# RIVAChain: Blockchain-based Integrity Verification for File Transfers

Ahmed Alhussen and Engin Arslan
Computer Science and Engineering, University of Nevada, Reno
Email: aalhussen@nevada.unr.edu, earslan@unr.edu

***Abstract*—File transfer integrity verification is used to detect silent data corruption by calculating and comparing the checksum of files using secure hash functions, such as SHA-256. However, it incurs significant performance overhead due to I/O and compute-intensive checksum calculation process. In this paper, we present blockchain-based ledger architecture to store the checksum of frequently accessed scientific datasets to minimize the overhead of integrity verification. In the proposed architecture, the checksum of files is calculated and pushed to a private blockchain when they are first created such that future transfers will not require data source to recalculate checksum. As scientific datasets are typically generated in one location (e.g., observatory) and streamed to many geographically distributed locations to enable collaboration, eliminating checksum calculation for data sources will save a significant amount of resource consumption. Moreover, we find that blockchain-based integrity verification reduces transfer time by up to $50\%$ when data source is the bottleneck in the integrity verification process. Finally, we show that private blockchains can scale to thousands of transactions per second thus they are better fit for scientific applications in which data generation rate can easily outpace the transaction confirmation rates of public blockchains.**

## I. Introduction

We are witnessing a data deluge in science as increasing number of projects started to produce large volumes of data. As an example, Large Hadron Collider experiment ALICE studies heavy-ion collisions at a centre of mass energy and generates three terabytes of data per second [1]. Similarly, Vera Rubin Observatory (also known as Large Sky Survey Telescope) will soon take high-quality pictures of the southern universe using a $3,200$ megapixel camera and is expected to produce 15TB data every night [2]. This massive volumes of data often moved to geographically distributed locations for collaboration, processing, and archival purposes. Although high-speed networks with up to 100Gbps bandwidth are established to overcome increasing transfer rates, the reliability of underlying technology in detecting and recovering from data corruption has not improved significantly over the years. For example, studies show that TCP checksum misses bit errors once in 16 million to 10 billion packets [3], causing missed errors once in every 30 minutes to 300 hours in 1 Gbps networks, and once in every 18 seconds to 3 hours in 100 Gbps networks. Thus, end-to-end integrity verification is proposed to improve the robustness of file transfers against silent data corruption.

End-to-end integrity verification works as follows: Transfer sender first reads a file and sends it to the transfer receiver. Upon the completion of the transfer, the sender reads the file again to compute its checksum using a secure hash algorithm, such as SHA-256. The receiver also computes the checksum of the file and sends it to the sender to compare. If the checksum values of the sender and the receiver match, then the transfer is considered successful. Otherwise, copy of file at the receiver side is deemed corrupt, so the file is transferred again.

While end-to-end integrity verification is crucial for many applications to capture silent errors, it imposes significant overhead due to requiring I/O and CPU-intensive checksum computation [4]. To alleviate its overhead, Liu et al. [5] proposed *block-level pipelining* to divide large files into smaller blocks and overlap transfer and checksum operations for different blocks. In a previous work, we proposed FIVER to execute transfer and checksum operations simultaneously for same file blocks to minimize the cost of integrity verification. By pipelining checksum and transfer operations for same file partition, FIVER allows I/O sharing between the checksum and transfer operations to avoid reading files twice.

In another work, we introduced a Robust Integrity Verification algorithm (RIVA) to enhance the robustness of integrity verification [6], [7]. While existing integrity verification algorithms calculate the checksum of files immediately after their transfer, RIVA evicts file pages from cache memory before reading them back from disk for checksum calculation to be able to detect and recover from undetected disk write errors. Despite its robustness in detecting silent errors, RIVA requires both source and destination sides to calculate file checksum each time a file is transferred, causing nonnegligible increase in transfer time. While checksum calculation by the transfer receiver is essential to capture errors, source-side checksum computation for every transfer causes unnecessary repetitive work when the same file is transferred multiple times. Since many science projects involve transmitting gathered data to multiple geographically distributed locations for collaboration purposes, current approach for integrity verification incurs significant compute and I/O overhead at data source that can be avoided by calculating file checksums once and saving them in a persistent repository.

In this paper, we tackle this problem through blockchain-based ledger system, RIVAChain, to store the integrity information of scientific datasets such that the correctness of file transfers can be validated without requiring data source to calculate file checksum for every transfer. In the proposed architecture, the source of data (e.g., observatory) calculates

the checksum of files when they are first created and pushes them to cloud-hosted private blockchain such that transfer receivers can refer to blockchain to detect silent data corruptions. Although the idea of storing the checksum of critical data in blockchain is not new, *this paper, to the best of our knowledge, makes the first attempt to integrate it to file transfer integrity verification process.* In summary, we make following contributions in this paper:

- We design and develop blockchain-based integrity verification architecture, RIVAChain, to minimize the overhead of integrity verification for data source.
- We compare the performance of RIVAChain against the state-of-the-art integrity verification solutions in terms of transfer time in multiple networks using various dataset types.
- We evaluate the scalability of RIVAChain in terms of publishing and querying integrity information using three different approaches with cloud instance types.

The rest of the paper is organized as follows: Section II describes related work and Section III presents the design of RIVAChain as well as the steps it takes for integrity verification. Section IV presents experimental results and Section V concludes the paper with a summary.

## II. Related Work

Integrity verification is used widely in many areas including storage outsourcing [8]–[10], long term archives [11], [12], file systems [13]–[15], databases [16], provenance [17] and data transfer [4]–[7] to detect and avoid silent data corruption. Globus transfer service employs end-to-end integrity verification for file transfers by computing and comparing the checksum of files after their transfers [18]. In the presence of multiple files in transfer dataset, Globus starts transferring next file in the queue only after previous one's transfer and integrity verification is completed successfully. This sequential approach incurs significant performance penalty due to pausing the transfer while checking the integrity [5]. Liu et al. [5] proposed block-level pipelining to optimize integrity verification by pipelining transfer and checksum operations for different files. It reduces execution time considerably especially when the dataset is composed of files with mixed sizes.

In previous work, we presented FIVER [4] and RIVA [6] to reduce the overhead and increase the robustness of integrity verification. FIVER reads files once and runs the transfer and checksum computation processes simultaneously, eliminating the need to read files twice. RIVA, on the other hand, aims to detect undetected disk write errors that might happen while flushing file data from memory to disk [6], [7]. RIVA does this by enforcing cache eviction immediately after the transfer such that checksum computation has to read files directly from disk. While RIVA offers stronger integrity verification coverage, it can lead to longer execution times in networks where I/O throughput is slower than transfer speed.

Blockchain technology has gained attention in recent years due to offering an immutable ledger architecture in a decentralized environment. Bitcoin [19] is the first implementation of blockchain technology for financial transactions. Replacing the central authority with a distributed ledger does not only offer identity protection for users but also improves the reliability of the system against cyber attacks. Therefore, it is adapted to many areas including but not limited to election voting [20]–[22], healthcare [23], [24], data management [25], and supply chain [26]. For example, TrialChain implements blockchain for data governance in biomedical research where the integrity information of files is stored in blockchain so that the authenticity of research results can be validated later.

Barinov et al. presented blockchain-based integrity verification framework for storage systems by saving files hashes and transaction logs such that any attempts to manipulate data can be detected and avoided [27]. Fisher et al. proposed digital data verification and authentication using blockchain [28]. The method hashes the digital content to a unique value then submits it to blockchain. As a result, blockchain is widely used to store sensitive information such that malicious attempts to alter them can be avoided. We, therefore, adopt this idea to implement blockchain-based integrity verification for scientific data transfers where accidental data corruption happens frequently.

## III. System Design

Blockchain has become popular in recent years due to its decentralized and tamper-resistant architecture. In blockchain, users submit transactions to the transaction pool, which are then combined in blocks and added to the ledger. Ledger in blockchain links blocks to each other to ensure that no single transaction in the ledger can be altered without requiring to recreate all blocks. To further improve resilience against data manipulation attacks, some implementations (e.g., Bitcoin) require participants (aka miners) to run computationally intensive calculations (mining) to become eligible for block generation. The mining process prevents malicious users to exploit the system for their benefit (e.g., double spending attack) since no single entity can dominate the mining process unless it can outpower the rest of the miners in terms of computation, also known as 51% attack.

Proof of Work (PoW) is a popular mining process that requires users to find a random number (i.e., nonce) to satisfy hash requirements when appended to blocks. Although PoW improves the robustness of the system against malicious users, it has adverse impact on the system performance. For instance, it takes an average of 10 minutes to mine a block in Bitcoin, resulting in an average of 2-3 transactions per second throughput. Thus, it is a major impediment in the adoption of blockchain technology especially for delay sensitive workload. On the other hand, private blockchain implementations offer high throughput by eliminating PoW in mining process. Since this would pose a threat to the immutability of blockchain when malicious users are present, permission is required for users to participate in mining process. Therefore, we utilize a private blockchain solution to implement the proposed framework. While this may cause security concerns due to potential

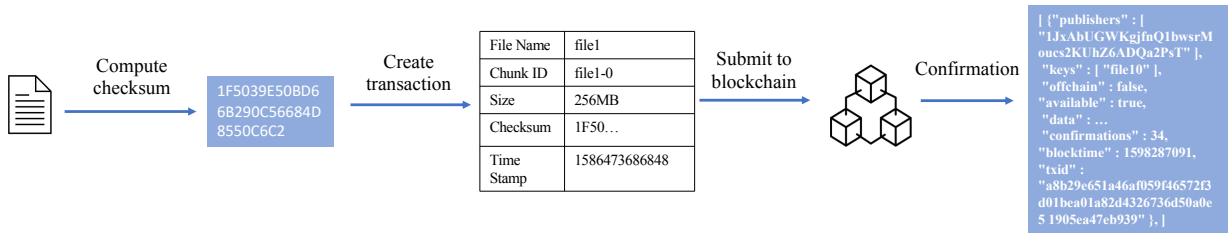



Fig. 1: Illustration of transaction preparation and verification process in RIVAChain.

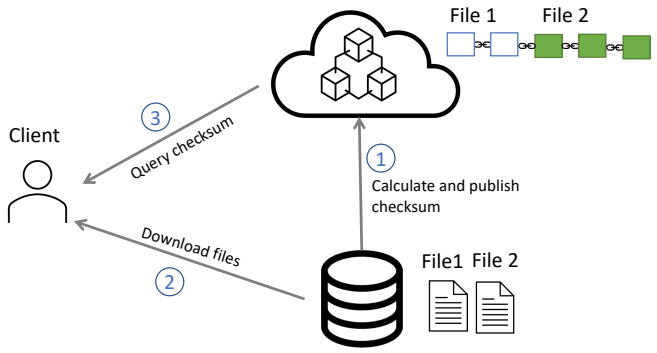



Fig. 2: RIVAChain obviates the need for checksum computation for data source as clients can verify the integrity of transfer using blockchain.

breaches, the impact of such attacks can be minimized by synchronizing the contents of the private blockchain with public blockchains periodically. We leave the implementation of a private-public blockchain integrated solution as future work and use MultiChain [29] in this paper to store the integrity information of scientific datasets.

MultiChain is an open-source private blockchain that is derived from Bitcoin Core software. It allows participants to have different roles as publisher and subscriber. The creator of the first block (i.e., genesis block) owns the root privileges and can add others as publisher or subscriber. Separation of publisher and subscriber roles offers enhanced security in transaction submission and block generation process while enabling read access to many clients. When a node joins to MultiChain, a unique identifier is assigned to it to keep track of the publishers of transactions and blocks.

RIVAChain works as follows: When a file is first generated, its checksum is calculated and published in MultiChain that is deployed in the cloud. Then, whenever a client wants to download a file, s/he can transfer the file from data source and query the file's checksum from blockchain. Upon the completion of the transfer operations, the client then calculates the checksum of the downloaded file and compares it against the one obtained from RIVAChain to make sure that the file is not corrupted during the transfer. If checksum values match, then the transfer is considered successful. Otherwise, the file is downloaded from the data source again. Figure 2 illustrates these steps for two files whose checksum values are computed and published in the cloud-hosted private blockchain (step 1) such that the client downloads data from the source (step 2) and validate the integrity of transfers by comparing their checksum against the ones saved in the blockchain (step 3).

When the checksum of a file does not match with the one in the blockchain due to data corruption, the client needs to download the file again. This is true even if the corruption alters only a single bit of a large file. This file-level integrity verification incurs significant recovery overhead, especially for large file transfers [4]. As an example, failure of integrity verification would require entire 100GB file to be downloaded again even if only single bit is corrupted. Thus, we adopt chunk-level integrity verification to confirm the integrity of large files [4], [6]. In the chunk-level integrity verification, the checksum is calculated for a fixed-size of file segments. For instance, a chunk size of 128MB will create 8 file segments (i.e., chunks) for a 1GB file. The checksum of each segment is not dependent on other segments such that the integrity of each segment transfer can be verified independently. Thus, when data corruption takes place, we can localize the error by finding the chunk whose integrity has failed so that only that portion of the file is retransferred.

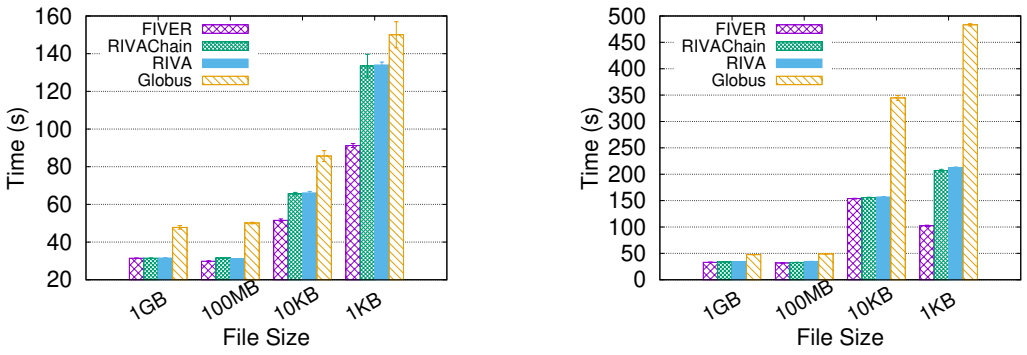
Once the checksum of a file chunk is calculated, it is then submitted to blockchain for confirmation as shown in Figure 1. In a transaction, we include file name, size, chunk ID, checksum, and timestamp information. Size refers to the size of the chunk, which would be equal to file size when the chunk size is equal or larger than file size. Checksum field stores the hash of a chunk calculated using secure hash algorithms, such as SHA-256. We finally append the timestamp of the checksum calculation for clients to be able to confirm the time of the checksum computation. Once the transaction is formed, it is submitted to the blockchain by the publisher nodes. When a transaction is included in a block and confirmed by the chain, publisher id, confirmation id, block time, and transaction id are assigned for future references.

## IV. Evaluations

We compared the performance of RIVAChain against the traditional integrity verification methods in terms of execution time. FIVER overlaps the transfer of a file with the checksum of the same file to share I/O between the two. Although FIVER keeps the overhead of integrity verification low, it falls short to prevent the detection of disk write errors [6], [7]. RIVA enforces checksum computation to read files from disk after their transfer to capture disk write errors. It incurs

| Testbed | Storage | CPU | Memory Size | Bandwidth | RTT | Disk Write Speed |
|---|---|---|---|---|---|---|
| HPCLab | NVMe SSD | 16 x Intel Xeon E5-2623 @2.60GHz | 64 GB | 40G | 0.2 ms | $<$ 3GB/s |
| Chameleon-WAN | SATA HDD | 12 x Intel Xeon E5-2650 @2.30GHz | 64 GB | 1G | 32 ms | $<$ 100MB/s |
| Chameleon-LAN | SATA SSD | 12 x Intel Xeon E5-2670 @2.30GHz | 128 GB | 10G | 0.2 ms | $<$ 100MB/s |
| Pronghorn | GPFS | 16 x Intel Xeon E5-2683 @2.10GHz | 192 GB | 10G | 0.1 ms | $<$ 3GB/s |

TABLE I: Specifications of test environments.




(a) HPCLab (b) Pronghorn

Fig. 3: Comparison of integrity verification algorithms in HPCLab and Pronghorn networks.

slightly higher overhead compared to FIVER in exchange of increased robustness against silent disk errors. Globus is a widely-used data transfer service that, by default, runs integrity verification for all file transfers. Globus runs transfer and integrity verification operations sequentially, thus transfer of a file starts only after the integrity of previous file is successfully completed. This in turn causes Globus to perform worse when there are lots of small files in the dataset due to causing a delay between transfers. Similar to RIVAChain, FIVER and RIVA use chunk-level integrity verification to avoid transferring large files as a whole when checksum mismatch is detected.

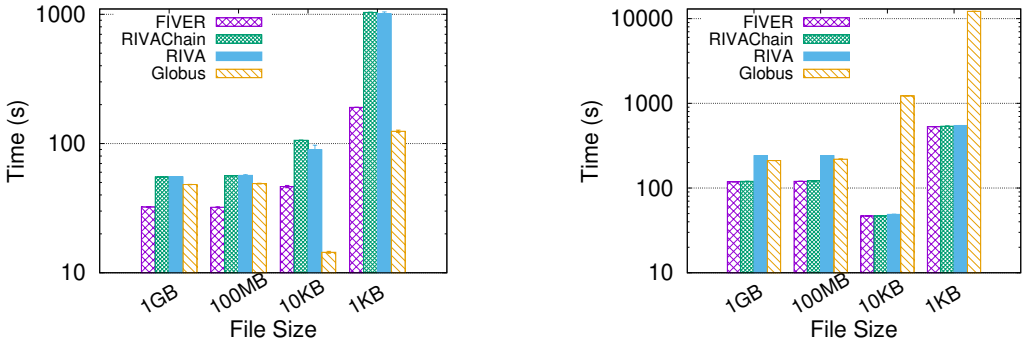
We conducted experiments in HPCLab, Chameleon, and Pronghorn networks whose hardware and network specifications are listed in Table I. Chameleon is an academic cloud service that offers instances in two sites, Chicago, IL and Austin, Texas. We name the experiments that run transfers between the instances of the same site as Chameleon-LAN and different sites as Chameleon-WAN. HPCLab is comprised of two data transfer nodes in the same local area network with 40 Gbps connectivity. Finally, Pronghorn is a campus cluster and its transfer nodes are connected with 10G links.

We run the experiments using four datasets that contain same-size files. The dataset consists of (i) 10x1GB files, 100x100MB files, $10,000$x10KB files, and $100,000$x1KB files. We repeated the experiments at least five times and report average and standard deviation values in the figures.

### A. Transfer Time

Figure 3 shows the results of experiments in HPCLab and Pronghorn networks. The speed of network transfers is faster than checksum calculation speed in these networks due to slow hash calculation speed ($<$ 3Gbps). Since FIVER runs checksum calculation of files using cache data, its performance is better than other solutions. Please note that FIVER does not offer the same level of protection for file transfers as provided by RIVA and RIVAChain. The performance difference widens for small files as the cost of cache eviction process by RIVAChain and RIVA is exacerbated when the number of files in the dataset is large. For instance, it takes 92 seconds for FIVER to complete the transfer of $100,000$ 1KB files with integrity verification whereas it lasts more than 120 seconds for the others in HPCLab network. Similar performance gap can be observed in Pronghorn network as shown in Figure 3(b). On the other hand, it takes the longest duration for Globus to finish the transfers, which can be attributed to the fact that it does not pipeline transfer and checksum calculations processes for different files. As an example, Globus spends around 450 seconds to complete 1KB dataset while the other algorithms can finish it in less than 220 seconds. It is clear that RIVAChain performs similar to RIVA in terms of transfer duration. Even though data source does not calculate checksum of files after the transfer, the receiver still needs to compute it to compare it against the version in the blockchain. Consequently, despite saving I/O and computation overhead for data source, RIVAChain does not shorten the transfer time in HPCLab and Pronghorn networks.

Figure 4 shows the results of experiments in Chameleon network. Since the network bandwidth of Chameleon-LAN is faster ($>$ 10 Gbps as shown in Table I) than the speed of checksum computation (around 2.4 Gbps using a single CPU core), checksum calculation becomes the bottleneck. Globus outperforms FIVER, RIVA, and RIVAChain since




(a) Chameleon-LAN (b) Chameleon-WAN

Fig. 4: Comparison of algorithms in Chameleon-LAN and Chameleon-WAN networks.

cache eviction takes a long time when disk speed is slow. The performance gap between Globus and the others reach to 10x for 1KB files. Again, this reduction in the transfer times comes at the expense of reduced coverage in integrity verification. Specifically, while RIVA and RIVAChain offer full end-to-end coverage for data corruptions, FIVER only detects errors that happen in the network. On the other hand, Globus error coverage depends on file size. Since it runs the integrity verification right after the transfer without evicting files from cache memory, its checksum calculation will utilize cache data when file size is small enough to fit in the memory. Since all the files we used in the experiments can fit into the memory of the servers in the Chameleon network, its execution time is shorter. It is important to note that the benefit of cache-based checksum calculation is reflected on the transfer time only when disk read/write speed is slower than checksum computation. As a result, Globus falls short to take advantage of this in HPCLab and Pronghorn networks where storage system is powered by RAID arrays on flash drives.

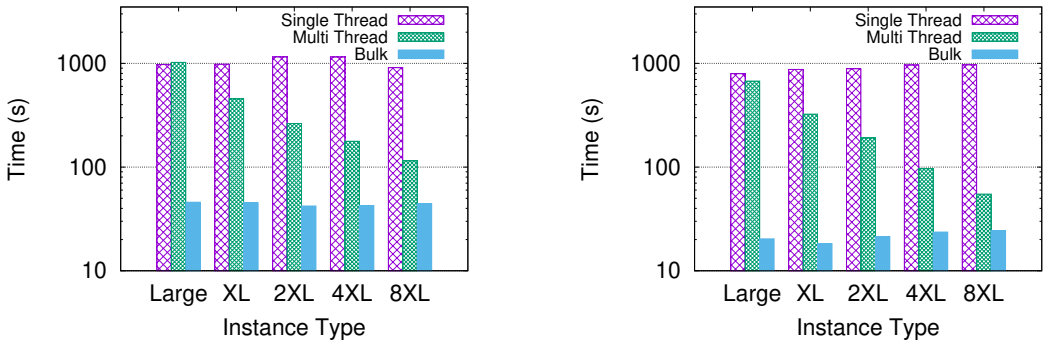
For large file transfers in Chameleon-WAN, RIVAChain outperforms RIVA as presented in Figure 4(b). This is mainly due to using nodes with slow I/O performance for the sender in the Chameleon-WAN experiments. While both sender and receiver nodes have SSD drives in the Chameleon-LAN network, the sender has hard drive and receiver has SSD drive in Chameleon-WAN experiments. As a result, sender-side checksum calculation takes longer than receiver-side checksum calculation thus becomes the bottleneck for RIVA transfers. On the other hand, RIVAChain does not require transfer senders to calculate the file checksum at the time of the transfer, hence runs at the checksum speed of receiver and yields higher throughput. Finally, performance of Globus degrades in Chameleon-WAN network due to slow transfer performance. Specifically, despite using cache data to calculate the checksum of files, it runs transfers and integrity verification sequentially hence causes short pauses between the transfer of files. This pause causes TCP congestion window to reset [30], necessitating TCP to go through slow start phase for each file. While this takes negligible amount of time in local area networks, it can dominate total transfer time in wide-area networks especially when file size is small. Therefore, despite saving time in checksum calculation process, Globus falls behind due to poor transfer performance. For instance, Globus finishes the transfer of 10KB dataset in more than $1,000$ seconds while other algorithms keep it under 50 seconds. Similarly, it takes more than $12,000$ seconds for Globus to transfer 1KB dataset as opposed to less than 550 seconds by the other solutions.

| Instance type | vCPUs | Memory (GB) |
|---|---|---|
| Large | 2 | 16 |
| XLarge | 4 | 32 |
| 2XLarge | 8 | 64 |
| 4XLarge | 16 | 128 |
| 8XLarge | 32 | 256 |

TABLE II: Specifications of AWS Cloud instance types.

### B. Scalability Analysis

As RIVAChain performs similar or better compared to RIVA in terms of transfer time, we next evaluate its ability to handle a large number of requests. To do so, we first measured its performance in accepting and publishing $100,000$ file checksum data. Table II lists the instance types and their main hardware specifications that we evaluated in this experiment. When users submit checksum data for new files, RIVAChain first places them into a queue. Then, the items in the queue are fetched to be submitted to MultiChain. We implemented three strategies to process queries in the queue as follows: The first and simplest approach uses a *single thread* to process requests one by one. The second approach uses *multiple threads* to pull requests from the queue and process them. The final and third approach uses a single thread for *bulk* processing in which multiple checksum requests are bundled in a one transaction. For instance, if there are 100 publish requests in the queue, *bulk* query can bundle them in a single transaction to reduce the number of total transactions.




(a) Publish (b) Query

Fig. 5: Scalability analysis of RIVAChain to publish (a) and query (b) 100,000 file checksum data using different methods. MultiChain's ability to combine many keys (i.e., file checksum) in single transaction (i.e., Bulk Method) offers low-latency for high-frequency operations.

Figure 5(a) shows the time it takes to process $100,000$ publish requests for file checksum data using the three approaches. It is clear that, it takes more than $1000s$ for *single thread* approach to process all the requests for all instance types. While *multi-threaded* approach improves the performance by lowering the time to around $100s$ when using 8XL instance, it still falls behind *bulk* method. Note that we decided the number of threads in multi-threaded approach by dynamically increasing the count until observed performance stops increasing. Finally, we find that *bulk* publish approach offers scalable performance regardless of the instance type. Specifically, it can complete processing all $100,000$ publish requests in less than $50$ seconds, bringing its speed to $1,500$ operations/sec.

We also evaluated the impact of using *single thread*, *multi-thread*, and *bulk* method to receive a response for $100,000$ queries in RIVAChain. We find that response time for a single query is about $0.1$ms, thus single thread approach returns the slowest execution time. It takes $795$ seconds to $968$ seconds for single thread approach to evaluate $100,000$ requests. Multi-thread approach yields significantly better results by processing multiple requests simultaneously. However, improvement ratio is not same for all instance types due to the difference in hardware specifications. Specifically, the execution time reduces from $968.7$ seconds to $54.9$ seconds (over $90\%$ decrease) when it is applied in 8XLarge instance type whereas execution time falls by less than $15\%$ for Large instance type. This can be attributed to the difference in vCPUs in instance types as multithreading cannot improve the performance much when the threads have to share the same core.

Although multithreading improves the performance of running queries in MultiChain, threads execute a separate query for each checksum request, causing its performance to lag behind the bulk querying approach as can be seen in Figure 5(b). In the bulk query method, we specify $100$ requests in a single query thus reduce the total number of queries to execute. The duration of querying $100,000$ transactions with bulk query method ranges between $18$ to $25$ seconds. Compared to single thread and multi-thread approaches, bulk querying improves the performance by $50-99\%$.

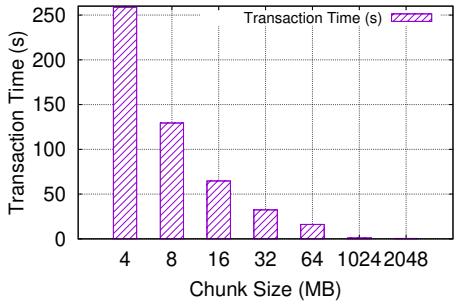



Fig. 6: Impact of chunk size on the transaction confirmation duration. Larger chunk sizes lead to smaller transactions times due to reducing the number of transactions for large files.

We analyzed the impact of chunk size on the performance of RIVAChain in Figure 6. Chunk size defines the number of checksum values will be created for large files. For example, chunk size of $128$MB will create $8$ checksum values for a $1$GB file. Lowering chunk size has an advantage for recovery time in the case of data corruption due to requiring to resend smaller amount of data. On the other hand, keeping the chunk size too low has disadvantage of creating too many transactions to be published/saved in the blockchain. Figure 6 presents the duration to publish checksum information of a $100$GB file using a single thread approach when chunk size is set to values between $4$MB and $2048$MB. As small chunk size values create more chunks for same amount of data, it takes longer to confirm the corresponding transactions. Specifically, while it takes $259$ seconds to publish the checksum information of all chunks when chunk size is set to $4$MB, the same takes around $1$ second when chunk size is set to $2048$MB.

## V. Conclusion

In this paper, we propose RIVAChain to utilize blockchain technology to store checksum information of scientific dataset such that cost of integrity verification process for file transfers can be minimized. We show that RIVAChain reduces significant amount of overhead on frequently accessed data repositories by delegating integrity verification process of file transfers to blockchain. Moreover, transfer time can be reduced by up to $50\%$ with RIVAChain when transfer sender is slower than transfer receiver in checksum calculation due to slow I/O and CPU throughput. Finally, we evaluated the scalability of RIVAChain in responding to checksum queries and find that it can process $100,000$ requests in around 20 seconds, achieving $5,000$/s query processing throughput. As a future work, we intend to adapt RIVAChain to high-speed data transfer applications [31], [32] to facilitate its adoption by science community.

## References


[1] "Alice experiment," https://home.cern/science/experiments/alice, 2020.

[2] "Large Synoptic Survey Telescope," https://www.lsst.org/, 2020.

[3] J. Stone and C. Partridge, "When the CRC and TCP checksum disagree," in *ACM SIGCOMM computer communication review*, vol. 30, no. 4. ACM, 2000, pp. 309–319.

[4] E. Arslan and A. Alhussen, "A low-overhead integrity verification for big data transfers," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 4227–4236.

[5] S. Liu, E.-S. Jung, R. Kettimuthu, X.-H. Sun, and M. Papka, "Towards optimizing large-scale data transfers with end-to-end integrity verification," in *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE, 2016, pp. 3002–3007.

[6] B. Charyyev, A. Alhussen, H. Sapkota, E. Pouyoul, M. H. Gunes, and E. Arslan, "Towards securing data transfers against silent data corruption," in *IEEE/ACM International Symposium in Cluster, Cloud, and Grid Computing, IEEE/ACM*, 2019.

[7] B. Charyyev and E. Arslan, "RIVA: Robust integrity verification algorithm for high-speed file transfers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1387–1399, 2020.

[8] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM conference on Computer and communications security*. Acm, 2007, pp. 598–609.

[9] Y. Zhu, H. Hu, G.-J. Ahn, and M. Yu, "Cooperative provable data possession for integrity verification in multicloud storage," *IEEE transactions on parallel and distributed systems*, vol. 23, no. 12, pp. 2231–2244, 2012.

[10] C. Liu, C. Yang, X. Zhang, and J. Chen, "External integrity verification for outsourced big data in cloud and IoT: A big picture," *Future generation computer systems*, vol. 49, pp. 58–67, 2015.

[11] P. Maniatis, M. Roussopoulos, T. J. Giuli, D. S. Rosenthal, and M. Baker, "The LOCKSS peer-to-peer digital preservation system," *ACM Transactions on Computer Systems (TOCS)*, vol. 23, no. 1, pp. 2–50, 2005.

[12] M. Vigil, J. Buchmann, D. Cabarcas, C. Weinert, and A. Wiesmaier, "Integrity, authenticity, non-repudiation, and proof of existence for long-term archiving: a survey," *Computers & Security*, vol. 50, pp. 16–32, 2015.

[13] A. Ma, C. Dragga, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, and M. K. Mckusick, "Ffsck: The fast file-system checker," *ACM Transactions on Storage (TOS)*, vol. 10, no. 1, p. 2, 2014.

[14] Y. Zhang, D. S. Myers, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Zettabyte reliability with flexible end-to-end data integrity," in *Mass Storage Systems and Technologies (MSST), 2013 IEEE 29th Symposium on*. IEEE, 2013, pp. 1–14.

[15] Y. Zhang, A. Rajimwale, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "End-to-end data integrity for file systems: A ZFS case study." in *FAST*, 2010, pp. 29–42.

[16] M. U. Arshad, A. Kundu, E. Bertino, A. Ghafoor, and C. Kundu, "Efficient and scalable integrity verification of data and query results for graph databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 5, pp. 866–879, 2018.

[17] R. Hasan, R. Sion, and M. Winslett, "The Case of the Fake Picasso: Preventing History Forgery with Secure Provenance." in *FAST*, vol. 9, 2009, pp. 1–14.

[18] "Globus," https://www.globus.org/.

[19] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.

[20] N. Kshetri and J. Voas, "Blockchain-enabled e-voting," *IEEE Software*, vol. 35, no. 4, pp. 95–99, 2018.

[21] A. B. Ayed, "A conceptual secure blockchain-based electronic voting system," *International Journal of Network Security & Its Applications*, vol. 9, no. 3, pp. 01–09, 2017.

[22] F. P. Hjalmarsson, G. K. Hreioarsson, M. Hamdaqa, and G. Hjalmtysson, "Blockchain-based e-voting system," *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pp. 983–986, 2018.

[23] M. Mettler, "Blockchain technology in healthcare: The revolution starts here," in *2016 IEEE 18th international conference on e-health networking, applications and services (Healthcom)*. IEEE, 2016, pp. 1–3.

[24] X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang, "Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control," *Journal of medical systems*, vol. 40, no. 10, p. 218, 2016.

[25] G. Zyskind, O. Nathan *et al.*, "Decentralizing privacy: Using blockchain to protect personal data," in *2015 IEEE Security and Privacy Workshops*. IEEE, 2015, pp. 180–184.

[26] F. Tian, "An agri-food supply chain traceability system for china based on rfid & blockchain technology," in *2016 13th international conference on service systems and service management (ICSSSM)*. IEEE, 2016, pp. 1–6.

[27] I. Barinov, V. Lysenko, S. Beloussov, M. Shmulevich, and S. Protasov, "System and method for verifying data integrity using a blockchain network," Oct. 30 2018, uS Patent 10,114,980.

[28] J. Fisher and M. H. Sanchez, "Authentication and verification of digital data utilizing blockchain technology," Sep. 29 2016, uS Patent App. 15/083,238.

[29] G. Greenspan, "Multichain private blockchain-white paper," *URl: http://www. multichain. com/download/MultiChain-White-Paper. pdf*, 2015.

[30] "TCP Congestion Control," https://tools.ietf.org/html/rfc2581.

[31] E. Arslan and T. Kosar, "High Speed Transfer Optimization Based on Historical Analysis and Real-Time Tuning," *IEEE Transactions on Parallel and Distributed Systems*, 2018.

[32] E. Arslan, B. A. Pehlivan, and T. Kosar, "Big data transfer optimization through adaptive parameter tuning," *Journal of Parallel and Distributed Computing*, vol. 120, pp. 89–100, 2018.