nothing):

```
claimAllRewards: one reward type: 50167 -> 50311 (144)
claimAllRewardsToSelf: one reward type: 49963 -> 50107 (144)
claimRewards partial: one reward type: 48299 -> 48430 (131)
```

```
claimRewards: one reward type: 48037 -> 48168 (131)
configureAssets: one reward type: 264184 -> 263847 (-337)
getUserAccruedRewards: one reward type: 2186 -> 2090 (-96)
setDistributionEnd: 5972 -> 5940 (-32)
setEmissionPerSecond: one reward one emission: 11541 -> 11455 (-86)
```

`Collector.json` total gas reduction -10172:

```
cancelStream: by funds admin: 18522 -> 15718 (-2804)
cancelStream: by recipient: 49489 -> 46456 (-3033)
createStream: 211680 -> 211518 (-162)
withdrawFromStream: final withdraw: 43594 -> 41449 (-2145)
withdrawFromStream: intermediate withdraw: 42252 -> 40224 (-2028)
```

Summary

Project Name	Aave V3.3 Origin
Repository	aave-v3-origin
Commit	464a0ea5147d...
Audit Timeline	May 5th - May 8th
Methods	Manual Review

Summary of Findings

[G-01] Cache `currentReserve.configuration` in `GenericLogic::calculateUserAccountData`	Open
[G-02] Cache `usersConfig[params.user]` in `LiquidationLogic::executeLiquidationCall`	Open
[G-03] Cache `collateralReserve.configuration` in `LiquidationLogic::executeLiquidationCall`	Open
[G-04] Cache `collateralReserve.id` in `LiquidationLogic::executeLiquidationCall`	Open
[G-05] Use cached `vars.collateralAToken` in `LiquidationLogic::_liquidateATokens`	Open
[G-06] Only read from and write to storage once for `usersConfig[params.user]` in `LiquidationLogic::executeLiquidationCall`	Open
[G-07] Reduce memory usage and gas by using named return variables in `LiquidationLogic::_calculateAvailableCollateralToLiquidate`	Open
[G-08] Remove memory struct `AvailableCollateralToLiquidateLocalVars` and use only local variables in `LiquidationLogic::_calculateAvailableCollateralToLiquidate`	Open
[G-09] Move 3 variables from `LiquidationCallLocalVars` struct into body of `LiquidationLogic::executeLiquidationCall`	Open
[G-10] Used named return variables in `GenericLogic::calculateUserAccountData`	Open

[G-11] Remove 5 variables from context struct `CalculateUserAccountDataVars` used in `GenericLogic::calculateUserAccountData`	Open
[G-12] Use named returns in `ReserveLogic::cache` and `cumulateToLiquidityIndex`	Open
[G-13] Misc used named returns to eliminate local variables or for `memory` returns	Open
[G-14] Only read from and write to `userConfig` storage once in `SupplyLogic::executeWithdraw`	Open
[G-15] Cache `usersConfig[params.from]` in `SupplyLogic::executeFinalizeTransfer`	Open
[G-16] Cache `userConfig` in `BorrowLogic::executeBorrow`	Open
[G-17] In `RewardsDistributor`, cache `storage` array lengths and when expected to read it `>= 3` times also for `memory`	Open
[G-18] Cache event emission parameters in `RewardsDistributor:::setDistributionEnd`, `setEmissionPerSecond`	Open
[G-19] Read and increment `availableRewardsCount` in same statement in `RewardsDistributor::_configureAssets`	Open
[G-20] Exit quickly in `RewardsDistributor::_updateData` when `numAvailableRewards == 0`	Open
[G-21] Read and increment `streamId` in same statement, also use named return in `Collector::createStream`	Open
[G-22] Don't copy entire `Stream` struct from `storage` to `memory` in `Collector::deltaOf`	Open
[G-23] Remove struct `BalanceOfLocalVars` and use local variables in `Collector::balanceOf`	Open
[G-24] Remove struct `CreateStreamLocalVars` and use local variables in `Collector::createStream`	Open
[G-25] Don't copy entire `Stream` struct from `storage` to `memory` and refactor `onlyAdminOrRecipient` modifier into internal function in `Collector::withdrawFromStream` and `cancelStream`	Open
[G-26] Cache `oldId` prior to `require` check to save 1 storage read in `PoolAddressesProviderRegistry::unregisterAddressesProvider`	Open

7 Findings

7.1 Gas Optimization

7.1.1 Cache `currentReserve.configuration` in `GenericLogic::calculateUserAccountData`

Description: Cache `currentReserve.configuration` in `GenericLogic::calculateUserAccountData` as this is a `view` function which doesn't change state.

Impact: `snapshots/Pool.Setters.json`:

```
-  "setUserEMode: enter eMode, 1 borrow, 1 supply": "140836",
-  "setUserEMode: leave eMode, 1 borrow, 1 supply": "112635",
+  "setUserEMode: enter eMode, 1 borrow, 1 supply": "140695",
+  "setUserEMode: leave eMode, 1 borrow, 1 supply": "112494",
```

`snapshots/Pool.Operations.json`:

```
-  "borrow: first borrow->borrowingEnabled": "256480",
-  "borrow: recurrent borrow": "249018",
+  "borrow: first borrow->borrowingEnabled": "256479",
+  "borrow: recurrent borrow": "248877",
   "flashLoan: flash loan for one asset": "197361",
-  "flashLoan: flash loan for one asset and borrow": "279057",
+  "flashLoan: flash loan for one asset and borrow": "279056",
   "flashLoan: flash loan for two assets": "325455",
-  "flashLoan: flash loan for two assets and borrow": "484439",
+  "flashLoan: flash loan for two assets and borrow": "484295",
   "flashLoanSimple: simple flash loan": "170603",
-  "liquidationCall: deficit on liquidated asset": "392365",
-  "liquidationCall: deficit on liquidated asset + other asset": "491921",
-  "liquidationCall: full liquidation": "392365",
-  "liquidationCall: full liquidation and receive ATokens": "368722",
-  "liquidationCall: partial liquidation": "383166",
-  "liquidationCall: partial liquidation and receive ATokens": "359520",
+  "liquidationCall: deficit on liquidated asset": "392223",
+  "liquidationCall: deficit on liquidated asset + other asset": "491638",
+  "liquidationCall: full liquidation": "392223",
+  "liquidationCall: full liquidation and receive ATokens": "368581",
+  "liquidationCall: partial liquidation": "383024",
+  "liquidationCall: partial liquidation and receive ATokens": "359378",
   "repay: full repay": "176521",
   "repay: full repay with ATokens": "173922",
   "repay: partial repay": "189949",
   "supply: first supply->collateralEnabled": "176366",
   "withdraw: full withdraw": "165226",
   "withdraw: partial withdraw": "181916",
-  "withdraw: partial withdraw with active borrows": "239471"
+  "withdraw: partial withdraw with active borrows": "239329"
```

Recommended Mitigation: See commit 3cd6639.

7.1.2 Cache `usersConfig[params.user]` in `LiquidationLogic::executeLiquidationCall`

Description: In `LiquidationLogic::executeLiquidationCall`, `usersConfig[params.user]` can be cached and the copy can be safely passed to view functions `GenericLogic::calculateUserAccountData` and `ValidationLogic::validateLiquidationCall`.

A more intrusive optimization is to use and update the cached copy throughout the entire liquidation process, then write the copy to storage at the end. This has been implemented separately in G-06 as it is more intrusive.

Impact: `snapshots/Pool.Operations.json`:

```
-  "liquidationCall: deficit on liquidated asset": "392223",
-  "liquidationCall: deficit on liquidated asset + other asset": "491638",
-  "liquidationCall: full liquidation": "392223",
-  "liquidationCall: full liquidation and receive ATokens": "368581",
-  "liquidationCall: partial liquidation": "383024",
-  "liquidationCall: partial liquidation and receive ATokens": "359378",
+  "liquidationCall: deficit on liquidated asset": "392182",
+  "liquidationCall: deficit on liquidated asset + other asset": "491597",
+  "liquidationCall: full liquidation": "392182",
+  "liquidationCall: full liquidation and receive ATokens": "368539",
+  "liquidationCall: partial liquidation": "382983",
+  "liquidationCall: partial liquidation and receive ATokens": "359337",
```

Recommended Mitigation: See commit 4ce346c.

7.1.3 Cache `collateralReserve.configuration` in `LiquidationLogic::executeLiquidationCall`

Description: In `LiquidationLogic::executeLiquidationCall`, `collateralReserve.configuration` can be safely cached and passed to child functions saving many identical storage reads.

Impact: `snapshots/Pool.Operations.json`:

```
-  "liquidationCall: deficit on liquidated asset": "392182",
-  "liquidationCall: deficit on liquidated asset + other asset": "491597",
-  "liquidationCall: full liquidation": "392182",
-  "liquidationCall: full liquidation and receive ATokens": "368539",
-  "liquidationCall: partial liquidation": "382983",
-  "liquidationCall: partial liquidation and receive ATokens": "359337",
+  "liquidationCall: deficit on liquidated asset": "391606",
+  "liquidationCall: deficit on liquidated asset + other asset": "491021",
+  "liquidationCall: full liquidation": "391606",
+  "liquidationCall: full liquidation and receive ATokens": "367841",
+  "liquidationCall: partial liquidation": "382408",
+  "liquidationCall: partial liquidation and receive ATokens": "358639",
```

Recommended Mitigation: See commit 414dc2d.

7.1.4 Cache `collateralReserve.id` in `LiquidationLogic::executeLiquidationCall`

Description: In `LiquidationLogic::executeLiquidationCall`, `collateralReserve.id` can be safely cached and the copy passed to child functions.

Impact: `snapshots/Pool.Operations.json`:

```
-  "liquidationCall: deficit on liquidated asset": "391606",
-  "liquidationCall: deficit on liquidated asset + other asset": "491021",
-  "liquidationCall: full liquidation": "391606",
-  "liquidationCall: full liquidation and receive ATokens": "367841",
-  "liquidationCall: partial liquidation": "382408",
-  "liquidationCall: partial liquidation and receive ATokens": "358639",
+  "liquidationCall: deficit on liquidated asset": "391560",
+  "liquidationCall: deficit on liquidated asset + other asset": "490975",
+  "liquidationCall: full liquidation": "391560",
+  "liquidationCall: full liquidation and receive ATokens": "367673",
+  "liquidationCall: partial liquidation": "382476",
+  "liquidationCall: partial liquidation and receive ATokens": "358585",
```

Recommended Mitigation: See commit a82d552.

7.1.5 Use cached `vars.collateralAToken` in `LiquidationLogic::_liquidateATokens`

Description: Use cached `vars.collateralAToken` in `LiquidationLogic::_liquidateATokens`; there's no need to read it from storage again.

Impact: `snapshots/Pool.Operations.json`:

```
-  "liquidationCall: full liquidation and receive ATokens": "367673",
+  "liquidationCall: full liquidation and receive ATokens": "367553",
   "liquidationCall: partial liquidation": "382476",
-  "liquidationCall: partial liquidation and receive ATokens": "358585",
+  "liquidationCall: partial liquidation and receive ATokens": "358465",
```

Recommended Mitigation: See commit f6af2e1.

7.1.6 Only read from and write to storage once for `usersConfig[params.user]` in `LiquidationLogic::executeLiquidationCall`

Description: `usersConfig[params.user]` can be read once at the start of `LiquidationLogic::executeLiquidationCall`, then the cached copy can be passed around using `memory` and modified as needed, and finally storage can be written once at the end of the function.

Impact: `snapshots/Pool.Operations.json`:

```
-  "liquidationCall: deficit on liquidated asset": "391560",
-  "liquidationCall: deficit on liquidated asset + other asset": "490975",
-  "liquidationCall: full liquidation": "391560",
-  "liquidationCall: full liquidation and receive ATokens": "367553",
-  "liquidationCall: partial liquidation": "382476",
-  "liquidationCall: partial liquidation and receive ATokens": "358465",
+  "liquidationCall: deficit on liquidated asset": "391305",
+  "liquidationCall: deficit on liquidated asset + other asset": "489972",
+  "liquidationCall: full liquidation": "391305",
+  "liquidationCall: full liquidation and receive ATokens": "367366",
+  "liquidationCall: partial liquidation": "382734",
+  "liquidationCall: partial liquidation and receive ATokens": "358795",
```

This change seems to benefit everything apart from partial liquidations where it results in slightly worse performance.

Recommended Mitigation: See commit f419f3c.

7.1.7 Reduce memory usage and gas by using named return variables in `LiquidationLogic::_calculateAvailableCollateralToLiquidate`

Description: Reduce memory usage and gas by using named return variables in `LiquidationLogic::_calculateAvailableCollateralToLiquidate`.

Impact: `snapshots/Pool.Operations.json`:

```
-  "liquidationCall: deficit on liquidated asset": "391305",
-  "liquidationCall: deficit on liquidated asset + other asset": "489972",
-  "liquidationCall: full liquidation": "391305",
-  "liquidationCall: full liquidation and receive ATokens": "367366",
-  "liquidationCall: partial liquidation": "382734",
-  "liquidationCall: partial liquidation and receive ATokens": "358795",
+  "liquidationCall: deficit on liquidated asset": "391070",
+  "liquidationCall: deficit on liquidated asset + other asset": "489736",
+  "liquidationCall: full liquidation": "391070",
+  "liquidationCall: full liquidation and receive ATokens": "367131",
```

```
+  "liquidationCall: partial liquidation": "382507",
+  "liquidationCall: partial liquidation and receive ATokens": "358569",
```

Recommended Mitigation: See commit af61e44.

7.1.8 Remove memory struct `AvailableCollateralToLiquidateLocalVars` and use only local variables in `LiquidationLogic::_calculateAvailableCollateralToLiquidate`

Description: Using in-memory "context" structs to store variables is a nice trick to get around "stack too deep errors", but also uses significantly more gas than using local in-function variables.

When in-memory "context" structs are not required, it is cheaper to not use them. Hence remove memory struct `AvailableCollateralToLiquidateLocalVars` and use only local variables in `LiquidationLogic::_calculateAvailableCollateralToLiquidate`.

Impact: `snapshots/Pool.Operations.json`:

```
-  "liquidationCall: deficit on liquidated asset": "391070",
-  "liquidationCall: deficit on liquidated asset + other asset": "489736",
-  "liquidationCall: full liquidation": "391070",
-  "liquidationCall: full liquidation and receive ATokens": "367131",
-  "liquidationCall: partial liquidation": "382507",
-  "liquidationCall: partial liquidation and receive ATokens": "358569",
+  "liquidationCall: deficit on liquidated asset": "390891",
+  "liquidationCall: deficit on liquidated asset + other asset": "489556",
+  "liquidationCall: full liquidation": "390891",
+  "liquidationCall: full liquidation and receive ATokens": "366954",
+  "liquidationCall: partial liquidation": "382335",
+  "liquidationCall: partial liquidation and receive ATokens": "358397",
```

Recommended Mitigation: See commit 84d3925.

7.1.9 Move 3 variables from `LiquidationCallLocalVars` struct into body of `LiquidationLogic::executeLiquidationCall`

Description: Similar to G-07, at least 3 variables can be moved from the in-memory "context" struct `LiquidationCallLocalVars` into the function body of `LiquidationLogic::executeLiquidationCall` without triggering "stack too deep" errors.

Impact: `snapshots/Pool.Operations.json`:

```
-  "liquidationCall: deficit on liquidated asset": "390891",
-  "liquidationCall: deficit on liquidated asset + other asset": "489556",
-  "liquidationCall: full liquidation": "390891",
-  "liquidationCall: full liquidation and receive ATokens": "366954",
-  "liquidationCall: partial liquidation": "382335",
-  "liquidationCall: partial liquidation and receive ATokens": "358397",
+  "liquidationCall: deficit on liquidated asset": "390795",
+  "liquidationCall: deficit on liquidated asset + other asset": "489459",
+  "liquidationCall: full liquidation": "390795",
+  "liquidationCall: full liquidation and receive ATokens": "366840",
+  "liquidationCall: partial liquidation": "382220",
+  "liquidationCall: partial liquidation and receive ATokens": "358265",
```

Recommended Mitigation: See commit 8bf12e2.

7.1.10 Used named return variables in `GenericLogic::calculateUserAccountData`

Description: Similar to G-7, cheaper to use named return variables in `GenericLogic::calculateUserAccountData`.

Impact: `snapshots/Pool.Operations.json`:

```
-  "borrow: first borrow->borrowingEnabled": "256479",
-  "borrow: recurrent borrow": "248877",
+  "borrow: first borrow->borrowingEnabled": "256117",
+  "borrow: recurrent borrow": "248451",
   "flashLoan: flash loan for one asset": "197361",
-  "flashLoan: flash loan for one asset and borrow": "279056",
+  "flashLoan: flash loan for one asset and borrow": "278694",
   "flashLoan: flash loan for two assets": "325455",
-  "flashLoan: flash loan for two assets and borrow": "484295",
+  "flashLoan: flash loan for two assets and borrow": "483384",
   "flashLoanSimple: simple flash loan": "170603",
-  "liquidationCall: deficit on liquidated asset": "390795",
-  "liquidationCall: deficit on liquidated asset + other asset": "489459",
-  "liquidationCall: full liquidation": "390795",
-  "liquidationCall: full liquidation and receive ATokens": "366840",
-  "liquidationCall: partial liquidation": "382220",
-  "liquidationCall: partial liquidation and receive ATokens": "358265",
+  "liquidationCall: deficit on liquidated asset": "390368",
+  "liquidationCall: deficit on liquidated asset + other asset": "489010",
+  "liquidationCall: full liquidation": "390368",
+  "liquidationCall: full liquidation and receive ATokens": "366414",
+  "liquidationCall: partial liquidation": "381793",
+  "liquidationCall: partial liquidation and receive ATokens": "357839",
-  "withdraw: partial withdraw with active borrows": "239329"
+  "withdraw: partial withdraw with active borrows": "238904"
```

Recommended Mitigation: See commit f6f7cb6.

7.1.11 Remove 5 variables from context struct `CalculateUserAccountDataVars` used in `GenericLogic::calculateUserAccountData`

Description: Similar to G-7 and G-8, it is cheaper to remove these variables from the in-memory struct `CalculateUserAccountDataVars` without triggering a "stack too deep" error.

Impact: `snapshots/Pool.Operations`:

```
-  "borrow: first borrow->borrowingEnabled": "256117",
-  "borrow: recurrent borrow": "248451",
+  "borrow: first borrow->borrowingEnabled": "255807",
+  "borrow: recurrent borrow": "248112",
   "flashLoan: flash loan for one asset": "197361",
-  "flashLoan: flash loan for one asset and borrow": "278694",
+  "flashLoan: flash loan for one asset and borrow": "278384",
   "flashLoan: flash loan for two assets": "325455",
-  "flashLoan: flash loan for two assets and borrow": "483384",
+  "flashLoan: flash loan for two assets and borrow": "482657",
   "flashLoanSimple: simple flash loan": "170603",
-  "liquidationCall: deficit on liquidated asset": "390368",
-  "liquidationCall: deficit on liquidated asset + other asset": "489010",
-  "liquidationCall: full liquidation": "390368",
-  "liquidationCall: full liquidation and receive ATokens": "366414",
-  "liquidationCall: partial liquidation": "381793",
-  "liquidationCall: partial liquidation and receive ATokens": "357839",
+  "liquidationCall: deficit on liquidated asset": "390029",
+  "liquidationCall: deficit on liquidated asset + other asset": "488641",
+  "liquidationCall: full liquidation": "390029",
+  "liquidationCall: full liquidation and receive ATokens": "366075",
+  "liquidationCall: partial liquidation": "381454",
+  "liquidationCall: partial liquidation and receive ATokens": "357501",
-  "withdraw: partial withdraw with active borrows": "238904"
```

```
+  "withdraw: partial withdraw with active borrows": "238566"
```

Recommended Mitigation: See commits 01e1024, 48d773e.

7.1.12 Use named returns in `ReserveLogic::cache` and `cumulateToLiquidityIndex`

Description: Using named returns in `ReserveLogic::cache` and `cumulateToLiquidityIndex` provides nice gas reductions across many functions.

Impact: `snapshots/Pool.Operations.json`:

```
-  "borrow: first borrow->borrowingEnabled": "255775",
-  "borrow: recurrent borrow": "248098",
-  "flashLoan: flash loan for one asset": "197361",
-  "flashLoan: flash loan for one asset and borrow": "278352",
-  "flashLoan: flash loan for two assets": "325455",
-  "flashLoan: flash loan for two assets and borrow": "482562",
-  "flashLoanSimple: simple flash loan": "170603",
-  "liquidationCall: deficit on liquidated asset": "390014",
-  "liquidationCall: deficit on liquidated asset + other asset": "488644",
-  "liquidationCall: full liquidation": "390014",
-  "liquidationCall: full liquidation and receive ATokens": "366060",
-  "liquidationCall: partial liquidation": "381439",
-  "liquidationCall: partial liquidation and receive ATokens": "357486",
-  "repay: full repay": "176521",
-  "repay: full repay with ATokens": "173922",
-  "repay: partial repay": "189949",
-  "repay: partial repay with ATokens": "185129",
-  "supply: collateralDisabled": "146755",
-  "supply: collateralEnabled": "146755",
-  "supply: first supply->collateralEnabled": "176229",
-  "withdraw: full withdraw": "165226",
-  "withdraw: partial withdraw": "181916",
-  "withdraw: partial withdraw with active borrows": "238552"
+  "borrow: first borrow->borrowingEnabled": "255438",
+  "borrow: recurrent borrow": "247760",
+  "flashLoan: flash loan for one asset": "197044",
+  "flashLoan: flash loan for one asset and borrow": "278015",
+  "flashLoan: flash loan for two assets": "324816",
+  "flashLoan: flash loan for two assets and borrow": "481887",
+  "flashLoanSimple: simple flash loan": "170288",
+  "liquidationCall: deficit on liquidated asset": "389335",
+  "liquidationCall: deficit on liquidated asset + other asset": "487617",
+  "liquidationCall: full liquidation": "389335",
+  "liquidationCall: full liquidation and receive ATokens": "365723",
+  "liquidationCall: partial liquidation": "380761",
+  "liquidationCall: partial liquidation and receive ATokens": "357148",
+  "repay: full repay": "176189",
+  "repay: full repay with ATokens": "173590",
+  "repay: partial repay": "189617",
+  "repay: partial repay with ATokens": "184797",
+  "supply: collateralDisabled": "146423",
+  "supply: collateralEnabled": "146423",
+  "supply: first supply->collateralEnabled": "175897",
+  "withdraw: full withdraw": "164894",
+  "withdraw: partial withdraw": "181583",
+  "withdraw: partial withdraw with active borrows": "238216"
```

Recommended Mitigation: See commit f51ced5.

7.1.13 Misc used named returns to eliminate local variables or for `memory` returns

Description: Generally using named returns is more efficient either when it can remove a local variable and/or when the return is `memory`.

Impact: `snapshots/Pool.Operations.json`:

```
-  "borrow: first borrow->borrowingEnabled": "255438",
-  "borrow: recurrent borrow": "247760",
+  "borrow: first borrow->borrowingEnabled": "255409",
+  "borrow: recurrent borrow": "247710",
   "flashLoan: flash loan for one asset": "197044",
-  "flashLoan: flash loan for one asset and borrow": "278015",
+  "flashLoan: flash loan for one asset and borrow": "277986",
   "flashLoan: flash loan for two assets": "324816",
-  "flashLoan: flash loan for two assets and borrow": "481887",
+  "flashLoan: flash loan for two assets and borrow": "481858",
-  "repay: full repay": "176189",
-  "repay: full repay with ATokens": "173590",
-  "repay: partial repay": "189617",
-  "repay: partial repay with ATokens": "184797",
+  "repay: full repay": "176156",
+  "repay: full repay with ATokens": "173565",
+  "repay: partial repay": "189587",
+  "repay: partial repay with ATokens": "184775",
   "supply: collateralDisabled": "146423",
   "supply: collateralEnabled": "146423",
-  "supply: first supply->collateralEnabled": "175897",
+  "supply: first supply->collateralEnabled": "175868",
```

`snapshots/RewardsController.json`:

```
-  "claimAllRewards: one reward type": "50167",
-  "claimAllRewardsToSelf: one reward type": "49963",
-  "claimRewards partial: one reward type": "48299",
-  "claimRewards: one reward type": "48037",
-  "configureAssets: one reward type": "264184",
+  "claimAllRewards: one reward type": "50131",
+  "claimAllRewardsToSelf: one reward type": "49927",
+  "claimRewards partial: one reward type": "48250",
+  "claimRewards: one reward type": "47988",
+  "configureAssets: one reward type": "264175",
```

`snapshots/StataTokenV2.json`:

```
-  "claimRewards": "359669",
+  "claimRewards": "359522",
```

Note: there were more benefits as well from later commits.

Recommended Mitigation: See commits ff2f190, 460e574, 47933e7, 2ad50db, 9e76a0b, 757b9a7.

7.1.14 Only read from and write to `userConfig` storage once in `SupplyLogic::executeWithdraw`

Description: Only read from and write to `userConfig` storage once in `SupplyLogic::executeWithdraw`.

Impact: `snapshots/Pool.Operations.json`:

```
-  "supply: first supply->collateralEnabled": "175868",
-  "withdraw: full withdraw": "164894",
-  "withdraw: partial withdraw": "181583",
-  "withdraw: partial withdraw with active borrows": "238208"
+  "supply: first supply->collateralEnabled": "175872",
```

```
+  "withdraw: full withdraw": "164728",
+  "withdraw: partial withdraw": "181499",
+  "withdraw: partial withdraw with active borrows": "237972"
```

Recommended Mitigation: See commit ba371a8.

7.1.15 Cache `usersConfig[params.from]` in `SupplyLogic::executeFinalizeTransfer`

Description: Caching `usersConfig[params.from]` in `SupplyLogic::executeFinalizeTransfer` and putting `uint256 reserveId = reserve.id;` inside the `if` statement provides a gas reduction across a number of functions.

Impact: `snapshots/AToken.transfer.json`:

```
-  "full amount; sender: ->disableCollateral;": "103316",
-  "full amount; sender: ->disableCollateral; receiver: ->enableCollateral": "145062",
-  "full amount; sender: ->disableCollateral; receiver: dirty, ->enableCollateral": "132989",
+  "full amount; sender: ->disableCollateral;": "103282",
+  "full amount; sender: ->disableCollateral; receiver: ->enableCollateral": "145028",
+  "full amount; sender: ->disableCollateral; receiver: dirty, ->enableCollateral": "132955",
-  "partial amount; sender: collateralEnabled;": "103347",
-  "partial amount; sender: collateralEnabled; receiver: ->enableCollateral": "145093"
+  "partial amount; sender: collateralEnabled;": "103208",
+  "partial amount; sender: collateralEnabled; receiver: ->enableCollateral": "144954"
```

`snapshots/StataTokenV2.json`:

```
-  "depositATokens": "219313",
+  "depositATokens": "219279",
-  "redeemAToken": "152637"
+  "redeemAToken": "152498"
```

`snapshots/WrappedTokenGatewayV3.json`:

```
-  "withdrawETH": "258800"
+  "withdrawETH": "258766"
```

Recommended Mitigation: See commit aba92d2.

7.1.16 Cache `userConfig` in `BorrowLogic::executeBorrow`

Description: Cache `userConfig` in `BorrowLogic::executeBorrow`.

Impact: `snapshots/Pool.Operations.json`:

```
-  "borrow: first borrow->borrowingEnabled": "255409",
-  "borrow: recurrent borrow": "247710",
+  "borrow: first borrow->borrowingEnabled": "255253",
+  "borrow: recurrent borrow": "247555",
   "flashLoan: flash loan for one asset": "197044",
-  "flashLoan: flash loan for one asset and borrow": "277986",
+  "flashLoan: flash loan for one asset and borrow": "277830",
   "flashLoan: flash loan for two assets": "324816",
-  "flashLoan: flash loan for two assets and borrow": "481858",
+  "flashLoan: flash loan for two assets and borrow": "481547",
```

Recommended Mitigation: See commit 204a894.

7.1.17 In `RewardsDistributor`, cache `storage` array lengths and when expected to read it `>= 3` times also for `memory`

Description: Cache array lengths for `storage` and when expected to read it `>= 3` times also for `memory`.

Impact: `snapshots/RewardsController.json`:

```
-  "getUserAccruedRewards: one reward type": "2182"
+  "getUserAccruedRewards: one reward type": "2090"
```

Recommended Mitigation: See commit a3117ba.

7.1.18 Cache event emission parameters in `RewardsDistributor:::setDistributionEnd`, `setEmissionPerSecond`

Description: Cache event emission parameters in `RewardsDistributor:::setDistributionEnd`, `setEmissionPerSecond`.

Impact: `snapshots/RewardsController.json`:

```
-  "setDistributionEnd": "5972"
+  "setDistributionEnd": "5940"
-  "setEmissionPerSecond: one reward one emission": "11541"
+  "setEmissionPerSecond: one reward one emission": "11455"
```

Recommended Mitigation: See commits 8f488dc, f516e0c, c932b96, 38408b1.

7.1.19 Read and increment `availableRewardsCount` in same statement in `RewardsDistributor::_configureAssets`

Description: Read and increment `availableRewardsCount` in same statement in `RewardsDistributor::_configureAssets`.

Impact: `snapshots/RewardsController.json`:

```
-  "configureAssets: one reward type": "264175",
+  "configureAssets: one reward type": "263847",
```

Recommended Mitigation: See commit 654ecad.

7.1.20 Exit quickly in `RewardsDistributor::_updateData` when `numAvailableRewards == 0`

Description: Exit quickly in `RewardsDistributor::_updateData` when `numAvailableRewards == 0` to avoid unnecessary work prior to exiting.

This optimization improves performance of any functions where the most common case is `numAvailableRewards = 0`, but results in worse performance when claiming rewards when `numAvailableRewards != 0`.

Hence it is a trade-off that should be considered based on what is the most likely case.

Impact: `snapshots/AToken.transfer.json`:

```
-  "full amount; receiver: ->enableCollateral": "144885",
-  "full amount; sender: ->disableCollateral;": "103282",
-  "full amount; sender: ->disableCollateral; receiver: ->enableCollateral": "145028",
-  "full amount; sender: ->disableCollateral; receiver: dirty, ->enableCollateral": "132955",
-  "full amount; sender: collateralDisabled": "103139",
-  "partial amount; sender: collateralDisabled;": "103139",
-  "partial amount; sender: collateralDisabled; receiver: ->enableCollateral": "144885",
-  "partial amount; sender: collateralEnabled;": "103208",
-  "partial amount; sender: collateralEnabled; receiver: ->enableCollateral": "144954"
+  "full amount; receiver: ->enableCollateral": "144757",
```

```
+  "full amount; sender: ->disableCollateral;": "103154",
+  "full amount; sender: ->disableCollateral; receiver: ->enableCollateral": "144900",
+  "full amount; sender: ->disableCollateral; receiver: dirty, ->enableCollateral": "132827",
+  "full amount; sender: collateralDisabled": "103011",
+  "partial amount; sender: collateralDisabled;": "103011",
+  "partial amount; sender: collateralDisabled; receiver: ->enableCollateral": "144757",
+  "partial amount; sender: collateralEnabled;": "103080",
+  "partial amount; sender: collateralEnabled; receiver: ->enableCollateral": "144826"
```

`snapshots/Pool.Operations.json`:

```
-  "borrow: first borrow->borrowingEnabled": "255253",
-  "borrow: recurrent borrow": "247555",
+  "borrow: first borrow->borrowingEnabled": "255189",
+  "borrow: recurrent borrow": "247491",

-  "flashLoan: flash loan for one asset and borrow": "277830",
+  "flashLoan: flash loan for one asset and borrow": "277766",

-  "flashLoan: flash loan for two assets and borrow": "481547",
+  "flashLoan: flash loan for two assets and borrow": "481419",

-  "liquidationCall: deficit on liquidated asset": "389335",
-  "liquidationCall: deficit on liquidated asset + other asset": "487617",
-  "liquidationCall: full liquidation": "389335",
-  "liquidationCall: full liquidation and receive ATokens": "365723",
-  "liquidationCall: partial liquidation": "380761",
-  "liquidationCall: partial liquidation and receive ATokens": "357148",
-  "repay: full repay": "176156",
-  "repay: full repay with ATokens": "173565",
-  "repay: partial repay": "189587",
-  "repay: partial repay with ATokens": "184775",
-  "supply: collateralDisabled": "146423",
-  "supply: collateralEnabled": "146423",
+  "liquidationCall: deficit on liquidated asset": "389079",
+  "liquidationCall: deficit on liquidated asset + other asset": "487297",
+  "liquidationCall: full liquidation": "389079",
+  "liquidationCall: full liquidation and receive ATokens": "365403",
+  "liquidationCall: partial liquidation": "380505",
+  "liquidationCall: partial liquidation and receive ATokens": "356828",
+  "repay: full repay": "176092",
+  "repay: full repay with ATokens": "173437",
+  "repay: partial repay": "189523",
+  "repay: partial repay with ATokens": "184647",
+  "supply: collateralDisabled": "146359",
+  "supply: collateralEnabled": "146359",
```

`snapshots/RewardsController.json`: (claiming worse)

```
-   "claimAllRewards: one reward type": "50131",
-   "claimAllRewardsToSelf: one reward type": "49927",
-   "claimRewards partial: one reward type": "48250",
-   "claimRewards: one reward type": "47988",
+  "claimAllRewards: one reward type": "50311",
+  "claimAllRewardsToSelf: one reward type": "50107",
+  "claimRewards partial: one reward type": "48430",
+  "claimRewards: one reward type": "48168"
```

`snapshots/StataTokenV2.json`: (some worse, some better)

```
-  "claimRewards": "359522",
-  "deposit": "280209",
```

```
-  "depositATokens": "219279",
-  "redeem": "205420",
-  "redeemAToken": "152498"
+  "claimRewards": "359882",
+  "deposit": "280145",
+  "depositATokens": "219151",
+  "redeem": "205356",
+  "redeemAToken": "152370"
```

`snapshots/WrappedTokenGatewayV3.json`:

```
-  "borrowETH": "249186",
-  "depositETH": "222292",
-  "repayETH": "192572",
-  "withdrawETH": "258766"
+  "borrowETH": "249122",
+  "depositETH": "222228",
+  "repayETH": "192508",
+  "withdrawETH": "258574"
```

Recommended Mitigation: See commit be7f13c.

7.1.21 Read and increment `streamId` in same statement, also use named return in `Collector::createStream`

Description: Read and increment `streamId` in same statement, also use named return in `Collector::createStream`.

Impact: `snapshots/Collector.json`:

```
-  "createStream": "211680",
+  "createStream": "211600",
```

Recommended Mitigation: See commit a008981.

7.1.22 Don't copy entire `Stream` struct from `storage` to `memory` in `Collector::deltaOf`

Description: Don't copy entire `Stream` struct from `storage` to `memory` in `Collector::deltaOf`, since only 2 variables are required.

Impact: `snapshots/Collector.json`:

```
-  "cancelStream: by funds admin": "18522",
-  "cancelStream: by recipient": "49489",
+  "cancelStream: by funds admin": "16710",
+  "cancelStream: by recipient": "47635",
-  "withdrawFromStream: final withdraw": "43594",
-  "withdrawFromStream: intermediate withdraw": "42252"
+  "withdrawFromStream: final withdraw": "42656",
+  "withdrawFromStream: intermediate withdraw": "41326"
```

Recommended Mitigation: See commit 78c8150.

7.1.23 Remove struct `BalanceOfLocalVars` and use local variables in `Collector::balanceOf`

Description: Remove struct `BalanceOfLocalVars` and use local variables in `Collector::balanceOf`.

Impact: `snapshots/Collector.json`:

```
-  "cancelStream: by funds admin": "16710",
-  "cancelStream: by recipient": "47635",
```

```
+  "cancelStream: by funds admin": "16483",
+  "cancelStream: by recipient": "47407",
-  "withdrawFromStream: final withdraw": "42656",
-  "withdrawFromStream: intermediate withdraw": "41326"
+  "withdrawFromStream: final withdraw": "42560",
+  "withdrawFromStream: intermediate withdraw": "41230"
```

Recommended Mitigation: See commit cc31124.

7.1.24 Remove struct `CreateStreamLocalVars` and use local variables in `Collector::createStream`

Description: Remove struct `CreateStreamLocalVars` and use local variables in `Collector::createStream`.

Impact: `snapshots/Collector.json`:

```
-  "createStream": "211600",
+  "createStream": "211518",
```

Recommended Mitigation: See commit dc3da97.

7.1.25 Don't copy entire `Stream` struct from `storage` to `memory` and refactor `onlyAdminOrRecipient` modifier into internal function in `Collector::withdrawFromStream` and `cancelStream`

Description: Don't copy entire `Stream` struct from `storage` to `memory` and refactor `onlyAdminOrRecipient` modifier into internal function in `Collector::withdrawFromStream` and `cancelStream`.

Impact: `snapshots/Collector.json`:

```
-  "cancelStream: by funds admin": "16483",
-  "cancelStream: by recipient": "47407",
+  "cancelStream: by funds admin": "15718",
+  "cancelStream: by recipient": "46456",
-  "withdrawFromStream: final withdraw": "42560",
-  "withdrawFromStream: intermediate withdraw": "41230"
+  "withdrawFromStream: final withdraw": "41449",
+  "withdrawFromStream: intermediate withdraw": "40224"
```

Recommended Mitigation: See commit 73e6123.

7.1.26 Cache `oldId` prior to `require` check to save 1 storage read in `PoolAddressesProviderRegistry::unregisterAddressesProvider`

Description: Cache `oldId` prior to `require` check to save 1 storage read in `PoolAddressesProviderRegistry::unregisterAddressesProvider`:

Recommended Mitigation:

```
  function unregisterAddressesProvider(address provider) external override onlyOwner {
-   require(_addressesProviderToId[provider] != 0, Errors.ADDRESSES_PROVIDER_NOT_REGISTERED);
    uint256 oldId = _addressesProviderToId[provider];
+   require(oldId != 0, Errors.ADDRESSES_PROVIDER_NOT_REGISTERED);
    _idToAddressesProvider[oldId] = address(0);
    _addressesProviderToId[provider] = 0;
```

See commit 93935b8.