You've got a Friend in ME (Mobile Edge): Blockchain Processing with Cloud Node Backup

Zane Karl[1]*, Hayden Freedman[1]*, Ahmad Showail[13], Abhishek Singh[1], Samaa Gazzaz[2], Faisal Nawab[1]
zkarl@uci.edu, hfreedma@uci.edu, ahmad.showail@kaust.edu.sa, abhishas@uci.edu, sgazzaz@ucsc.edu, nawabf@uci.edu
[1]Donald Bren School of Information and Computer Science, University of California, Irvine, Irvine, CA, 92697, USA
[2]Baskin School of Engineering, University of California, Santa Cruz, Santa Cruz, CA, 95064, USA
[3]Department of Computer Engineering, Taibah University, Madinah, Saudi Arabia

Abstract - **The growing ubiquity of Blockchain has led to the experimental transplantation of classical web applications into the Blockchain space. Notoriously, running computationally-expensive processes on-chain is less than ideal. Therefore, it is crucial to find ways to utilize the security and decentralization provided by Blockchain technology while reducing on-chain computational load. To that end, recent work has shown the benefit of Multi-stage Transaction architectures. The benefit of such systems is that they provide requested data quickly to the client while still ensuring that accurate results are eventually returned. In this paper, we describe a prototype of a multi-stage transaction system, which we call Friend in Mobile Edge (FiME), that intelligently coordinates client requests between a Blockchain and a central cloud node. We select the use case of stock price prediction and evaluate based on response time to the client as well as computational load on the various components. Our results involve an analysis of the performance of such a system in delivering timely results to the end user, as well as a discussion of the drawbacks of handling large data requests on-chain. We also include recommendations for how future systems can build on these findings.**

I. Introduction

Many modern computing applications require both intensive computation and rapid response time [1], [2], [3]; however providing both requires expensive, state-of-the-art hardware which in many cases may not be feasible. While modern cloud computing has been able to make high-powered computers more accessible to end users, cloud-based resources are usually centralized and thus can suffer from high latency, outages, and variability in performance depending on the geographic location of the client [4], [5], [6], [7]. Furthermore, computing on a single, centralized cloud node does not provide the capacity for distributed computing across a network, which can improve efficiency by optimizing resource allocation [8], [9].

One way to mitigate issues of cloud server latency and availability is to deploy multi-stage transactions with edge and cloud components [10]. The basis for this technique is rooted in the realization that not all client requests will require intense computation; some may be handled easily by an edge node without high-powered computing resources at its disposal, while others may require the high-powered computing that a cloud-based data center can provide. Such multi-stage transaction systems may operate with a network of edge nodes as the client-facing component; this architecture provides increased reliability due to decentralization of resources [11] as well as reduced latency compared to the centralized cloud computing model [12].

Previous work has focused on prototyping multi-stage transaction edge-cloud systems that can handle complex tasks such as image recognition [10], [13], [14]. However, we believe that such architectures can provide support for a wide variety of computational tasks in various domains. In this paper we describe a multi-stage transaction system called FiME, and explore an application of this system implemented to rapidly predict market price for a given stock some arbitrary number of days in the future. As an example, consider a day trader who wants to get a quick intuition about how the price of a stock will behave over the next few days. They submit a request to the system and receive a prediction nigh instantaneously, allowing them to act quickly. At the same time, their request is processed asynchronously by a cloud data center, which uses a more advanced model to make more accurate predictions. When it is available, an updated prediction from the cloud will be issued to the trader, allowing them to refine their behavior if necessary.

Current multi-stage transaction systems have generally not taken advantage of Blockchain technology [10]. FiME works to build off of previous work in multi-stage transactions by applying them to Blockchain Smart Contracts that wish to use off-chain resources. Our results indicate that this is plausible from a performance perspective and provides theoretical advantages such as immutable transaction records inherent in Blockchain technology [15].

Our prototype system is called Friend in ME (Mobile Edge), inspired by our Blockchain-based solution to multi-stage transactions. In our results, we simulate the average response time of FiME compared to systems that use only an edge network or only a cloud-based server to answer client requests. We also examine the average time for the cloud node to verify responses to clients based on the rate of requests received, as well as the performance improvement achieved by adding a cache mechanism to the Blockchain component further improve the response time to the client.

*These authors contributed equally to this work

The rest of this paper is structured as follows: Section II discusses related work, Section III provides relevant background, Section IV gives an overview of FiME, Section V considers FiME's design applied to stock prediction, Section VI defines our evaluative framework and displays the results of our empirical evaluation, Section VII discusses our findings in more detail and provides recommendations for future work, and Section VIII concludes and prefaces following work.

II. Related Work

Our implementation of a multi-stage transaction system is primarily motivated by CROESUS [10]. The CROESUS system introduces multi-stage processing for image recognition in video streams. The paper demonstrates the efficacy of coordinating requests between an edge and cloud node to provide rapid responses to client requests. In the first stage, the edge node responds to the client request the best that it can given limited computational resources. In the second stage, the edge response is asynchronously validated by the cloud node, which may then issue an apology and correction to the client if the new result is significantly different than the original response.

CROESUS was not the first to tackle the multi-stage computation setting. multi-stage transactions can really be viewed as special cases of Long-lived Transactions, such as the Sagas system [16]. The goal of Sagas is to facilitate the interleaving of sequences of long-lived transactions. Both our prototype system and CROESUS process transactions like these on different hosts, allowing for faster processing of initial transaction sections and the possibility of dedicating the entirety of an individual host's computational resources to the transaction assigned to it.

Additionally, there is a large body of prior research that relates to the coordination of edge and cloud nodes. For instance, Chen et al. [13] seek to provide accurate real-time image recognition using the cameras on mobile devices, by supplementing the computational power of the mobile device with dedicated server machines. Similarly, both Grulich and Nawab [14] and Kang et al. [17] work to improve the accuracy of Internet of Things (IoT) devices' abilities to perform complex machine learning computations by adding a computational aid in the form of dedicated cloud hosts.

There have also been numerous efforts to use Blockchain technology in conjunction with a network of edge devices. For example, Stanciu [18] proposes a Blockchain-based control system for distributing and coordinating work amongst a network of edge nodes. Abadi et al. [19] discuss a decentralized platform for unifying IoT devices that uses the Blockchain to maintain a global state and arbitrate disputes. Yang et al. [20] use the Blockchain to coordinate an edge-cloud network of Industrial IoT devices and servers, using a novel consensus algorithm to improve the efficiency of resource allocation. Kathirevelu et al. [21] note the growing prevalence of small-scale data centers at the edge, which can provide low-latency response times to clients, and propose a Blockchain-based framework to improve network service discovery and interoperability of network components. A more current system that uses multi-stage transactions in Blockchain is LiftChain [22], which builds a scalable NFT transaction protocol that first performs NFT transactions off-chain and eventually propagates them on-chain.

Building on related work, the research presented in this paper makes the following unique contributions. First, we explore the practicality of the CROESUS idea in the Blockchain space. Where CROESUS computes its first stage result on an edge node, we instead compute the result on the Blockchain, specifically within an Ethereum Smart Contract[*]. Second, we provide recommendations to further improve the practicality of this architecture in a real-world setting, which could be used to help create a framework for handling such transactions generally.

III. Background

This paper seeks to serve use cases that require complex computational analysis and also rapid response. In order to do so, we run a rapid, simple model on the Blockchain, and a more complex, more accurate model on a cloud-based node. These components coordinate in order to deliver rapid results to the client, as well as potential corrections to those results based on an asynchronous verification process. The resulting architecture, demonstrated by our prototype software, provides a computationally efficient mechanism for handling client requests that require intensive use of computational resources. It also takes advantage of the decentralization and security benefits inherent in Blockchain technology.

In this section, we will briefly review some of the fundamental concepts on which our paper is based, with the acknowledgement that space permits the inclusion of only a fraction of available discourse on these topics.

A. *Multi-stage Transactions*

Multi-stage transactions divide transactions into an initial stage and secondary stage commit. They may be implemented in order to allow for greater flexibility, maximum capacity, and/or efficiency in the flow of transactions. For instance, Wang et al. [23] show that their multi-stage Blockchain provides increased maximum capacity for the flow of transactions, as well as reduces storage requirements. In the case of CROESUS, multi-stage transactions consist of two stages: a fast and less accurate response, and a slower and more accurate response. [10]. The advantage of this transaction archetype is that it does not overextend computational resources in an effort to quickly perform intensive computations. Instead, the initial stage is executed by a lightweight device (perhaps an IoT device or a Blockchain node), and its result is asynchronously validated by a higher-power device once the device becomes available.

[*]Without loss of generality we use the Ethereum Blockchain in this paper due to its popularity and community support. We emphasize that the system itself is the focus of this report and the Blockchain used can be considered as a black box with the necessary infrastructure, liveness, and security guarantees.

B. *Ethereum Smart Contract*

An Ethereum Smart Contract is represented as an address on the Ethereum Blockchain, the programmatic implementation of which can be called when an auxiliary account executes a transaction with the contract address as the recipient. It provides an interface for a client to communicate with the Blockchain and execute transactions [24]. In our use case, the client queries the contract to ask for stock predictions and receives a prediction in return. The contract contains the methods needed to execute a simple Exponential Moving Average (EMA) calculation to get a prediction to give to the client. The cloud node also uses the Smart Contract interface to validate the results of the contract's EMA calculations, and then to asynchronously provide it updated stock prediction results.[*]

IV. System Overview

The FiME prototype system architecture consists of three primary components: 1) an (on-chain) Ethereum Smart Contract that executes a simple model to predict stock prices, 2) an (off-chain) cloud node that executes a more complex machine learning model to verify the results, and 3) a client program to initiate a series of requests. This prototype performs a simulation of a series of client requests and monitors the interactions between the various components of the system.

A. *Model Assumptions*

The client is assumed to be an edge-system with limited computational resources; for example, a laptop or desktop computer that makes requests and receives responses from the Blockchain. To the client, our system is a black box which they treat as an oracle that can predict future stock prices. The Ethereum Smart Contract is located at an address, or account, attributed to the Ethereum Blockchain hosted locally via the Ganache test network application [25].

Upon the receipt of client requests, the Smart Contract executes its code using the limited computational power of the Ethereum Virtual Machine (EVM). The cloud node is assumed to have computational power greater than or equal to that of the client. Though it would be to our system's benefit for the cloud node to have a large degree of computational power in comparison to the client, we make this assumption so as to give a lower bound to this system's capabilities as well as to more accurately represent the setting under which it will be tested and implemented.

B. *Security Model*

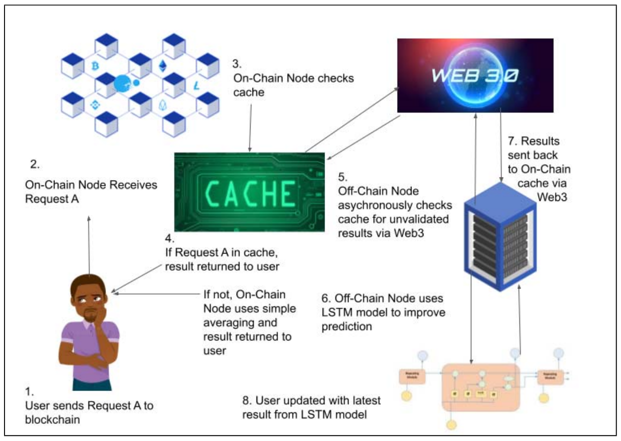
Our system's security model has a quite interesting property intrinsic to it by virtue of its reliance on Blockchain. Assuming a safe and correct implementation of the Smart Contract (that is, one that does not permit abuse due to programmer error) the entirety of the Blockchain "virtual node" may be safely assumed to be insulated from a malicious attacker. The fact that the Smart Contract's code would exist immutably on the Ethereum Blockchain inhibits any attacker's ability to modify its capabilities, and the assumption that it was implemented correctly allows us to conclude that it is not liable to some kind of query language injection. In essence, the Blockchain node is analogous to a Trusted Platform Module [26] or Trusted Execution Environment [27] in hardware/software, respectively.

V. System Design

The goal for our system is to provide a quick response to the client and then verify the validity of the response asynchronously. The Smart Contract receives stock prediction requests made by a client, and uses a subset of the data to simply take an average of the previous days' prices, which is returned to the client as the initial prediction. As a proxy for confidence of this prediction, we simply check the number of days in the future for which the prediction was requested. If the confidence value of the request processed by the Smart Contract exceeds a fixed confidence threshold, the request will be labeled as "pending" until it can be verified by the cloud node, which asynchronously polls the Smart Contract's cache to find results that require verification. The cloud node uses a pre-trained Long Short-Term Memory (LSTM) model to make more robust predictions, and its results are sent back to the Smart Contract. If the returned results differ significantly from the initial prediction, the Smart Contract issues an apology to the client and sends the revised results.

Figure 1 shows the workflow to which our prototype conforms. This section details each of the 8 steps of the workflow in detail. In step 1, a user sends Request A to the Blockchain. In step 2, the on-chain node receives Request A and prepares a response. In step 3, the on-chain node checks its cache to see if the requested ["stock", "day"] pair has recently been requested, and if so, it returns the result of the prior computation without having to re-process. If the data already exists in the cache, it is simply returned, and if not, the system proceeds to step 4. In step 4, the Smart Contract processes the request. The model accepts the two parameters ["stock", "day"] as input and uses them to predict the price of that stock on that day, using the EMA formula that employs a subset of the available training data. An initial response is returned by the model. The Smart Contract sends the predicted price to the client, thus fulfilling the promise of rapid response to Request A. In step 5, the edge node asynchronously polls the Smart Contract cache, checking for unverified results. It also checks against the "confidence" reported by the Smart Contract. Client requests that fall below the hard-coded confidence threshold are passed to the off-chain node for further processing. In step 6, the ML model running on the off chain node predicts the result for Request A. The prediction of the off-chain ML model is then sent back to the Etheureum Smart Contract and stored in the Smart Contract cache. In step 7, the off-chain ML model returns the result to the Smart Contract, which stores the result in its cache. The result is labeled as "verified". In step 8, the user is updated with the most accurate results, along with an apology if necessary.

[*]The cloud node is performing the data validation step in our system. To do so, it is querying the smart contract for previous results which the cloud node subsequently uses as validation input.



3.
On-Chain Node checks cache
WEB 3.0
2.
On-Chain Node Receives Request A
CACHE
5.
Off-Chain Node asychronously checks cache for unvalidated results via Web3
7. Results sent back to On-Chain cache via Web3
4.
If Request A in cache, result returned to user
If not, On-Chain Node uses simple averaging and result returned to user
6. Off-Chain Node uses LSTM model to improve prediction
1.
User sends Request A to blockchain
8. User updated with latest result from LSTM model

Figure 1. Workflow showing all high-level stages of the multi-stage transaction setup for Blockchain system with an off-chain node component. The process shows the handling of a user request which is then forwarded through the appropriate on-chain and off-chain components, after which a response is returned to the client.

VI. Evaluation

A. Experimental Setup

In order to determine the efficacy and performance of our system, we ran several experiments using a simulated client launching random requests.

In our experimental setup, the client, cloud node, and Blockchain all run on the same computer. It should be noted that due to this limitation, all performance results do not include network latency, which is an important caveat. In reality, such artifacts would exist when sending and receiving requests between separate computers.

B. Experimental results

1) Experiment 1: Response Time Comparison

In this experiment, the results of which are shown in Figure 2, we evaluate the response time of three variations of our system: 1) the Smart Contract's EMA executes without cloud node intervention/correction (red bar), 2) FiME executes with a confidence threshold such that 50% of requests are sent to the cloud node for correction (green bar), and 3) all requests are sent to the cloud node without use of the Smart Contract's EMA model (blue bar). Although exclusively using the Smart Contract's EMA model provides the fastest response time, this approach leads to very low accuracy results due to the simplicity of the model. FiME provides a better response time than the cloud node to return a verified result, assuming that we have selected an appropriate confidence threshold. Furthermore, we see that the FiME system's response time can be flexible based on the confidence threshold.

2) Experiment 2: Off-Chain Validation Lag

In this experiment, we varied the time between requests from our client to verify how the cloud node was able to keep up with demand. The cloud node takes about 15 seconds per prediction using its LSTM model. Figure 3 shows that when the time between client requests is small, the cloud node becomes overwhelmed and the average time until verification is very high. As expected, as the time between client requests increases, the average time for verification drops significantly. This finding indicates that creating a request limit based on the resources available to the cloud node could be helpful for a production system implementing this architecture.

3) Experiment 3: Off-Chain Validation Cache Hits

One of the primary obstacles in performing computation on the cloud node is speed. To that end, the Friend in ME system implements a caching mechanism in the Smart Contract to limit the number of calls to the cloud node. This modification also means that fewer client requests change the state on the Blockchain, which helps to reduce the average gas cost of a client request. As we can see from Figure 4, the client invariably requests the same data over time and having previously cached that data results in more stateless exchanges between the client and the Blockchain node.

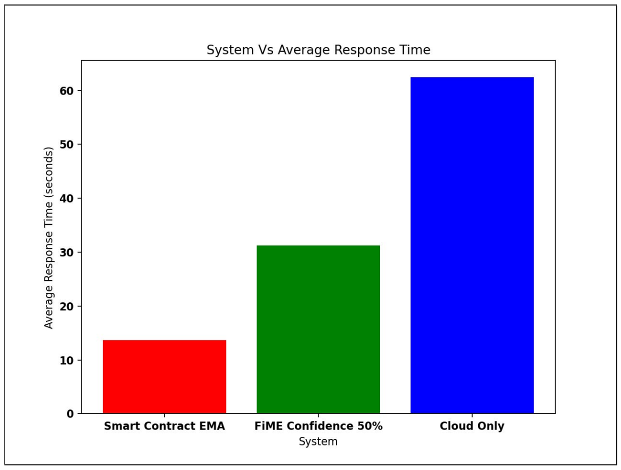


Figure 2. This bar graph shows the average response time of three variations of our system for stock prediction

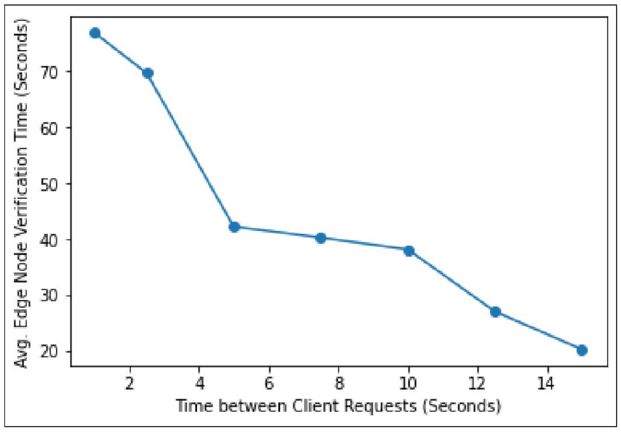


Figure 3. This graphic shows the average cloud node validation time as a function of the rate of requests from the client. When the client makes a new request every second, the average delay until the the results are verified by the cloud node is relatively high; however when the delay is 15 seconds, the cloud node can keep up with demand easily.

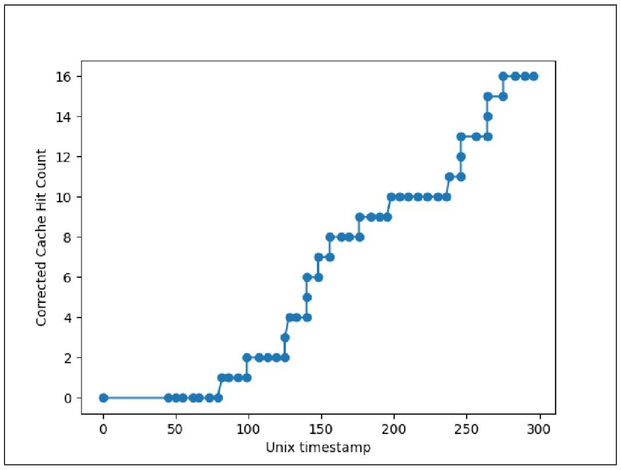


Figure 4. This graphic shows the utility of implementing a results cache in the Smart Contract. As time increases the number of client requests for a previously corrected value residing in the cache increases as well, reducing the load on the cloud node.

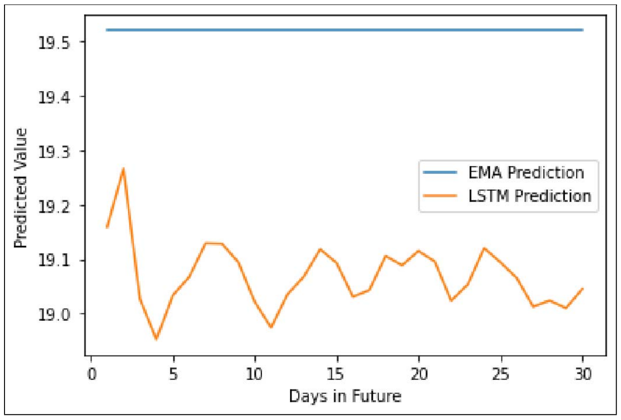


Figure 5. This graphic illustrates the differences between the on-chain EMA prediction and off-chain LSTM prediction for Hewlett-Packard Stock (HPQ).

4) *Experiment 4: EMA vs LSTM Predictions*

The goal of this experiment was to demonstrate the necessity of the off-chain corrective requests/apologies, by highlighting the difference in computational capacity of the Blockchain node and a cloud node. The on-chain EMA model is simple, and, as we can see from Figure 5, does not change its prediction after the first day in the future. The LSTM model was able to act on significantly more data than the EMA model, as well as incorporate more complex machine learning techniques, allowing it to predict stock prices with a much higher level of nuance than the EMA model.

VII. Discussion/Recommendations

The goal of this research is to evaluate the plausibility of a multi-stage transaction system using a Blockchain component in place of an edge node network. Our proof-of-concept demonstrates that such a system is viable; however, future systems should consider the drawbacks that we encountered with our approach. Based on this initial work, we make the following recommendations for ways that future systems using similar architectures could be improved.

One of the primary issues we encountered while implementing the FiME system was the high gas cost of storing and processing data in the Smart Contract. The Solidity programming language, in which the Smart Contract was written, has only rudimentary support for data processing, making it a somewhat unwieldy tool even for the simple data averaging technique that we deployed. Because the EMA technique was only a stand-in for a more useful model, any production-level system using FiME would need to deploy a significantly more complex predictive model within a Smart Contract implying higher gas usage. Because of this, our primary recommendation for future work is to limit the amount of data that will be pushed through the Smart Contract. This could potentially be accomplished by combining Blockchain with a network of edge nodes to which the Smart Contract could connect to and request tasks.

Another potentially useful paradigm for utilizing a Blockchain node as part of a multi-stage transaction is to compute results off-chain but store results in an on-chain cache. This would take advantage of the Blockchain's ability to immutably store transaction results while offloading the actual computation. These cached results could be stored via a more intelligent system than the in-contract storage used by our prototype. One idea in this direction is the incentives-based mechanism for data storage on the Blockchain described by Ren et al. [28]. Future work should examine the trade-offs between Blockchain-based cache storage, cost, and overall system performance.

There are a few limitations to our experimental approach that should be addressed in future iterations of this work. Since all of our evaluation occurred with just one client node, we are not sure how FiME would scale to multiple client nodes executing simultaneously. This would be important behavior to characterize as part of future work, since it is a more realistic scenario for how FiME could be used in production. Furthermore, future work interested in replicating or extending our approach should focus on incorporating latency into the performance evaluation, as operating nodes on different machines spread out across the globe is a more realistic scenario than all of the nodes operating on a single computer, as in our evaluation. The effects of network latency could even be tested by using servers that are close to each other and then progressively farther away. This would help give a better idea about the overall usability of the system.

VIII. Conclusion

The Friend in ME system illustrates a first step in using Blockchain technology to assist multi-stage transactions. FiME uses computation within an Ethereum Smart Contract to return rapid results to the user, while offloading the verification of results to a cloud node with more computing power. However, due to the difficulties inherent in ingesting, storing, and computing on data within a Smart Contract, we recommend that future iterations of this research offload work from the Smart Contract to an edge network, and use the on-chain component as a client interface, task delegator, and immutable cache of results. The immediate next step would be to implement FiME using geographically disparate end systems for each of the parties involved. This will better allow us to validate the practicality of the system at a more meaningful scale. Following that, or included as a part of it, we will next need to demonstrate a use case that more specifically takes advantage of the Blockchain's unique characteristics.

IX. Acknowledgement

This research is supported in part by the NSF under grant CNS-1815212.

References

[1] O. Villa, D. P. Scarpazza, and F. Petrini, "Accelerating real-time string searching with multicore processors," *Computer*, vol. 41, no. 4, pp. 42–50, 2008.

[2] D. Y. Parkinson, K. Beattie, X. Chen, J. Correa, E. Dart, B. J. Daurer, J. R. Deslippe, A. Hexemer, H. Krishnan, A. A. MacDowell *et al.*, "Real-time data-intensive computing," in *AIP Conference Proceedings*, vol. 1741, no. 1. AIP Publishing LLC, 2016, p. 050001.

[3] R. T. Kouzes, G. A. Anderson, S. T. Elbert, I. Gorton, and D. K. Gracio, "The changing paradigm of data-intensive computing," *Computer*, vol. 42, no. 1, pp. 26–34, 2009.

[4] M. Ryden, K. Oh, A. Chandra, and J. Weissman, "Nebula: Distributed edge cloud for data intensive computing," in *2014 IEEE International Conference on Cloud Engineering*. IEEE, 2014, pp. 57–66.

[5] S. M. Habib, S. Ries, and M. Muhlhauser, "Cloud computing landscape and research challenges regarding trust and reputation," in *2010 7th International conference on ubiquitous intelligence & computing and 7th international conference on autonomic & trusted computing*. IEEE, 2010, pp. 410–415.

[6] L. Tawalbeh, N. Alassaf, W. Bakheder, and A. Tawalbeh, "Resilience mobile cloud computing: features, applications and challenges," in *2015 Fifth International Conference on e-Learning (econf)*. IEEE, 2015, pp. 280–284.

[7] A. Chandra, J. Weissman, and B. Heintz, "Decentralized edge clouds," *IEEE Internet Computing*, vol. 17, no. 5, pp. 70–73, 2013.

[8] H. Hussain, S. U. R. Malik, A. Hameed, S. U. Khan, G. Bickler, N. Min-Allah, M. B. Qureshi, L. Zhang, W. Yongji, N. Ghani *et al.*, "A survey on resource allocation in high performance distributed computing systems," *Parallel Computing*, vol. 39, no. 11, pp. 709–736, 2013.

[9] Y.-H. Jia, W.-N. Chen, T. Gu, H. Zhang, H.-Q. Yuan, S. Kwong, and J. Zhang, "Distributed cooperative co-evolution with adaptive computing resource allocation for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 188–202, 2018.

[10] S. Gazzaz, V. Chakraborty, and F. Nawab, "Croesus: Multi-stage processing and transactions for video-analytics in edge-cloud systems," *arXiv preprint arXiv:2201.00063*, 2021.

[11] J. Zhang, S. Zhong, T. Wang, H.-C. Chao, and J. Wang, "Blockchain-based systems and applications: a survey," *Journal of Internet Technology*, vol. 21, no. 1, pp. 1–14, 2020.

[12] M. Tahir, M. H. Habaebi, M. Dabbagh, A. Mughees, A. Ahad, and K. I. Ahmed, "A review on application of blockchain in 5g and beyond networks: Taxonomy, field-trials, challenges and opportunities," *IEEE Access*, vol. 8, pp. 115 876–115 904, 2020.

[13] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, "Glimpse: Continuous, real-time object recognition on mobile devices," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, 2015, pp. 155–168.

[14] P. M. Grulich and F. Nawab, "Collaborative edge and cloud neural networks for real-time video processing," *Proceedings of the VLDB Endowment*, vol. 11, no. 12, pp. 2046–2049, 2018.

[15] B. S. White, C. G. King, and J. Holladay, "Blockchain security risk assessment and the auditor," *Journal of Corporate Accounting & Finance*, vol. 31, no. 2, pp. 47–53, 2020.

[16] H. Garcia-Molina and K. Salem, "Sagas," *ACM Sigmod Record*, vol. 16, no. 3, pp. 249–259, 1987.

[17] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Computer Architecture News*, vol. 45, no. 1, pp. 615–629, 2017.

[18] A. Stanciu, "Blockchain based distributed control system for edge computing," in *2017 21st International Conference on Control Systems and Computer Science (CSCS)*. IEEE, 2017, pp. 667–671.

[19] D. Abadi, O. Arden, F. Nawab, and M. Shadmon, "Anylog: a grand unification of the internet of things," in *Conference on Innovative Data Systems Research (CIDR '20)*, 2020.

[20] F. Yang, J. Tian, T. Feng, F. Xu, C. Qiu, and C. Zhao, "Blockchain-enabled parallel learning in industrial edge-cloud network: a fuzzy dpost-pbft approach," in *2021 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2021, pp. 1–6.

[21] P. Kathiravelu, Z. Zaiman, J. Gichoya, L. Veiga, and I. Banerjee, "Towards an internet-scale overlay network for latency-aware decentralized workflows at the edge," *Computer Networks*, vol. 203, p. 108654, 2022.

[22] H. Chaparala, S. Doddala, A. Showail, A. Singh, S. Gazzaz, and F. Nawab, "Liftchain: A scalable multi-stage nft transaction protocol," in *First International Symposium on Recent Advances of Blockchain Evolution: Architecture, Intelligence, Incentive, and Applications (BlockchainEvo)*, 2022.

[23] J. Wang, S. Fan, A. Alexandridis, K. Han, G. Jeon, Z. Zilic, and Y. Pang, "A multistage blockchain-based secure and trustworthy smart healthcare system using ecg characteristic," *IEEE Internet of Things Magazine*, vol. 4, no. 3, pp. 48–58, 2021.

[24] S. Wang, Y. Yuan, X. Wang, J. Li, R. Qin, and F.-Y. Wang, "An overview of smart contract: architecture, applications, and future trends," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 108–113.

[25] W.-M. Lee, "Testing smart contracts using ganache," in *Beginning Ethereum Smart Contracts Programming*. Springer, 2019, pp. 147–167.

[26] W. Arthur, D. Challener, and K. Goldman, "History of the tpm," in *A Practical Guide to TPM 2.0*. Springer, 2015, pp. 1–5.

[27] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: what it is, and what it is not," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1. IEEE, 2015, pp. 57–64.

[28] Y. Ren, Y. Liu, S. Ji, A. K. Sangaiah, and J. Wang, "Incentive mechanism of data storage based on blockchain for wireless sensor networks," *Mobile Information Systems*, vol. 2018, 2018.