SciChain: Blockchain-enabled Lightweight and Efficient Data Provenance for Reproducible Scientific Computing

Abdullah Al-Mamun*, Feng Yan†, and Dongfang Zhao‡
University of Nevada, Reno
*aalmamun@nevada.unr.edu, {†fyan,‡dzhao}@unr.edu

2021 IEEE 37th International Conference on Data Engineering (ICDE) | 978-1-7281-9184-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICDE51399.2021.00166

***Abstract*—The state-of-the-art for auditing and reproducing scientific applications on high-performance computing (HPC) systems is through a data provenance subsystem. While recent advances in data provenance lie in reducing the performance overhead and improving the user's query flexibility, the fidelity of data provenance is often overlooked: there is no such way to ensure that the provenance data itself has not been fabricated or falsified. This paper advocates leveraging blockchains to deliver immutable and autonomous data provenance services such that scientific discoveries are trustworthy. The challenges for adopting blockchains to HPC include designing a new blockchain architecture compatible with the HPC platforms and, more importantly, a set of new consensus protocols for scientific applications atop blockchains. To this end, we have designed the proof-of-scalable-traceability (POST) protocol and implemented it in a blockchain prototype, namely SciChain, the very first practical blockchain system for provenance services on HPC. We evaluated SciChain by comparing it with multiple state-of-the-art systems; experimental results showed that SciChain guaranteed trustworthy data provenance while incurring orders of magnitude lower overhead than existing solutions.**

***Index Terms*—Blockchain, provenance, fault tolerance, HPC.**

I. Introduction

A. Motivation

A fundamental means to reproduce computational scientific results is through data provenance, which tracks the entire lifespan of the data during the experiments and simulation at various phases such as data creation, data changes, and data archival. Data provenance plays a critical role in guaranteeing the validity of scientific discoveries and research results, as data fabrication and falsification could happen to meet research objectives or personal interests, or both. For instance, the National Cancer Institute found 0.25% of trial data are fraudulent in the year of 2015 [1], [2]. In earth sciences, scientists emphasized the importance of maintaining data provenance in achieving the transparency of scientific discoveries [3].

Conventional provenance systems can be categorized into two types: centralized provenance systems and distributed provenance systems. One popular centralized provenance system is SPADE [4], where the provenance (from various data sources) is collected and managed by a centralized relational database management systems (RDBMS). Domain-specific systems based on such centralized design paradigms are also available in biomedical engineering [5], computational chemistry [6], among others. Although having been reasonably adopted by various disciplines, the centralized provenance systems are being increasingly criticized due to the exponentially-grown data: the centralized provenance system becomes a performance bottleneck and a single point of failure, and to this end, we witness the inception of various distributed approaches toward *scalable provenance* [7], [8]. Indeed, those distributed provenance systems, mostly built upon distributed file systems as opposed to centralized databases, eliminated the performance bottleneck and proved to deliver orders of magnitude higher performance than centralized approaches.

As a double-edged sword, however, distributed provenance systems raise a new concern [9] on the provenance itself: *while the provenance is supposed to audit the execution of the application, who then should audit the provenance?* Do we need to build the provenance of provenance? So the recursion goes on and on, indefinitely. Note that this concern was not that critical in a centralized approach as long as we can, which is the case, apply robust reliability mechanisms to the centralized node. However, it turns to be an extremely challenging problem for all the participating nodes in a (large-scale) distributed system: if any single node of the entire deployment is compromised, the provenance as a whole becomes invalid. To this end, *distributed provenance systems* were recently proposed, inspired by blockchains. These systems (e.g., ProvChain [10], SmartProvenance [11], LineageChain [12]) are also called blockchain-based provenance systems that are both tamper-evident and autonomous, thus guarantee trustworthiness of the provenance data. They share the same key idea: *instead of storing the data on a single node or splitting the data into n exclusively distinct chunks on n nodes, let us* ***replicate*** *the data and maintain a* ***hashed linkedlist*** *for each copy of the data*. The *replication* guarantees the provenance is tolerant to a certain degree of fault (e.g., $\lfloor \frac{n-1}{3} \rfloor$ in a Byzantine system, where the fault is arbitrary, including malicious activities and even possible coalition among participants), and the *hashed linkedlist* guarantees that the provenance data cannot be tampered with without being noticed by a simple hash verification.

In the context of scientific applications and high-performance computing (HPC), however, we encounter unique challenges in employing blockchain-based provenance services. There is a series of concerns on resource utilization: the

978-1-7281-9184-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDE51399.2021.00166

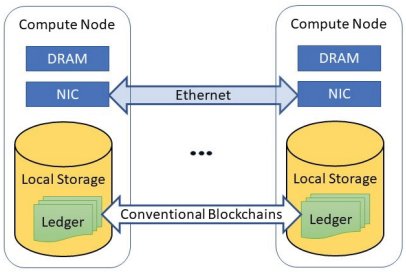


Fig. 1. Conventional blockchain architecture on shared-nothing platforms.

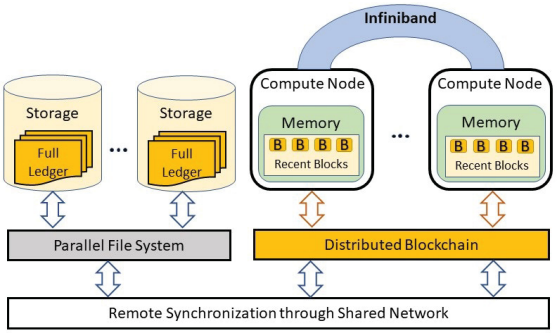


Fig. 2. Proposed HPC blockchain architecture with shared storage.

space efficiency is low, the network bandwidth consumption is high, the CPU cycles are "wasted" for meaningless mining, to name a few. Besides, existing blockchain-based provenance systems are built in such a way that the underlying blockchain infrastructure is a black box, and the provenance service works as a higher-level application by calling the programming interfaces provided by the blockchain infrastructure such as Hyperledger Fabric [13] and Ethereum [14]. In the best case, the provenance service might miss optimization and customization opportunities because it cannot modify the lower blockchain layer; to make it worse, the applicability of those blockchain-based provenance systems is constrained by the underlying blockchain infrastructure, which is not optimized for, or not applicable to, HPC platforms. For instance, (i) current blockchain-based provenance system becomes useless when compute nodes have no local disks, (ii) inappropriate consensus protocols with either compute-intensive or communication-intensive design, and (iii) incompatibility with multi-tiered storage architecture such as burst buffers, I/O nodes, remote parallel file systems. Figure 1 shows the system architecture of conventional blockchains deployed to a shared-nothing cluster. Regardless of private (Hyperledger [13]) or public (Ethereum [14]), all existing blockchain systems assume that the underlying computer infrastructure is shared-nothing: the memory subsystems and I/O subsystems are all independent of the participant nodes who are often connected through commodity networking.

In summary, a highly desired provenance system for scientific applications should be crafted with a balance between scalability, reliability, and applicability. Unfortunately, existing provenance systems failed to meet the above requirements from scientific computing and HPC communities. Of note, a few recent works indeed proposed a blockchain-like provenance system deployed to HPC systems. These works are either a preliminary study [15] or solely depends on conventional compute-intensive proof-of-work (PoW) protocol [16] which is hardly applicable to real-world settings for the HPC architecture. What is desired is a protocol that is particularly crafted to adopting blockchain-backed provenance in HPC systems and overcomes the resource utilization challenges from various perspectives, such as replacing the conventional compute-intensive consensus by more cost-effective ones, enforcing memory constraints on compute nodes, and limiting inter-node communications to reduce network overhead.

B. Proposed Approach

This paper proposes a new distributed approach to manage the data provenance of scientific applications deployed to HPC systems. Rather than only taking an existing blockchain system as a block box, we hack into blockchain internals to improve the applicability and performance of provenance services built upon blockchains. Specifically, as shown in Figure 2, we propose a new blockchain architecture supporting multi-tier storage and then design new consensus protocols aiming to optimize the distributed provenance services in an HPC environment. The proposed architecture applies to any resource-constrained environment, where the nodes need to operate with limited memory and require no local disks (i.e., lightweight replication support) but supported by a distributed remote-storage that persists the blockchain. Besides, the secured lightweight privacy mechanism applied in the newly proposed consensus protocol (i.e., HPC compatible lightweight custom protocol) allows the proposed system to be able to work in the blockchain-based HPC ecosystem.

In summary, this work makes the following contributions:

- We propose a new architecture for secure and reliable distributed data provenance on HPC systems. The new architecture is tailored to the HPC environment: compute nodes can maintain the blockchain in local memory with minimum overhead (i.e., lightweight ledger support) while using distributed shared ledger as a persistent medium (i.e., multi-tiered storage support) for enhanced reliability and as a precaution for any catastrophes (e.g., compute nodes failure or restart).
- We design a set of consensus protocols, namely, proof-of-scalable-traceability (POST), for validating applications' data provenance following a *push-pull* mechanism that promises memory optimization (i.e., HPC compatible lightweight custom protocol). The key idea of POST is that the consensus comes not only from the fellow compute nodes but also from the remote shared storage through proof-of-extended-traceability (POET). POET comes into action only if the compute nodes are unable to reach consensus.

- We implement a system prototype, SciChain, and experimentally verify the system's effectiveness (i.e., reliability in III-B, lightweightedness in III-C, scalability in III-D) with more than one million transactions derived from both micro-benchmarks on up to 100 nodes.

II. System Design

A. Architecture Overview

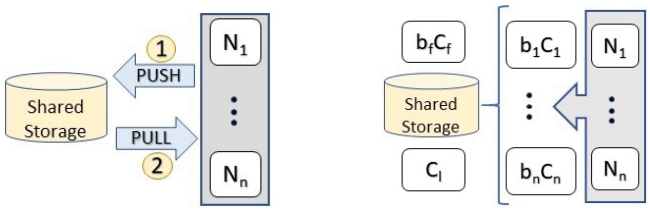
Our proposed system consists of four key modules: a lightweight distributed blockchain, shared storage persistence, a ledger synchronization protocol, and a consensus protocol. We will discuss each of them in more detail in the following.

1) Lightweight Distributed Blockchain: The first module is a distributed blockchain specially crafted to alleviate the memory constraints in compute nodes. The module helps enable the compute nodes to perform the block validation process while keeping a minimum number of recent blocks (e.g., 100-200 blocks) in memory. As we are interested in permissioned blockchains for HPC, a high-performance network infrastructure (e.g., Infiniband) interconnects all the compute nodes to speed up the communication.

2) Shared Storage Persistence: The second module acts as a persistent medium to hold the entire blockchain. As we assume that the compute nodes are generally volatile and store the distributed blockchain in memory, persistent shared storage is essential to provide more reliable backup in case of any catastrophes (e.g., more than 50% compute nodes crash or restart or lose their in-memory blocks).

3) Ledger Synchronization Protocol: The third module is an extended validator protocol (i.e., proof-of-extended-traceability) that supports a faster validation mechanism. Whenever a block is broadcasted, first, the compute nodes come forward to validate the block with the support of the in-memory blocks. If more than 50% of nodes fail to validate the block, the protocol then employs the shared storage to provide the validation service. The protocol also supports the synchronization among the distributed ledger of the compute nodes and the shared storage. The benefit of this protocol is two-fold: (i) it synchronizes the compute nodes in case of any catastrophes; (ii) it provides a reliable guaranteed validation through the shared storage. We will discuss more details in Section II-B1.

4) Consensus Protocol: The fourth module manages the consensus mechanism among the compute nodes and the shared storage. The consensus protocol (i.e., proof-of-scalable-traceability) leverages the third module (i.e., ledger synchronization protocol) to employ the nodes in the validation process, and finally, helps in aggregating the compute nodes votes to provide the system the consensus for a block. The benefit of this protocol is that it minimizes the extensive communication overhead between the compute nodes and the shared storage during the consensus gathering process. This is because the consensus process stops as soon as 51% of nodes can attain the consensus for a block. Section II-B2 provides more details on this protocol.



(a) Push-Pull (b) Consensus

Fig. 3. Some primitives in the proposed protocol. Note that the 'shared storage' is a cluster of nodes, in despite of being drawn as a single cylinder for simplicity here.

B. Protocols

The proposed architecture is designed to work through two parallel protocols. Protocol 1 (i.e., proof-of-extended-traceability) helps in validating a block consists of new provenance records, while Protocol 2 (i.e., proof-of-scalable-traceability) works with gathering consensus from the nodes after the validation and helps Protocol 1 in making the final decision about storing the validated block both in the shared storage's disk and compute nodes' in-memory. The proposed protocol optimizes memory consumption following two phases, as shown in Figure 3(a). In Phase 1 (i.e., *Push method*), the shared storage is being leveraged to persist the entire ledger replica with minimum communication overhead as all ledgers on compute nodes are necessarily volatile and only keep the recent blocks in a limited amount of memory, as the nodes are usually disk-less. In Phase 2 (i.e., *Pull method*), *proof-of-extended-traceability* (i.e., Protocol 1) pushes the storage node into a more active position: whenever a new block is created, it will be validated by the compute nodes first, and if more than 50% of compute nodes are compromised or unable to validate the new block with the in-memory blocks, the shared storage participates in the validation process.

To reduce the communication overhead, the overall consensus (i.e., 51% votes) aggregation is controlled by the shared storage, as shown in Figure 3(b). The compute nodes forward their *hashed-votes* (b_iC_i, where b indicates a block and C indicates a vote by a node for that block) to the shared storage after validating a block, where it (i.e., shared storage) has the aggregation mechanism to generate the final consensus for a block (b_fC_f). During the aggregation process, the shared-storage randomly picks one of the compute nodes' hashes from the *hashed-votes* list to set the *new-id* for the newly validated block. As soon as 51% consensus is reached, the shared storage broadcasts the final consensus along with the hashed-votes list to the compute nodes to suspend further validation. A node's hashed-vote is encrypted with its private key.

To keep the compute nodes up-to-date, we leverage the *Pull method*, as shown in Figure 3(a). That is if a compute node is unable to validate a block and provide the vote, it is then added to the C_l list (i.e., shown in Figure 3(b)) in the shared storage. If more than 50% of compute nodes fail to provide vote during a block validation process, the shared storage will

Protocol 1 Proof-of-Extended-Traceability on storage

Require: Compute nodes C where the i-th node is C^i; C^i_B the in-memory recent block list on C^i; b_o oldest block in the i-th compute node's C^i_B; a newly mined block b; shared storage D; D_B the blockchain copy on the storage; Synced compute nodes C_l where the k-th node is C^k_l;

Ensure: b is validated by both the local SciChain ledger C^i_B and the shared ledger D_B, and then appended to all C_B's and D_B.

1: **function** PROOF-OF-EXTENDED-TRACEABILITY(b, C, D)
2: $i \leftarrow 1$
3: **while** $|Consensus| <= \frac{C}{2}$ **do** ▷ Call Protocol 2
4: Validate b with recent block list C^i_B
5: **if** C^i_B can not validate b **then**
6: $C_l \leftarrow C_l \cup C_i$
7: **end if**
8: $i \leftarrow i + 1$
9: **end while**
10: **if** $|Consensus| <= \frac{C}{2}$ **then** ▷ Call Protocol 2
11: Validate b with D_B ▷ Pull method
12: **for** $C^k_l \in C_l$ **do**
13: Update with recent block from D_B
14: **end for**
15: **end if**
16: **if** $|Consensus| > \frac{C}{2}$ **then** ▷ Call Protocol 2
17: $D_B \leftarrow D_B \cup b$ ▷ Push method
18: **for** $C_i \in C$ **do**
19: **if** $b \notin C^i_B$ **then**
20: $C^i_B \leftarrow C^i_B - b_o$
21: $C^i_B \leftarrow C^i_B \cup b$
22: **end if**
23: **end for**
24: **end if**
25: **end function**

Protocol 2 All compute and storage nodes reach consensus

Require: Compute nodes C where the i-th node is C^i; C^i_B the in-memory recent block list on C^i; a new block b; shared storage D; D_B the blockchain copy on the storage;

Ensure: At least 51% nodes provide $Consensus$ who validate b both with in memory latest block list and with shared ledger D_B.

1: **function** POST-CONSENSUS(b, C, D)
2: $Consensus \leftarrow \emptyset$
3: **if** $b \notin D_B$ **then**
4: **for** $C_i \in C$ **do**
5: **if** C^i_B can validate b **then** ▷ Call Protocol 1
6: $Consensus \leftarrow Consensus \cup C_i$
7: **if** $|Consensus| > \frac{C}{2}$ **then**
8: break ▷ Stop consensus process.
9: **end if**
10: **end if**
11: **end for**
12: **if** $|Consensus| <= \frac{C}{2}$ **then** ▷ 51% consensus?
13: **if** D_B can validate b **then** ▷ Call Protocol 1
14: $Consensus \leftarrow Consensus \cup D$
15: **end if**
16: **end if**
17: **end if**
18: **end function**

update the nodes enlisted in C_l with the recent block from the storage node's blockchain.

1) Proof of Extended Traceability (POET) : POET (i.e., shown in Protocol 1) helps in both validating and storing a block and works in two steps. First, it checks whether it can validate a block (e.g., new provenance records) with the help of the recent blocks stored in the compute nodes memory, as shown in Line 4. If a compute node is unable to validate a block, it is enlisted to the C_l (Line 6). If more than 50% compute nodes fail (Line 10), which is rare, the storage node then comes forward (i.e., *Pull method*) to proceed further with the block validation process as shown in Line 11. The block is stored both in the compute nodes (Line 21) as well as in the shared storage (*Push method* in Line 17) only if more than 50% nodes jointly from compute nodes and shared storage provide consensus (Line 16). We remove the oldest block from the in-memory block list (Line 20) of a compute node before appending the newest block. Besides, we stop the block validation process as soon as 51% consensus is attained (Line 3).

The benefits of this new validation method are two-fold: (i) A faster validation process achieved by the agreement of the compute nodes, thanks to the in-memory support, (ii) Extended and stable validation support is achieved from the shared storage that serves as a reliable persistent medium if more than 50% compute nodes are compromised or fail to provide consensus. The time complexity of Protocol 1 is $O(|C|)$, which is on par with the original PoW consensus. However, most of the time, it is possible to reduce the number of iterations, as we stop the validation process as soon as 51% consensus is attained.

2) Proof of Scalable Traceability (POST): To achieve the consensus, as shown in Figure 3, Protocol 2 is applied in parallel with Protocol 1. Protocol 2 ensures that most of the compute nodes (more than 50%) guarantee the validity of the newly proposed block from all respective in-memory blocks previously-stored in all compute nodes and the shared storage. If 51% compute nodes get compromised, it is guaranteed that at-least the shared storage can ensure validity. To avoid duplication, the new block validation and consensus collection process start only if the block is not already validated, as shown in Line 3 of Protocol 2. If the provenance records in the newly proposed block are new, all of the compute nodes in the network attempt to validate the block (Line 4). Each compute node provides a vote after validating the block (i.e., Line 6). If more than 50% nodes agree on the validity of the new block, the block will be appended both in the shared storage and the compute nodes, as shown in Lines 17

and 21 of Protocol 1 respectively. However, if at-least 51% compute nodes are unable to reach the consensus (i.e., Line 12), then the remote storage node starts to validate the block through Protocol 1 (i.e., Line 13 in Protocol 2) to achieve 51% consensus jointly from compute nodes and remote storage node. To minimize the block validation cost, we stop the consensus attaining process as soon as at least 51% compute nodes provide consensus, as shown in Line 7.

III. Evaluation

We have implemented a prototype system of the proposed blockchain architecture and consensus protocols with Python. At this point, we only release the core modules of the prototype; some complementary components and plug-ins are still being tested with more edge cases and will be released once they are stable enough. The source code is currently hosted on Github. Figure 2 illustrates the overview of the proposed SciChain architecture. The prototype system currently runs on a high-performance computing (HPC) system consist of 58 nodes where (i) each node is equipped with 32 cores; hence each node can be emulated with up to 32 individual nodes through user-level threads, (ii) ledgers are stored on in-memory, (iii) shared storage's ledgers are stored on as files.

A. Experiment Setup

1) Testbed: All experiments with the proposed system are carried out on a high-performance computing (HPC) cluster comprised of 58 nodes interconnected with FDR InfiniBand. Each node is equipped with an Intel Core-i7 2.6 GHz 32-core CPU along with 296 GB 2400 MHz DDR4 memory. There is no local disk on compute nodes, which is a typical configuration on HPC systems; yet, a remote 2.1 PB storage system is available managed by GPFS [17]. Each node is installed with Ubuntu 16.04, Python 3.7.0, and NumPy 1.15.4. We deploy the system prototype mostly on 100 cores.

2) Evaluated Systems: We evaluate the SciChain prototype against two other blockchain systems. The first blockchain is a *Conventional Blockchain* system deployed to a shared-nothing cluster comprised of 10 nodes interconnected with Ethernet. Each node in the cluster is equipped with an Intel Core-i7 2.6 GHz 32-core CPU along with 128 GB 2400 MHz DDR4 memory. The second blockchain is a *Memory-only Blockchain* deployed to the same cluster (i.e., HPC cluster with 58 nodes) same as the proposed system with high-performance networking interconnections (i.e., InfiniBand, RDMA) without any persistent storage; this is not a practical solution due to the lack of data persistence but is considered as the performance upper bound of the proposed *SciChain*. We implement all three blockchain systems in Python and make a reasonable effort in optimizing all of them.

3) Workload: For micro-benchmarks, we use YCSB workload [18] commonly used for transaction evaluation. We map each provenance record (e.g., file creation or modification) in a YCSB transaction format before start processing. At the beginning of the transaction, the system checks whether the submitted transaction is valid. If so, the statuses of two

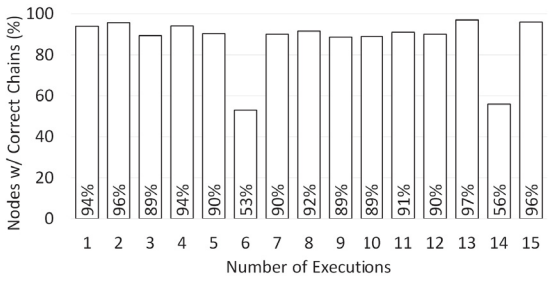


Fig. 4. POST guarantees more than 50% of nodes holding valid blocks.

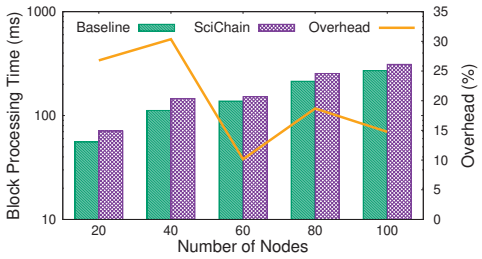


Fig. 5. Latency overhead.

nodes are updated accordingly, followed by the propagation of the updates to all other nodes in the network. On average, each block contains about a thousand transactions in our experiments. We deploy more than one million transactions (1,036,303) to the system prototype, and compare it to the other two baseline systems.

B. Reliability and Trustworthiness

This section demonstrates that the proposed POST consensus can be achieved by more than 50% of participants. In other words, we want to show that introducing the shared storage as an additional node does not reduce the portion of good compute nodes to under 51%. To put it another way, we want to verify, if we try to inject tampered blocks (e.g., provenance records) randomly, whether the new consensus protocol leads to the same longest valid blockchain compared to other consensus protocols. Note that we prove the safety and liveness in Section II-B; we will experimentally verify the trustworthiness and reliability here.

To demonstrate the trustworthiness and reliability, we run the system prototype given random transactions for 10 minutes and repeat the execution 15 times, as shown in Figure 4. All of the 15 executions lead to more than 50% nodes holding the correct blockchains: 13 out of 15 executions yield more than 90% validity, while two executions exhibit lower ratios because we terminate the execution once more than 50% of nodes hold the correct blocks. The bottom line is that we need to guarantee at least 51% of nodes' data are not tampered with, which is the case.

C. Overhead

This section reports the provenance overhead incurred by the proposed SciChain. The memory-only baseline system

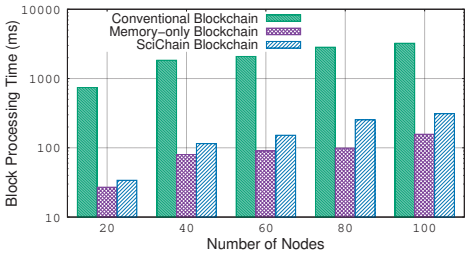


Fig. 6. Scalability of three blockchain systems.

persists the data provenance to the disk with no security or audibility guarantees. We turn on SciChain atop the baseline and measure the end-to-end block processing time compared to the baseline performance.

As shown in Figure 5, we observe the overhead incurred by the proposed SciChain is noticeable (25% – 30%) at small scales of 20 and 40 nodes. This is the price we have to pay to achieve high security. The good news is, however, the overhead is reduced to under 20% on larger scales; in particular, the overhead is only 15% on 100 nodes. That is, unlike conventional provenance systems whose overhead increases proportionally to the number of nodes, SciChain's overhead ratio does not significantly increase, thanks to the POST consensus. In the POST mechanism: (i) a block gets persisted only once after it gets qualified by majority votes; (ii) the larger the scale, the fewer chances the POST will redirect the validation to the shared storage that incurs reasonable, if not negligible, overhead.

D. Scalability

Figure 6 reports the performance of the three systems on 20-, 40-, 60-, 80-, and 100-node scales. The memory-only blockchain, as expected, achieves the highest performance (i.e., lowest processing time). The proposed SciChain with shared storage does not exhibit a significant slowdown than the upper bound; for instance at 100-node scale, the comparison is 157 *ms* vs. 311 *ms*, at the same order of magnitude. However, the conventional blockchain on 100 nodes appends a new block in 3,231 *ms*, significantly slower than SciChain. Specifically, SciChain shows significant speedup in performance compared with existing systems: $\frac{3,231}{311} = 10.4\times$.

IV. Conclusion and Future Work

This paper proposes a new blockchain consensus protocol, namely POST, to enable immutable and autonomous data provenance for scientific applications deployed to HPC systems. POST is implemented in a prototype system called SciChain, the first practical HPC-blockchain system toward trustworthy data provenance in HPC. The effectiveness and efficiency of POST experimentally demonstrated through both micro-benchmarks on up to 100 nodes.

Our future work is two-fold. Firstly, the current POST protocol does not take into account the energy consumption of the underlying data movement. We plan to design and integrate a subsystem for application-specific optimizer regarding energy consumption. Secondly, at this point, it is unclear how to migrate the data stored in one specific SciChain instance to another. This is also a challenge in the blockchain community: ensuring the atomicity of the transaction migration is far more complicated than it looks.

Acknowledgement

This work is in part supported by the U.S. DOE under contract number DE-SC0020455. This work is also supported in part by the following grants: National Science Foundation CCF-1756013, IIS-1838024.

References

[1] R. B. Weiss, N. J. Vogelzang, B. A. Peterson, L. C. Panasci, J. T. Carpenter, M. Gavigan, K. Sartell, E. Frei, and O. R. McIntyre, "A successful system of scientific data audits for clinical trials: a report from the cancer and leukemia group b," *JAMA*, vol. 270, no. 4, pp. 459–464, 1993.

[2] Data fraud in clinical trials, "https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4340084/," Accessed 2020.

[3] The Importance of Data Set Provenance for Science, "https://eos.org/opinions/the-importance-of-data-set-provenance-for-science," Accessed 2020.

[4] A. Gehani and D. Tariq, "SPADE: Support for Provenance Auditing in Distributed Environments," in *Proceedings of the 13th International Middleware Conference (Middleware)*, 2012.

[5] T. Clark, P. N. Ciccarese, and C. A. Goble, "Micropublications: a semantic model for claims, evidence, arguments and annotations in biomedical communications," *Journal of Biomedical Semantics*, vol. 5, no. 1, p. 28, Jul 2014.

[6] E. Pettersen, T. Goddard, C. Huang, G. Couch, D. Greenblatt, E. Meng, and T. Ferrin, "Ucsf chimera–a visualization system for exploratory research and analysis," *Journal of Computational Chemistry*, vol. 25, no. 13, pp. 1605–1612, Oct 2004.

[7] D. Zhao, C. Shou, T. Malik, and I. Raicu, "Distributed data provenance for large-scale data-intensive computing," in *IEEE International Conference on Cluster Computing (CLUSTER)*, 2013.

[8] D. Dai, Y. Chen, P. Carns, J. Jenkins, and R. Ross, "Lightweight provenance service for high-performance computing," in *International Conference on Parallel Architectures and Compilation Techniques*, 2017.

[9] A. Bates, D. J. Tian, K. R. Butler, and T. Moyer, "Trustworthy whole-system provenance for the linux kernel," in *24th USENIX Security Symposium*, 2015, pp. 319–334.

[10] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, "Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability," in *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, 2017.

[11] A. Ramachandran and M. Kantarcioglu, "Smartprovenance: A distributed, blockchain based data provenance system," in *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '18, 2018, pp. 35–42.

[12] P. Ruan, G. Chen, T. T. A. Dinh, Q. Lin, B. C. Ooi, and M. Zhang, "Fine-grained, secure and efficient data provenance on blockchain systems," *Proceedings of the VLDB Endowment*, vol. 12, no. 9, pp. 975–988, 2019.

[13] Hyperledger, "https://www.hyperledger.org/," Accessed 2018.

[14] Ethereum, "https://www.ethereum.org/," Accessed 2018.

[15] A. Al-Mamun, T. Li, M. Sadoghi, L. Jiang, H. Shen, and D. Zhao, "Hpchain: An mpi-based blockchain framework for data fidelity in high-performance computing systems," in *Supercomputing*, 2019.

[16] A. Al-Mamun, T. Li, M. Sadoghi, and D. Zhao, "In-memory blockchain: Toward efficient and trustworthy data provenance for hpc systems," in *IEEE International Conference on Big Data (BigData)*, 2018.

[17] F. Schmuck and R. Haskin, "GPFS: A shared-disk file system for large computing clusters," in *Proceedings of the 1st USENIX Conference on File and Storage Technologies (FAST)*, 2002.

[18] YCSB, "https://github.com/brianfrankcooper/YCSB/wiki/Core-Workloads," Accessed 2018.