

Research Report

Scalability of Bulk Synchronization in the Cardano Blockchain

Alex Sierkov
alex.sierkov@gmail.com

June 20, 2023

Contents

Introduction	2
Bulk-synchronization constraints of the Cardano network protocol	3
2.1 Overview of the protocol's design objectives	3
2.2 Performance of personal wallets is not a priority	3
2.3 Unsustainable cost of network traffic	4
2.4 Insufficient financial incentives for stake-pool operators	4
2.5 Limited number of simultaneous high-bandwidth clients	8
2.6 Limitations of Chain-Sync and Block-Fetch mini-protocols	8
2.7 Limited peer discovery	9
Analysis of Mithril as a bulk-synchronization method	12
3.1 Overview of Mithril's design objectives	12
3.2 Scalable data distribution not within the project's scope	12
3.3 Snapshot format not suitable for parallel and incremental processing	13
An alternative high-performance bulk-synchronization method	14
4.1 Requirements for high-bandwidth bulk synchronization	14
4.2 Data sharing by downloading clients	16
4.3 Optional rewarded access to high-bandwidth data distribution	20
4.4 Hierarchical data discovery	22
4.5 Transport compression of blockchain data	23
4.6 Parallel validation of blockchain data	27
Discussion	31
5.1 Handling a billion clients	31
5.2 A measurement framework for bulk-synchronization performance	32
5.3 Suggested implementation plan	32
5.4 Recommended adjustments to the Cardano network protocol	33
Conclusion	34
References	35

Introduction

The primary objective of a blockchain is to establish a shared view of a transaction history among its participants. As a public blockchain, Cardano must be able to do this for an unconstrained number of clients. Many Cardano software clients are personal wallets that do not track blockchain updates 24/7. For them, the time of bulk synchronization, the process of synchronizing large chunks of the blockchain, is a major quality-of-life attribute. At the same time, Cardano's bulk-synchronization time keeps degrading under the pressure of two factors growing exponentially over time: number of participants and data size. This report aims to answer two questions:

- Can Cardano's current design of bulk synchronization handle growth to one billion clients and one terabyte in size? ¹
- Can the pressure exerted by the exponential growth be neutralized by active countermeasures?

A good way to tame an exponentially-growing factor is to counter it with another exponentially-growing one. Thus, this report analyzes the impact of applying two countermeasures:

- Leveraging the growth in the number of personal-wallet clients, more specifically, by making personal wallets participate in the data distribution;
- Leveraging the continuing growth in the power of consumer hardware, more specifically, the growth of parallel-processing capabilities and the speed of Internet access.

The contributions of this report are

- Providing evidence that the performance and scalability of bulk synchronization in Cardano are limited by the design of its network protocol and by Cardano's structure of stake-pool rewards;
- Proposing specific adjustments to scale Cardano's bulk synchronization to the aforementioned growth targets;
- Proposing an evaluation framework for bulk-synchronization optimizations;
- Providing an analysis of Mithril as a method for bulk synchronization;
- Presenting a parallel method for validation of Cardano blockchain data.

¹One terabyte is the current scale of Ethereum, the older and bigger blockchain, with which Cardano competes.

Bulk-synchronization constraints of the Cardano network protocol

2.1 Overview of the protocol's design objectives

The design of the Cardano network protocol ([1] and [2]) is focused on security against various attacks, with particular attention given to the timeliness of data distribution among block-producing nodes. The reason for that is that Cardano's consensus protocol [3] operates in time-bound slots, so an untimely delivery of blocks to slot leaders induces blockchain forks and reduces the security of the protocol.

The protocol's design document makes several assumptions:

- The total number of nodes is expected to be between ten thousand and hundred thousand;
- One thousand largest stake pools are expected to control 80% of the total stake;
- The number of hops between large block-producing nodes is expected to be five or less;
- New blocks are expected to reach 95% of large block-producing nodes within five seconds.

The following sections highlight the problems that flow out of the design and affect the performance of bulk synchronization.

2.2 Performance of personal wallets is not a priority

Personal wallets are the most widespread use case among general users of Cardano, and their requirements are noticeably different from those of block-producing nodes. The most important difference is that the number of personal wallets can grow far above the upper bound of a hundred thousand nodes considered by the protocol's design: ten million and even a billion personal wallets are conceivable numbers.

There are further differences:

- Personal wallets are synchronized occasionally and do not stay online 24/7, so the speed of bulk synchronization is of high importance;
- Wallets are less sensitive to the timeliness of data delivery: a 20-second delivery delay to a slot leader may lead to a blockchain fork, but a 20-second delivery delay to a personal wallet will likely not even be noticed since end users are accustomed to waiting for 20–30 block-generation periods for their funds to be recognized by crypto exchanges and other

counterparties;

- Users can have multiple devices, such as a laptop for travel and a desktop computer for home use, leading to multiple software configurations per user and thus additional demand for bulk synchronization;
- Users can regularly upgrade or replace their devices. Thus, reinstalling their wallet software and also increasing the demand for bulk synchronization.

2.3 Unsustainable cost of network traffic

The Cardano blockchain is operated by independent stake-pool operators, many of whom provision servers from major cloud providers, such as Amazon, Google, and Microsoft. These providers charge their clients for the outgoing Internet traffic. Given the exponential growth in client numbers and data size, this becomes a major cost driver.

To make this argument more specific, it is best to estimate the cost of the network traffic necessary to support the scenario set in the research question. One can estimate the traffic cost, c , using the formula

$$c = p \times v \times n, \quad (2.1)$$

where

- p = the price of traffic in US\$ per terabyte (TB going forward);
- v = the traffic volume that one client consumes in TB per year;
- n = the number of clients.

The following parameters are put into Equation (2.1):

- p = US\$10 per TB, taking the price set by smaller and more affordable cloud providers, such as Vultr [4]. In contrast, the prices set by larger providers, such as Amazon [5], Google [6], or Microsoft Azure [7], start from US\$20 per TB and can go up to US\$200 per TB depending on the geographic destination of traffic;
- v = 1 TB per year, taking the blockchain size from the research question and an assumption that clients download a full blockchain copy only once per year;
- c = 1 billion, taking the target number of clients from the research question.

The result of the calculation shows that stake-pool operators may need to pay **US\$10 billion per year** to support the scenario, a cost that is clearly beyond the financial capabilities of stake-pool operators, as is shown in the next section.

2.4 Insufficient financial incentives for stake-pool operators

Cardano's stake pools are rewarded for every epoch in which they produce at least one new block. The reward consists of a fixed amount of 340 ADA, Cardano's native coin, and the margin, a pool-defined percentage of the total reward attributable to the pool in that epoch. Usually, margins are set between 0% and 5%. Moreover, there is strong pressure to lower them because stake-pool rewards depend on the stake delegated to a given pool. Therefore, stake-pool

operators are naturally interested in attracting stake delegators. On the other hand, the key interest of stake delegators is simple return on investment. Therefore, delegators naturally prefer stake pools with low margins. Thus, without an offer of additional value to delegators (a topic discussed in section 4.3), stake pools are pressed to reduce their margins to the lowest level possible. Consequentially, they must reduce operational expenses to a level sustainable with just the minimum reward.

To better understand what level of expenses a stake pool can dedicate to network traffic, an example pool that operates at a 0% margin is analyzed. Its income per month is roughly 2040 ADA ² (340×6 since a Cardano epoch takes five days). Between June 2022 and June 2023, ADA price ranged between US\$0.24 and US\$0.64. Thus, rounding to hundred-dollar values, this stake pool would have earned between US\$500 and US\$1300 per month. With US\$900 per month being the average of the minimum and the maximum monthly reward.

Expense	Amount	Description
Rent of Cloud Instances	US\$498	Usually, Cardano stake pools use three servers to operate: one block-producing server and two relay servers distributing new blocks to the rest of the Cardano network. For optimal performance, such servers should have at least 32 GB of memory and 200 GB of SSD storage. The most popular cloud provider, Amazon, offers r6gd.xlarge instances meeting these requirements for US\$0.2304 per hour [8] or roughly US\$166 per month. Thus, US\$498 for three servers
Time of Technical Personnel	US\$200	Each stake pool must perform regular technical activities, such as upgrades to new Cardano Node versions and reactions to ongoing monitoring events ensuring high-availability of services necessary for stake-pool operations. Taking two hours of technical work each month at an hourly rate of US\$100 per hour, results in US\$200 per month
Reserve for ADA Price Volatility / Profit Margin	US\$200	Each pool owner must make a profit to have an economic sense to continue its operation. At the very minimum that means that the pool breaks even on average: during periods of high ADA prices it creates reserves covering regular expenses during periods of low ADA prices

Table 2.1: Estimated monthly expenses of a stake pool

Table 2.1 shows how the average monthly reward of US\$900 dollars can be allocated. It is clear that such a pool does not have a lot of unallocated financial resources. Moreover, it could not allocate more than US\$200 dollars, or roughly 20% of its total reward, to pay for additional network-bandwidth expenses.

Cexplorer.io [9] tracks cumulative per-epoch rewards of all stake pools in its Pool Profitability report. Between June 2022 and June 2023, the total per-epoch rewards fluctuated between US\$350,000 and US\$950,000. Taking the average of US\$650,000 and multiplying it by the number of epochs in a year, 73, the total rewards for that year can be estimated at US\$47 million. 20% of that is US\$9.4 million.

²The amount may slightly diverge for months with the number of days different from 30.

To simplify further calculations and given the natural error in this type of estimations, this number is rounded up to **US\$10 million per year**. It will be used as the maximum realistic budget for network traffic at the current reward level. **This number is a thousand times smaller than US\$10 billion** necessary to support the scenario defined in the research question. Thus, **the current structure of stake-pool rewards cannot support one billion clients**. The following subsections provide additional calculations to further develop this point.

2.4.1 Maximum number of blockchain transfers

At a cost of US\$10 per TB, US\$10 million can purchase a million TB of traffic, which is sufficient to deliver roughly eight million copies of the blockchain at its current size of 123 GB and just a million copies when its size grows to 1 TB.

2.4.2 Purchasing network bandwidth instead of network traffic

One way to reduce network-traffic costs is to use cloud providers that allow purchasing network bandwidth instead of paying for actual network traffic. Prices for 1 Gbps of outgoing bandwidth start around US\$200 per month or US\$2400 per year ([10], [11]). Furthermore, purchasing larger amounts of bandwidth, such as 10 Gbps or 40 Gbps, provides for even bigger savings. However, the low monthly rewards discussed above prevent stake pools from leveraging this optimization.

With an annual budget of US\$10 million, stake pools can purchase 4166 Gbps of network bandwidth at the price of \$2400 dollars per 1 Gbps per year. One can estimate the number of blockchain copies, N , that can be transmitted in one year with the formula

$$N = \frac{B \times 86400 \times 365}{O \times S}, \quad (2.2)$$

where

- B = the available network bandwidth in gigabits per second;
- 86400 = the number of seconds in a day;
- 365 = the number of days in a year;
- S = the blockchain size in gigabytes;
- O = the network traffic, including overhead needed to transfer one byte of blockchain, in bits.

Putting the bandwidth of 4166 Gbps, the blockchain size of 1 TB, and 10 bits of traffic for every byte of blockchain data into Equation (2.2), one arrives at 13 million copies. This is notably better than a million copies in the case of purchasing network traffic. However, that assumes full utilization of the purchased bandwidth.

In practice, client demand for network traffic is subject to strong fluctuations due to daily, weekly, and seasonal cycles and the impact of special events. Thus, to ensure that a given level of average network bandwidth is available, stake-pool operators must purchase reserve bandwidth that is three to five times that level. With an adjustment factor of three, that is four million blockchain copies given the same budget.

Therefore, even after adjusting for the necessary bandwidth reserves and without taking into account discounts from purchasing bandwidth in larger volumes, **purchasing network bandwidth is four times more efficient than purchasing network traffic**. However, implementing this optimization requires an increase in the minimum reward for stake pools so that they can afford it. This topic is further discussed in section 4.3.

2.4.3 Impact of ADA price growth on the projections

It is important to discuss one more factor that directly influences stake-pool rewards: the price of ADA. It is perfectly reasonable to assume that the growth in the number of clients will drive up its price. However, growth at the thousand-fold level, the difference between the currently possible and target number of blockchain transfers, seems unlikely. To obtain a quantitative estimate of the price growth potential, the following analysis was conducted:

- Bitcoin was chosen as the benchmark cryptocurrency. It is the oldest cryptocurrency with the largest amount of historical data. Moreover, it is the most successful cryptocurrency with regard to market capitalization and thus price growth.
- Bitcoin's historical prices were downloaded from coinmarketcap.com [12], and the corresponding annual average prices calculated. The results are presented in Figure 2.1.
- After that, years in which Bitcoin's market capitalization had not yet passed a billion dollars were excluded. That is because larger price movements are easier at lower market capitalization levels since they do not require much financial capital.
- Finally, five-year growth factors were calculated as the average price after five years divided by the price in a given year. Results for years with insufficient future data were excluded.

Figure 2.2 shows the computed five-year growth factors, which range between 2.8 and 40.9. This comparison makes it clear that expecting a thousand-time growth in ADA price is unrealistic. These numbers will be used as the realistic range of ADA price growth over the next five years.

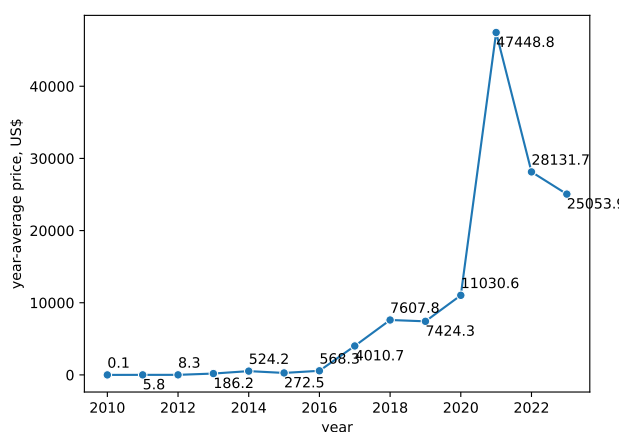


Figure 2.1: Bitcoin's historical year-average prices

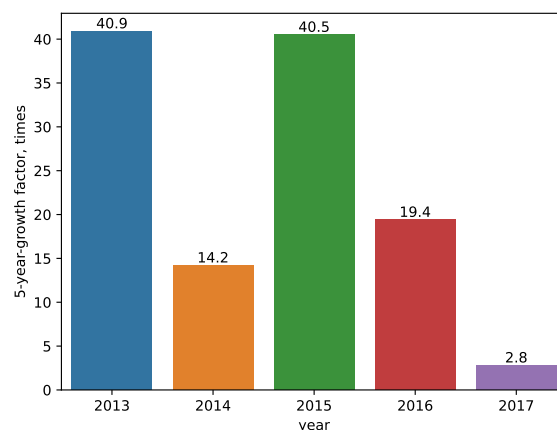


Figure 2.2: Bitcoin's five-year growth factors after reaching a billion dollars in market capitalization

2.5 Limited number of simultaneous high-bandwidth clients

Another important topic is the ability of the network to support continuing growth in the power of consumer hardware so that clients who are willing to invest more in their setups can obtain better performance. The maximum number of simultaneous clients, N , can be estimated using the formula

$$N = \left\lfloor \frac{B}{C} \right\rfloor, \quad (2.3)$$

where

B = the cumulative upload bandwidth of all stake pools;
 C = the peak download bandwidth of a single client.

A paper on highly-parallel wallet-history-reconstruction method [13] has shown that it is possible to process blockchain data at speeds of up to 4000 GB/sec, which with the assumption of 10 bits of network traffic needed for every byte of blockchain data means 40 Gbps in peak client download bandwidth. The cumulative network bandwidth of all stake pools has been already estimated in section 2.4.2 at 4166 Gbps. Thus, after putting the above parameters into Equation (2.3), with the current reward structure of stake-pool operators and the current ADA price level, **Cardano can provide blockchain data to only 104 (one hundred and four) high-bandwidth clients simultaneously**, which appears rather low for a global blockchain.

2.6 Limitations of Chain-Sync and Block-Fetch mini-protocols

The Cardano network protocol [1] is organized as a set of mini-protocols operating over a single TCP connection. For the problem of bulk synchronization, two mini-protocols are of interest: **Chain-Sync**³ and **Block-Fetch**. Brief descriptions of each mini-protocol, along with a discussion of whether each one constrains the performance of bulk synchronization, are presented in the following subsections.

2.6.1 Chain-Sync mini-protocol

The Chain-Sync mini-protocol begins with a data consumer sending a **MsgFindIntersect** message to a data-distributing node that lists potential blockchain points (pairs of a slot number and a block hash) from which the data consumer would like to receive data. The data-distributing node responds with either a **MsgIntersectFound** message if one of the points is known to it or with a **MsgIntersectNotFound** if synchronization must start from scratch. Then, the data consumer sends **MsgRequestNext** messages to request additional block headers, and the data-distributing node responds with **MsgRollForward** messages containing the header of each next block and adjusting its internal state.

³The ChainSync protocol has two versions: node-to-node, in which only block headers are transmitted, and node-to-client, which transmits full blocks. Since this report focuses on bulk synchronization over the Internet, the node-to-node version of the protocol is assumed.

The main problem with this mini-protocol is its sequential nature: block headers are transmitted one by one, and data consumers must know a slot number and the corresponding block hash to request data from anywhere beyond the blockchain’s start, which prevents data consumers from retrieving multiple segments of the blockchain in parallel. This means that data consumers are limited by the available data-upload bandwidth of a single data-distributing node until all headers have been fetched.

Headers in the Cardano blockchain constitute around 6% ⁴ of its total size. So, when the blockchain grows to 1 TB, the size of the headers will be around 60 GB. For a consumer Internet connection with a download speed of 250 Mbps assuming full utilization, that is over 40 minutes in the fully sequential phase. Moreover, full bandwidth utilization is an unrealistic expectation for a number of reasons:

- A data-distributing node may already be overloaded with other requests;
- A data-distributing node may have a smaller upload bandwidth than that of the client in the case of enterprise clients;
- One of the intermediate network segments (e.g., a cross-continental link) may be overloaded;
- The software or hardware of a data-distributing node may not be configured to maximize upload bandwidth.

Moreover, the best bandwidth maximization strategy for data consumers using Chain-Sync is to keep switching between data-distributing nodes until one can upload data at the desired speed. However, that assumes that data consumers can easily find those other nodes, and that is not the case, especially during the early stages of synchronization. Section 2.7 further discusses this point.

2.6.2 Block-Fetch mini-protocol

Block-Fetch mini-protocol starts with a data consumer sending a **MsgRequestRange** message describing the requested blockchain segment by using its start and end points (pairs of a slot number and a block hash). The data-distributing node then either declines the request with **MsgNoBlocks** if it does not have the requested data or responds with a sequence of **MsgStartBatch**, **MsgBlock**, and **MsgBatchDone** messages delivering the requested data.

This protocol can, in principle, support parallel transfer of blockchain data, but under the assumption that a data consumer knows

- the addresses of relevant peers, a point further discussed in section 2.7;
- the precise points in the blockchain to request, which recalls the constraints of Chain-Sync discussed above.

2.7 Limited peer discovery

Efficient utilization of each client’s download bandwidth depends on the ability to fetch data from multiple nodes and dynamically adjust the active peer list depending on the effective network

⁴The value has been estimated by analyzing Cardano’s mainnet data up to the end of epoch 415.

conditions. For that to work, clients must have a good overview of available peers. Fundamentally, clients can learn about those from one of these sources:

- Static configuration bundled with the client’s software;
- Analysis of stake-pool registrations and retirements in the downloaded blockchain segments;
- Requesting peer lists from already-known peers.

To understand how in practice the selection of peers works in Cardano Node, the official implementation of the Cardano network protocol, three types of activities were performed:

- An analysis of the technical documentation ([1] and [2]), which present a peer-discovery mechanism named "gossip".
- An experiment where all network traffic of a Cardano Node is captured and analyzed for IP addresses of peers. The experiment revealed that in the default configuration, Cardano Node did not use any form of peer discovery beyond static configuration. Details are presented in section 2.7.1.
- An analysis of the source code of Cardano Node’s release 1.35.5. Contrary to the documentation, the source code did not fully implement active peer discovery, as is discussed with specific evidence in section 2.7.2.

Thus, **the default configuration of Cardano Node 1.35.5 does not use any form of peer discovery during bulk synchronization.**

2.7.1 Network traffic analysis

To investigate how peer discovery works effectively, the following experiment was conducted (all configuration files are published on GitHub [14]):

- An official docker image of Cardano Node was taken: **inputoutput/cardano-node:1.35.5**;
- An instance of it was started with the default configuration and kept running until the node was synchronized;
- All network traffic from the container was recorded with the **tcpdump -n -q -l tcp** command.
- The recorded data were analyzed for unique IP addresses to establish the effective number of peers with which the node communicated during its synchronization process.

The surprising outcome was that the only IP addresses discovered in the log were those of the Cardano relay node defined in the static configuration: **relays-new.cardano-mainnet.iohk.io**. Therefore, not only did the software not download data in parallel from multiple stake pools, but it also did not connect to Cardano nodes not defined in its static configuration.

2.7.2 Source code analysis

To confirm the discovery of the network traffic analysis, an analysis of the source code of Cardano Node corresponding to release 1.35.5 was performed. The package responsible for the networking component is **ouroboros-network** [15]. Version **0.3.0.0** was analyzed (commit 926d785efb6686ee46356cf6304c688a56e1493d). That is the latest tagged version in the repository

before release 1.35.5 of Cardano Node ⁵. Two discoveries confirming the conclusions of the network traffic analysis were made:

- No network mini-protocol implementing peer discovery is implemented within a namespace dedicated to mini-protocols, **Ouroboros.Network.Protocol**.
- Analysis of **Ouroboros.Network.PeerSelection** module identified `requestPeerGossip`, the function responsible for the implementation of actual peer discovery activities. However, that function is defined only within the `Ouroboros.Network.PeerSelection.Simple` module, with its value always returning an empty list of new peers, as shown in Listing 2.1.

Thus, the source code analysis confirms the fact that peer discovery is not used by Cardano Node.

Listing 2.1: `ouroboros-network/src/Ouroboros/Network/PeerSelection/Simple.hs` lines 74-80

```
let peerSelectionActions = PeerSelectionActions {  
    readPeerSelectionTargets = readTargets ,  
    readLocalRootPeers = toList <$> readTVar localRootsVar ,  
    requestPublicRootPeers = requestPublicRootPeers ,  
    requestPeerGossip = \_ -> pure [] ,  
    peerStateActions
```

⁵Cardano Node's source code for version 1.35.5 does not set a specific version requirement for the `ouroboros-network` package in its dependency list.

Analysis of Mithril as a bulk-synchronization method

3.1 Overview of Mithril’s design objectives

Mithril is a protocol for stake-based threshold multisignatures presented in [16]. One of its applications is fast bootstrapping of proof-of-stake blockchains, or bulk synchronization in terms of this report. An open-source implementation of Mithril for this use case is being actively developed in [17] and its pre-release for Cardano mainnet is expected in 2023.

Schematically Mithril’s multisignatures work as follows:

- A set of nodes whose stake can be independently verified register themselves with a Mithril aggregator service;
- Periodically, a publicly verifiable lottery is conducted, and a subset of nodes are selected as its winners;
- The winning nodes sign a cryptographic hash of the blockchain’s immutable data at an agreed-upon point in time and send the signature to the aggregator;
- The aggregator combines the individual signatures into a multisignature, verifies that there is a quorum among the individual signatures, and makes the multisignature public;
- Since each multisignature verifies a snapshot of Cardano’s immutable data, which contains the most recent stake distribution, it can be used as a trusted source of the stake distribution for the next lottery, thus, creating a chain of multisignatures that allows anyone validate authenticity of snapshots.

Overall, Mithril is a sound protocol for validating authenticity of snapshots of Cardano’s immutable data. However, data validation is only one component of bulk synchronization. The following sections discuss some specifics of the current implementation with regard to two other components: data distribution over the Internet and the subsequent data processing. The analysis was conducted using Mithril’s source code as of April 5, 2023 (commit 333fbe581e25f355f23c1cddda6e8c9eb5db5ebb).

3.2 Scalable data distribution not within the project’s scope

The primary objective of the Mithril team is to provide a reference implementation of the protocol for stake-based multisignatures. Even though the team has chosen fast bootstrapping of Cardano

Node as their target use case, the development of a scalable data-distribution infrastructure is currently considered to be out of scope.

Data distribution is currently implemented as a publication of snapshots using Google’s Cloud CDN service. A solution that leads to the following problems:

- The prices charged by Google CDN [6] for Internet-egress traffic are high. They range between US\$20 and US\$200 per TB, depending on the geographic destination of traffic. Therefore, the cost of blockchain distribution to a billion personal wallets becomes even less feasible.
- The use of a centralized service from a single commercial vendor is not entirely aligned with the principle of decentralization behind blockchain phenomenon.

The Mithril team acknowledges these issues but believes that the development of a scalable data-distribution infrastructure should be carried out by the community if there is sufficient demand for it.

3.3 Snapshot format not suitable for parallel and incremental processing

Currently, each Mithril snapshot is a **tar/gzip** archive that includes immutable data of all epochs and a corresponding snapshot of the ledger. This approach has certain disadvantages:

- Gzip has a suboptimal compression ratio. In particular, Zstandard library (further discussed in section 4.5) offers better compression speed and ratio for the whole range of compression-speed settings [18]. Moreover, Zlib library [19] used by Gzip has nothing to compete against high-compression settings provided by Zstandard.
- A tar/gzip archive must be fully decompressed by Gzip before meta-information about the archive’s content can be accessed. This makes it impossible neither to decompress nor process archived data in parallel.
- Each snapshot contains all blockchain data up to a specific time point. Thus, clients that need only a few most-recent epochs of data (a frequent case for bulk synchronization of personal wallets), still must download and decompress the full snapshot, increasing the data-distribution costs and data-delivery delays.

The Mithril team acknowledges these effects and has created a GitHub issue [20] to resolve them in the future.

An alternative high-performance bulk-synchronization method

4.1 Requirements for high-bandwidth bulk synchronization

Bulk synchronization consists of three parts: data transmission, data validation, and use-case-specific data processing. Moreover, data validation and use-case-specific data processing can be further grouped together into general data processing since they happen after data transmission and on a client's side. The following subsections discuss requirements for these two general components.

4.1.1 Sufficient outgoing network bandwidth

The first requirement for high-bandwidth bulk synchronization is that the network bandwidth of all data-consuming nodes can be fully utilized. That is needed to maximize the speed of data transmission. To meet this goal, the cumulative upload bandwidth of data-supplying nodes must be greater than or equal to the cumulative download bandwidth of data-consuming nodes,

$$\sum_{s \in S} O_s \geq \sum_{c \in C} I_c, \quad (4.1)$$

where

- S = the set of all data-supplying nodes;
- C = the set of all data-consuming nodes;
- O_i = the outgoing network bandwidth of a given node;
- I_i = the incoming network bandwidth of a given node.

In addition, the bandwidth relation (4.1) must hold for special cases, such as

- sudden demand spikes, such as after a release of a new client version;
- sudden supply reductions, such as during outages of major cloud providers;
- temporary overload of cross-continental links.

In practice, the above means that the cumulative outgoing bandwidth of data-supplying nodes should have additional reserves to sustain operations in such special conditions.

4.1.2 Sufficient data-processing performance

The second requirement is that data-consuming nodes can process data faster than they download them

$$\sum_{c \in C} P_c \geq \sum_{c \in C} I_c, \quad (4.2)$$

where

- C = the set of all data-consuming nodes;
- P_i = the data-processing throughput of a given node;
- I_i = the incoming network bandwidth of a given node.

In simpler words that means that network performance is the ultimate bottleneck. Normally, that is the case since networking is the slowest component of a modern computer when compared to the performance of a CPU, RAM, or SSD. However, unbalanced hardware configurations or poorly optimized software could break this requirement.

4.1.3 Practical considerations

Even though the requirements set by equations (4.1) and (4.2) are simple, they can be harder to fulfill in practice:

- Every increase in the download bandwidth of data-consuming nodes must be matched by the corresponding increase in the outgoing bandwidth of data-distributing nodes;
- Data-consuming nodes must effectively leverage the available bandwidth of multiple data-distributing nodes to not be constrained when an individual data-distributing node performs poorly. For example, by implementing features, such as robust peer discovery, multi-source download, and active connection management;
- Data-processing capabilities of data-consuming nodes must be monitored for unbalanced hardware configurations or poorly optimized software.

The following sections analyze several solutions fulfilling the requirement set by Equation (4.1):

- Section 4.2 looks into data sharing by downloading clients to compensate for the demand they create;
- Section 4.3 discusses optional rewarded access to high-bandwidth data distribution;
- Section 4.5 presents an approach to transport compression of blockchain data that both improves bulk-synchronization time and reduces the cost of network traffic for stake pool operators.

To fulfill the requirement set by Equation (4.2) the following solutions are offered:

- Section 4.6 presents an approach to parallel validation of blockchain data that helps scale the data-processing throughput of data consumers by leveraging all available CPU cores;
- For personal wallets, the data-processing approach presented in the paper on highly-parallel wallet-history-reconstruction method [13] is recommended. When supported by sufficiently

powerful hardware, it can process blockchain data at speeds of up to 4000 MB/sec. That is roughly equivalent to the incoming network bandwidth of 40 Gbps, which is orders of magnitude faster than most consumer Internet connections today.

4.2 Data sharing by downloading clients

In the analysis of data sharing by downloading clients, it is important to consider a number of factors:

- Consumer Internet connections can be asymmetric, meaning that their download bandwidth is better than their upload bandwidth. For example, DSL connections with a download speed of 250 Mbps and an upload speed of 50 Mbps are widespread;
- Personal wallets are normally used for very short periods, typically measured in single-digit numbers of minutes, and those are the only times when they can contribute to data sharing;
- Consumer computers can be located behind a firewall or NAT ⁶, creating hurdles for other clients to connect to them;
- The most recent segments of blockchain data are in greater demand by clients than the older ones. Thus, to maximize the contribution of downloading clients to data sharing, it is preferable that they adopt the "rarest blocks first" download policy. This policy ensures that the network prioritizes availability of the most recent blocks for distribution since in a blockchain the rarest blocks are the most recent ones. The following analysis assumes that this policy is active and consequentially that each client can fully utilize its outgoing network bandwidth.

The above factors can be combined into a formula that estimates the self-sufficiency of scalability of a network. A derivation of this formula is presented in the next subsection.

4.2.1 Sharing-self-sufficiency coefficient

The derivation starts with a relation of the volume of uploaded data by a network's client to the volume of downloaded data by the same client

$$s = \frac{v_u}{v_d}, \quad (4.3)$$

where

v_u = the volume of uploaded data during a session in MB;
 v_d = the volume of downloaded data during a session in MB.

When each software client of a network uploads more data than it downloads, such a network can scale indefinitely in terms of bandwidth.

Since a data volume is a product of a data-transfer bandwidth and a transfer time, the volumes in Equation (4.3) can be replaced by the respective products. In addition, the time of data upload

⁶Network address translation - a method of mapping of IP-address space [21]

can be expressed as the sum of the time a client waits for its synchronization to finish, t_{sync} , and the time a client keeps a personal-wallet application open after the synchronization, t_{use} ,

$$s = \frac{b_u \times t_{use} + b_u \times t_{sync}}{b_d \times t_{sync}}, \quad (4.4)$$

where

b_u = the upload bandwidth in MB/sec;
 b_d = the download bandwidth in MB/sec.

Equation (4.4) can be regrouped and t_{sync} replaced by a division of the data volume to be synchronized, v_{sync} , by the download bandwidth of a client, b_d ,

$$\begin{aligned} s &= \frac{b_u \times t_{use}}{b_d \times t_{sync}} + \frac{b_u \times t_{sync}}{b_d \times t_{sync}} \\ &= \frac{b_u}{b_d} \times \left(\frac{t_{use}}{t_{sync}} + 1 \right) \\ &= \frac{b_u}{b_d} \times \left(\frac{t_{use}}{v_{sync}/b_d} + 1 \right) \\ &= \frac{b_u}{b_d} \times \left(t_{use} \times \frac{b_d}{v_{sync}} + 1 \right). \end{aligned} \quad (4.5)$$

Finally, Equation (4.5) is adjusted for the probability of a client being unavailable for peer requests, c , by replacing b_u with $b_u \times c$

$$s = \frac{b_u \times c}{b_d} \times \left(t_{use} \times \frac{b_d}{v_{sync}} + 1 \right). \quad (4.6)$$

Equation (4.6) will be used to estimate the self-sufficiency of a network's scalability, or its sharing-self-sufficiency coefficient, for various scenarios. When s is greater than or equal to one, the cumulative outgoing bandwidth of the network can scale to any number of downloading clients. In addition, the value of $s + 1$ can be interpreted a boost factor for the network's upload bandwidth that the additional data distribution by downloading clients creates.

To understand the range of potential values of the sharing-self-sufficiency coefficient, two scenarios were selected: one representing a home user, and the other representing a startup operating from a coworking space. The following selection criteria explain this choice:

- The scenarios differ in their inputs to highlight the potential variability in outcomes;
- The scenarios present realistic cases to ensure that the results of the analysis are of practical use.

4.2.2 Scenario 1: weekly personal use over a residential DSL connection

The inputs for this scenario are as follows:

- The Internet connection is an asynchronous DSL one with a download speed of 250 Mbps and an upload speed of 50 Mbps;
- The network and protocol overhead is assumed to be 2 extra bits for every byte of blockchain data. That means that the respective net download speed is 25 MB/sec (megabytes per second) and the upload speed is 5 MB/sec;
- Users launch their wallet apps once per week (52 times per year) and spend five minutes using the apps after data synchronization is finished;
- Two blockchain sizes are analyzed: the current, 123 GB, and the one from the research question, 1000 GB.
- The blockchain size is expected to keep doubling further each year in all cases;
- The probability of a client being available for external requests is assumed to be 0.5;
- Users synchronize blockchain from scratch only once per year.

There is one caveat that needs to be taken care of before calculating a sharing-self-sufficiency coefficient for this case. Clients use the network differently during an initial synchronization from scratch and during weekly updates. To recognize this fact, the final coefficient is computed as a weighted average over all synchronizations of a user in a given year

$$s = s_{init} \times \frac{1}{f} + s_{update} \times \frac{f-1}{f}, \quad (4.7)$$

where

- s_{init} = the sharing-self-sufficiency coefficient during an initial full synchronization;
- s_{update} = the sharing-self-sufficiency coefficient during further partial synchronizations;
- f = the number of blockchain synchronizations per year.

In addition, the average data volume during weekly synchronizations is derived from the assumption of the blockchain doubling in size every year. This effectively means that new data produced in a given year corresponds to 50% or $\frac{1}{2}$ of the blockchain's size

$$v_{update} = \frac{v_{full}}{2} \times \frac{1}{f}, \quad (4.8)$$

where

- v_{full} = the full size of the blockchain in a scenario.

Figure 4.3 presents the computed values of sharing-self-sufficiency coefficients for this scenario. They were computed as follows:

- First, v_{update} values were computed for both blockchain sizes using Equation (4.8);
- Then, using Equation (4.6) s_{init} and s_{update} were calculated;
- Last, Equation (4.7) was used to obtain the values of sharing-self-sufficiency coefficients.

4.2.3 Scenario 2: daily small-business use over an enterprise network

The inputs for this scenario are as follows:

- The Internet connection is a synchronous one with a download and upload bandwidth of 1000 Mbps. The respective net upload and download speeds equal to 100 MB/sec with the same assumption of network overhead as in the previous scenario;
- Users launch their wallet apps every business day (260 times per year) and spend ten minutes using them after synchronization is completed;
- The probability of a client being available for external requests is assumed to be 0.75;
- Other parameters remain the same as in the previous scenario.

Calculations estimating sharing-self-sufficiency coefficients are the same as in the previous scenario. The results of the computations are presented in Figure 4.3.

4.2.4 Scenario analysis takeaways

The results in Figure 4.3 highlight the immense variability in the values: they range from 0.18 to 190.29. A reasonable conclusion from that is that the network is likely to fluctuate between self-sufficient and non-self-sufficient states. Thus, data-sharing by downloading clients is a necessary component of making bulk synchronization scalable, but another component may be necessary to support the network during periods when it is not in self-sufficient state. One such approach is presented in section 4.3. Furthermore, the values 0.18 and 24.06 corresponding to the blockchain size of 1 TB will be used in further calculations to estimate the respective boost factors ($s + 1$) from data-sharing by downloading clients: 1.18 and 25.06.

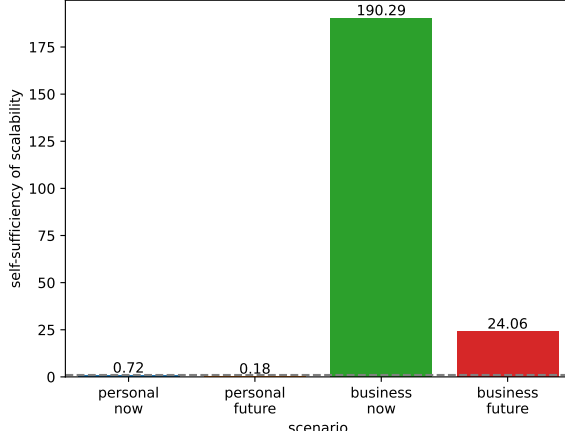


Figure 4.3: Sharing-self-sufficiency coefficients for the analyzed scenarios

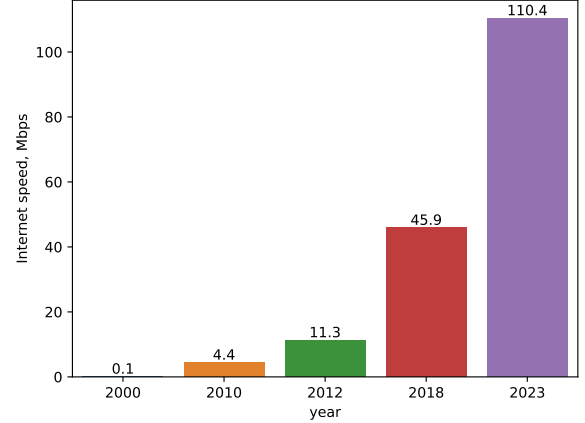


Figure 4.4: History of the average global Internet connection speed

4.2.5 Historical development of the speed of Internet connections

The scenario analysis highlights that growth in the blockchain size negatively influences self-sufficiency coefficients⁷. Surprisingly, the global average speed of Internet connections keeps growing at rates comparable to the rates of blockchain data growth. Figure 4.4 shows the historical

⁷The values of the scalability-self-sufficiency coefficients are lower at the future values of the blockchain size.

growth since 2000 ([22] [23] [24] [25]). The key takeaway is that the Cardano community has a natural exponential factor working in favor of the network’s self-sufficiency of scaling. All that is necessary to leverage it is to enable data sharing by downloading clients.

4.3 Optional rewarded access to high-bandwidth data distribution

4.3.1 Conflict of interest between stake delegators and stake-pool operators

At the moment, the main factor influencing the choice of a stake pool by a delegator is the expected return on investment, which depends primarily on the pool’s margin parameter. Moreover, the lower the pool’s margin, the better it is for the delegator. Thus, a natural conflict of interest exists between delegators and stake-pool operators. Furthermore, delegators have the upper hand because of the simplicity with which they can change stake pools and the availability of stake pools offering a 0% margin. Consequentially, stake-pool operators must keep their operating expenses under control, which influences them to choose cloud providers offering the lowest fixed cost for renting cloud instances rather than those offering the most bandwidth per dollar, thus, limiting the total available bandwidth of the Cardano network.

4.3.2 A potential solution

The key concept behind the solution is to find a way to change the dynamics outlined above. In addition, avoiding changes in Cardano’s fee structure is preferable. Fortunately, such a solution exists, and it is quite simple: provide additional value to delegators that can outweigh their preference for low margins. This additional value can be bulk-synchronization performance since it is one of the major quality-of-life attributes for owners of personal wallets. This approach could work as follows:

- Pools supporting optional high-bandwidth data delivery will set higher values of the margin parameter but provide their delegators with access to high-speed bulk synchronization;
- Such pools can reserve 90% of their outgoing network bandwidth for the exclusive use of their delegators;
- To identify themselves, delegators can send a special request containing their current IP address signed by their private stake key to one of the pool’s servers. After verifying the signature, the pool will mark requests from that IP address so that they are served from the reserve bandwidth capacity;
- All requests in the paid-bandwidth section are to be additionally prioritized relative to the weight of the stake among delegators currently issuing requests. This is required to prevent an attack when new delegators stake only one ADA to obtain full access to the paid-bandwidth capacity. In addition, this aligns the serving priority with the size of “subsidies” that a given delegator provides to the pool to purchase its bandwidth;
- All requests in the free section of the network bandwidth are to be prioritized using a fair-share policy. This allows the pool to continue contributing to the total capacity of the Cardano network while ensuring the sustainability of the financial model.

The above approach is simple, easy to implement, and requires no changes to Cardano’s network protocol or reward structure. Moreover, it allows such pools to provide support for additional bandwidth- and time-optimizing features, such as the transport data compression discussed in section 4.5 and hierarchical data discovery, discussed in section 4.4.

4.3.3 Scaling of stake-pool rewards as a function of margin

Before discussing the impact on network bandwidth, it is necessary to understand how changes in the margin parameter influence the rewards that can be allocated to a stake pool. To meet that aim, the following analysis was conducted:

- Two pools with 0% margin, delegated stakes above 68 million ADA, and the highest values of Cardano’s non-myopic-profitability factor were chosen ⁸ from the cExplorer.io pool list [26];
- The cumulative rewards over six consecutive epochs ⁹ were taken and their average was computed. The six-epoch period is roughly equivalent to one month, a typical billing cycle of cloud providers;
- Using the computed average, estimated monthly rewards were computed for a range of margin parameter values between 0% and 20%.

Figure 4.5 presents the estimated monthly rewards in ADA. The relative scaling factor of the rewards, which is the relationship of rewards to the rewards at 0% margin, is presented in Figure 4.6. The rewards scale well at a rate slightly below of a linear increase. Overall, stake pools can dramatically increase their operational budgets by simply changing the margin parameter, as long as their delegators are willing to support that decision.

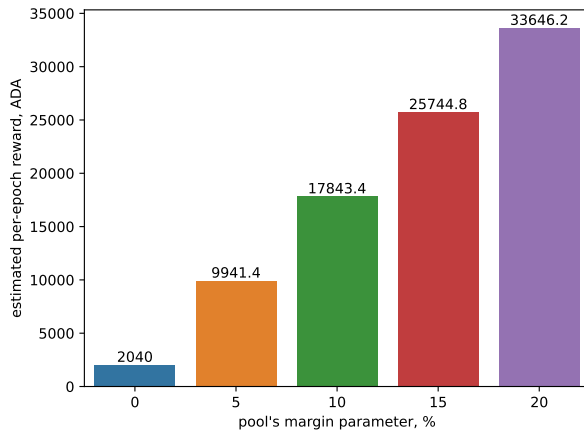


Figure 4.5: The estimated impact of a pool’s margin on its monthly reward

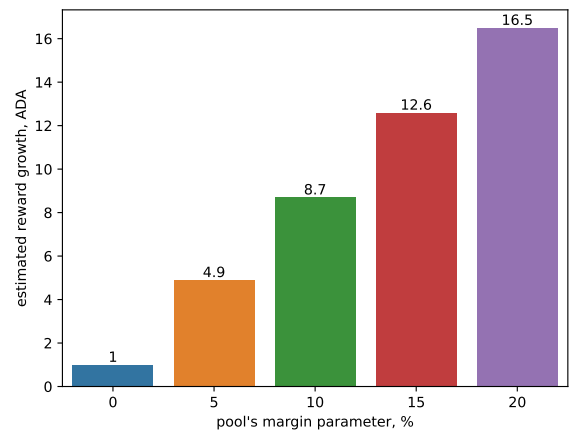


Figure 4.6: The increase in a pool’s reward as a function of its margin

⁸Pool ids pool1zz7x8ern02chznayj9fjds2gk5q6prmpdr49faspd65kzqsrsja and pool12t3zmafjqms7cuun86uwc8se4na07r3e5xswe86u37djr5f0lx.

⁹Epochs from 410 to 416.

4.3.4 The impact of higher rewards on network bandwidth

The solution presented above scales perfectly with the demand through its marketplace-like dynamics: the higher the demand for fast bulk synchronization, the higher the margins that high-bandwidth stake pools can command. Consequently, driven by the economic opportunity, additional stake pools will offer high-bandwidth services. This process will continue until an equilibrium is reached. Potentially, all Cardano stake pools could support high-bandwidth data distribution.

Since the discussion is about a potential impact, it is reasonable to examine an optimistic-but-conceivable scenario:

- The extra rewards are driven by the demand for fast bulk synchronization, so stake-pool operators are incentivized to allocate a larger portion of their rewards to bandwidth expenses. Thus, a situation in which 50% of the total rewards go to the bandwidth budget is conceivable;
- Even without the high-bandwidth feature, there are Cardano stake pools commanding 5%¹⁰ and even 8%¹¹ margins with a delegated stake above 60 million ADA. Thus, an increase of the margin to 20% for high-bandwidth pools is conceivable.

Using the scaling factors from Figure 4.6, a 20% margin will increase the rewards 16.5 times. In addition, due to the 2.5-times higher reward-to-bandwidth-budget rate (50% instead of the 20% discussed in section 2.4), the bandwidth budget can increase by a factor of more than 40. Since the increase in the budget directly increases the available network bandwidth, **rewarded access can increase the Cardano network’s bandwidth up to 40 times.**

4.4 Hierarchical data discovery

Section 2.6 presented an issue limiting the performance of bulk synchronization in Cardano: Its Chain-Sync mini-protocol prohibits parallel downloads of meta information despite its multi-gigabyte size. A simple solution to this issue is to provide a similar protocol that shares aggregated data that can be further expanded through additional but parallel requests. Below are brief descriptions of two examples of mini-protocols illustrating how that might look. Their key goal is to dramatically reduce the sequential part of bulk synchronization during which a client can effectively use only a single connection to a data-distributing node.

4.4.1 Epoch-Sync mini protocol

The Epoch-Sync mini-protocol is a simplified version of Chain-Sync; instead of sending block headers, it sends information about epochs, allowing the client to fetch complete data about each epoch later in parallel. The Epoch-Sync protocol can send the following data instead of sending the headers of next blocks in its **MsgRollForward** message:

- the slot number and block hash of an epoch’s last block;

¹⁰pool1qv3velgldz529zq904jy9qfrhdh0akwkkhyly2rymd6sw9m02cc.

¹¹b7908ea0ef61c441e4dca0e756f335c33459bb8f2779cb8f6caf8eb.

- the slot number and block hash of an epoch's first block;
- the size of an epoch's data.

After receiving the above information about available epochs, the client can download epochs in parallel using the existing Block-Fetch protocol.

4.4.2 Segment-Sample mini-protocol

Another way to introduce hierarchical data discovery is through a Segment-Sample mini-protocol that can work as follows:

- A data consumer sends a request to a data-distributing node. The request consists of start and end points (a slot number and block hash) and a count of subsegments whose information is to be shared;
- The data-distributing node responds with the requested number of equal-sized subsegments within the blockchain segment delimited by the points. Each subsegment is described with its start point, end point, and size;
- The data consumer can then either send further Segment-Sample requests to obtain more granular data or fetch data using Block-Fetch requests.

The Segment-Sample mini-protocol reduces the sequential phase of bulk synchronization into a single request.

4.5 Transport compression of blockchain data

High-bandwidth data distribution is challenging, and whether using transport compression is beneficial to performance can change case by case. Inequality (4.9) illustrates the problem and reminds us that the gains from quicker transmission times due to compression can be offset by the extra time needed to compress and decompress the data:

$$t_{transmission}^{compressed} + t_{compression} + t_{decompression} < t_{transmission}^{uncompressed}, \quad (4.9)$$

where

- $t_{transmission}$ = the transmission time of the data, either compressed or uncompressed;
- $t_{compression}$ = the time it takes for a data-distributing node to compress the data;
- $t_{decompression}$ = the time it takes for a data-consuming node to decompress the data.

Fortunately, the delivery of blockchain data is a special case since most of the data are immutable. Thus, inequality (4.9) does not need to consider compression time since it is amortized over all data transfers, leading to a simpler formula

$$t_{transmission}^{compressed} + t_{decompression} < t_{transmission}^{uncompressed}. \quad (4.10)$$

The following subsections compare two compression methods in their maximum-compression settings:

- LZ4 [27]: a compression algorithm designed specifically for high-bandwidth data transfers that features extremely fast compression and decompression at the expense of compression ratios;
- ZStandard [28] [29]: a modern compression algorithm offering excellent compression ratios and compression and decompression speeds faster than those of the extremely popular UNIX gzip compressor based on the Zlib compression library [19].

4.5.1 Experimental setup

All experiments were conducted with the following settings; the reported numbers are averages of five runs:

- Cardano’s blockchain data through the end of epoch 415, which ended on June 4, 2023, were taken, and each immutable-chunk file of the blockchain was compressed independently;
- An eight-core CPU-optimized virtual instance with 16 GB of RAM and 300 GB of NVMe SSD storage was rented at Vultr [4] and operated under Ubuntu Linux 22.04 LTS.

4.5.2 Compression ratio

Figure 4.7 compares the compression ratios of the two algorithms. This is a very important characteristic since it directly affects the transmission time in inequality (4.10). Zstandard is clearly the better option, boasting two times better compression ratio.

At the same time, it is important to note that the gain in the compression ratio comes at the expense of compression times, as presented in Figure 4.8. However, as previously noted, this factor is of minor importance for the distribution of immutable data since its cost is amortized over all data transfers from a given data-distributing node.

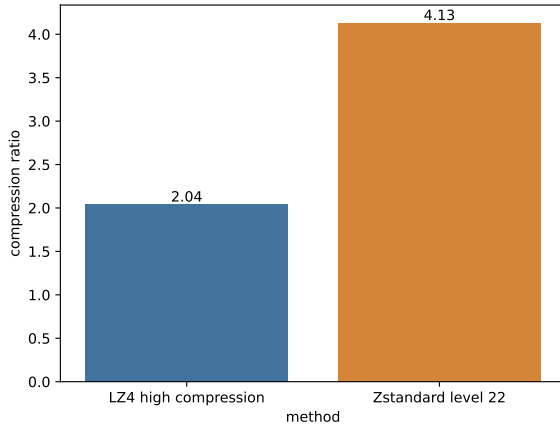


Figure 4.7: Comparison of compression ratios

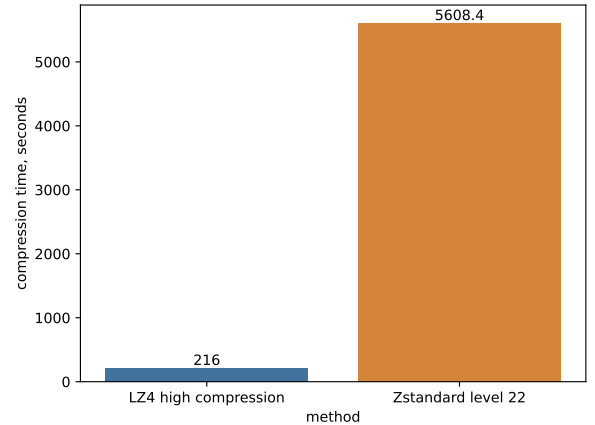


Figure 4.8: Comparison of compression times

4.5.3 Decompression time

Figure 4.9 compares the decompression times of the two algorithms, which are very similar, although Zstandard has a slight edge over LZ4 in decompression time. This may seem surprising since the LZ4 algorithm was specifically designed for fast compression and decompression. However, the explanation is simple. In addition to compute efficiency, the algorithms are constrained by the I/O bandwidth; more specifically by the sequential access speed of the local SSD. Although Zstandard consumes more CPU resources, it also consumes less SSD bandwidth due to its better compression ratio. Consequently, it achieves a slightly better decompression throughput. Figure 4.10 compares the decompression throughputs of the two algorithms. The dashed red line presents the single-threaded sequential read speed of the local SSD as measured by Linux’s `hdparm -t` command.

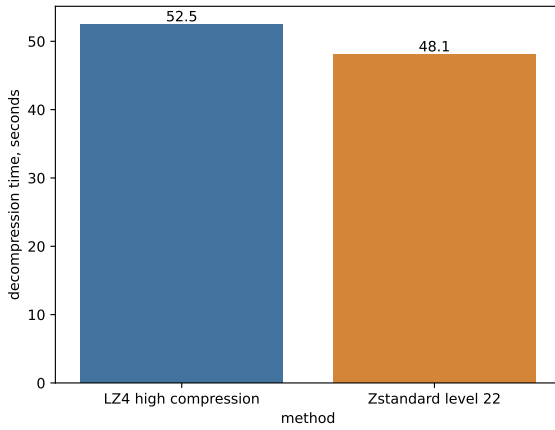


Figure 4.9: Comparison of decompression times

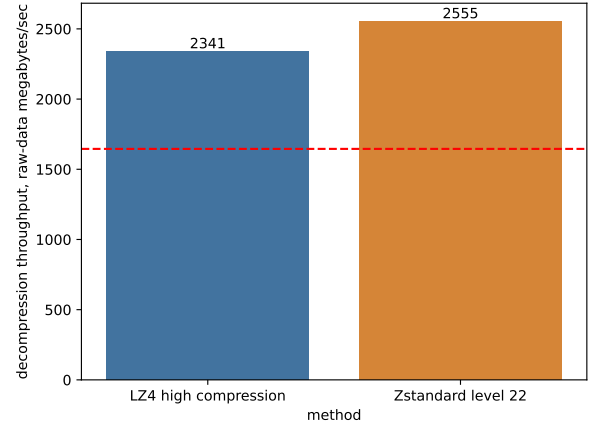


Figure 4.10: Comparison of decompression throughputs

4.5.4 End-to-end transfer time

Using the transfer and decompression times from the previous subsections, it is now possible to compare end-to-end transfer times according to inequality (4.10). This comparison is done in four situations to highlight the variability in outcomes:

- Consumer-grade 250 Mbps Internet connection and blockchain sizes of 123 GB and 1000 GB;
- Enterprise-grade 40 Gbps Internet connection and blockchain sizes of 123 GB and 1000 GB.

Figure 4.11 and Figure 4.13 compare the 250 Mbps case and Figure 4.12 and Figure 4.14 compare the 40 Gbps case. It is interesting to note that the transfer of uncompressed data over a 40 Gbps network is roughly two times faster than that of compressed data. The explanation is simple: the decompression throughput is limited by both CPU and SSD performance in this case.

However, that specific configuration is extremely rare: a server with high-end network but low-end

CPU and SSD performance. A typical configuration of such a server would normally have four times better CPU and SSD performance, reversing the inequality’s result to favor compressed transfers.

Therefore, leveraging Zstandard compression is beneficial in cases when network bandwidth is the bottleneck, which for all consumer and many enterprise configurations is ultimately the case. The expected improvement in end-to-end-transfer times for such cases is close to the compression ratio; for Cardano’s blockchain data, it is around a factor of four.

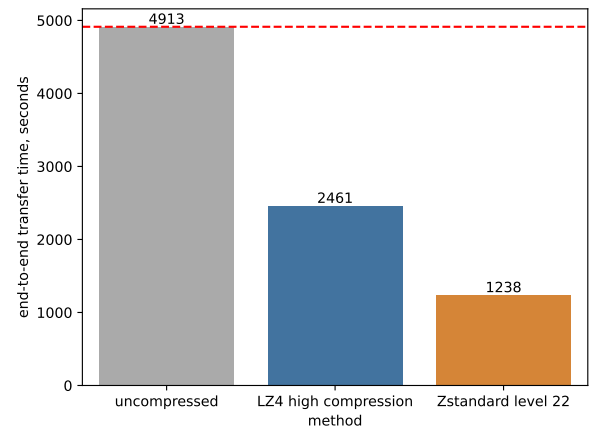


Figure 4.11: Total transfer time for 123 GB of blockchain data over a 250 Mbps connection

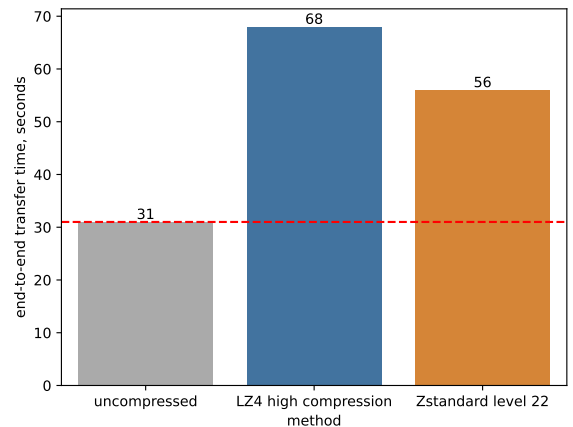


Figure 4.12: Total transfer time for 123 GB of blockchain data over a 40 Gbps connection

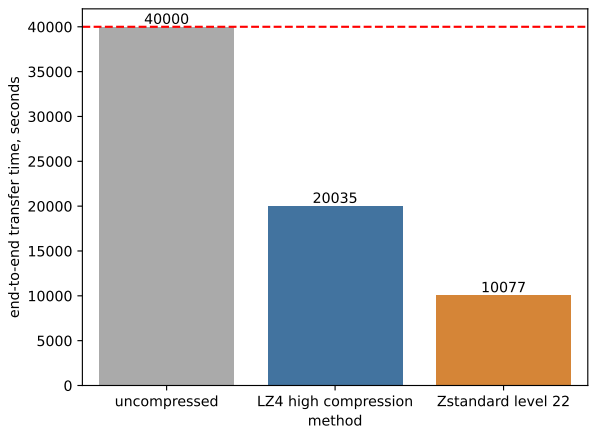


Figure 4.13: Total transfer time for 1000 GB of blockchain data over a 250 Mbps connection

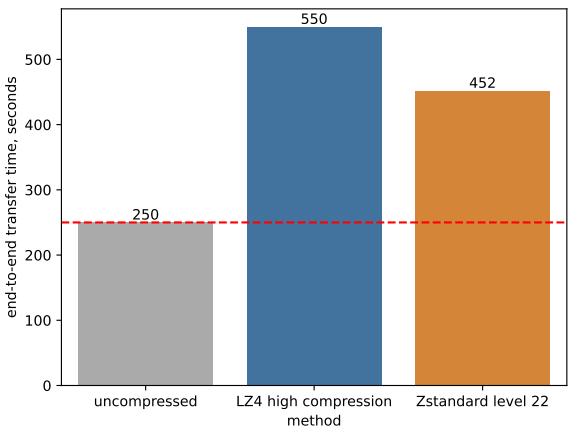


Figure 4.14: Total transfer time for 1000 GB of blockchain data over a 40 Gbps connection

4.6 Parallel validation of blockchain data

Cardano actively leverages header-based validation of blockchain data. For example, the data-diffusion process among block-producing nodes (explained in section 5.1.2 of [2]) relies on it. In header-based validation, a successful verification that an eligible slot leader has signed a given block is considered sufficient evidence of block validity. Unfortunately, the sequential nature of the current implementation of block-header validation in Cardano makes it difficult to fulfill the requirement set by Equation (4.2). At the same time, if it is possible to make such validation faster than the download speeds of consumer-grade Internet connections, that will offer an additional method ¹² for high-performance validation of blockchain snapshots.

This section investigates the feasibility of leveraging parallel processing for bulk header-based validation. In addition, methods that allow for out-of-order execution are preferred since that unlocks additional performance improvements because computations can be started during the data download and be largely completed by the time all the blockchain data have been transferred.

The specific checks ¹³ necessary for header-based validation are as follows:

- check that blocks' previous-hash fields match the respective header hashes;
- check that blocks' slot numbers are monotonically increasing;
- check that the first blockchain's block refers to the well-known genesis-block hash in its previous hash field;
- check that blocks' block-body-hash fields match the actual hashes of block bodies;
- check the validity of block-header signatures;
- check the slot-leadership-eligibility of block-signing nodes.

4.6.1 Parallel verification of header-hash and slot-number invariants

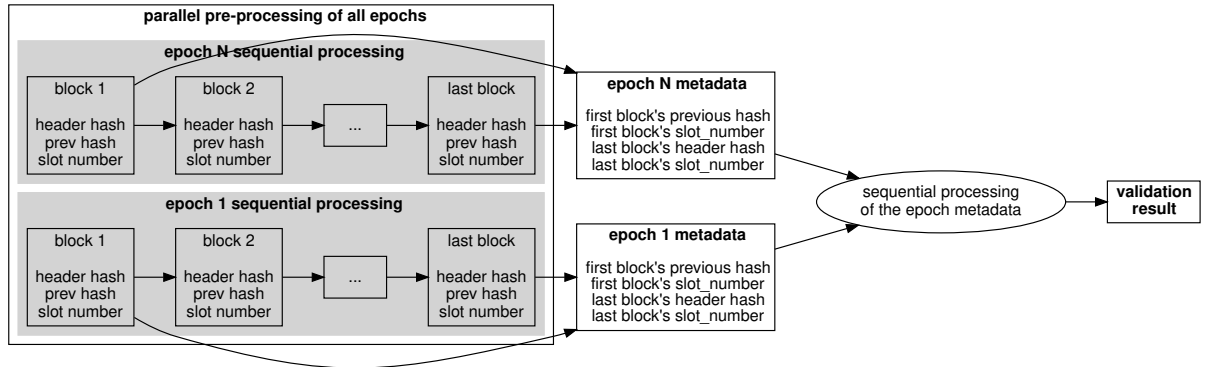


Figure 4.15: Overview of the parallel verification of header-hash and slot-number chains

Figure 4.15 depicts the parallel verification of header-hash and slot-number invariants, which operates as follows:

¹²In addition to Mithril, described in section 3.1.

¹³This list assumes the modern (Shelley-era or more recent) format of Cardano block headers.

- Epochs are pre-processed in parallel, while the blocks of each epoch are processed sequentially by a single process;
- After verifying the invariants within the blocks of a given epoch, the process saves the previous hash and the slot number of the epoch's first and last blocks for future reference;
- Once the pre-processing of all epochs is finished, a single process verifies that the saved values of the epochs' boundary blocks also satisfy the invariants;
- If any check at any stage fails, processing is immediately terminated.

4.6.2 Parallel verification of block-body hashes

The verification of block-body hashes does not require any context and can be integrated with the initial block-parsing activities. Thus, it can be carried out fully in parallel.

4.6.3 Parallel verification of block-header signatures

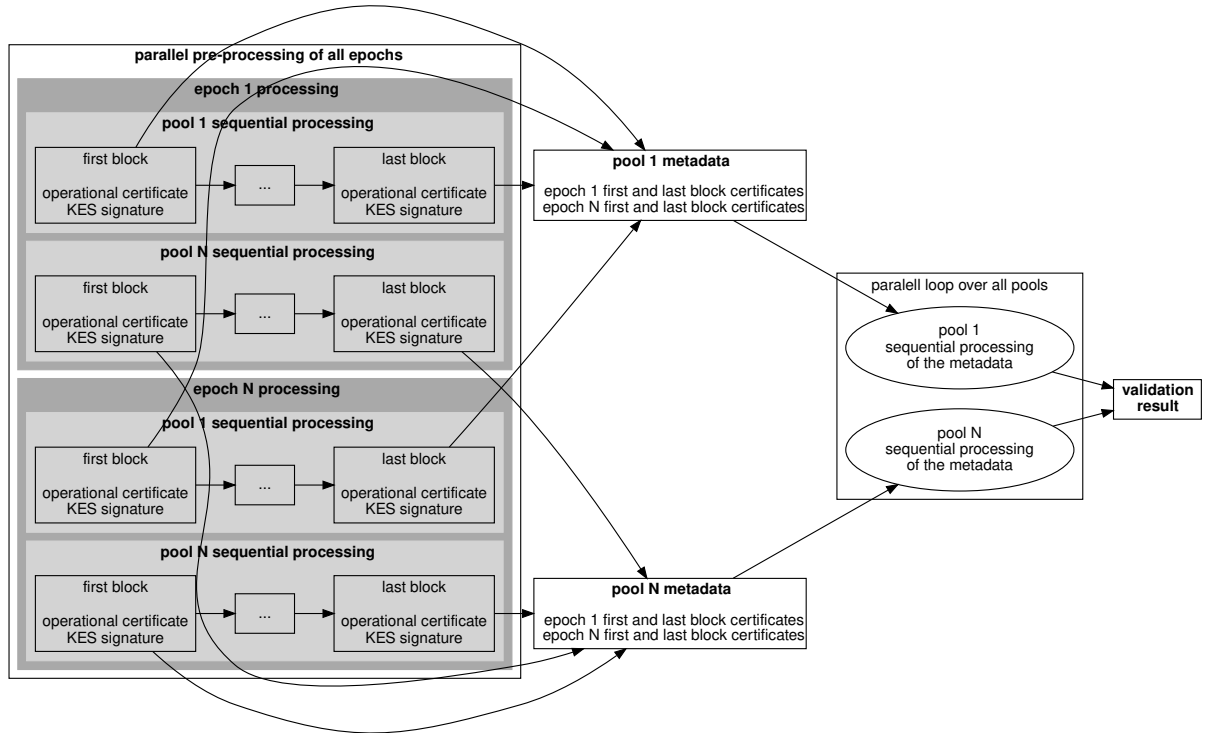


Figure 4.16: Overview of the parallel verification of block signatures

Figure 4.16 presents an overview of the parallel verification process of block-header signatures, which does the following:

- Epochs are pre-processed in parallel, while the blocks of each epoch are processed sequentially by a single process;
- The process groups the epoch's block headers by the id of the block-signing pool;

- For each pool, the process verifies the validity of the signature and the fulfillment of the signature invariants, then saves certificates from the first and last blocks for future reference;
- When all epochs are processed, a parallel loop over all pools is started to check the validity of the certificate invariants on the epochs' boundaries from the saved information;
- If any check at any stage fails, processing is immediately terminated.

4.6.4 Parallel verification of slot-leadership eligibility

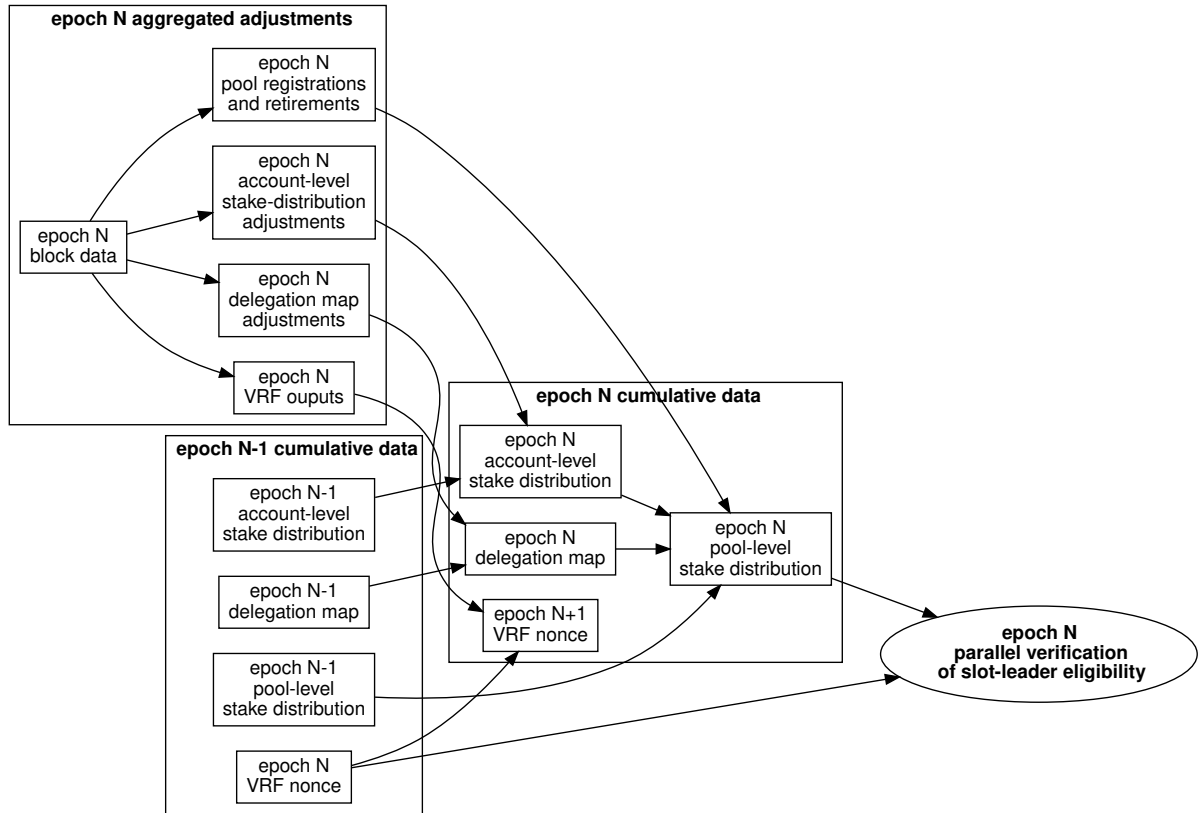


Figure 4.17: Overview of the second step of the parallel verification of slot-leader eligibility

The eligibility of slot leaders to sign their blocks is verified in three steps. First, per-epoch per-account stake adjustments are computed by following the parallel computation of per-epoch stake distributions, as presented in section 4.2.3 of [13]. In addition, three extra per-epoch indices are created: pool registrations, pool retirements, and outputs of block-issuers' verifiable random functions.

Second, the following per-epoch computations, visualized in Figure 4.17, are performed:

- The aggregated per-epoch data from the first stage are processed sequentially by a single process;
- To compute the cumulative data of the next epoch, the process takes the cumulative data of the previous epoch (all start empty at epoch 0) and applies the aggregated adjustments

computed in the first stage;

- The outputs of each per-epoch calculation are
 - cumulative stake delegation map;
 - cumulative per-account stake distribution;
 - cumulative per-pool stake distribution;
 - the next epoch’s nonce value for the slot-leader-eligibility verification computed from the outputs of verifiable random functions located in the headers of the epoch’s blocks.

Third, the verification of slot-leader eligibility starts as a parallel loop over all epochs. The process for each epoch iterates over the epoch’s blocks and for each block prepares three inputs: per-pool stake distribution, the epoch’s nonce value, and the slot number of a given block. With those inputs ready, it checks the following:

- the output of the verifiable random function (VRF) of the block-signing pool has been constructed from the respective epoch’s nonce and the block’s slot number;
- the VRF’s output value when seen as a 2^{512} natural number meets a dynamic threshold that depends on the pool’s relative stake taken from the epoch’s per-pool stake distribution;
- If any check at any stage fails, processing is immediately terminated.

4.6.5 Security and future plans

The proposed method is a parallel reimplementation of Ouroboros Praos [3] and thus retains its general corruption-resistance properties without change. At the same time, in exchange for performance, this method may require more steps before detecting an attack. In the worst case, an attack will be detected after processing the whole blockchain. If this method were intended for use in real time on block-producing nodes, that might have been a problem. However, the intended use is with personal wallets during bulk synchronization. For this use case, these properties are fully appropriate since they do not create any new risks for the network’s block-production process.

Finally, it is important to highlight that it is possible to further extend this method to other tasks, such as validating transaction graphs referenced by transactions of a given user. However, both the additional functionality and a detailed performance evaluation require more serious attention; thus, a dedicated report on the topic of parallel validation is planned in the future.

Discussion

5.1 Handling a billion clients

The analysis presented in section 2.4.1 shows that the current level of stake-pool rewards is sufficient to pay for a million transfers of a one-terabyte-sized blockchain. This section analyzes how factors discussed in this report can cumulatively influence the network's capacity to distribute blockchain copies.

Factor	Impact	Outcome
Transport compression (section 4.5)	4x cost reduction	1 million transfers become 4
Paying for bandwidth (section 2.4.2)	4x cost reduction	4 million transfers become 16
Data sharing by all clients (section 4.2)	increase in bandwidth between 1.18 and 25.06 times depending on the network's self-sufficiency coefficient: $s + 1$	16 million transfers become 19 to 400 million
Rewarded access to high-bandwidth data distribution (section 4.3)	increase in bandwidth up to 40 times	19 to 400 million transfers become 760 million to 16 billion
Growth in ADA price due to growth in the number of Cardano users (section 2.4.3)	prices increase stake-pool rewards; the purchasable bandwidth increases from 2.8 to 40.9 times	760 million to 16 billion transfers become 2.1 billion to 654 billion

Table 5.2: Impact of various factors on the number of supported blockchain transfers

Table 5.2 summarizes the impact of individual factors and estimates their combined influence. Even without the ADA price factor, the proposed solutions can reach levels close to the billion transfers set in the research question. Moreover, when the ADA price factor is taken into account, the number of supported transfers can go far beyond a billion.

5.2 A measurement framework for bulk-synchronization performance

Quantitative measurement is an important part of the continuous-improvement process necessary for most long-term projects. It enables precisely comparing the performance of alternative approaches, selecting the more promising ones, and preventing performance regressions. A robust performance-measurement framework should provide measurements that are easy to interpret. In addition, it must be comprehensive and take into account all major aspects of the problem area. The following set of metrics provides the right balance of simplicity and comprehensiveness and offers a solid basis for measuring future improvements and competing approaches:

- **End-to-end synchronization time.** Such time should include the times of network transfer, data decompression, and data validation;
- **Cost of blockchain data delivery.** This is measured by tracking the outgoing network traffic on reference stake-pool nodes, multiplying the result by a constant of US\$10 per TB of traffic, and dividing the result by the uncompressed size of the blockchain data delivered to clients that initiated transfers from those stake-pool nodes;
- **Data-propagation delay** calculated as the average difference between the time when a new block is added to the blockchain is available for download by clients.

5.3 Suggested implementation plan

Integrating new features into the official Cardano code base is a slow process that requires extensive testing and review. Thus, for a new feature that demands active experimentation and careful refinement, it is more expedient to begin its development as a standalone project and go through the integration process once its design becomes stable. An additional factor in favor of decision to start standalone and merge later is that many in the Cardano community (including some of the technical staff maintaining the official repositories) still believe that fast bulk synchronization is a myth. Thus, going the standalone path allows for the more rapid production of sufficient evidence of feasibility and practicality and helps win the broad community support, both of which facilitate a quicker integration into the official code.

High-level components of the alternative method for bulk synchronization have been presented in the previous chapter. The following list provides some additional details to make the implementation plan more specific:

- **Leverage BitTorrent V2.** To prove the effectiveness of robust peer discovery, data sharing by downloading clients, parallel download from multiple sources and active connection management features in satisfying the requirement for sufficient upload bandwidth, it is best to leverage an existing protocol with a high-quality implementation. The BitTorrent version 2 protocol has a battle-tested implementation of these features. Moreover, it has robust and well-tested implementations, such as the libtorrent library [30]; the development team can shorten the development time for proof of concept by leveraging this features;
- **Support multiple data validation options.** To ensure sufficient data-processing performance, it is best to support more than one data-validation option: the parallel validation presented in section 4.6 and Mithril discussed in section 3.1. This will hedge the

risks of one of them being only available with a delay or offering insufficient performance, which is a scenario that must be considered, given the early stage of both projects;

- **Leverage Zstandard compression.** To make it easier for data-distributing nodes to satisfy the requirement for sufficient upload bandwidth, the use of Zstandard compression can be integrated for distribution of blockchain data;
- **Launch an experimental Cardano stake pool.** To further support the requirement for sufficient upload bandwidth and test the optional rewarded access and hierarchical data-discovery features, a custom Cardano stake pool can be created serving as a real-world playground for the development and testing of fast bulk-synchronization software;
- **Develop an API-compatible replacement for Cardano Wallet.** To integrate fast bulk synchronization into Daedalus wallet and further satisfy the requirement for sufficient data-processing performance, it would best to move forward with the implementation vision presented in section 4.3 of paper on highly-parallel wallet-history-reconstruction method [13].

5.4 Recommended adjustments to the Cardano network protocol

The benefits of fast bulk synchronization can be made available to all Cardano clients if the fundamental features of the alternative method for bulk synchronization are implemented as extensions to the Cardano network protocol. Moreover, it is possible to implement only some of those features and still provide noticeable quality-of-life improvements to Cardano users. The list below is an overview of potential changes:

- Integrate support for transfer data compression, as described in section 4.5;
- Introduce a variant of hierarchical discovery of blockchain data, as described in section 4.4;
- Develop a more robust peer-discovery functionality reacting to the critique presented in section 2.7;
- Integrate support for high-performance bulk-validation methods, such as the Mithril or parallel validation, as presented in section 4.6;
- Increase the minimal stake-pool fees to the level that allows all stake pools to purchase networking capacity using the more efficient pay-for-bandwidth plans instead of pay-for-traffic ones, as presented in section 2.4.2;
- Introduce support for data sharing by synchronizing nodes to increase the scaling self-sufficiency of the network, as described in section 4.2.

Conclusion

This report presents evidence that Cardano's current network protocol and its structure of stake-pool rewards limit the performance and scalability of bulk synchronization. More specifically, with its current configuration, Cardano cannot support a billion full-data clients and the blockchain size of one terabyte. Furthermore, the potential number of simultaneously supported clients capable of the download speed of 40 Gbps can be as low as a hundred.

In addition, this report sets the requirements necessary for Cardano to successfully scale bulk synchronization, suggests a list of major improvements, and presents an implementation plan that can help Cardano to support a billion full-data clients.

Next, this report discusses alternative options for validation of blockchain data, such as Mithril and parallel validation. Leveraging them should make the speed of consumer Internet connections the bottleneck in most practical cases of bulk synchronization. That should not only increase the overall performance of bulk synchronization by an order of magnitude but also expose users to future improvements from continuing growth in the speed of Internet access.

Finally, this report presents a measurement framework for the evaluation of improvements and alternative approaches to bulk synchronization.

References

- [1] Duncan Coutts et al. “The Shelley Networking Protocol.” In: (2019). URL: <https://input-output-hk.github.io/ouroboros-network/pdfs/network-spec/network-spec.pdf>.
- [2] Duncan Coutts et al. “Introduction to the design of the Data Diffusion and Networking for Cardano Shelley.” In: (2020). URL: <https://input-output-hk.github.io/ouroboros-network/pdfs/network-design/network-design.pdf>.
- [3] Bernardo Machado David et al. “Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake protocol.” In: *IACR Cryptol. ePrint Arch.* (2017), p. 573. URL: <http://eprint.iacr.org/2017/573>.
- [4] *Vultr - Optimized Cloud Compute*. URL: <https://www.vultr.com/pricing/>.
- [5] *Amazon EC2 On-Demand Pricing*. <https://aws.amazon.com/ec2/pricing/on-demand/>.
- [6] *Google Cloud - All networking pricing*. <https://cloud.google.com/vpc/network-pricing>.
- [7] *Microsoft Azure - Bandwidth pricing*. <https://azure.microsoft.com/en-us/pricing/details/bandwidth/?cdn=disable>.
- [8] *Amazon - On Demand EC2 Instance Pricing*. <https://aws.amazon.com/ec2/pricing/on-demand/>.
- [9] *Cexplorer.io - Cardano pool operator profitability*. <https://cexplorer.io/profitability>.
- [10] *Interserver - Bandwidth options*. <https://www.interserver.net/dedicated/10gbps.html>.
- [11] *Interserver - Bandwidth options*. <https://www.datapacket.com/pricing>.
- [12] *CoinMarketCap - Historical prices of Bitcoin*. URL: <https://coinmarketcap.com/currencies/bitcoin/>.
- [13] Alex Sierkov. “Highly Parallel Reconstruction of Wallet History in Cardano Blockchain.” In: (2023). URL: https://github.com/sierkov/daedalus-turbo/blob/main/doc/2023_Sierkov_WalletHistoryReconstruction.pdf.
- [14] *Experiment configuration - actual peer discovery of Cardano Node*. <https://github.com/sierkov/daedalus-turbo/tree/main/experiment/cardano-peer-discovery>.
- [15] *Ouroboros Network official source code as referenced by Cardano Node release 1.35.5*. <https://github.com/input-output-hk/ouroboros-network/tree/926d785efb6686ee46356cf6304c688a>.
- [16] Pyrros Chaidos and Aggelos Kiayias. “Mithril: Stake-based Threshold Multisignatures.” In: *IACR Cryptol. ePrint Arch.* (2021), p. 916. URL: <https://eprint.iacr.org/2021/916>.

- [17] *Mithril - Github*. <https://github.com/input-output-hk/mithril>.
- [18] *Zstandard - Compression Speed vs Ratio Chart*. URL: <https://github.com/facebook/zstd/blob/dev/doc/images/CSpeed2.png>.
- [19] *Zlib - A massively spiffy yet delicately unobtrusive compression library*. <https://github.com/madler/zlib>.
- [20] *Mithril - improve archive format Github issue*. <https://github.com/input-output-hk/mithril/issues/876>.
- [21] *Wikipedia - Network address translation*. URL: https://en.wikipedia.org/wiki/Network_address_translation.
- [22] *Cisco: Global Internet Traffic to Quadruple by 2014*. 2010. URL: <https://www.bbcmag.com/breaking-news/cisco-global-internet-traffic-to-quadruple-by-2014>.
- [23] *Cisco's Visual Networking Index Forecast Projects Nearly Half the World's Population Will Be Connected to the Internet by 2017*. 2013. URL: <https://newsroom.cisco.com/c/r/newsroom/en/us/a/y2013/m05/cisco-s-visual-networking-index-forecast-projects-nearly-half-the-world-s-population-will-be-connected-to-the-internet-by-2017.html>.
- [24] *Cisco Annual Internet Report (2018–2023) White Paper*. 2020. URL: <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html>.
- [25] *Average fixed broadband speeds worldwide from 2018 to 2023*. 2023. URL: <https://www.statista.com/statistics/1190324/average-fixed-speeds-download-and-upload-in-worldwide/>.
- [26] *cExplorer.io - Cardano stake-pool list*. URL: <https://cexplorer.io/pool>.
- [27] *Lz4 - Extremely Fast Compression algorithm*. <https://github.com/lz4/lz4>.
- [28] Yann Collet. “Zstandard Compression and the ‘application/zstd’ Media Type.” In: (2021). URL: <https://datatracker.ietf.org/doc/html/rfc8878>.
- [29] *Zstandard - Fast real-time compression algorithm*. <https://github.com/facebook/zstd>.
- [30] *Libtorrent - an efficient feature complete C++ bittorrent implementation*. <https://github.com/arvidn/libtorrent>.