Mehran University Research Journal of Engineering and Technology

https://doi.org/10.22581/muet1982.2204.14

2022, 41(4) 138-145

Investigation on a price oracle problem

Kashif Mehboob Khan [*], Rida Taufique, Maryah Abdul Rauf

Department of Software Engineering, NED University of Engineering and Technology, Karachi Pakistan

[*] Corresponding author: Kashif Mehboob Khan, Email: kashifmehboob@neduet.edu.pk

Received: 05 February 2022, Accepted: 06 June 2022, Published: 01 October 2022

K E Y W O R D S

Smart Contracts

Oracles

Blockchain

Decentralized Finance

Peer to Peer Networks

A B S T R A C T

The core of blockchain smart contracts is the execution of business logic code in a decentralized architecture with all executing nodes trusting and agreeing on the results. Smart contracts are unable to get data from the outside world on their own. Smart contracts communicate with oracles, which are off-chain data sources whose primary function is to collect and give data feeds to smart contracts. The usage of oracle returns the blockchain to its centralization problem and also exposes the blockchain to the possibility of introducing corrupt, malicious and erroneous data. This problem is called 'Oracle problem'. This paper presents an investigation of this problem. To demonstrate this, we have considered the price oracle problem using chainlink's decentralized network of nodes that connects off-chain data to on-chain smart contracts through oracles. Finally, we performed an analysis and comparison on retrieving external off-chain data through external APIs (Application Programming Interfaces) and through decentralized oracles. This would be helpful in determining how decentralization (through oracles) may result in performance constraints in contrast to fetching data through our own built APIs where data source APIs of blockchain are not connected to the decentralized nodes of network. The results have shown the increase in transaction throughput of the overall system.

1. Introduction

As the number of blockchain-based applications are increasing, the oracle problem for pushing external data into the blockchain is becoming critical. Investigating oracle problem in blockchain is one of the key areas of today as it may lead towards some serious loss in terms of both data and money. In December 2017, an attacker was able to steal $226 thousand of cryptocurrency [1]. Similarly, in another attempt, a hacker successfully exposed vulnerability of oracles in 2020. A manipulation of data in price oracle caused a loss of over $30 million while surprisingly it did not make the network slow due to the manipulation [2]. The hackers have been able to steal around $120 million only in 2020 followed by draining $11 million of fund in February 2021 [3]. A common issue in such attacks include the poorly written smart contracts. This allowed the attackers to exploit the vulnerabilities and carry malicious transactions. These vulnerabilities are mostly due to the logical bugs in code and builds up a very severe impact due to the immutability of smart contracts [1-3]. Smart contracts require data from outside the blockchain system. This situation demand smart contracts to bring off-chain data into the blockchain system. These data feeds are performed through oracles. In the context of blockchain, an oracle is an external data agent that monitors real-world events and reports them back to the blockchain so that smart contracts can use them. The DeFi (Decentralized Finance) protocols which rely on these oracles provide a lot of incentives to malicious actors as they handle large transactions. A type of attack in view is the flash loan attack [4]. Flash loans allow users to utilize loans with the condition that the loan must be repaid within the same transaction. Another common attack on blockchain oracle is Sybil attack [5]. In this attack, the malicious user gains control over multiple nodes in the oracle network. Some other attacks exploiting these vulnerabilities include bZx margin and Warp Finance [6]. Due to the losses made by such attacks, researchers are making certain attempts to investigate various factors which are associated with blockchain oracles and may lead towards a successful attack. The paper aims to investigate factors such as decentralization attribute of oracles versus its scalability. This has been done by compared the price oracle to a blockchain system which is getting off-chain data from an external API (Application Programming Interface) and pushing it to the on-chain consensus data.

The following are the paper's significant contributions.

1. We have presented an empirical analysis on price oracles problem and its comparison with our own built APIs for retrieving data and pushing it to the blockchain. The empirical study aims to highlight the significant trade-offs between oracles problem and scalability of blockchain network upon compromising the source of data as a centralized source accessible through API.
2. We presented our own proposed API based data model to push data from external source into the blockchain. We implemented this model to make a comparison between decentralised oracles and off-chain external data source in the context of scalability.

2. Related Work

Egberts [7] demonstrated why oracles still lack the capability of handling the problem of decentralization. He also discussed how oracles could become a single point of failure for the entire blockchain system. Authors in [8] discussed price oracles for Defi. They carried out investigation over platform design for DeFi and highlighted certain issues in the design of the existing oracles. Grooteman et al. [9], proposed an oracle service using cryptographic scheme. The purpose was to publicly verify data feeds on the main consensus chain. Heiss et al. [10] enforce the need to ensure trust worthy on-chain data and then propose a set of rules to be implemented in oracles in accordance with the trust worthy on-chain data. Schaad et al. [11] implemented a hardware-based oracle (printer in that case) using existing software level protection scheme. Their major contribution included the protection of API calls which were cryptographically secured. Damjan [12] attempted to verify the integrity of off-chain data from the real world through oracles. The paper discusses the data in the context of its legal aspect. Beniiche [13] discusses the human oracle and its utilization to respond to a certain problem for identifying the truth against a problem. Murimi et al. [14] discuss the uncertainties that are associated with the outside world of the oracles. It also determines the size of the network that is appropriate to be served by the oracle. Mühlberger et al. [15] have discussed oracles in context of their data flows and their sequence to analyse their performance constraints. Pasdar et al. [16] worked on the feedback of oracles, which are based upon voting and reputation models to verify the correctness of external data. Caldarelli et al. [17] identified the patterns to define the oracle problem in DeFi and to present the most promising approaches to overcoming the related flaws. They addressed the problems which are associated with the use of oracles in DeFi, and how these problems are being investigated. To deal with the most common hacks, they proposed that standardization of oracle design and patterns is essential. Similarly, Caldarelli [18] expressed great concerns over the implications of not exploring or acting upon solving the oracles problem in most of the papers. Kaleem et al. [19] based their research on real data collection from blockchain network using Chainlink Oracles. Oracle price feeds and Oracle external APIs were considered for this collection. The results showed that the price feed oracles are increasing in use with the growing count of DeFi projects. However external API feature has not been utilized much. Caldarelli [20] has analysed the different conditions of oracles and their respective vulnerabilities and their implications on the security of real world blockchain applications. They examine healthcare, law, academics, energy, and supply-chain. It attributes the problem of oracle more to the social aspects than the technical ones. The impact of oracle problem varies with the application depending on the trustworthiness of the system where it is implemented. Al-Breiki et al. [21] examined and compared trust-enabling properties of the leading blockchain oracle techniques on the basis of two aspects. The first aspect is the solution's deployment (for example, whether the solution is implemented on-chain, off-chain, or on both sides). The second aspect is ‘Trust Model’, where a single node represents a centralized trust model and several nodes represent decentralized trust models. A study has been presented by Khan et al. [22], where authors have explored the scalability constraints of a blockchain based service level agreement system that retrieved off-chain data to evaluate the quality of service.

3. Data Models

We particularly want to compare performance constraint using oracles against our own built API in the context of scalability. We have chainlink's data access model for oracles as shown in Fig 1, while for the centralized data source, we have proposed and implemented our own data model. The implementation of the model has been performed on Multichain and the code has been uploaded on github with detail at section 6.2. We will therefore now discuss in brief these two types of models based on SLA (Service Level Agreement).

3.1 Data Model for Existing Oracles

Here we will discuss in short the execution of SLA based smart contracts that has been used for comparison with our own proposed SLA data model (to be discussed next) in terms of decentralization and scalability constraints. When smart contract requests for external data, it is called a Requesting contract. This request is registered as an event and the chainlink creates a contract called SLA- Service Level Agreement to get the required data. This SLA creates three more contracts which are Reputation, Order- Matching and Aggregating Contracts. The Chainlink Reputation Contract verifies an oracle provider's validity and performance history before evaluating and discarding malicious nodes. The Order-Matching Contract deals with the oracle nodes that are ready to bid on the particular request, generated by Requesting Contract. Its job is to select the correct type and number of nodes to fulfil the desired request. The Chainlink Aggregating Contract checks all of the data from the chosen oracles to provide an accurate result to the requesting smart contract.

3.2 Data Model for Proposed API Based Architecture

Fig. 2 shows the architecture of our proposed model which we have used to perform a comparative empirical analysis with exiting model of oracles using chainlink as platform. We have implemented this model using java. We controlled the data flows by customizing open source JSON based RPC APIs available for Multichain. These java based programmable remote clients were used to capture response time of hosted services and pushes it to the blockchain [22]. In this way any violation of the SLA may become a part of blockchain's immutable collection of records. This proposed model of SLA will be used in empirical analysis to investigate the performance constraints with the existing model of oracles for SLA contracts (as presented in Fig. 1).

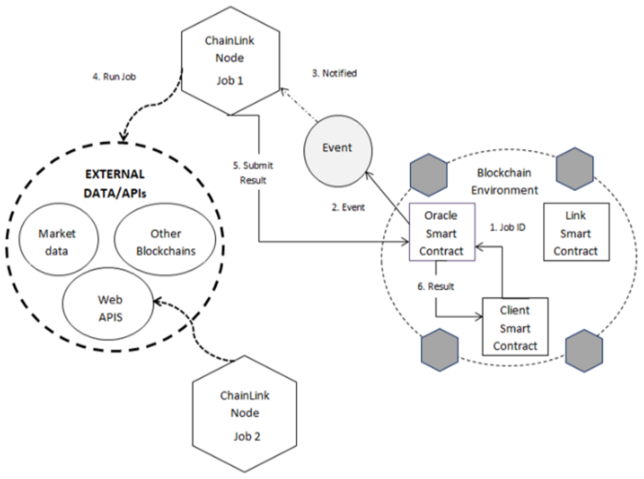


Fig. 1. System architecture

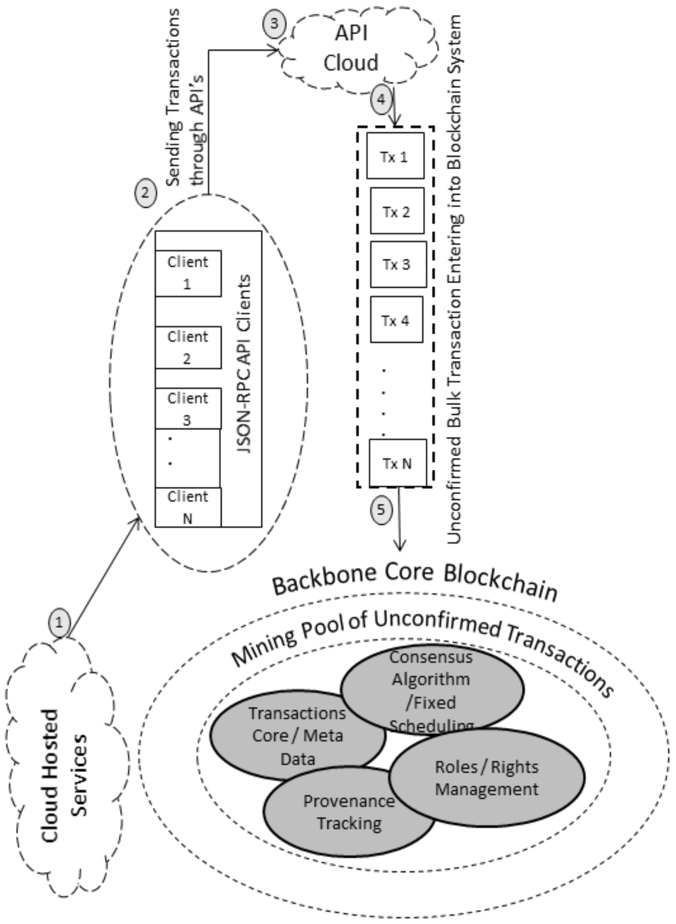

Sending Transactions through API's
API Cloud
Unconfirmed Bulk Transaction Entering into Blockchain System
Client 1
Client 2
Client 3
Client N
JSON-RPC API Clients
Tx 1
Tx 2
Tx 3
Tx 4
Tx N
Backbone Core Blockchain
Mining Pool of Unconfirmed Transactions
Consensus Algorithm /Fixed Scheduling
Transactions Core / Meta Data
Roles / Rights Management
Provenance Tracking
Cloud Hosted Services

Fig. 2. Proposed Blockchain based SLA Architecture without Oracles

4. Empirical Analysis

4.1 Decentralized Oracles

In this section, we would analyse the performance, particularly transaction throughput of the existing oracles using Chainlink as a platform for experimentation. Before we discuss, the experiment itself, it is important to iterate the aim of this experiment. We are seeking to figure out how Chainlink's oracle network performs for scalability while claiming to solve oracle's problem through decentralization of nodes.

Below are the steps, which were performed for the experimentation.

4.1.1 Creating wallet and getting link and ether

We created a wallet in MetaMask which is added as a browser extension. The test ethers was requested using the link from https://faucets.chain.link/ by entering the unique wallet hashed address. Ether is received directly but LINKs (chainlinks) require importing the tokens through the address.

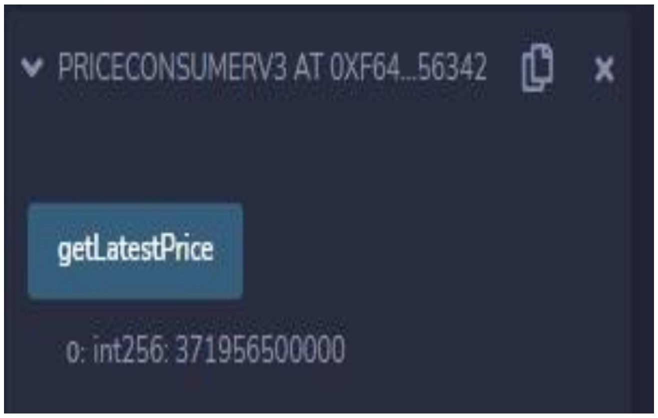
4.1.2 Accessing price feeds

We started with Chainlink price feeds to connect smart contracts to real-time asset market values. Smart contracts can use them to get the most up-to-date pricing of an asset in a single call. We have used 'Kovan Network' to get the most up-to-date ETH (Ether values) price inside smart contracts. Remix Ethereum IDE (Integrated Development Environment) was used to deploy the smart contract. The contract contains variables for the Ethereum price, round identifier, and timestamp. In order to consume price data, the smart contract should reference 'AggregatorV3Interface', which describes the external methods supported by Price Feeds.

After deploying the contract, and depositing the required gas fee, the contract returns the aggregated price as shown in Fig. 3. The price is aggregated from 31 oracles as shown in Fig. 4.

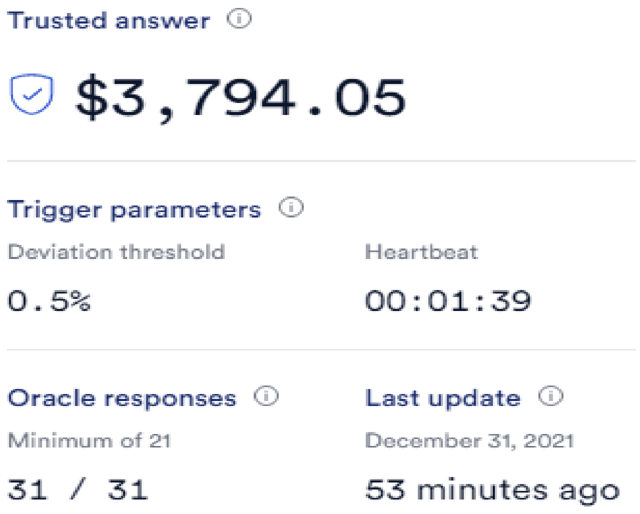
4.1.3 Monitoring API calls

In this part, we have attempted to investigate the response of external APIs. Apart from the approach as adapted in section 5.2, another way to access data from any source outside the blockchain is through external API calls as it already provided by chainlink. External APIs are more generalized in the sense that they allow the user to define any data source they would like to retrieve from. This is also passed through the decentralised network of oracles. We have used them for testing purposes. We retrieved ETH price for evaluation. A Chainlink client was imported into the smart contract from Github repository for connecting to their network. The same (Remix) IDE was used to deploy a different smart contract from before for API call. The contract contained variables for the Ethereum price, job ID, and the oracle network hash. The function for requesting Ethereum price creates a request to retrieve API response and multiplies the retrieved price by 100 to remove decimal points using different method calls. Here, URLs can also be set from which we retrieved the price along with the currency format such as 'USD'. Finally, the main API call is performed. The link token and Ethereum gas price is subtracted from our wallet and the price is as shown in Fig. 5.



```
0: int256: 371956500000
```

Fig. 3. Latest price function call



Trusted answer

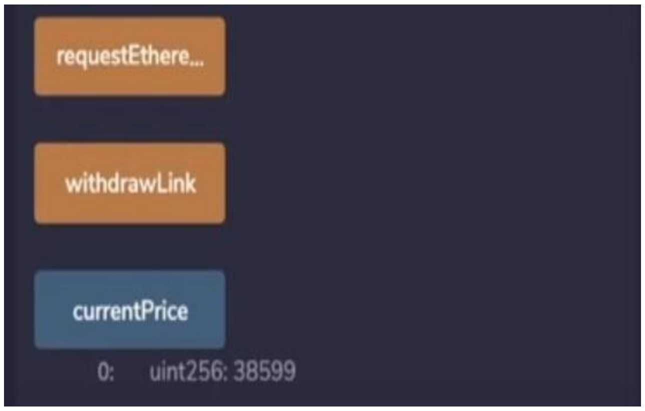
$3,794.05

Trigger parameters

Deviation threshold	Heartbeat
0.5%	00:01:39

Oracle responses	Last update
Minimum of 21	December 31, 2021
31 / 31	53 minutes ago

Fig. 4. Aggregated price of ETH/USD



```
0:    uint256: 38599
```

Fig. 5. Request for Ethereum price function call

4.1.4 Specifications of smart contract and network

The contracts were deployed using the specifications that are shown in Table 1.

Table 1

Development environment specifications

Parameter	Value
Network	Kovan Test Network
Language	Solidity
Faucet	Koven ChainLink Faucet
Development Environment	Remix
Wallet	MetaMask

4.2 Proposed Blockchain Based SLA Model

Here, we will investigate the implementation of using our own proposed Blockchain based SLA model. This model is based upon blockchain. The data in the model is not being fetched through decentralized oracles. In this way by analysing this model empirically and comparing it with the existing model of oracles, we may investigate the impact of decentralization of oracles towards the scalability and transaction throughput in blockchain based system.

In order to implement the system, we used Multichain as a blockchain platform. A total of 13 nodes were used to build the blockchain system for proposed SLA model. Out of which, there are 10 JSON based RPC remote clients, one seed node, and one connected node as shown in Table 4. Table 5 shows the basic configuration of blockchain. In order to capture the response time, we have programmed our remote clients to ping various web URLs. These web URLs have been mapped with blockchain through individual wallet addresses. Table 6 shows some of the services which we have used in the experimentation along with their respective wallet addresses. Client wallet address shows the sending wallet address which is being used by a particular remote client.

Table 2

Contract creation

Parameter	Values	
External Data	Price Feeds	API Call
Transaction Fee (Ether)	0.00055211002	0.00367573751
Gas Used by Transaction	216,844	1,470,295
Gas Price (Gwei)	2.500000007	2.500000007
Link Price	0	1
Lines of Code	20	26

4.1.5 Parameters observed

After deploying both the contracts i.e., Price Feed Contract and API Calling contract. We observed the following parameters from "Etherscan". The parameters have been shown in Tables 2 and 3.

Table 3

Transaction details

	Parameter	Values
Price Feeds	Response Time	8 seconds
	Burnt Ether	1.51791E-12
	Transaction Savings (Ether)	1.51791E-12
API Calls	Response Time	12 seconds
	Burnt Ether	1.02921E-11
	Transaction Savings (Ether)	1.02921E-11

Table 4

Node specifications for MultiChain 2.0 Blockchain

Parameter	Values		
No. of Nodes	1	2	10
Platform	Windows 10	Ubuntu 18.04	Windows 10
Type	Desktop PC (Full Running Node)	Laptop (Full Running Node)	Java Based Remote Clients
Processor	Core i7	Core i7	Core i7
RAM	8192 MB	8192 MB	8192 MB
Wallet Addresses	1	10	10

Table 5

Blockchain configuration

Parameter	Values
No. of Miners	10
Mining Diversity	0.5
Block Generation Rate (seconds)	15
Block Size (bytes)	8388608
No. of Bits (PoW)	8

4.2.1 Scalability testing

We initially started with 500 transactions by each java based API RPC remote clients. These clients were monitoring the response time of web services and then pushing the data to the blockchain along with response time and SLA token within transactions. Let's $T_{tx\,count/client}$ denotes number of transactions issued by a client. In order to determine average transaction throughput of the system, $T_{avg-txcount}$, we can divide average block size by the size of individual transaction. The observations have been recorded in Table 7. Here the size of the transaction is computed to be 578 bytes. The computation has been done using raw transaction as illustrated by Multichain officials [23].

Table 6

Some Registered Wallet Addresses for Remote Clients / Hosted Services

		Wallet IDs
Remote Clients	Client 1	1UjBijqcAeqERzGtUAbwta6WXJFtgdBH8yztnm
	Client 2	12GqrwcRmoX7Vw9cZV38Jz8YbdybhXfNGWdpV3
	Client 3	1TK95RZJ7JugMTJzjRkdRYTmnQ2xhVJWFpVBcD
		Wallet IDs
Hosted Services	SLA	14gtzt3XZVT3Mf61uAQoGEXxGgP6yc41awynzg
	Yahoo	1UuPVjnTvewdSaw9Xe8XL1HzaTBYVi2yG8xZVD
	Ebay	1VfSM1pC2qtFQaYHcYirGc8tMFUDLnJLjn5JPo

Table 7

Transaction Throughput without Decentralized Nodes

Parameter	Values					
$T_{t\,x\,count/client}$	500	1000	1500	2000	2500	3000
Total Transactions (* 10)	5000	10000	15000	20000	25000	30000
$Bl_{avg-datasize}$ (bytes)	5581	8499	10399	15380	16120	16218
$T_{avg-txcount}$ per block	9.65	14.70	17.99	26.60	27.88	28.05

5. Results and Discussion

This section discusses the findings of empirical evaluation that has been performed for scaling up the network.

5.1. Decentralized Nodes Scalability

In general, while considering the performance constraints in terms of requirement, it is very evident that external APIs require much more transaction fee and gas as compare to accessing data without calling APIs as in the case of price feeds smart contracts (refer to Fig. 6). We are more interested for the response or execution time using oracles due to decentralization. As mentioned before, we captured response time for accessing external data using smart contracts and API call. While both of these approaches use decentralized network of nodes, external APIs were not very responsive in comparison as shown in Fig. 7. The price feed fetched the required value in nearly 8 secs whereas external API consumed 12 seconds for the same which is the best throughput of our evaluation in the given condition.

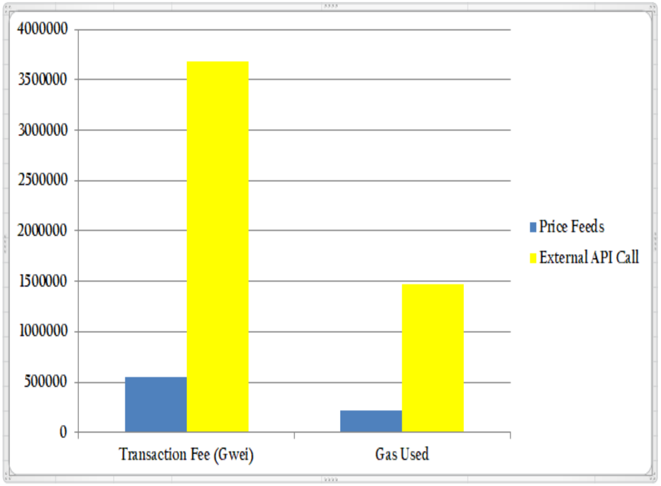


Fig. 6. Comparison of price feeds and external API contract

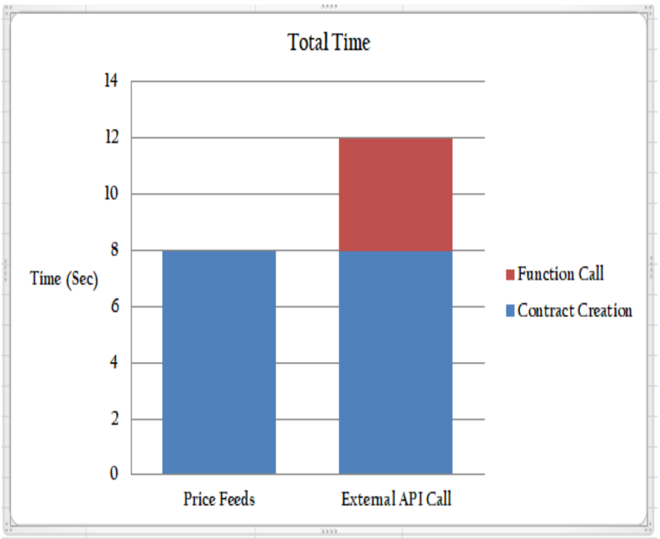


Fig. 7. Comparison of total response time

5.2. Scalability for Blockchain Based SLA Model

Observing the graph for Fig. 8, it may be seen that scalability is much higher if decentralization is compromised to access external data outside blockchain. An interesting fact to note down here is that the graph has started to get almost same values in the last three reading although the number of overall transactions were increasing in the same way as they were for the first three readings. This happened because we have set 0.5 value for mining diversity (refer to Table 5). We have developed and made this project available at [24].This value implies that at a time 50% of the mining power (or miners) may be used to confirm transactions into the block. Therefore at this stage, all the miners were fully occupied due higher number of incoming transactions from remote clients.

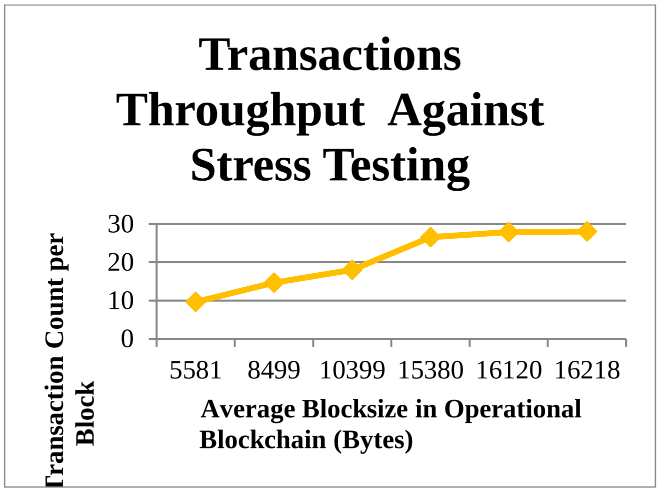


Fig. 8. Scalability for proposed model without oracles

6. Conclusion

The investigation in this research study aimed at the performance constraint of oracles problem in terms of scalability. It has been observed that under the given controlled environment as mentioned in this paper, decentralized oracles do not allow the blockchain network to scale up as much as those blockchain networks which access the off-chain data without decentralized oracles.

7. Future Work

We are interested to propose a new time efficient yet effective consensus algorithm to extend our blockchain based SLA model by adding decentralized external sources. The purpose is to build highly scalable network with the ability to solve oracle's problem.

8. References

[1] C. Forrest, "How one hacker stole $226K worth of cryptocurrency from Oracle servers", TechRepublic, https://www.techrepublic.com/article/how-one-hacker-stole-226k-worth-of-cryptocurrency-from-oracle-servers [Accessed on 11 Jan 2018].

[2] https://www.bitcoin.com [Accessed on 10 January 2022].

[3] T. A. Xu and J. Xu, "A short survey on business models of decentralized finance (defi) protocols," arXiv preprint 07742, 2022.

[4] D. Wang et al., "Towards a first step to understand flash loan and its applications in DeFi ecosystem", Proceedings of the 9th International Workshop on Security in Blockchain and Cloud Computing, pp. 23-28, 2021.

[5] M. Kumar, N. Nikhil, and R. Singh, "Decentralising finance using decentralised blockchain oracles", IEEE International Conference for Emerging Technology, pp. 1-4, 2020.

[6] K. Qin, L. Zhou, and A. Gervais, "Quantifying blockchain extractable value: how dark is the forest?", IEEE Symposium on Security and Privacy, 2022.

[7] A. Egberts, "The oracle problem - an analysis of how blockchain oracles undermine the advantages of decentralized ledger systems", SSRN, http://dx.doi.org/10.2139/ssrn.3382343, pp. 1-54, 2017.

[8] B. Liu, P. Szalachowski, and J. Zhou, "A first Look into DeFi Oracles", IEEE International Conference on Decentralized Applications and Infrastructures, 2021.

[9] B. Grooteman and T. U. Eindhoven, "Providing Trusted Datafeeds to the Blockchain". Master's Thesis, Eindhoven University of Technology, Netherlands, 2019.

[10] Heiss, J. Eberhardt, and S. Tai, "From oracles to trustworthy data on-chaining systems", IEEE International Conference on Blockchain, 2019.

[11] A. Schaad, T. Reski, and O. Winzenried, "Integration of a secure physical element as a trusted oracle in a hyperledger blockchain", Proceedings of the 16th International Joint Conference on e-Business and Telecommunications, Czech Republic 2019.

[12] M. Damjan, "The interface between blockchain and the real world", Ragion Pratica, vol. 2, pp. 379–406, 2018.

[13] A. Beniiche, "A study of blockchain oracles", preprint arXiv:2004.07140, 2020.

[14] R. M. Murimi and G. G. Wang, "On elastic incentives for blockchain oracles", Journal of Database Management, vol. 32, no. 1, pp. 1–26, 2021.

[15] R. Mühlberger et al., "Foundational oracle patterns: Connecting blockchain to the off-chain world", International Conference on Business Process Management. Springer, pp. 35–51, 2020.

[16] A. Pasdar, Z. Dong, and Y. C. Lee, “Blockchain Oracle Design Patterns”, preprint arXiv:2106.09349, 2021.

[17] G. Caldarelli and J. Ellul, “The blockchain oracle problem in decentralized finance - A multivocal approach”, Applied Sciences, vol. 11, no. 16, p. 7572, 2021.

[18] G. Caldarelli, “Understanding the blockchain oracle problem: A call for action”, Information (Basel), vol. 11, no. 11, p. 509, 2020.

[19] M. Kaleem and W. Shi, “Demystifying pythia: a survey of chainlink oracles usage on ethereum”, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 115–123, 2021.

[20] G. Caldarelli, “Real-world blockchain applications under the lens of the oracle problem. A systematic literature review”, IEEE International Conference on Technology Management, Operations and Decisions, 2020.

[21] H. Al-Breiki, M. H. U. Rehman, K. Salah, and D. Svetinovic, “Trustworthy blockchain oracles: Review, comparison, and open research challenges”, IEEE Access, vol. 8, pp. 85675–85685, 2020.

[22] K. M. Khan, J. Arshad, and M. M. Khan, “Simulation of transaction malleability attack for blockchain-based e-Voting”, Computers and Electrical Engineering, vol. 83, no. 106583, p. 106583, 2020.

[23] G. Greenspan, “MultiChain Private Blockchain - White Paper”, Multichain.com [Accessed on 10 January 2022].

[24] K. M. Khan, “SLA Monitoring”, https://github.com/KashifMehboob/SLA-Monitoring/tree/master [Accessed on 28 May 2021].