Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 53 (2021) 594–605

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

49th SME North American Manufacturing Research Conference, NAMRC 49, Ohio, USA

Hybrid Blockchain Architecture for Cloud Manufacturing-as-a-service (CMaaS) Platforms with Improved Data Storage and Transaction Efficiency

Mahmud Hasan[a], Kemafor Ogan[b], Binil Starly[a]*

[a]*Edward P. Fitts Department of Industrial and Systems Engineering, North Carolina State University, Raleigh, NC 27695, United States of America.*
[b]*Department of Computer Science, North Carolina State University, Raleigh, NC 27695, United States of America*

* Corresponding author. Tel.: +1 919 515 1815; fax: +1-919-515-5281. *E-mail address:* bstarly@ncsu.edu.

Abstract

Blockchain based decentralized Cloud Manufacturing-as-a-Service (CMaaS) platforms enable customers to gain access to a large capacity of manufacturing nodes over cryptographically secure networks. In recent times, the Ethereum network has emerged as a popular blockchain framework for providing provenance and traceability of proprietary manufacturing data in decentralized CMaaS. However, the Ethereum ecosystem was only designed to store and transmit low volume financial transaction data and little has been done to make it an efficient repository of large manufacturing data streams in CMaaS systems. In this paper, the authors build on their previous work and report the design, implementation, and validation of middleware software architectures that allow Ethereum based distributed CMaaS platforms to harness the benefits of the secure asset models of the Ethereum ecosystem and the immutable big data storage capabilities of the decentralized BigchainDB database platform. A novel hybrid blockchain architecture enabled by efficient communication protocols and blockchain oracles is proposed. This architecture allows the transfer and immutable storage of large manufacturing data streams onto global BigchainDB nodes allowing data rich manufacturing transactions to bypass the transaction fees of the Ethereum ecosystem. Additionally, a machine learning based time series inference model is proposed which enables the forecast of Ethereum gas price into the future. This allows the CMaaS platform smart contracts to judiciously assign gas price limits and hence save on transactions ensuing from transfer or creation of assets. The outcomes of this research show that the designed hybrid architecture can lead to the reduction of significant number of computational steps and hence transaction fees on Ethereum by offloading large volume data onto BigchainDB nodes. A Random Forest regressor based time series inference model has been shown to exhibit superior performance in the prediction of Ethereum gas price, that allows the CMaaS to avoid executing transactions in periods of high gas prices within the Ethereum ecosystem.

© 2021 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)
Peer-review under responsibility of the Scientific Committee of the NAMRI/SME

Keywords: blockchain; smart contracts; distributed database; time series; regression; lstm; gas price prediction.

1. Introduction

Recent trends in manufacturing suggest that modern manufacturing systems are increasingly becoming digitalized and gradually moving towards cloud manufacturing ecosystems [1]. Connectivity enabled by the presence of highly scalable and networked architectures [2] and the rise of Cyber-Physical Systems (CPS) [3] are playing an important role into transforming conventional manufacturing into Cloud Manufacturing-as-a-Service (CMaaS) platforms [4] wherein the assimilation and the subsequent deployment on the cloud, of hard and soft manufacturing resources have immense potentials. This also promises to allow the democratization of manufacturing services where consumers can come up with highly customized products reflecting their individual needs [5]. Modern cloud manufacturing platforms like Xometry [6], CloudNC [7] etc. now have the capability to cater to evolving customer requirements and can provide request for quote replies within minutes.

1.1. CMaaS and digital threads: issues and concerns

The development of Digital Thread (DT) centric manufacturing system architectures has led to the adoption

2351-9789 © 2021 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)
Peer-review under responsibility of the Scientific Committee of the NAMRI/SME
10.1016/j.promfg.2021.06.060

CMaaS platforms [8]. A typical DT in a broad CMaaS organization would span across multiple stakeholders and intermediary agents. This thread would expose to all participating entities digitalized manufacturing data right from the inception of a product till its release. However, the collection of this data throughout the product lifecycle has opened business, legal and privacy concerns. Competing manufacturing service providers are often concerned about how their proprietary product and manufacturing data are shared over highly networked CMaaS platforms and DTs [9]. A key concern of manufacturing entities on CMaaS platforms is over modalities of data security, privacy and automated contract negotiation. It spans primarily over the broad issues of cyber-threats, data theft, loss of intellectual property etc. [10]. Existing gaps in the construction of a cyber-threat resistant DT originates from the fact that most of these participating agents would have disparate and incompatible middleware stacks and this lack of standardization often lays bare major points of attrition in the overall security framework [9]. Many extant DT architectures for CMaaS platforms today would alarmingly not even adequately satisfy the basic security controls laid out in the National Institute of Standards and Technology (NIST) standard for security and privacy controls for federal information systems and organizations [11]. Another issue is the ownership and provenance of data traversing through these platforms. Contemporary CMaaS platforms are primarily centralized systems wherein the right of ownership and onus of provenance lies concentrated into the control of the central entity managing the CMaaS. The problem with this paradigm is in the inherent lack of trust of participating agents over the centralized control of proprietary data and a pervasive lack of universal consensus over election of authority.

1.2. Blockchain based CMaaS platforms

To address many of the standardization issues of DT's in contemporary CMaaS platforms, the NIST through its "Digital Thread for Smart Manufacturing Project" has been actively involved in the development of systems and protocols for DT's in manufacturing, design and allied sub-processes [12]. There have also been attempts to formulate models for better representation and transfer of manufacturing product data across DT's [13] that are encapsulated under different standard protocols like the STEP AP242 standards for manufacturing information exchange [14], or the use of Quality Information Framework standards for the exchange of metrology data [15]. Commensurately, a lot of research has also been invested into solving the issues surrounding manufacturing data integrity, ownership, provenance, and contract negotiation in CMaaS platforms. A very recent yet revolutionary technology called the Blockchain has come as a promising contender for solving such issues in inherently trustless networks of CMaaS DTs [16, 17]. This is also bolstered from NIST's vision about the impact of blockchains in manufacturing wherein it is focused on not only its ability to enable tamper proof transmission of data but also due to its unparalleled ability in providing seamless traceability in these trustless networks [18]. Blockchains are a network of computer or digital manufacturing nodes, that are a part of a distributed, decentralized, universally synchronized and cryptographically intact data ledger, composed of chained blocks that can be validated by all the nodes participating in the network [19]. Blockchain based CMaaS platforms have the potential of reducing costs in supply chain and logistics by more than 15% through improving security and removing paperwork [20]. The advantage of decentralized blockchain based systems over centralized database management systems comes from the fact that centralized systems are centrally managed. As in decentralized systems, there are no algorithms and protocols that can ensure provenance and traceability in centralized systems. A centralized database management system provides a single point of attack for cyber threats making it a much less trustworthy alternative for establishing data intensive CMaaS platforms. Investigations into product data models and blockchain based CMaaS and CPS architectures are already present in literature to act as guidelines for convenient blockchain adoption in DT [21].

1.3. Decentralized CMaaS based on the Ethereum ecosystem

Many different contemporary blockchain platforms have been demonstrated to be apt for housing decentralized CMaaS platforms and DTs through numerous research and implementation use cases. Global permission-less blockchains like Ethereum [22], or distributed operating systems for permissioned blockchains like the Hyperledger [23] have been shown to perform quite effectively in maintaining the data security, privacy, provenance, traceability of products and information in a manufacturing supply chain typical in CMaaS platforms [24]. The Ethereum platform has by far been one of the most popular [25] and highly standardized platforms for the deployment of decentralized apps [26] through the implementation of smart contracts [27] which can be used to stringently regulate security issues and automate contract negotiations. The Ethereum platform also provides an efficient and highly secure fungible asset transfer model through the implementation of its ERC-20 standard token model that forms the basis behind the digital cryptocurrency 'Ether' [28]. The security and space constraints inherent in the statically typed smart contract coding protocols of Ethereum makes sure that any transfer of fungible assets occurs in purely deterministic fashion, which is otherwise not guaranteed by the dynamically typed coding protocols of other blockchain platforms. The Ethereum ecosystem also allows for the modelling of data and non-fungible tokens through the implementation of the ERC-721 [29] token standard and this has been the de facto standard for the construction of data models of manufacturing assets on blockchain based CMaaS platforms. Contemporary Ethereum based implementations of decentralized CMaaS DT and product models fall short in several ways: (i) While such systems do benefit from the adoption of the standardized and secure fungible and non-fungible asset transfer models of the Ethereum ecosystem, the sheer magnitude of data originating from manufacturing nodes cannot be efficiently stored on the Ethereum platform since blockchains are non-ideal data storage solutions. The computational complexity and inefficiency of data storage arising from the Proof of Work [30] mining protocol of the Ethereum blockchain makes data storage an expensive affair, both computationally and financially. (ii)

CMaaS platforms based on the Ethereum ecosystem run on seamless inter-communication of autonomous smart contracts. Any exchange of data or addition of data to the Ethereum Virtual Environment (EVM) [31] requires the participating nodes in the blockchain that are invoking functions from the smart contracts pay gas fees (gas is a unit of computation on the EVM) for executing commands on the EVM realized through transactions [32]. Therefore, transactions cannot be indiscriminately executed on the EVM. However, generation of high frequency large volume data transfer events is common in large CMaaS organizations. Therefore, there is a need of an efficient, intelligent, and autonomous system that would be able to judiciously decide in real time when to execute cost-effective and computationally efficient Ethereum transactions over the EVM.

In this paper, the authors, as an improvement of their previous work, propose the design, implementation, and validation of middleware software architectures and blockchain oracles that allow the transfer of large volume manufacturing data onto decentralized and immutable big data storage platforms like the BigchainDB [33]. BigchainDB is a distributed database with decentralized control and provides immutability feature of recorded data. A hybrid blockchain architecture comprising of the Ethereum network and the BigchainDB platform is proposed. In this hybrid architecture, a parallel BigchainDB global distributed network runs along the more public Ethereum chain. The BigchainDB global database network records high volume immutable data originating from manufacturing nodes. The authors also propose the design and validation of a machine learning based, new inference middleware, that enables the forecast of Ethereum gas price into the future based on historical time series data of Ethereum gas prices. The goal is to allow the CMaaS platform smart contracts to judiciously assign gas price limits on transactions ensuing from transfer or creation of assets on the blockchain so that Ethereum transactions executed by the smart contracts are more efficient and economical.

This paper is organized as follows: section 2 presents a brief overview of the existing work in recent literature and elaborately describes the prior work of the authors that this paper is based on. Section 3 outlines the design and implementation of the hybrid blockchain architecture that has been proposed. Section 4 delineates the design and implementation of the new machine learning based inference middleware used for Ethereum gas price forecasts. Section 5 describes the results and outcomes achieved from the research in this paper. Conclusions, limitations and future research directions are laid out in section 6.

2. Related work

There has been considerable research in the past on blockchain based decentralized manufacturing systems. Lee et. al [21] in their research, have proposed a three level blockchain architecture for the construction of highly connected cyber physical production systems. Mandolla et. al [34] have explored the application of blockchain for CPS integrated additive manufacturing for the highly regulated aircrafts parts industry. In a case study of blockchain in manufacturing, comprising of real manufacturing nodes, the authors Angrish et. al [35] have introduced the “FabRec” model which comprises a consortium of manufacturing and computer nodes, autonomously communicating and negotiating with each other by means of computer coded smart contracts. Most of the aforementioned research has proposed important architectures and data models for blockchain based cloud manufacturing infrastructures. However, there have been significant research gaps in extant literature. Lack of proper data, communication models and appropriate implementation details for digital assets and smart contracts are major sectors that need significant investigation. Fungible asset transfer models that are needed to model assets of monetary value is another sector where current research does not put appropriate focus on. Appropriate security and platform consideration required for blockchain based manufacturing systems and appropriate middleware that allows seamless communication of different parts of the system, are all thrust areas where not much work has been done in the past. Many of the existing research have also been found to be limited in their proposal of security measures. These measures are integral in the design of decentralized CMaaS that deal with assets of monetary value. They also do not demonstrate implementation of any case study on any known, industry grade, fault-tolerant public or consortium blockchain network.

To address some of these knowledge gaps, the authors presented a decentralized CMaaS platform architecture in their previous research [4]. The architecture presents the design and implementation of several manufacturing and blockchain middleware, an efficient object-oriented design pattern for smart contracts in constrained Ethereum environments, an ERC-721 standard digital asset for manufacturing product data models and a novel algorithm to improve security of fungible assets on CMaaS platforms. The CMaaS platform architecture proposed by the authors is primarily based on the Ethereum ecosystem to take advantage of a standardized infrastructure with a relatively stringent fungible asset transfer model that is otherwise not present in other blockchain platforms. Conventional permissioned blockchain platforms unlike the Ethereum EVM would allow the construction of smart contract codes in Turing complete [36], high level dynamically typed languages as opposed to the statically typed Solidity language [37] in Ethereum. The use of high-level, dynamically typed programming to encode blockchain logic as opposed to using Turing-incomplete (Bitcoin) scripting or highly secure Turing-complete (Ethereum) scripting is a risk factor for CMaaS platforms since it makes the process outcomes of code execution in the blockchain ecosystem less deterministic.

Although blockchain platforms like Ethereum can assist in the avenues of data integrity and provenance, constrained environments of the Ethereum ecosystem ensuing from its security considerations render them as non-ideal data storage solutions particularly in relation to the sheer magnitude of data being produced by manufacturing nodes on decentralized CMaaS platforms. This was a limitation of the proposed platform pointed out by the authors in their work [4]. The importance for the encapsulation of big data in manufacturing is immense and subsequent mining on this large volume of data allows for the extraction of important analytics that can be harnessed for important business and logistic decisions. The limitations of the Ethereum ecosystem also renders it incapable of encapsulating important product life cycle data that can be harnessed for better traceability. Bonnard et. al [38] have shown in their research a hierarchical object-oriented model for digital chains in additive manufacturing platforms. The authors propose the model for better encapsulation of closed-loop manufacturing data that are generated in typical additive manufacturing processes. Lu et. al [13] in their research have demonstrated the importance of capturing service-oriented product data through models that can efficiently capture changing requirements of downstream production activities. Lu and Xu [39] have proposed a generic system architecture for the integration of cloud-based manufacturing equipment and big data analytics required for on-demand, near real-time manufacturing services. Therefore, it is evident that state of the art research and on demand cloud manufacturing processes on contemporary CMaaS platforms would be in increasing need of big data storage and analysis capabilities.

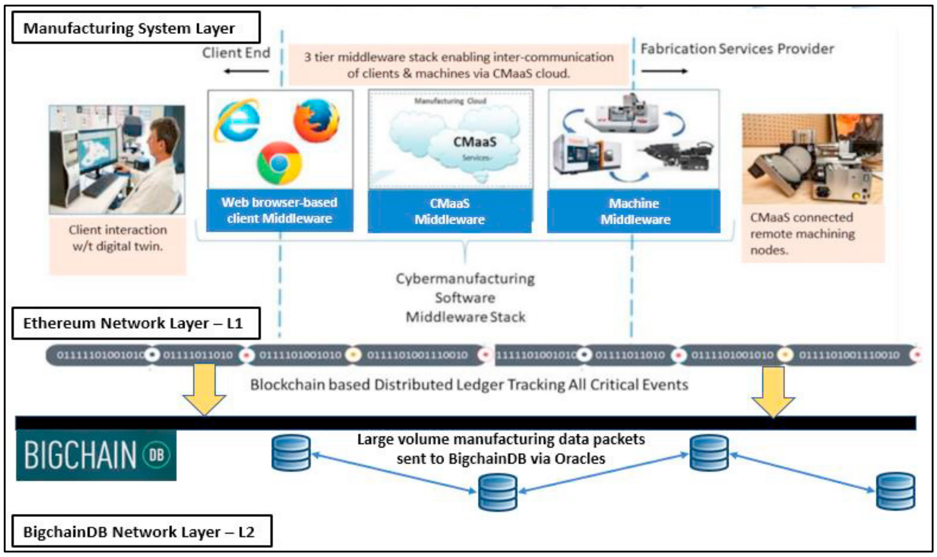
A potential solution to the dilemma of the constrained storage capabilities of the Ethereum ecosystem is to relinquish it for permissioned blockchain platform solutions like the Hyperledger. The Hyperledger platform allows the construction of smart contract codes in high level, dynamically typed languages that are more prone to non-deterministic outcomes and hence pose more security pitfalls that can be a concern to CMaaS stakeholders. Other smart contract based blockchain platforms could also have come as solution to this problem. Smart Contract platforms like NEM [40] that comes with the Proof of Importance consensus protocol is a promising candidate but its reliance on the extent of investment or stake as a measure of credibility of a mined transaction is postulated to affect small and medium enterprises on a decentralized CMaaS adversely. Similarly, blockchain platforms like Eosio [41, 42] claims to have eliminated transaction fees. They limit smart contracts by only devoting a small amount of computational time devoted to them. This is postulated to affect large enterprise smart contracts requiring more computational steps of execution adversely since now verbose smart contracts will always be rejected by the blockchain platform. Many existing major, decentralized cloud manufacturing platforms are based on the public Ethereum ecosystem or its variants and hence the decision for forsaking the platform for better data storage and big data capabilities while jeopardizing security and standardization is not justifiable. In view of all the mentioned limitations and roadblocks, there is therefore a: (i) need of the evolution of a new, universal blockchain architecture that is able to cater to the big data storage requirements of any existing blockchain based CMaaS platform without the need of any major architectural changes to an existing infrastructure. (ii) need of a time and cost efficient blockchain database query service. Many read, query and write operations in the Ethereum global database are carried out in the form of transactions which have to be mined. Mining of these transactions is time consuming and requires the transfer of fungible assets to the miners in a public blockchain network, demanded as a part of the consensus protocol. This would cause unnecessary financial depletion for a large CMaaS platform with millions of write and query operations taking place on its data. Therefore, there is a need of intelligent, preferably autonomous decision-making protocols within the CMaaS platform that would be able to issue transactions in a cost effective and efficient manner.

In this paper, the authors have attempted to address these outstanding issues pertaining to decentralized CMaaS platforms. The contribution of this paper is: (i) to propose a hybrid blockchain architecture composed of an Ethereum backbone that manages data integrity, asset provenance, contract negotiation and fungible asset transfer through its secure, public network as mentioned in section 1.3. The second element in the hybrid architecture is a concurrent BigchainDB global database that communicates seamlessly with the Ethereum backbone through the gateways established by a designed middleware. The function of the middleware is to get activated by outbound oracles initiated by data storage event signals from the Ethereum backbone. This eventually leads to the storage of big data pertaining to manufacturing product specifications onto a decentralized, immutable BigchainDB database, transactions on which do not require any transfer of fungible assets, making it economical by default. The second contribution of this paper is: (ii) to design and validate a machine learning based inference middleware that would be able to predict and infer Ethereum gas price into the future to a reasonable degree of accuracy. In their previous work cited in [4], the authors deployed CMaaS smart contracts with default allowable gas price limits which were high enough to allow any transaction from the contracts occurring within the Ethereum ecosystem to get readily mined. To improve the efficiency of these transactions, the authors have proposed the architecture of an inference middleware that forecasts Ethereum gas price based on historical time series data using machine learning models. The underpinning idea is to enable the CMaaS platform to automatically conserve resources in terms of gas fees paid for mining. The inference model forecasts possible allowable average gas price for a certain transaction event in the future. This average price is then broadcasted along with the transaction onto the Ethereum network. The conjecture is that, by continuously broadcasting low enough gas price limits, the inference middleware allows to save costs associated with gas fees for Ethereum transactions. It does this by making sure that the limit is not too high that it is above the average gas price limit of a certain day leading to monetary losses, nor is the limit too low that the transaction is rejected by the network of miners and is not preferentially mined.

Machine learning based Ethereum transaction gas price prediction models have been looked into in past literature. Liu et. al. [43] have explored the application of a suite of machine learning regression-based prediction models to predict the lowest gas price in the next block of transaction being mined within the Ethereum blockchain. To motivate the requirement to accurately predict future Ethereum gas prices, the authors have performed cursory analysis on the data collected over one day's worth of mined blocks in the Ethereum EVM. The authors were able to show that, accurate prediction models had the potential of saving costs as high as around $60 per block in fiat currency. In their proposed approach [43], the authors have evaluated few gas price influencing factors namely block difficulty, block gas limit, transaction gas limit, ether price, miner reward etc. to identify how these features affect transaction gas price by assigning them as input features to a machine learning model. While such an approach have led to reasonably accurate gas price prediction models as quoted by the authors, the limitation of the approach lies in the fact that a real time system implementing such models would need to know accurate values of these influencing factors in real time to decide on the gas price limit on the next transaction. This is often not a practical proposition since features like block difficulty level or miner reward are subject to stochastic variations or non-deterministic events like blockchain hard forks [44] or protocol changes. The authors in this paper have on the other hand, used time series machine learning models and trained them on actual, historical Ethereum gas price data to come up with prediction models that can predict future average gas price. The advantage of this approach is that it does not require the accurate knowledge of a plethora of EVM variables and relies on much simpler, one dimensional time series data to make predictions to a reasonable degree of accuracy.

3. Hybrid blockchain architecture for improved data storage capabilities

Figure 1 shows the improved version of the hybrid decentralized CMaaS platform architecture adopted from the previous work of the authors. For a detailed explanation of the components of the system, the reader is referred to the work cited in [4]. The system shows a top physical manufacturing system layer showing important stakeholders and data flow processes that interact with a product. The entire CMaaS system hosts parametrically configurable consumer or industrial parts which customers want to customize and manufacture. The process starts with a client modifying the parameters of the digital twin of a product hosted on the CMaaS platform through a front-end app on a web browser. When the client is satisfied with the final parameters of the altered digital twin of the product, a Request for Quote (RFQ) command can be initiated. This is when the client middleware shown in Figure 2, starts making Remote Procedural Calls (RPC) to the functions encoded within a manufacturing system smart contract that has been deployed on the Ethereum network layer – L1, represented by the second layer below the manufacturing system layer in Figure 1. Depending on the specifications of the part sent to the blockchain layer, functions from within the smart contract return a quote to the client. On approval of this quote, the client can initiate a make order through the client middleware. This information is duly sent to the Ethereum network and a make order event is recorded on the blockchain. This also starts a sequence of blockchain events that the client middleware subscribes to. The trigger of a make order event generated from the Ethereum blockchain layer automatically prompts the client middleware to send off toolpath regeneration commands to the CNC toolpath regeneration engine hosted by the CMaaS platform. This is where the CMaaS middleware takes over and another session of client and CMaaS middleware communication leads to a regenerated toolpath that captures the customized design of the client. Details on the internal components of the CMaaS, manufacturing middleware and intricacies of the implemented manufacturing system smart contracts are not focus of this research and is delineated in fine detail in reference [4]. Once the custom part is successfully manufactured, a complete takeover of the blockchain network layer takes place. The end of the machining process is recorded on the blockchain as an event which eventually leads to the creation of an ERC-721 token representation of the part on the blockchain network. The blockchain network also takes care of the automatic payment of fees by the client due to the manufacturing service availed and autonomously regulates issues relevant to refunds and returns via functions encoded within the smart contracts of the manufacturing system. The contribution of this research into converting the CMaaS platform into a hybrid blockchain architecture can be seen from the new, bottom most layer L2 in Figure 1. This layer, aptly named the BigchainDB network layer comprises of a parallel consortium of global nodes of computers executing instances


Manufacturing System Layer
Client End
3 tier middleware stack enabling inter-communication of clients & machines via CMaaS cloud.
Fabrication Services Provider
Manufacturing Cloud
CMaaS
Services
Client interaction w/t digital twin.
Web browser-based client Middleware
CMaaS Middleware
Machine Middleware
CMaaS connected remote machining nodes.
Cybermanufacturing Software Middleware Stack
Ethereum Network Layer – L1
Blockchain based Distributed Ledger Tracking All Critical Events
BIGCHAIN DB
Large volume manufacturing data packets sent to BigchainDB via Oracles
BigchainDB Network Layer – L2
Fig. 1. Improved Hybrid architecture of decentralized CMaaS with BigchainDB layer for big data storage.

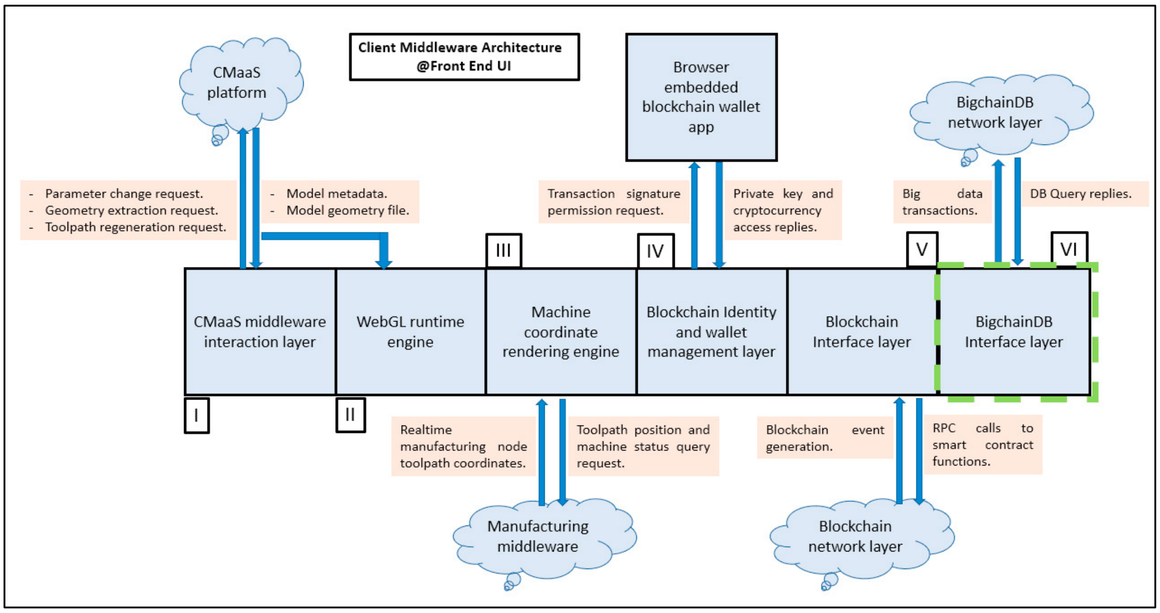

Client Middleware Architecture @Front End UI
CMaaS platform
Browser embedded blockchain wallet app
BigchainDB network layer
- Parameter change request.
- Geometry extraction request.
- Toolpath regeneration request.
- Model metadata.
- Model geometry file.
Transaction signature permission request.
Private key and cryptocurrency access replies.
Big data transactions.
DB Query replies.
I
II
III
IV
V
VI
CMaaS middleware interaction layer
WebGL runtime engine
Machine coordinate rendering engine
Blockchain Identity and wallet management layer
Blockchain Interface layer
BigchainDB Interface layer
Realtime manufacturing node toolpath coordinates.
Toolpath position and machine status query request.
Blockchain event generation.
RPC calls to smart contract functions.
Manufacturing middleware
Blockchain network layer

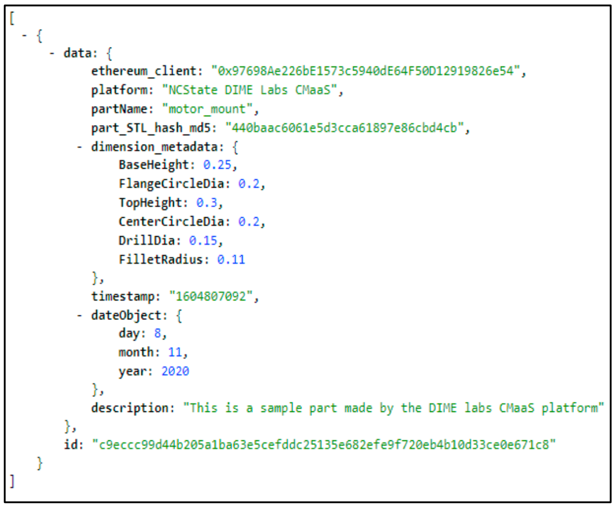
Fig. 2. Improved client middleware with BigchainDB interface.

of decentralized, distributed, immutable BigchainDB database servers. Communication between layers L1 and L2 is established by a new addition to the client middleware shown in Figure 2 section VI as the BigchainDB interface layer. This layer is primarily a middleware software module encoded in JavaScript. This layer resides within the client middleware software architecture and houses event subscribers that subscribe to specific transaction events. These transaction events are emitted by the oracles designed as a part of the CMaaS smart contracts. The event of the completion of the manufactured part issues transactions on the Ethereum network. The oracles on issuance of these transactions in turn emit their own events which trigger subscribers in the BigchainDB layer to complete the collection of data. The BigchainDB layer collects large volume metadata about the part as it keeps on getting manufactured. This information includes a cryptographically hashed signature of the design file of the final part, detailed dimensional metadata of the part and necessary information about both the client and the CMaaS platform in terms of their Ethereum identities i.e., wallet addresses.

It is needless to mention that, without the presence of this newly added BigchainDB network layer L2, storage of this complex set of information pertinent to product and manufacturing data would have been a gargantuan task so far as the Ethereum blockchain database is concerned. Most of this information quite naturally would involve data of different types and precision. Due to the restrictive nature of the Ethereum ecosystem, there is no default support of many complex datatypes like variable length strings which have to be used to record modalities like product name or description. Even if there were support of these data types, the low storage capabilities of the Ethereum ecosystem would not have allowed the capture of sufficient product information. Additionally, immutable registration of these information in the form of Ethereum transactions would have led to significant monetary contributions in terms of payable transaction fees for miners.

The logical conclusion to this imminent problem is to store complex, type and precision diverse information about a product and its manufacturing process on a database platform that provides much larger degree of freedom as far as storage space is concerned. However, at the same time, a compromise on the immutability and security of the stored data cannot be made and this makes the decentralized, distributed BigchainDB database layer an ideal solution to this problem. In addition to storing complex information in an immutable fashion, the BigchainDB layer L2 also provides highly efficient, low latency query functionalities to read from the database as is common in many conventional centralized database platforms. The communication and information handshake between layers L1 and L2 in Figure 1, is enabled by the new BigchainDB interface layer in Figure 2 section VI of the improved client middleware. The completion of the manufacture of a part ordered by the client is represented by the creation of an ERC-721 token on the Ethereum network L1, and this in turn initiates outbound Ethereum oracles encoded as smart contracts [45]. This eventually triggers communication between L1 and L2 layers via the new interface in the client middleware. Large volume, complex data relevant to the product is then sent to L2 layer to be stored securely and immutably. This is how, the Ethereum and the BigchainDB networks in this hybrid architecture, work in tandem to allow for the continuous registration of complex product data. The BigchainDB decentralized database while storing this information, also allows for the fast and efficient execution of complex database queries that would have otherwise not been possible on the Ethereum chain database.

Figure 3 shows the result of a complex query made against the BigchainDB database that was implemented as a part of this research. The query allows for the search and retrieval of



```
[
  - {
      - data: {
            ethereum_client: "0x97698Ae226bE1573c5940dE64F50D12919826e54",
            platform: "NCState DIME Labs CMaaS",
            partName: "motor_mount",
            part_STL_hash_md5: "440baac6061e5d3cca61897e86cbd4cb",
          - dimension_metadata: {
                BaseHeight: 0.25,
                FlangeCircleDia: 0.2,
                TopHeight: 0.3,
                CenterCircleDia: 0.2,
                DrillDia: 0.15,
                FilletRadius: 0.11
            },
            timestamp: "1604807092",
          - dateObject: {
                day: 8,
                month: 11,
                year: 2020
            },
            description: "This is a sample part made by the DIME labs CMaaS platform"
        },
        id: "c9eccc99d44b205a1ba63e5cefddc25135e682efe9f720eb4b10d33ce0e671c8"
    }
]
```

Fig. 3. BigchainDB database query result for a sample part made on CMaaS.

information corresponding to an embedded search string or tag that could be present anywhere within the body of the information stored in the database. In Figure 3, the result can be seen as a query that was made using a search string tag of "DIME" and all the transactions on the BigchainDB which contained the tag "DIME" anywhere within the body of the JSON [46] converted information was retrieved. As is evident from the figure, the BigchainDB database is able to capture complex data of different types and precision in an immutable format and this is a direct upshot of the hybrid blockchain architecture that has been proposed as an improvement in this paper.

4. Machine learning based inference middleware for economic Ethereum transactions

Section 3 proposed an indirect solution to carry out economic Ethereum transactions in CMaaS platforms by allowing a CMaaS to offload large volume manufacturing data streams onto BigchainDB nodes instead of on the Ethereum EVM thereby saving transaction fees. In this section, the authors wanted to investigate if manufacturing transactions on such platforms could be further optimized. The goal of this section was to present a machine learning based inference middleware that can be proposed to forecast Ethereum transaction gas price in the future. As has been mentioned previously, an accurate gas price prediction model would allow the decentralized CMaaS platform to judiciously assign gas price limits on upcoming blockchain transactions instead of using default maximum values that guarantee transaction mining and acceptance by the miners in the Ethereum network. Assigning a gas price limit that is not too high from the average time series trend or nor too low ensures that the transactions would be mined at economical rates thereby saving cumulative costs. Since a decentralized CMaaS could see millions of such transactions happening over the course of time, the need of such a system is of paramount importance.

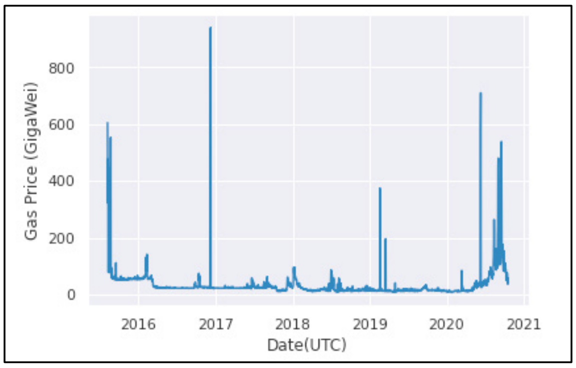


Fig. 4. Historical time series data of Ethereum gas price.

4.1. Data collection

In order to train machine learning (ML) models on historical data for time series prediction, there is need of chronological Ethereum gas price values over a significantly wide time span such that it allows for the capture of trend and seasonality patterns by the ML models. The Ethereum foundation has made daily time series data of past gas price values [47] available over a period of 4 years and it was this data that was used for extracting patterns from for the forecasting task. Figure 4 shows a snapshot of this historical time series data of Ethereum gas price in GigaWei shown along the vertical axis. The time index of the series starts from a date towards the end of the year 2016 and ends in the year 2020 as represented by the horizontal axis of the chart.

4.2. Problem framing

Given the time series trend of Ethereum gas prices, the problem statement of this section was constructed as follows: "*Given recent Ethereum gas price trends, what is the expected gas price for the week (7 days) ahead*?" That means, ML models that would be able to predict gas price values for at least 7 days into future would be required. Technically, this framing of the problem is referred to as a multivariate, multi-step time series forecasting model. This type of a model could be helpful for the decentralized CMaaS system to do predictions of 7 days into the future thereby enabling the platform to anticipate and prepare, much in advance, for any major changes of future gas price. Quite evidently, the choice of a 7-day prediction window is a hyperparameter that can be tuned to cater to differing system needs. For the purpose of this research however, this value was found to be able to do satisfactory forecasting to reasonable degree of accuracy.

4.3. Evaluation metric

Based on the assumptions of the model laid out in section 4.2, a forecast of Ethereum gas price would comprise of seven values, one for each day of the week ahead. It is common for multi-step forecasting problems to evaluate each time step separately [48]. This is usually done to contrast the performance of different models based on their capabilities at different lead times. It is always useful to have an error metric that has the same unit as the feature being predicted. Consequently, the quintessential continuous regression metric of Root Mean Square Error (RMSE) was adopted to compare the forecasting performance of different ML models. To deal with a single score identifying a model, an average RMSE value across all the 7 forecast days was evaluated.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y_i^*)^2} \tag{1}$$

Equation (1) shows the formula for the RMSE metric. It essentially is used to measure the deviation between the predicted or forecasted gas price y_i^* and the actual gas price y_i. The smaller the RMSE metric of an ML model on a data point, the higher is the accuracy of forecast of the model on that data point.

4.4. Train/Test Subset

The ML training scheme was fashioned after a typical supervised ML training protocol. The first 3 years' worth of data from the time series were used for training the ML models and the last year of 2020 was used to evaluate the performance of the models. This was in contrary to how train/test splits are made on normal datasets where the protocol is to randomly split the data. It is to be noted that the basis of this problem was time series data which has temporal structure and hence the conventional randomly sliced, train/test split is not acceptable. This is the reason why a train data subset of the first 3 years was chosen and a subsequent test data subset of the last year was chosen to make sure the temporal structure of the input data remains intact. The time series data was divided into standard weeks. This was a useful way for using the chosen framing of the model mentioned in section 4.2 where the gas price for the week ahead could be predicted.

4.5. Walk forward validation

The ML models eventually chosen to be trained on the train dataset were evaluated using a scheme called walk forward validation. In this scheme, an ML model being trained is required to make a prediction in the future over 7 days. Subsequently, the actual data of those 7 days of the week is appended to the train set and is made available to the ML model. This is done so that this new data can now be used as the basis for making predictions for the subsequent week in the forecasting window.

4.6. ML model choice and training

In typical ML based time series prediction problems, it is a norm to first start by training a naïve prediction model [49]. The results from such a naïve model provide a quantitative idea as to how difficult the forecasting problem is and also provides a baseline performance that can be used to compare other, more complex ML models trained on the same data. Three naïve

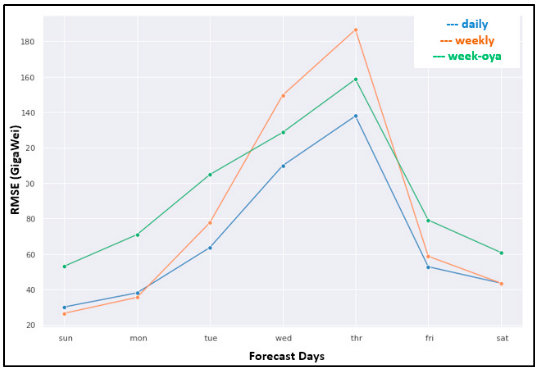


Figure 5. Performance of Naive forecasting models.

forecasting models were developed as a part of this process. They are listed down as follows:

- Daily persistence model.
- Weekly persistence model.
- Weekly one year ago persistence model.

The daily persistence model takes the Ethereum gas price from the last day prior to the period of forecast and uses that value as the value for each day in the forecast period. Similarly, the weekly persistence model uses the Ethereum gas price from the entire prior week as the forecast for the week ahead, and the weekly one year ago persistence model uses the same week last year to the predict next week's gas price. It becomes evident as to why these models are referred to as naïve models. Using prior days' or weeks' values for forecasting would obviously lead to less than optimum prediction models but acts as a fine baseline for other models to outperform. Any model performing worse in terms of RMSE value than the best naïve model can be readily rejected from consideration. A comparison of the forecasting prediction performance of the naïve models was subsequently performed to choose the final naïve model that would be used as the baseline performer. Figure 5 shows the average RMSE errors of forecasting performance for the 3 naïve models across a forecasting window of 7 days. It can be observed that the lowest RMSE scores were obtained for the daily persistence model across all days in the forecasting window and hence this model was eventually chosen as the model with the best baseline performance. An average RMSE of 77.6 GigaWei was obtained for the daily persistence model.

A suite of 12 supervised ML algorithms known to be quite robust for regression forecasting were chosen as candidate models to be trained on the data. The chosen models included the lasso regularized polynomial regression [50], the elastic net regressor [51], the extreme gradient boosted regressor [52], the decision tree regressor [53], the random forests regressor [54], the K nearest neighbor regressor [55], the support vector regressor [56], the extra tree regressor [57], the gradient boosting regressor [58], the ada-boosted ensemble regressor [58], 1D convolutional neural network regressor [59] and a sequence modelled LSTM regressor [60].

Figure 6 shows the neural network architecture for the prediction model based on the 1D convolutional regressor. It

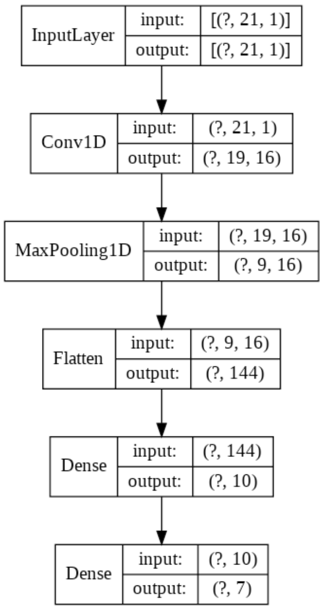


Fig. 6. Architecture of the 1D convnet.

can be observed that an input shape of 21 features representing the past 21 days' worth of Ethereum gas price values were used to forecast the next 7 day's Ethereum gas price represented by the output dense feature vector of size 7. These were hyperparameters of the model and the specific values of the input, output and layer sizes of the neural network were found after a series of trial and errors that were used to tune the model to its best possible performance.

Figure 7 shows the neural network architecture of the LSTM model. This model also takes in a sequence of 21 values of Ethereum gas prices and uses them to forecast the next set of values as the output. It is to be noted that the suit of the ML models chosen did not include more conventional statistical time series models like the auto regressive moving average models. This was a conscious decision since such statistical models are generally parametric and hence are much harder to tune and train on unconventional time series data with less profound trend and seasonality features as is the case for the historical Ethereum gas price data.

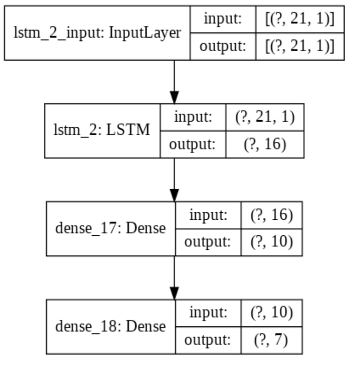


Fig. 7. Architecture of LSTM network.

5. Results

The proposed hybrid blockchain architecture was implemented using Ethereum as the blockchain network layer, L1 and the global BigchainDB test net as the layer L2 with reference to Figure 2. It was made to communicate in tandem with a CMaaS manufacturing layer deployed on a Flask [61] server infrastructure. Oracles encoded within the smart contracts deployed on L1 allowed the intercommunications between L1 and L2. It is to be noted that the global BigchainDB node is a global instance of the database and only retains information for a certain period of time, apt for testing purposes. It is a challenging task to accurately assess the potential computational and financial benefits provided by the hybrid blockchain architecture introduced in section 3. There are no standardized means to find a single score that could be used to assess the improved performance of a decentralized CMaaS platform after the implementation of the proposed architecture. However, some insight can be drawn from a sample data encapsulation by the BigchainDB network layer. As has been mentioned previously, Figure 3 shows the result of a complex query made against the BigchainDB database that is now used to store large volume processing and manufacturing data. This JSON encoded data shown in the Figure 3 was stored in the BigchainDB layer after the information was sent by the client middleware when it was triggered by events emitted by outbound Ethereum oracles. A cursory measure of performance could be derived from the calculation of computational steps i.e. gas units saved from being expended if the information stored by the JSON data object were otherwise to be stored directly on the Ethereum network layer instead of on the BigchainDB layer. The amount of gas units expended is directly proportional to the size of information encapsulated by the JSON data object.

Table 1. Ethereum fee schedule for data storage [22].

Name	Description	Value (Gas)	Total Cost (Gas)
G_{SSET}	SSTORE operation cost	20,000	
G_{TXDATA}	Non-zero-byte data cost	68 / byte	80,508
$G_{TRANSACTION}$	Flat transaction cost	21,000	

The JSON data object pertinent to the manufacturing information shown in Figure 3 was found to have a size of 581 bytes. Table 1 shows the Ethereum fee schedule for data storage [22]. This schedule determines how much gas units are expended when storing information of a certain size on the Ethereum network. Each information registration transaction on the Ethereum network is a function of the three fee elements shown in Table 1. The total cost of an information storage operation in terms of total gas units can thus be expressed in the form of equation 2.

$$Cost = G_{SSET} + G_{TRANSACTION} + Size * G_{TXDATA} \tag{2}$$

Using equation 2, the total gas cost for storing the information encapsulated by the JSON data object directly on the Ethereum network would have been 80,508 gas units. This is the amount of the computational steps that can be saved if the information can be stored on the BigchainDB layer. The amount of gas units can easily be converted to fiat currency to gauge the amount of financial savings. Using the average per unit gas price of 55 GigaWei for the month of November 2020, this amounts to a total of around 4,427,940 GigaWei required for the storage of the information. This value in fiat currency is equivalent to around $2. While this calculated cost might seem to appear low, it must be remembered that a decentralized CMaaS platform would typically execute thousands of such data storage transactions every day. Additionally, the amount calculated in this section is under the conservative assumption that the JSON data object is only 581 bytes. A CMaaS platform could store manufacturing and process metadata which could be much larger than 581 bytes. Hence the total computational and financial cost savings ensuing from the implementation of this hybrid blockchain architecture can easily grow exponentially.

Table 2. RMSE comparison of trained ML models with Naïve model.

ML Model	Test RMSE	Naïve RMSE
Random Forest	70.6	
Extra Tree	72.6	
Elastic Net	72.9	
Lasso	72.9	
LSTM	76.4	
XGB	76.5	77.6
KNN	76.7	
Gradient Boost	78.7	
CNN	80.5	
Ada Boost	83.4	
Support Vector	89.7	
Decision Tree	107.3	

To assess how well the trained ML models introduced in section 4.6 were able to forecast the time series Ethereum gas price, the performance of the models on the test set were compared to that of the Naïve daily persistence model in terms of RMSE values. Table 2 shows the RMSE comparison of the 12 models with the Naïve daily persistence model. It can be observed that the best performing model was a Random Forest regressor model with the least RMSE score of 70.6 on the test set, which was lower than the naïve RMSE of 77.6. The random forest model was therefore able to forecast Ethereum gas prices with larger degree of accuracy when compared to the naïve model. This trained random forest model was serialized and then deployed on the decentralized CMaaS platform server encapsulated in an inference middleware, so that the client middleware would then be able to use forecasted gas prices from this inference middleware to judiciously assign gas price limits on future Ethereum transactions, thereby leading to eventual cost savings. The models starting from Gradient Boosting and ending in Decision Trees had RMSE values worse than the naïve model and hence were rejected from the available pool of forecasting regressors. Figure 8 shows the predicted trends forecasted by the top 2 performing models on the test data set. The trends were overlaid on the actual test data set for comparison of deviations. It can be observed that the random forest regressor model does in fact closely resemble the actual time series trend of the test data set.

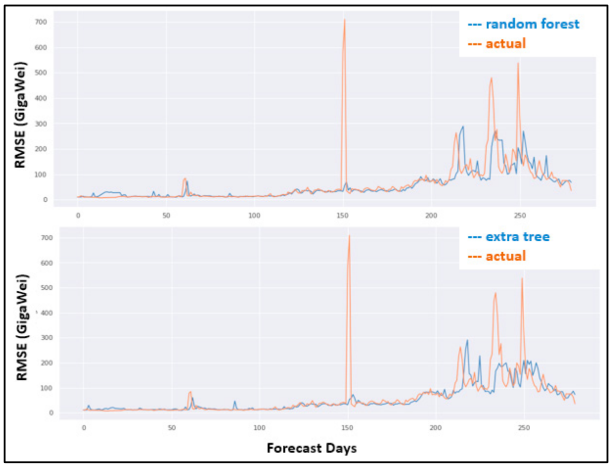


Figure 8. Predicted trends of top 2 ML models overlaid with actual trends.

6. Conclusion

In this paper, the framework of a hybrid blockchain architecture for an improved decentralized CMaaS platform was proposed as a continuation of the previous work of the authors. It was shown that, by integrating a parallel BigchainDB network of distributed database alongside the Ethereum blockchain backbone of the decentralized CMaaS, significant portion of the data storage payload could be offloaded to the BigchainDB layer. Through the design and implementation of server based plug-and-play middleware and outbound Ethereum oracles, large volume manufacturing and process metadata could be transferred to be stored seamlessly on the BigchainDB layer without the need of storing data intensive information onto the Ethereum network through computationally intensive, high cost transaction processes. The novel hybrid blockchain architecture comprising of the aforementioned elements was shown to reduce significant costs associated with data storage events on Ethereum blockchain based CMaaS platforms.

In a quest to further reduce costs from manufacturing transactions on blockchain based CMaaS platforms, a machine learning based inference middleware was also designed and deployed as a part of this research. The middleware houses a trained machine learning model based on the Random Forest Regressor algorithm that can accurately predict and forecast Ethereum gas prices. It was shown that the trained prediction model was able to forecast gas price to reasonable degree of accuracy with a low RMSE score of 70.6, outperforming a naïve daily persistence model. The ability of the decentralized CMaaS platform to predict Ethereum gas price allows it to judiciously assign allowable gas price limit to upcoming transactions, making sure that they are not too high as is the case when default modes of transactions are adopted. This allows the CMaaS platform to autonomously decide when to broadcast a transaction to the Ethereum network with an aim to minimize transaction costs. If the Ethereum ecosystem is going through a period of relatively higher gas prices, the inference middleware can temporarily hold the transaction from being broadcasted so that it can later be released in times of lower gas prices. This does not pose any additional time restrictions for quick delivery products in the CMaaS since these fluctuations of gas prices occur over temporal frequencies in the order of seconds. Such a system allows the platform to save a significant amount in terms of mining fees, on thousands of transactions that occur in blockchain based CMaaS platforms.

The hybrid blockchain architecture proposed in this paper is enabled by an improved client middleware and Ethereum oracles. This means, there are several steps involved before large volume data can make its way to the BigchainDB network. The inter-communication of different middleware and time-consuming oracles can introduce unwanted latency if real time performance is mandated. To conquer this limitation, there is need of much simpler architectures where there are lesser number of information exchange steps. The need of a single, universal blockchain platform with secure asset transfer models and big data capabilities is thus imperative and remains an avenue of future research.

Acknowledgements

This work was supported by the National Science Foundation grant #1764025.

References

[1] Helo, P., Hao, Y., Toshev, R., & Boldosova, V. (2021). Cloud manufacturing ecosystem analysis and design. Robotics and Computer-Integrated Manufacturing, 67, 102050.

[2] Li, Y., Tao, F., Cheng, Y., Zhang, X., & Nee, A. Y. C. (2017). Complex networks in advanced manufacturing systems. Journal of Manufacturing Systems, 43, 409-421.

[3] Napoleone, A., Macchi, M., & Pozzetti, A. (2020). A review on the characteristics of cyber-physical systems for the future smart factories. Journal of Manufacturing Systems, 54, 305-335.

[4] Hasan, M., & Starly, B. (2020). Decentralized cloud manufacturing-as-a-service (CMaaS) platform architecture with configurable digital assets. Journal of Manufacturing Systems, 56, 157-174.

[5] Starly, B., Angrish, A., Pahwa, D., Hasan, M., Bharadwaj, A., & Cohen, P. (2020). Democratizing Innovation through Design Automation, Manufacturing-As-A-Service Marketplaces and Intelligent Machines.

[6] Custom manufacturing on demand. (n.d.). Retrieved March 01, 2021, from https://www.xometry.com/

[7] The worlds most Intelligent Cnc Factory: Groundbreaking AI. (2021, February 19). Retrieved March 01, 2021, from https://cloudnc.com/.

[8] Helu, M., Hedberg Jr, T., & Feeney, A. B. (2017). Reference architecture to integrate heterogeneous manufacturing systems for the digital thread. CIRP journal of manufacturing science and technology, 19, 191-195.

[9] Lubell, J. (2015, August). Extending the cybersecurity digital thread with XForms. In Balisage: the markup conference.

[10] Sturm, L. D., Williams, C. B., Camelio, J. A., White, J., & Parker, R. (2017). Cyber-physical vulnerabilities in additive manufacturing systems: A case study attack on the. STL file with human subjects. Journal of Manufacturing Systems, 44, 154-164.

[11] Force, J. T., & Initiative, T. (2013). Security and privacy controls for federal information systems and organizations. NIST Special Publication, 800(53), 8-13.

[12] Thompson, K. (2020, December 11). Digital thread for smart manufacturing. Retrieved March 01, 2021, from https://www.nist.gov/programs-projects/digital-thread-smart-manufacturing.

[13] Lu, Y., Wang, H., Xu, X. "ManuService ontology: a product data model for service-oriented business interactions in a cloud manufacturing environment". (2019) Journal of Intelligent Manufacturing, 30 (1), pp. 317-334.

[14] Fischer K, Rosche P, Trainer A, Feeney AB, Hedberg TD. Investigating the impact of standards-based interoperability for design to manufacturing and quality in the supply chain (No. Grant/Contract Reports (NISTGCR)-15-1009). 2015.

[15] Michaloski, J., Hedberg, T., Huang, H., Kramer, T., & Michaloski, J. (2016). End-to-end quality information framework (QIF) technology survey. US Department of Commerce, National Institute of Standards and Technology.

[16] Yaqoob, I., Salah, K., Uddin, M., Jayaraman, R., Omar, M., & Imran, M. (2020). Blockchain for digital twins: Recent advances and future research challenges. IEEE Network, 34(5), 290-298.

[17] Huang, S., Wang, G., Yan, Y., & Fang, X. (2020). Blockchain-based data management for digital twin of product. Journal of Manufacturing Systems, 54, 361-371.

[18] Krima, S., Krima, S., Hedberg, T., & Feeney, A. B. (2019). Securing the digital threat for smart manufacturing: A reference model for blockchain-based product data traceability. US Department of Commerce, National Institute of Standards and Technology.

[19] Arcenegui, J., Arjona, R., & Baturone, I. (2020, October). Secure Management of IoT Devices Based on Blockchain Non-fungible Tokens and Physical Unclonable Functions. In International Conference on Applied Cryptography and Network Security (pp. 24-40). Springer, Cham.

[20] Kumar, A., Abhishek, K., Nerurkar, P., Ghalib, M. R., Shankar, A., & Cheng, X. (2020). Secure smart contracts for cloud ‐ based manufacturing using Ethereum blockchain. Transactions on Emerging Telecommunications Technologies, e4129.

[21] Lee, J., Azamfar, M., & Singh, J. (2019). A blockchain enabled Cyber-Physical System architecture for Industry 4.0 manufacturing systems. Manufacturing Letters, 20, 34-39.

[22] Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper, 151(2014), 1-32.

[23] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., ... & Muralidharan, S. (2018, April). Hyperledger fabric: a distributed operating system for permissioned blockchains. In Proceedings of the thirteenth EuroSys conference (pp. 1-15).

[24] Zyskind, G., & Nathan, O. (2015, May). Decentralizing privacy: Using blockchain to protect personal data. In 2015 IEEE Security and Privacy Workshops (pp. 180-184). IEEE.

[25] Thomson, G. (2020, July 03). Dapp volume hits $12 billion as Ethereum dominates. Retrieved November 03, 2020, from https://decrypt.co/34494/dapp-volume-hits-12-billion-as-ethereum-dominates.

[26] Lee, T. N. (2017). Traditional firms are opening up to blockchain and its decentralised apps. LSE Business Review.

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

[28] Victor, F., & Lüders, B. K. (2019, February). Measuring ethereum-based erc20 token networks. In International Conference on Financial Cryptography and Data Security (pp. 113-129). Springer, Cham.

[29] Entriken, W., Shirley, D., Evans, J., & Sachs, N. (2018). Erc-721 non-fungible token standard. Ethereum Foundation.

[30] Gervais, A., Karame, G. O., Wüst, K., Glykantzis, V., Ritzdorf, H., & Capkun, S. (2016, October). On the security and performance of proof of work blockchains. In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security (pp. 3-16).

[31] Hirai, Y. (2017, April). Defining the ethereum virtual machine for interactive theorem provers. In International Conference on Financial Cryptography and Data Security (pp. 520-535). Springer, Cham.

[32] Grech, N., Kong, M., Jurisevic, A., Brent, L., Scholz, B., & Smaragdakis, Y. (2018). Madmax: Surviving out-of-gas conditions in ethereum smart contracts. Proceedings of the ACM on Programming Languages, 2(OOPSLA), 1-27.

[33] McConaghy, T., Marques, R., Müller, A., De Jonghe, D., McConaghy, T., McMullen, G., ... & Granzotto, A. (2018). BigchainDB: a scalable blockchain database. white paper, BigChainDB (2016). Query date, 02-26.

[34] Mandolla, C., Petruzzelli, A. M., Percoco, G., & Urbinati, A. (2019). Building a digital twin for additive manufacturing through the exploitation of blockchain: A case analysis of the aircraft industry. Computers in Industry, 109, 134-152.

[35] Angrish, A., Craver, B., Hasan, M., & Starly, B. (2018). A case study for Blockchain in manufacturing: “FabRec”: A prototype for peer-to-peer network of manufacturing nodes. Procedia Manufacturing, 26, 1180-1192.

[36] Veldhuizen, T. L. (2003). C++ templates are turing complete.

[37] Dannen C. Introducing ethereum and solidity Vol. 1. Berkeley: Apress.; 2017.

[38] Bonnard, R., Hascoët, J. Y., Mognol, P., Zancul, E., & Alvares, A. J. (2019). Hierarchical object-oriented model (HOOM) for additive manufacturing digital thread. Journal of Manufacturing Systems, 50, 36-52.

[39] Lu, Y., & Xu, X. (2019). Cloud-based manufacturing equipment and big data analytics to enable on-demand manufacturing services. Robotics and Computer-Integrated Manufacturing, 57, 92-102.

[40] NEM Blockchain. (2021, February 26). Retrieved March 01, 2021, from https://nem.io/platforms/

[41] Eosio blockchain software &amp; services. (2021, February 25). Retrieved March 01, 2021, from https://eos.io/

[42] Huang, Y., Wang, H., Wu, L., Tyson, G., Luo, X., Zhang, R., ... & Jiang, X. (2020). Characterizing eosio blockchain. arXiv preprint arXiv:2002.05369.

[43] Liu, F., Wang, X., Li, Z., Xu, J., & Gao, Y. (2020, January). Effective GasPrice Prediction for Carrying Out Economical Ethereum Transaction. In 2019 6th International Conference on Dependable Systems and Their Applications (DSA) (pp. 329-334). IEEE.

[44] Kiffer, L., Levin, D., & Mislove, A. (2017, November). Stick a fork in it: Analyzing the ethereum network partition. In Proceedings of the 16th ACM Workshop on Hot Topics in Networks (pp. 94-100).

[45] Adler, J., Berryhill, R., Veneris, A., Poulos, Z., Veira, N., & Kastania, A. (2018, July). Astraea: A decentralized blockchain oracle. In 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) (pp. 1145-1152). IEEE.

[46] Introducing JSON. (n.d.). Retrieved November 08, 2020, from https://www.json.org/json-en.html

[47] Historical Ethereum Gas Price Values. (n.d.). Retrieved November 08, 2020, from https://etherscan.io/chart/gasprice

[48] Brownlee, J. (2018). Deep learning for time series forecasting predict the future with MLPs, CNNs and LSTMs in Python. Machine Learning Mastery Publishers, Melbourne, Australia.

[49] Akpinar, M., & Yumuşak, N. (2017). Naive forecasting of household natural gas consumption with sliding window approach. Turkish Journal of Electrical Engineering & Computer Sciences, 25(1), 30-45.

[50] Owen, A. B. (2007). A robust hybrid of lasso and ridge regression. Contemporary Mathematics, 443(7), 59-72.

[51] Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. Journal of the royal statistical society: series B (statistical methodology), 67(2), 301-320.

[52] Chen, T., He, T., Benesty, M., Khotilovich, V., & Tang, Y. (2015). Xgboost: extreme gradient boosting. R package version 0.4-2, 1-4.

[53] Dobra, A. (2002). Classification and regression tree construction. Retrieved September, 18, 2011.

[54] Grömping, U. (2009). Variable importance assessment in regression: linear regression versus random forest. The American Statistician, 63(4), 308-319.

[55] Peterson, L. E. (2009). K-nearest neighbor. Scholarpedia, 4(2), 1883.

[56] Drucker, H., Burges, C. J., Kaufman, L., Smola, A. J., & Vapnik, V. (1997). Support vector regression machines. In Advances in neural information processing systems (pp. 155-161).MIT Press.

[57] Ahmad, M. W., Reynolds, J., & Rezgui, Y. (2018). Predictive modelling for solar thermal energy systems: A comparison of support vector regression, random forest, extra trees and regression trees. Journal of cleaner production, 203, 810-821.

[58] He, H., Yang, Y., & Pan, Y. (2019). Machine learning for continuous liquid interface production: Printing speed modelling. Journal of Manufacturing Systems, 50, 236-246.

[59] Kucukoglu, I., Atici-Ulusu, H., Gunduz, T., & Tokcalar, O. (2018). Application of the artificial neural network method to detect defective assembling processes by using a wearable technology. Journal of manufacturing systems, 49, 163-171.

[60] Zhang, J., Wang, P., Yan, R., & Gao, R. X. (2018). Long short-term memory for machine remaining life prediction. Journal of manufacturing systems, 48, 78-86.

[61] Grinberg, M. (2018). Flask web development: developing web applications with python. " O'Reilly Media, Inc.".