Triastore: A Web 3.0 Blockchain Datastore for Massive IoT Workloads

Panagiotis Drakatos*, Erodotos Demetriou*, Stavroulla Koumou*,
Andreas Konstantinidis‡* and Demetrios Zeinalipour-Yazti*

* Department of Computer Science, University of Cyprus, 1678 Nicosia, Cyprus
‡ Department of Computer Science & Engineering, Frederick University, 1036 Nicosia, Cyprus
{pdraka01, edemet01, skoumo01, akonstan, dzeina}@cs.ucy.ac.cy

***Abstract*—The Internet of Things (IoT) revolution has introduced sensor-rich devices to an ever growing landscape of smart environments. A key component in the IoT scenarios of the future is the requirement to utilize a shared database that allows all participants to operate collaboratively, transparently, immutably, correctly and with performance guarantees. Blockchain databases have been proposed by the community to alleviate these challenges, however existing blockchain architectures suffer from performance issues. In this short paper we propose Triastore, a novel permissioned blockchain database system that carries out machine learning on the edge, abstracts machine learning models into primitive data blocks that are subsequently stored and retrieved from the blockchain. Triastore comprises of two internal routines, namely: (i) *Proof of Federated Learning (PoFL)*, which trains in a distributed manner a global model for the ingested data; and (ii) *Blockchain Consensus*, which commits this generated model data on permissioned blockchain database. We present a detailed explanation of our data ingestion algorithm with relevant examples and carry out an experimental evaluation with image data from MNIST. The evaluation shows that our proposed data ingestion framework retains high levels of accuracy with low loss in data quality.**

***Index Terms*—blockchain, machine-learning, federated-learning, fabric, consensus, databases, privacy.**

I. Introduction

Internet of Things (IoT) refers to a large number of physical devices being connected to the Internet that are able to see, hear, think, perform tasks as well as communicate with each other using open protocols [1]–[4]. IoT devices are connected to Cloud and Edge computing appliances through massively parallel I/O channels (e.g., 5G, Wi-Fi 6) with millisecond latency offering new opportunities in industrial optimization, human health, and well-being as well as safety. In absolute numbers, the IoT revolution is expected to bring the number of such devices close to a staggering 40 billion in 2020, more than double from 2019 [5]. This will procreate tremendous opportunities for IoT applications between multiple parties, such as collaborative multitasking techniques [6], machine learning [7], cooperative benchmarking [8], and augmented reality technology [9].

A key component in the IoT scenarios of the future is the requirement to utilize a shared database that allows all participants to operate collaboratively with more functionality.

Fig. 1. Triastore organizes massive IoT data on the edge over distributed blockchains, which capture the machine learning models behind the data. This offers immutability, transparency, correctness, performance and privacy to future Web 3.0 applications.

The shared database can bridge the actual gap between the data generated from the IoT applications [10] and the rate that these are processed and analyzed in real-time. The objective is to enable users execute updates and queries on the collaborative database while preserving a consistent view among all users maintaining the system consistency and transparency. Moreover, it is essentially common to be compromised by malicious outsources. To mitigate the problem described, an innovative design of a shared database with high performance is required for all the participants, in order to collaborate among each other with trust. Blockchain databases have been proposed by the community to alleviate these challenges, however existing blockchain architectures suffer from performance issues measured in terms of throughput and latency. In this situation, the transactions are basically executed in a sequential manner and this, in conjunction with confidentiality issues, does not leave much space for scaling.

It is imperative to devise a database architecture that can withstand billions of transactions per second, as opposed to thousands transactions per second that is currently the case for typical blockchains due to the expensive verification cost.

In order to motivate our description, we now explain two visionary Web 3.0 scenarios in which the above type of blockchain database can find utility in the future:

Telco Big Data: consider a smart city scenario in which telecommunication companies (telcos) in a city [11], [12] aim to share the network health data from cell towers (e.g.,

signal strength, call drops, bandwidth measurements) with public authorities for monitoring and compliance (e.g., EMF-compliance). Huawei alone reports 5TBs/day for 10M clients (i.e., 2PB/year) for Shenzhen, China, for a respective telco big data scenario, so we are in the realm of massive and big data. From an architectural perspective the challenge is how to transparently and immutably store the collected massive velocity data at the edge of each telecommunication network in order to facilitate efficient and scalable data sharing and access. Storing big data in a centralized way is not a preferable choice, because it doesn't fulfill any of these requirements.

Edge Camera Network: consider again a smart city scenario in which local authorities aim to collaboratively poll local road traffic data in shared information spaces that can be useful for public transit planning at scale. Currently, such a task relies on either disconnected hardware infrastructure or on privacy-invasive smartphone solutions that are outside the administrative control of local authorities and public jurisdictions. Contributing to improvement of road traffic is critical as it is well known that congestion costs U.S. drivers nearly $300 Billion in 2020, which is an average of $1400 per driver per year. In the envisioned scenario, aim to capture road traffic on the edge, extract through deep learning accurate city traffic models that can subsequently be shared among local authorities. The problem again from an architectural perspective is how to transparently and immutably store the collected massive velocity data at the edge in order to facilitate efficient and scalable data sharing and access.

In this short paper we propose *Triastore* (inspired from Greek "Tria", meaning "three"), being a storage layer for our complete Triabase database architecture designed for the Web 3.0 era. Triabase is a permissioned blockchain database system that carries out machine learning at the edge, abstracts machine learning in primitive blocks that are subsequently stored and retrieved from the blockchain. In Triabase, we have two types of nodes those that store the entire shared database, and the others that use the database for their own operations, such as sending query and update requests to the blockchain shared ledger. We expect the blockchain nodes to be synchronized under the decentralized blockchain network. The clients that use the blockchain only for database operations store only the appropriate block header in contrast with the full nodes that store the entire blockchain ledger.

For this purpose, the key challenge is to find a robust design that is able to: *i Execute machine learning algorithms at the edge; ii Operate on a distributed environment; and iii Mitigate issues related to data privacy protection.* Our main goal is to guarantee the following aspects:

- **Immutability:** We want to ensure that any update committed to the blockchain is immutable and will not be tampered by any malicious node;
- **Transparency:** We oblige the shared database to strictly update according to the committed transactions. All database operations e.g., insertions, deletions, updates are transparent to nodes because users are able at any time to get all historical data of the transactions committed on the blockchain;
- **Correctness:** Performing all the required operations with minimal computational requirements and without the excessive energy consumption, when a client receives a query, results from a server node;
- **Performance:** Our system must support a wide range of queries and indexes. As a result, we should allow Triastore to achieve better performance to scale; and thus improve the overall throughput of the network in order to minimize any unnecessary overhead that causes latency;
- **Privacy:** Centralized artificial intelligence algorithms demand from the clients to provide whole trained models, which incurs high data leakage risks and this must be taken into account by Triastore.

To enhance user security and privacy in Triabase, we propose federating learning [13] that has appeared as a new wise choice for distributed machine learning. This technique differs from the traditional artificial intelligence algorithms because the federated learning models demand transmitting only the appropriate parameters from the locally trained models and, thus, transmit local models to the server while the full amount of data is stored in the user endpoint for mitigating security and privacy risks.

In order to test the validity of the system, we have implemented an initial version of our architecture using the hyperledger fabric technology, which enables us to measure the latency as well as the throughput of different parts of our implementation during the ingestion load and during the searching query process. Our preliminary results are very encouraging as they reveal that in our proposed architecture, the tradeoff between the learning accuracy and the efficiency of the trained models from the federated learning approach achieve comparable results.

The main contributions of our paper are as follows:

- We introduce Triastore, a storage layer for a permissioned blockchain database system that is enhanced with federating learning containing the running states and behavior models of the blockchain nodes to ensure the security and data privacy of users;
- We propose a new consensus empowering collaborative mechanism namely Proof of Federated Learning (PoFL) to share parameters over distributed multiple parties to reduce the risk of data leakage and to protect federated nodes from being tampered;
- We also implement our proposal with the integration of the fabric open-source platform to provide a more realistic blockchain assessment.

The rest of the paper is organized as follows. Background and Related work is presented in Section II. In Section III, the proposed system architecture and the communication protocol of Triastore is presented. In Section IV, we present an experimental evaluation study of the presented ideas over Fabric that exposes the accuracy and performance of our approach. Finally, Section V concludes the paper.

II. Background And Related Work

A. Blockchain and Artificial Intelligence for IoT environments

The main usage of the blockchain architecture is to keep records on an immutable chain of blocks, so later on, nodes agree on the shared state across a network of untrusted participants. Thus, it forms the blockchain platform that can be viewed as a distributed (transaction-log or) database system. The blocks are agreed by the majority of validators according to the consensus protocols that tolerate Byzantine faults. The most well-known platforms include Capera [14], Hyperledger [15], Monoxide [16]. This design does not require a centralized server and operates in untrusted environments of arbitrary nodes.

The authors of [14] introduce a system named Caper, a permission blockchain architecture based on an acyclic graph and on three consensus protocols to support internal and all cross-application transactions. Moreover, [17] introduces a novel framework, called vChain, which is able to improve the storage and computing costs of the user and employs verifiable queries to ensure the system integrity. The design of a privacy-preserving contact tracing framework to ensure the integrity of the tracing procedure has not been sufficiently studied and remains a challenge. In paper [18], the authors propose P2B-Trace, a privacy-preserving contact tracing initiative based on blockchain and privacy-preserving principles are a future direction of our proposed architecture.

Moreover, a considerable interest in the blockchain field is the scalability and performance characteristics of blockchain networks. Algorand [19] and RandHound [20] achieve high scalability by randomly selecting a subset of validators to participate in the consensus, while they maintain and guarantee the same security level with other blockchain infrastructure. Other works [21] use directed acyclic graphs instead of a blockchain structure and they ensure that the average amount of time for each transaction is reduced. Blockbench [22] was the first to look for permissioned blockchain in the context of benchmarking. They present an approach for comparing the performance of different platforms including Ethereum Parity, and Hyperledger Fabric by using a set of micro and macro benchmarks. Furthermore, [15] introduces the architecture of fabcoin which presents the performance of bitcoin in Fabric.

III. The Triastore Algorithm

A. Overview

In this section, we introduce the proposed Triastore algorithm and discuss its two internal routines, namely: (i) *Proof of Federated Learning (PoFL) routine*, which trains in a distributed manner a global model for the ingested data; and (ii) *Blockchain Consensus* routine, which commits this generated model data on permissioned blockchain database. The core functionality of our proposition is illustrated at a high level in Algorithm 1. The first routine of Triastore is the *PoFL*, which utilizes a convolution network loss function to train the local models across multiple decentralized edge nodes holding local data samples, without exchanging them. The final goal is to compute an average model and to converge fast with high learning accuracy. The second routine of Triastore is the blockchain process that is triggered after a respective leader election process takes place. The blockchain process is responsible to collaboratively maintain the blockchain structure, endorse new transactions from blockchain nodes, and is partially responsible for the 2-step consensus protocol.

Algorithm 1 The Triastore Algorithm

Input: Data D on blockchain nodes N, Time Epoch t, Train. Weights W_t
Output: Blockchain TX identifier B_{id}

▷ **Routine 1: Proof-of-Federated-Learning (PoFL)**
1: $M = ml_{init}(W_t)$ ▷ **global model**
2: **for all** $n_i \in N$ **do** ▷ **Distributed Training**
3: *determine* $M_i = ml_{\text{training}}(M, d_i, n_i)$
4: *determine* $ml_{\text{merging}}(M, M_i)$
5: **end for**

▷ **Routine 2: Blockchain Consensus**
6: *determine* $l = leader(N, t)$ ▷ Leader / Orderer Election
7: *determine* $u = view\_number(N, t)$ ▷ Consensus Round
8: $T_{rx} \leftarrow init\_tx(t, u, W_t)$ ▷ initialize transaction
9: **while** $(!T_{rx})$ **do** ▷ **Fabric Consensus Phase**
10: **if** T_{rx}.PRE-PREPARE **then** ▷ Fabric stage PRE-PREPARE
11: *calculate* f = l.difficulty(W_t,alpha) ▷ $alpha = bc_{\text{depth(t-1)}}$
12: *construct* B_{id} = **l.build**(T_{rx}, W_t, u, f) ▷ **Triastore Block**
13: **for all** $n_i \in N$ **do**
14: l.send(PREPARE,t,l,f,B_{id}) ▷ lead by l
15: n_i.receive(PREPARE,t,l,f,B_{id}) ▷ 2-step consensus initiated
16: **end for**
17: **end if**
18: **if** T_{rx}.LEDGER-UPDATE **then** ▷ Fabric stage LEDGER-UPDATE
19: **for all** $n_i \in N$ **do**
20: l.send(COMMIT,t,l,f,B_{id}); ▷ lead by l
21: n_i.receive(COMMIT,t,l,f,B_{id}); ▷ Commit block in Triastore
22: **end for**
23: **end if**
24: **end while**
25: return B_{id}

B. Routine 1: Proof of Federated Learning (PoFL) Routine

The Bitcoin protocol uses a PoW (Proof-of-Work) consensus mechanism to validate users' transactions in the blockchain. This is associated with an extremely high energy consumption bill, which is unnecessary in a private (permissioned) blockchain where contributing nodes are of higher trust. Yet, provisioning a consensus mechanism is still necessary in order to provide an incentive to participating nodes to contribute to the transaction verification process. To this end, in this work we propose such a consensus mechanism that relies on Federating Learning, as such, is coined *Proof of Federated Learning (PoFL)*.

In Algorithm 1, lines 1 to 5, we show the overall execution of the PoFL proposition in this work. Initially, in line 1 a global training model (M) is constructed using a function ml_{init} taking as input some training weights W_t, where t is the epoch. Subsequently, in lines 2-5 this global training init model M is used by all $|N|$ nodes in the network in order to derive local refined models $M_i, i \leq |N|$. The local M_i models are eventually merged in the global M model in order to finalize the model construction process. In the above

process, it is apparent that every block generation in epoch t relies on the previous model generated in epoch $t-1$.

C. Routine 2: Blockchain Consensus Routine

In Algorithm 1, lines 6 to 21, we show the overall execution of the blockchain consensus routine. The process starts in lines 6-7 with a leader election routine followed by a view_number routine, both of which take as input the blockchain network N and the epoch t. The former yields the leader l while the later infers the fabric consensus round, which helps in the convergence of the consensus process and guarantees the liveness of the consensus protocol. Subsequently, in line 8 the transaction is boostrapped and passes through two stages: the PRE-PREPARE stage (lines 10-17) and the LEDGER-UPDATE stage (lines 18-23). The PRE-PREPARE stage starts out by having the leader l computing the blockchain *difficulty*, which is derived based on the length of the blockchain ($alpha = bc_{\text{depth(t-1)}}$). Particularly, longer chains are expected to be more difficult while shorter chains have a lower difficulty. Based on the above a triastore data block is constructed and broadcasted in the network for storage (i.e., lines 13-16). The LEDGER-UPDATE stage basically wraps up the communication by carrying out a final commit broadcast.

D. Triastore Example

The overall scheme for Triastore is shown in Figure 2. The process starts with the local training of the model on their user's data. After that, the communication process takes place where all users broadcast and upload the appropriately trained models to the blockchain nodes and store them as transactions to the distributed ledger. The blockchain node that was the winner from the previous round (depends on the blockchain difficulty) is responsible for initiating a 2-step consensus protocol and construct the blocks with all the cached transactions that are not validated yet.

In addition, the winner node is in charge of aggregating the local model of clients and producing a shared model by putting it as the first transaction in the block, so later on, the federated learning nodes can access it in the next round $r+1$. Our PoFL consensus protocol contemplates that users who participate in the blockchain process get rewarded with training coins. The coins of each user are awarded according to the performance in the training process. Particularly, federated nodes converging faster and achieving more accuracy are rewarded higher. The node that receives the most accuracy coupled with the difficulty of the block recognized as the winner of the given round r. Furthermore, in every training round the coins will be adjusted to the users depending on their work.

Nevertheless, to secure our protocol and ensure that every user will obey the protocol, we introduce a new hierarchy of nodes that we coin *peacemaker entity*. This entity is responsible to observe the correctness of the protocol followed by all the federating nodes. For example, users that refuse to cooperate with the protocol will get no payment for their work. Moreover, users that will try to get more rewards and try to

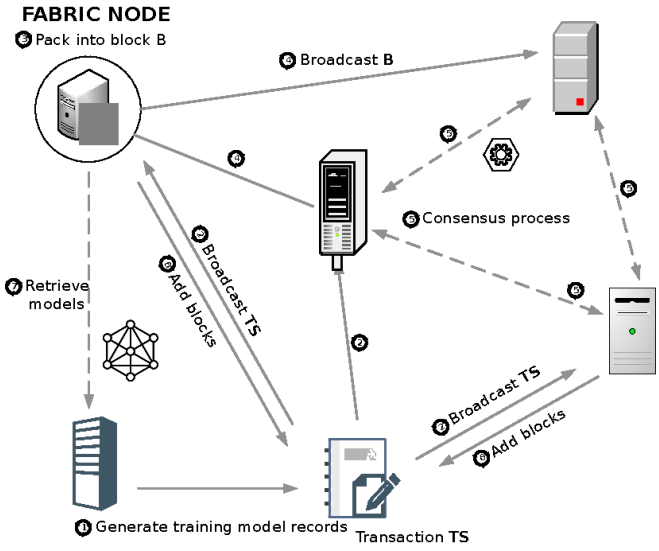

Fig. 2. Example Execution of a Transaction in Triastore

counterfeit the correctness of the whole process will not be rewarded by the peacemaker entity. The peacemaker will then claim the adjusted coins as their own reward for their effort in the protocol correctness.

E. Discussion

In this section we carry out a more detailed discussion around the proposed algorithm.

Consensus Protocol: The two steps of the consensus protocol, summarized in Algorithm 1, include the execution of the PBFT [23] [24] algorithm and the notation of the detector. The orderer collects all the received transactions along with some endorsement proposals, constructs a new block, and initiates the 2-step phase by sending the proposal block to all blockchain nodes for verification. In particular, the orderer broadcasts a pre-specified message to all fabric nodes to initiate the protocol. The message contains the proposed block B_{id}, the current epoch t, the current leader l based on the view number u (used for the PBFT [23] participation) and the block difficulty f that relies on the height of the blockchain. Then all fabric nodes $n \in N$ echo the same message until the majority of them receive at least a quorum of $3f+1$ valid messages. Each blockchain node checks the validity of all transactions that exist in a block, by analyzing the endorsement policy from the assigned peers. The detector is responsible for supervising that all the appropriate nodes comply with the endorsement policy and only the applicable peers join the process. The latter ensures that the client is not compromised and does not incorporate invalid results that may cause erroneous behavior to the Triastore blockchain.

Ledger Update After the fulfilment of the consensus protocol, the invoking process is called in which: (i) each client updates its copy of the ledger; and (ii) each client is notified about the ledger updates. In the proposed Triastore algorithm,

the learning process is executed locally, i.e., trains machine learning models locally. Furthermore at the edge division, all local models are aggregated iteratively (in multiple rounds) to construct the final models, which are then stored in the blockchain (during the invoking process). We assume that all edge server nodes have enough computing and caching resources for completing complex calculations and maintaining the blockchain, in order to store the federated learning parameters collected from the users.

IV. Experimental Methodology and Evaluation

This section presents an experimental evaluation of our proposed Triastore data ingestion algorithm. We start out with the experimental methodology and setup, followed by two experiments. In the first experiment, the performance of federated accuracy is compared against two baseline approaches: one for CPU bound and the other with the GPU bound, with respect to various metrics on a set of real-world datasets. The second experiment examines the loss function of the data that is compacted on the blockchain network due to the Triastore ingestion algorithm. For brevity in this short paper, we omit results relevant to the Consensus routine of the Triastore algorithm.

Implementation and Dataset: We have implemented Triastore by modifying the Tensorflow federated Framework [25] and measure it through the single-machine simulation runtime provided by TFF. The performance of the federated learning process is tested on a real-world dataset MNIST [26]. This dataset contains training examples for about 60,000 different scenarios and the testing metrics contain about 10,000 examples.

Computing Server: Our evaluation is carried out on the DMSL VCenter IaaS datacenter, a private cloud. Our computing server is an Ubuntu 16.04 server image, featuring 8GB of RAM with 2 virtual CPUs (@ 2.40GHz). The image utilize fast local 10K RPM RAID-5 LSILogic SCSI disks, formatted with VMFS 5.54 (1MB block size).

A. Experiment 1: PoFL approach

We compare the proposed approach with respect to the state of the art federated learning algorithm FedAvg [27].

Experimental Setup: We use the CNN convolution layer and two dense layers. The first two convolutional layers have 32 and 64 filters respectively and they are responsible for setting the communication channels dynamically based on the width and the height of the image. The pool size is set dynamically (2,2) and the kern size is 5. Moreover, convolution layers followed by a dropout [28] with a probability of 0.7. The second convolutional layer has also a flatten operation. The last two dense layers are fully connected layers with 512 units activated by ReLu and a softmax output layer.

Algorithmic settings: In all experiments the algorithmic parameters were configured as follows: local mini-batch $B = 20$, the trained local epochs $E = 10$, the total number of clients $K = 500$ and the fraction of clients that performs computation

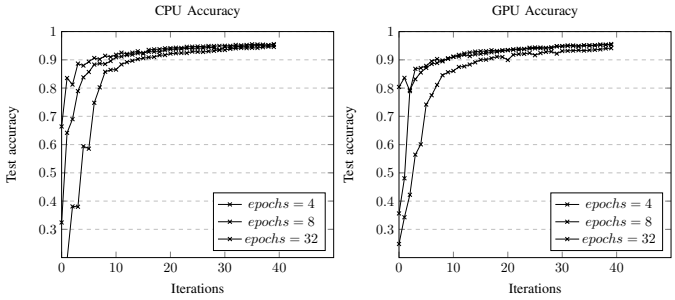

Fig. 3. Performance Evaluation: Triastore evaluation in terms of accuracy for both CPU and GPU on the federated learning data in the MNIST dataset

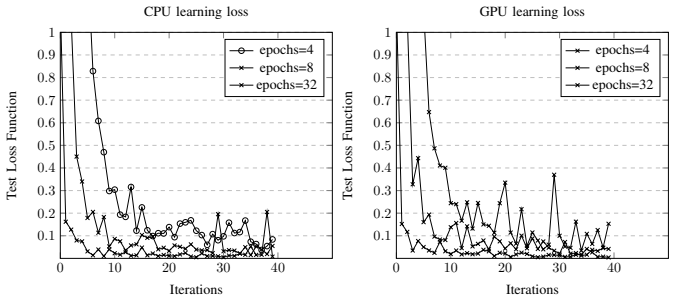

Fig. 4. Performance Evaluation: Triastore evaluation in terms of learning loss for both CPU and GPU on the federated learning data in MNIST dataset

at each round $C = 0.05$. The local training process for each client proceeds with the SGD optimizer with a learning rate $\eta = 0.001$ and no weight decay.

Figures 3 and 4 illustrate the performance of the proposed approach in terms of learning accuracy and learning loss, respectively, for various epochs over two different metrics. The results show that the proposed federated learning approach achieves high accuracy (95%) and low learning loss (10%) with a small set of iterations for both CPU and GPU metrics. Moreover, the federated learning is performed faster when the GPU version is used and increases while the number of epochs increase. In particular, in the first 10 rounds, the training of the model converges faster and the accuracy of the model increases with the increase of the epoch. After 35 rounds, the accuracy is slightly reduced (by 2-4%) or remains the same, especially for the models trained with larger epoch values. This is due to the overfitting of the CNN model.

Figure 4 shows that the learning loss is generally high at the beginning and it highly depends on the epoch value. At the beginning in round 5, the learning loss around 0.5, which is relatively high when the epoch value is low. In contrast, when the epoch value is high then the learning loss is reduced, which shows that the model converges. Moreover, the results show that when the epoch value is 32, the learning loss is reduced to almost zero, after round 10. There are also cases where the learning loss is high and this is because of overfitting and then is reduced again to a close to zero value after some iterations, e.g., in round 38.

V. Conclusion and Future Work

In this short paper we propose Triastore, a novel permissioned blockchain database system that carries out machine learning on the edge, abstracts machine learning models into primitive data blocks that are subsequently stored and retrieved from the blockchain. Triastore comprises of two internal routines, namely: (i) *Proof of Federated Learning (PoFL)*, which trains in a distributed manner a global model for the ingested data; and (ii) *Blockchain Consensus*, which commits this generated model data on permissioned blockchain database.

We present a detailed explanation of our data ingestion algorithm with relevant examples and carry out an experimental evaluation with image data from MNIST. The evaluation shows that our proposed data ingestion framework retains high levels of accuracy with low loss in data quality. We conducted experiments on a real world dataset where we observe the improved accuracy that our federated learning approach achieves.

In future, we aim to expand the triastore architecture with complete querying layers so that we can carry out experiments with more datasets at scale. We also aim to investigate the effectiveness of Triastore on an edge deep learning datacenter, equipped with powerful GPU cards. We finally also aim to investigate techniques and algorithms to achieve stronger privacy guarantees.

References

[1] L. Yao, Q. Z. Sheng, and S. Dustdar, "Web-based management of the internet of things," *in IEEE Internet Computing, vol. 19, iss. 4, pp. 60–67*, 2015.

[2] A. A. Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Comm. Surv. Tutor., vol. 17, no. 4, pp. 2347–2376*, 2015.

[3] S. Li, L. D. Xu, and S. Zhao, "The internet of things: a survey," *Information Systems Frontiers*, vol. 17, no. 2, pp. 243–259, 2015.

[4] L. Atzori, A. Iera, and G. Morabito., "The internet of things: A survey," *Comput. Netw., vol. 54, iss. 15, pp. 2787–2805*, 2010.

[5] Juniper Research, "IoT connected devices to almost triple to over 38 billion units by 2020," 2019. [Online]. Available: https://tinyurl.com/juniperresearchIoT

[6] Wenliang Du and M. J. Atallah, "Privacy-preserving cooperative scientific computations," in *Proceedings. 14th IEEE Computer Security Foundations Workshop*, 2001, pp. 273–282.

[7] D. Billsus and M. J. Pazzani, "Learning Collaborative Information Filters," in *Proceedings of the Fifteenth International Conference on Machine Learning*, ser. ICML '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Jul. 1998, pp. 46–54.

[8] M. Atallah, M. Bykova, J. Li, K. Frikken, and M. Topkara, "Private collaborative forecasting and benchmarking," in *Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, ser. WPES '04. New York, NY, USA: Association for Computing Machinery, Oct. 2004, pp. 103–114.

[9] J. Li, C. Wang, X. Kang, and Q. Zhao, "Camera localization for augmented reality and indoor positioning: a vision-based 3D feature database approach," *International Journal of Digital Earth*, vol. 13, no. 6, pp. 727–741, Jun. 2020, publisher: Taylor & Francis.

[10] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, "A Vision of IoT: Applications, Challenges, and Opportunities With China Perspective," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 349–359, Aug. 2014, conference Name: IEEE Internet of Things Journal.

[11] C. Costa and D. Zeinalipour-Yazti, "Telco big data research and open problems," in *Proceedings of the 35th IEEE International Conference on Data Engineering*, ser. ICDE'19. 8-12 April 2019, Macau SAR, China: IEEE Computer Society, 2019, conference, pp. 2056–2059.

[12] C. Costa, G. Chatzimilioudis, D. Zeinalipour-Yazti, and M. F. Mokbel, "Efficient Exploration of Telco Big Data with Compression and Decaying," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, Apr. 2017, pp. 1332–1343, 2375-026X.

[13] L. Corinzia and J. M. Buhmann, "Variational Federated Multi-Task Learning," *arXiv:1906.06268 [cs, stat]*, Jun. 2019, arXiv: 1906.06268.

[14] M. J. Amiri, D. Agrawal, and A. E. Abbadi, "CAPER: a cross-application permissioned blockchain," *Proceedings of the VLDB Endowment*, vol. 12, no. 11, pp. 1385–1398, Jul. 2019.

[15] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukoli, S. W. Cocco, and J. Yellick, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, ser. EuroSys '18. New York, NY, USA: Association for Computing Machinery, Apr. 2018, pp. 1–15.

[16] J. Wang and H. Wang, "Monoxide: Scale out Blockchains with Asynchronous Consensus Zones," 2019, pp. 95–112.

[17] C. Xu, C. Zhang, and J. Xu, "vChain: Enabling Verifiable Boolean Range Queries over Blockchain Databases," in *Proceedings of the 2019 International Conference on Management of Data*, ser. SIGMOD '19. New York, NY, USA: Association for Computing Machinery, Jun. 2019, pp. 141–158.

[18] Z. Peng, C. Xu, H. Wang, J. Huang, J. Xu, and X. Chu, "P^{2}B-Trace: Privacy-preserving blockchain-based contact tracing to combat pandemics," in *Proceedings of the 2021 ACM SIGMOD International Conference on Management of Data*, Xi'an, Shaanxi, China, Jun. 2021.

[19] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling Byzantine Agreements for Cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP '17. New York, NY, USA: Association for Computing Machinery, Oct. 2017, pp. 51–68.

[20] E. Syta, P. Jovanovic, E. K. Kogias, N. Gailly, L. Gasser, I. Khoffi, M. J. Fischer, and B. Ford, "Scalable Bias-Resistant Distributed Randomness," in *2017 IEEE Symposium on Security and Privacy (SP)*, May 2017, pp. 444–460, iSSN: 2375-1207.

[21] C. Li, P. Li, D. Zhou, W. Xu, F. Long, and A. Yao, "Scaling Nakamoto Consensus to Thousands of Transactions per Second," *arXiv:1805.03870 [cs]*, Aug. 2018, arXiv: 1805.03870.

[22] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "Blockbench: A Framework for Analyzing Private Blockchains," in *Proceedings of the 2017 ACM International Conference on Management of Data*, ser. SIGMOD '17. New York, NY, USA: Association for Computing Machinery, May 2017, pp. 1085–1100.

[23] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proceedings of the third symposium on Operating systems design and implementation*, ser. OSDI '99. USA: USENIX Association, Feb. 1999, pp. 173–186.

[24] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, Jul. 1982.

[25] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A System for Large-Scale Machine Learning," 2016, pp. 265–283.

[26] G. Cohen, S. Afshar, J. Tapson, and A. v. Schaik, "EMNIST: Extending MNIST to handwritten letters," in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 2921–2926, iSSN: 2161-4407.

[27] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Artificial Intelligence and Statistics*. PMLR, Apr. 2017, pp. 1273–1282, iSSN: 2640-3498.

[28] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of Neural Networks using DropConnect," in *International Conference on Machine Learning*. PMLR, May 2013, pp. 1058–1066, iSSN: 1938-7228.