

# Integrating Parallelized Algorithms into the Daedalus Wallet

## 1 INTRODUCTION

This note discusses the plan for integrating parallelized algorithms into the Daedalus Wallet [2]. The discussed algorithms are presented in the following papers:

- *Parallelized Ouroboros Praos* [8] discusses the key component of the trust model: the parallelized approach to the validation of consensus.
- *On the Security of Wallet Nodes in the Cardano Blockchain* [7] discusses changes to the trust model and the impact of these changes on the security of individual wallet nodes.
- *Highly Parallel Reconstruction of Wallet History in the Cardano Blockchain* [5] presents a parallel algorithm for wallet history reconstruction.
- *Scalability of Bulk Synchronization in the Cardano Blockchain* [6] discusses problems and solutions to the fast network delivery of blockchain data.

The scope of this note is the integration only, and readers are encouraged to refer to the above papers for details of individual algorithms.

## 2 INTEGRATION OPTIONS

Two integration options are proposed for discussion: generation of binary-compatible state data and emulation of the Cardano Wallet HTTP API. The pros and cons of each are discussed in the following subsections.

### 2.1 Generation of binary-compatible state data

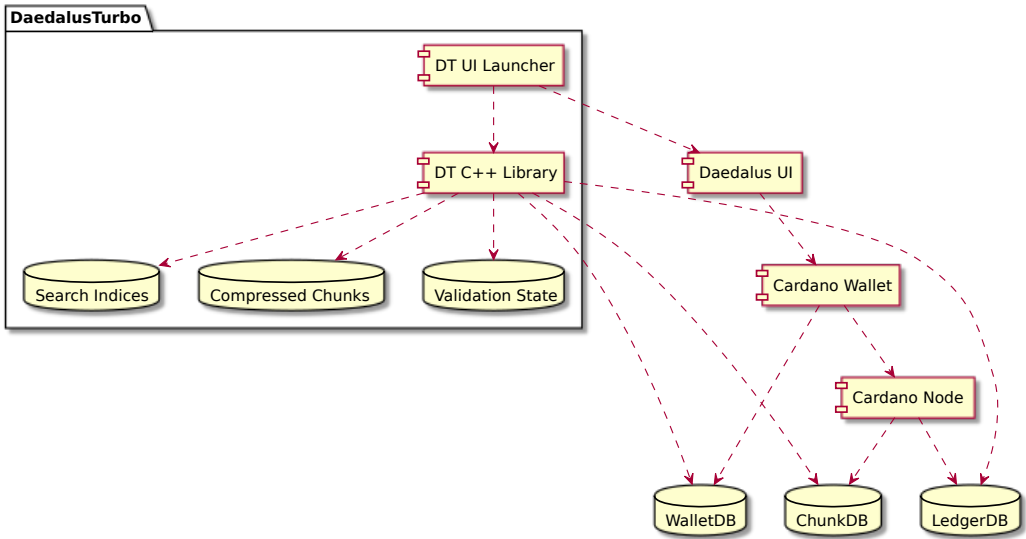


Fig. 1. A component diagram of the generation of binary-compatible data

Figure 1 presents an overview of the generation of binary-compatible state data. The essence of this option is in updating the state data (raw blockchain data, ledger snapshots, and wallet database files for individual wallets registered with the Daedalus instance) used by the Daedalus wallet before the application is started.

This integration option is the least intrusive and allows the maximum reuse of existing software components. In addition, it makes it easy to enable and disable the new algorithms and to switch between data-generation methods.

Conversely, this option requires additional storage space, as data structures used by the new algorithms and the existing software components must be stored simultaneously. Furthermore, this option does not allow to provide additional run-time security features, such as dynamic querying of Cardano network nodes, tail density estimation, and a local blockchain explorer.

## 2.2 Emulation of Cardano Wallet HTTP API

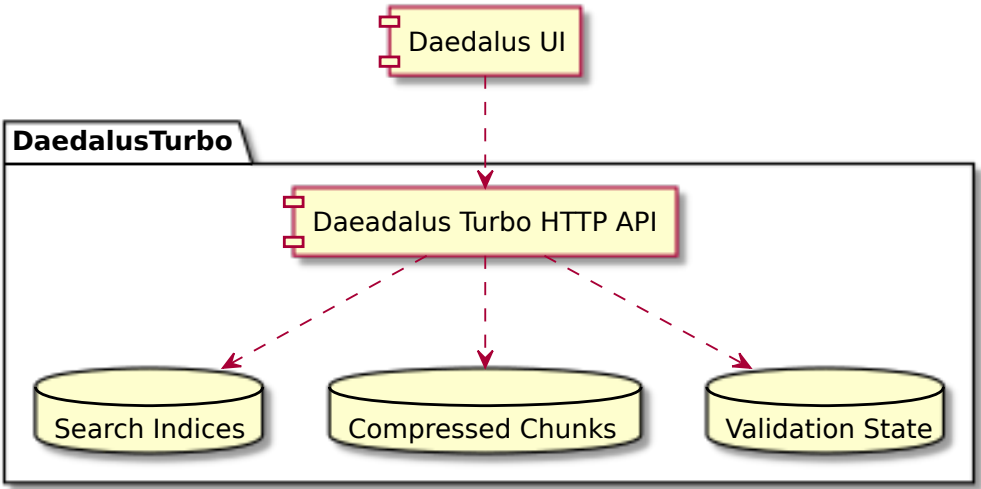


Fig. 2. A component diagram for emulation of Cardano Wallet HTTP API

Figure 2 presents a component diagram explaining the emulation of the Cardano Wallet HTTP API, whose essence is the creation of a local HTTP service providing an HTTP API emulating the API endpoints of the Cardano Wallet [1] used by the Daedalus Wallet.

The benefits of this approach are that it allows the integration of additional run-time verifications and features and it reduces storage requirements due to the use of compression by the new algorithms. Moreover, this option can allow partial reuse of existing functionality, as HTTP API requests can be forwarded to the original Cardano Wallet API.

Conversely, this option comes at the expense of additional software development and quality assurance costs.

## 3 EFFICIENT AND RELIABLE DELIVERY BLOCKCHAIN DATA

The effective use of data compression can noticeably reduce the cost of blockchain data delivery, as explained in the following subsection. Simultaneously, the Cardano network protocol does not currently support data compression, which means that during the introduction

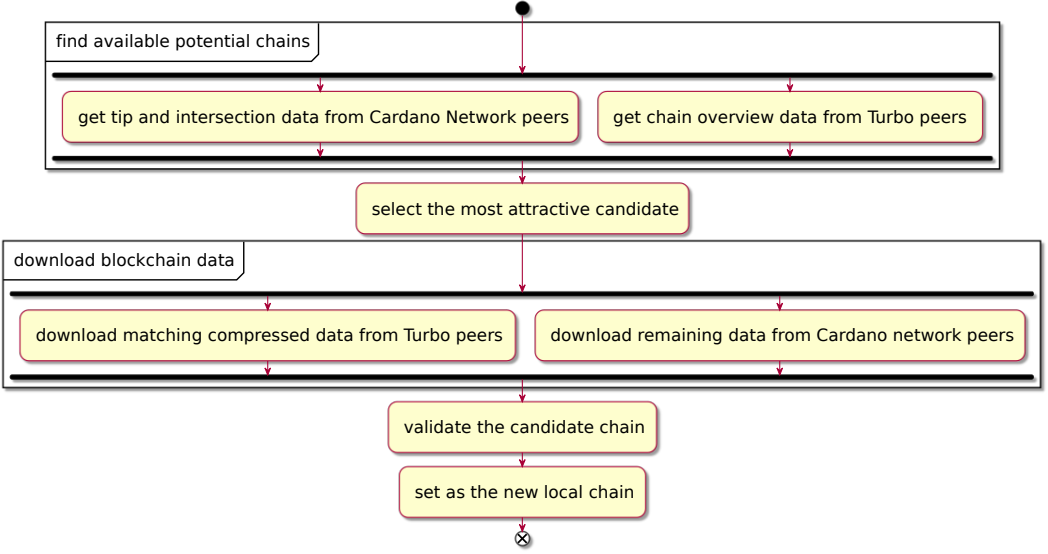


Fig. 3. Hybrid blockchain data discovery and delivery

of data compression, only a subset of Cardano network nodes may be used, potentially creating a problem with the centralization of data delivery.

However, the use of a hybrid data discovery and data delivery algorithm, as presented in Figure 3, eliminates this problem, because the algorithm functions as follows:

- To discover potential chains, a wallet node collects information from both nodes supporting the standard Cardano network protocol and compressed data delivery.
- The chain selection algorithm chooses the most attractive chain independently of the supported network protocol.
- Blocks from the selected candidate chain are checked for presence on peers supporting compressed data transfers via hashes of their content. In this case, these peers function as a pure content delivery network and do not influence chain selection in any way.
- Remaining blocks are downloaded directly using the standard Cardano network protocol.

The above algorithm allows use of the same chain selection rule as other nodes, allows accelerated data delivery when peers supporting compression have the necessary data, and can reliably function without any peers supporting compressed data transfers.

### 3.1 Cost of data delivery

Figure 4 presents a screenshot of live compression statistics from one of the compressing proxies. It shows that on April 29, 2024, data compression reduced necessary network traffic by a factor of 4.58. Assuming a cost of data distribution of U.S. \$10 per terabyte of traffic, delivering one full copy of the blockchain with compression enabled costs only U.S. \$0.39 and U.S. \$1.78 when compression is disabled. To highlight the scale of this difference further, it means more than a billion dollars less in networking expenses for a billion full blockchain data transfers.

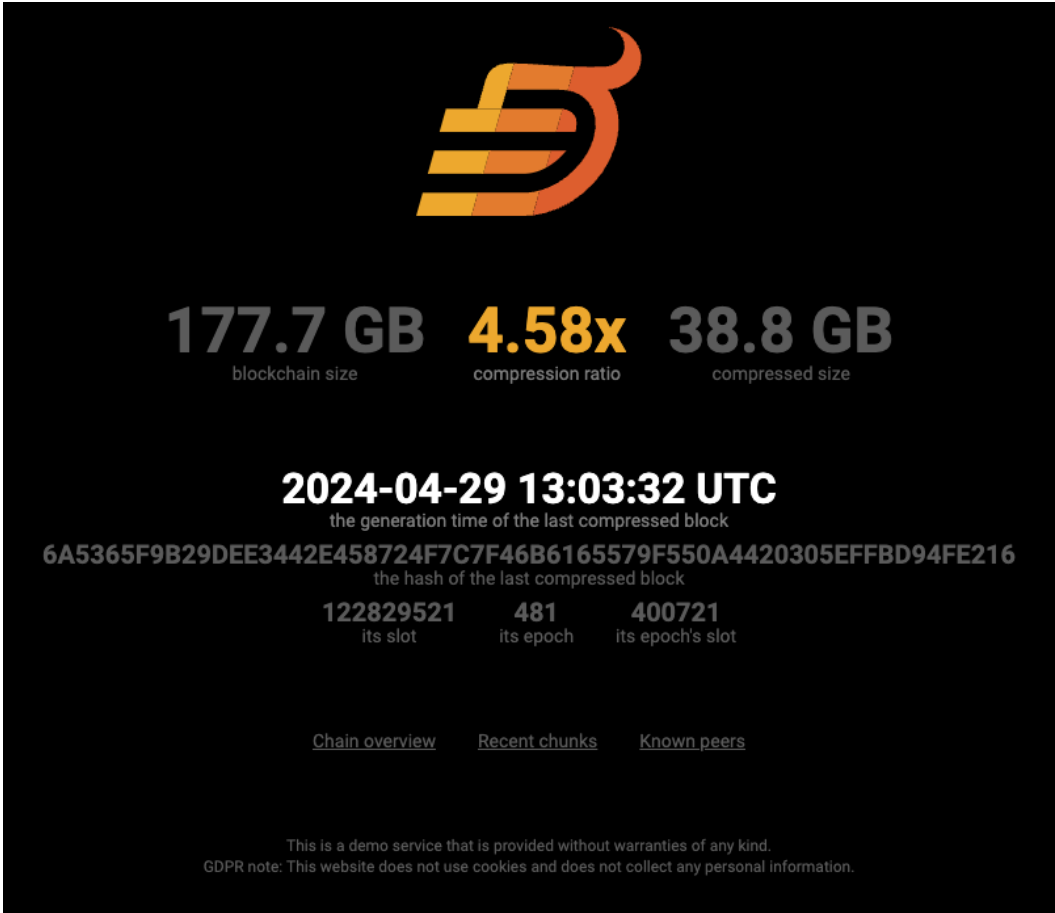


Fig. 4. A screenshot of live compression reporting by a compressing proxy

3.2 Data distribution delay

Statistic	Delay, seconds
50th percentile	2.88
95th percentile	4.31
99th percentile	5.26

Table 1. Compressed data distribution delay

Because data compression is currently not integrated into the Cardano network protocol, compressed data blocks become available with a delay. This delay is composed of the time necessary for new blockchain blocks to reach compressing nodes, data compression time, and the publication time of compressed data for further distribution. Table 1 presents the measurements [3] of the data distribution delay of a straightforward implementation providing support for data compression. The median delay is three seconds and that in the 99% percentile is five seconds. Both constitute only a fraction of the Cardano’s current

block-generation period of 20 seconds. Therefore, it presents no noticeable problems for owners of personal wallet nodes.

#### 4 TESTING AND SECURITY VERIFICATION

To ensure the security and quality of the integrated algorithms, milestone 4 of the project plans the following activities:

- Code review of modules that directly affect security, such data validation, networking, etc.
- Extension of existing unit tests with additional test cases, focusing on error-handling and rare-event handling.
- Creation of additional integration tests emulating various known attacks against the blockchain to provide empirical evidence of the security of the implementation.

#### 5 COMPONENTS TO BE IMPROVED FOR INTEGRATION

In addition to the components described in Section 2 support is planned for the following features:

- Full support of tail validation, as described in [7]
- Support for the expected upgrades to the Cardano blockchain: Ouroboros Genesis and Conway block format
- Improved support for the historical Byron block format

#### 6 MODIFICATION COSTS TO SUPPORT FUTURE CARDANO UPGRADES

Component	Code Lines, Thousands	Effort, Person-Months
Cardano block parsers	4.7	12
Blockchain storage and indexing	3.3	8
Data validation	2.6	6
Explorer User Interface	2.4	6
Synchronization	1.3	3
Cryptographic primitives	0.9	2
History reconstruction	0.5	1
Supporting code	6.0	16
<b>Total</b>	<b>21.7</b>	<b>54</b>

Table 2. Component complexity and the estimated development effort

Table 2 presents a high-level overview of the existing software components and their relative complexity. The estimates of the development effort were calculated using the SLOC-Count tool [4].

The Cardano blockchain has already undergone several upgrades: Shelley, Allegra, Mary, Alonzo, and Babbage. Therefore, it is possible to analyze the potential complexity of adjusting to upgrades based on past experience.

For more straightforward upgrades that involve only changes to block formats, modification complexity required less than 1,000 lines of code or two person-months of development effort.

For more extensive upgrades, such as the introduction of the Ouroboros Praos consensus or scripting support, in addition to the block format changes, the development of new software components was necessary, which in these cases required up to 2,000 lines of new code, thus totalling up to 3,000 lines of code or about eight person-months of development effort.

Given that the development effort can be spread across several engineers, it is reasonable to expect that preparing even for the most complex upgrades within six months is entirely feasible.

## 7 CONCLUSION

This note discusses options for the integration of the parallelized algorithms presented in [5–8] into the Daedalus wallets. In addition, it explains the planned testing and security verification procedures and evaluates the modification efforts in the case of future upgrades to the Cardano blockchain.

## REFERENCES

- [1] 2024. Cardano Wallet Backend API (v2024-03-27). <https://cardano-foundation.github.io/cardano-wallet/api/edge/>
- [2] 2024. Daedalus Wallet. <https://daedaluswallet.io/>.
- [3] 2024. GitHub: Daedalus Turbo Data Distribution Delay Experiment. <https://github.com/sierkov/daedalus-turbo/tree/main/experiment/distribution-delay>
- [4] 2024. SLOCCount. <https://dwheeler.com/sloccount/>
- [5] Alex Sierkov. 2023. Highly Parallel Reconstruction of Wallet History in the Cardano Blockchain. (2023). [https://github.com/sierkov/daedalus-turbo/blob/main/doc/2023\\_Sierkov\\_WalletHistoryReconstruction.pdf](https://github.com/sierkov/daedalus-turbo/blob/main/doc/2023_Sierkov_WalletHistoryReconstruction.pdf)
- [6] Alex Sierkov. 2023. Scalability of Bulk Synchronization in the Cardano Blockchain. (2023). [https://github.com/sierkov/daedalus-turbo/blob/main/doc/2023\\_Sierkov.CardanoBulkSynchronization.pdf](https://github.com/sierkov/daedalus-turbo/blob/main/doc/2023_Sierkov.CardanoBulkSynchronization.pdf)
- [7] Alex Sierkov. 2024. On the Security of Wallet Nodes in the Cardano Blockchain. (2024). <https://github.com/sierkov/daedalus-turbo/blob/main/doc/2024-sierkov-on-wallet-security.pdf>
- [8] Alex Sierkov. 2024. Parallelized Ouroboros Praos. (2024). <https://github.com/sierkov/daedalus-turbo/blob/main/doc/2024-sierkov-parallelized-ouroboros-praos.pdf>