ORIGINAL ARTICLE

Bali Medical Journal (*Bali MedJ*) 2022, Volume 11, Number 3: 1195-1201
P-ISSN.2089-1180, E-ISSN: 2302-2914

Blockchain integration for hospital information system management

Naufal Adi Satrio[1],Sritrusta Sukaridhoto[1,2], M. Udin Harun Al Rasyid[1,2], Rizqi Putri Nourma Budiarti[3*], Ilham Achmad Al-Hafidz[1], Evianita Dewi Fajrianti[2]

[1]Information and Computer Engineering, Politeknik Elektronika Negeri Surabaya, Jl. Raya ITS, Keputih, Kec. Sukolilo, Surabaya,60111, Jawa Timur, Indonesia;
[2]Electronic Engineering, Politeknik Elektronika Negeri Surabaya,Jl. Raya ITS, Keputih, Kec. Sukolilo, Surabaya, 60111, JawaTimur, Indonesia;
[3]Information System, Universitas Nadhlatul Ulama Surabaya, Jl.Raya Jemursari No.57, Jemur Wonosari, Kec. Wonocolo,Surabaya, 60237, Jawa Timur, Indonesia;

*Corresponding author:
Rizqi Putri Nourma Budiarti;
Information System, Universitas Nadhlatul Ulama Surabaya, Jl.Raya Jemursari No.57, Jemur Wonosari, Kec. Wonocolo,Surabaya, 60237, Jawa Timur, Indonesia;
rizqi.putri.nb@unusa.ac.id

Received: 2022-07-16
Accepted: 2022-08-20
Published: 2022-09-14

ABSTRACT

Introduction: A lot of hospital management systems, especially open-source ones, still lack interoperability which is holding back the effectiveness of administration. This happens due to different regulations from hospital organizations. Therefore, there must be a system that facilitates interoperability. Blockchain is a decentralized ledger system, to insert the data from the blockchain all parties must agree that the data is valid thus the data can be inserted so, all the data are immutable thus every change can be audited.

Methods: We integrate the Hyperledger fabric network, an open-source modular blockchain platform, into an open-source hospital management system, openEMR. We take the changed data using debezium connect and Kafka and inserted it into the blockchain.

Results: Based on our test, we managed to get average read latency at 27ms, average read throughput at 36 Read per Second, average transaction latency at 45ms, average transaction throughput at 22 Transaction per Second, and average integrated system data transfer at 111.36ms.

Conclusions: All services deployed successfully at the Kubernetes without any error. All services work as they should be. One service can integrate through the internal network of the Kubernetes and from the outside cluster using ingress. OpenEMR can be used normally as indicated in the official documentation and the data change is stored in the blockchain.

Keywords: management system, hospital, blockchain.
Cite This Article: Satrio, N.A., Sukaridhoto, S., Rasyid, M.U.H.A., Budiarti, R.p.N., Al-Hafidz, I.A., Fajrianti, E.D. 2022. Blockchain integration for hospital information system management. *Bali Medical Journal* 11(3): 1195-1201. DOI: 10.15562/bmj.v11i3.3540

INTRODUCTION

According to the Regulation of the Indonesian Minister of Health No. 82 SIMRS (Hospital Management Information System) is a communication information technology that includes health information, namely data, information, indicators, procedures, technology, devices, and human resources that are managed in an integrated manner to direct actions or decisions that are useful for health development.[1]

Various hospitals maintain to use conventional techniques in managing hospital information systems, even though the government has provided facilities in the form of SIMRS as a tool for coordination, reporting, and administrative procedures to obtain accurate, accurate, and fast information. With the freedom from the government for hospitals to build independent information systems, it can bring up a better hospital managerial platform. To create an information system that can be accepted by government agencies, the information system that is built at least supports health services in hospitals includes:

1. speed, accuracy, integration, service improvement, efficiency improvement, ease of reporting in operational implementation.
2. speed of decision making, accuracy and speed of problem identification, and ease in formulating strategies in managerial implementation; and
3. work culture, transparency, coordination between units, understanding the system, and reducing administrative costs in the implementation of the organization.

Interoperability in health institutions still has some barriers. Over 78% of the barriers to interoperability were organizational issues. Followed by 50% was the standard issue of how interoperability is implemented, and 28% was about what technology is suitable for this type of implementation.[2] Interoperability on security, privacy, incentives, and management must be developed on a large scale to overcome those barriers. So, it's time to look for innovations to answer common problems in health institutions. One of the technologies that are a solution to improve institutional interoperability is by proposing blockchain. Where blockchain is a string that contains a list of transactions that occur.[3] Hyperledger Fabric is an open-source business-grade permissioned distributed ledger technology (DLT) platform that is optimized for corporate usage. It offers many important advantages over other prominent distributed ledger or blockchain systems.[4] This provides an advantage for implementing business activities from the hospital but still provides transparency to

clients who use the application. In addition to blockchain technology, microservice technology is used where the application architecture is broken down into small tasks that can be escalated, tested, and launched individually. By using this architecture, application development can be carried out with continuous delivery and better time to market.[5]

Roehrs et al. examine the digital storage of patient health records, which aims to support distributed PHR (Personal Health Records), where patients can maintain their medical history from any device and anywhere. And contribute to the connection of patient data between various health organizations.[6] The method used is OmniPHR, a distributed model for integrating PHR, where health records are developed by implementing blockchain. The results of the research demonstrate the feasibility of the model in maintaining distributed health records in an architectural model that promotes PHR with the elasticity and scalability of the solution.

Hidde et al. analyze the problems that exist in financial applications.[7] The problem raised is the existence of an asymmetry of information between the subject and the economic object involved, which can lead to double payments. To solve this problem, a framework, DecReg (Decentralized Registry), is proposed, which consists of a dedicated blockchain, a network of certified nodes, and three protocols to be executed on the blockchain. This research also presents a performance analysis of the proposed framework.

Seyoung et al. conduct problems regarding the limitations and synchronization of thousands or tens of IoT (Internet of Things) devices between each other are raised in this paper.[2] Then it is proposed to use a client-server model, namely blockchain technology to build this IoT system, where we can control and configure IoT devices. The proposed method is the Ethereum model which is used as a blockchain platform because by using its smart contracts, we can write our own Turing complete code to run on top of Ethereum. The goal is to simplify the management of IoT device configurations and the construction of key management systems. The weakness of the Ethereum blockchain is that it's still not fast enough for some domains and it doesn't support light clients.

The discussion from Akihiro Fujihara related to transportation technology in smart cities where self-driving cars are interconnected and IoT devices are developed for more efficient and safe transportation.[8] o reduce traffic congestion that occurs by efficiently controlling mobility, a crowdsensing system is proposed to collect traffic information. The method used is PoWaP (Proof of Work at Proximity), which utilizes the geographical proximity of the device collaboratively and automatically records which vehicles pass on the road segment into the blockchain. Another contribution is to improve anti-interference performance by crossreferencing blockchains from multiple road segments to each other. By storing traffic data chronologically into blocks on the blockchain, each beacon can reference abnormal amounts of traffic and traffic, such as traffic jams and accidents.

Then there are several studies that discuss the application of Kubernetes. The research raised is about resource scheduling schemes on Google Kubernetes or K8s. The problem that arises is, that when node selection occurs, the module only considers the current optimal node without looking at the use of costs from its resources.[9] Authors design and implement simulation experiments for model extraction from their scheduling modules. The K8s scheduling model was developed by combining the ant colony algorithm and the particle swarm optimization algorithm, where the node with the smallest objective function is chosen to propagate the Pod. From the research results, it is found that the proposed method is different from the original scheduling scheme with the advantages of reducing resource costs and maximum load of nodes. The discussion of the platform architecture is the reason why this research is proposed. The author proposes a Cloud-training and Edge-prediction framework by combining the resource-rich advantages of cloud servers with the stable network performance of edge computing technologies.[10] To build an edge computing platform and implement a machine learning model (Inception V3), Docker container technology, and Kubernetes container choreography are used. From experiments, it was found that the Edge Computing platform with limited resources has the ability to implement machine learning services. With the advantage of not having to worry about network conditions between the cloud and users, the Edge Computing platform can provide users with more stable machine learning services. Assuming that Kubernetes hides the complexity of its microservices while managing their availability, this paper investigates more architectures, such as public and private clouds, and conducts more experiments to evaluate the availability that Kubernetes provides for its managed microservices.[11] The intended investigation is in the form of evaluating the available capacity that can be achieved and the impact of the addition of the availability redundancy. Evaluation is also done by comparing it to the Availability Management Framework (AMF). The result is that in certain cases,

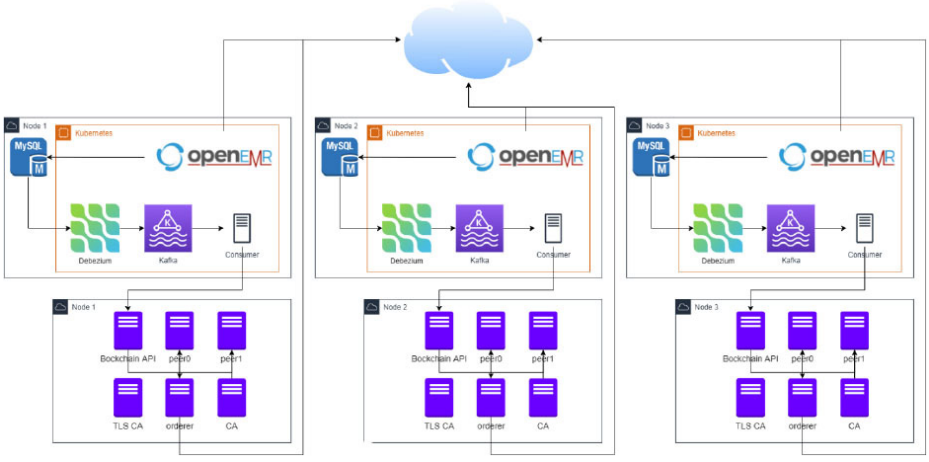


Figure 1. System Design.

there is a high probability of service outages on applications managed with Kubernetes.

METHODS

The system is built based on the system design shown in Figure 1. This research will utilize the openEMR hospital management system as an example of a hospital management system that has been integrated with a blockchain network. OpenEMR is an open-source hospital management system that is widely used. OpenEMR also has a large community that supports and contributes to its growth. In this journal, there will be three different blockchain nodes deployed, each representing a different hospital. We used Hyperledger Fabric for the blockchain platform because it is a permissioned blockchain. To integrate the hospital management system with the blockchain network, Kafka is used as a message broker, and debezium connect is used to monitor changes to the openEMR database, which are then received by the consumer service and transmitted to the existing blockchain API. The blockchain API will create a transaction containing a new value entered by the admin in the form of an asset.

To create the whole system, the following steps are required. I installed MySQL to store data input into openEMR. Create and mount an NFS folder that will be used as a certificate store. Installing Docker and Kubernetes to deploy the network and several additional services such as openEMR, kafka, Debezium Connect, and consumer service. It will deploy and configure blockchain networks. Create and deploy a blockchain API. Configure and deploy openEMR, kafka, and debezium connect. Develop and implement a consumer service.

Kubernetes Implementation

The first thing we did in the implementation process was to set up a k3s cluster. K3S is a lightweight Kubernetes distribution that is designed specifically for edge computing applications. Additionally, k3s offer a more straightforward setup process when compared to standard Kubernetes. We also used Docker to deploy hyper ledger fabric, which was done to ease the deployment of hyper ledger fabric while taking production considerations into account. After that, we create an NFS folder on one of the nodes and connect the rest of the nodes that will be inside the network. We also download the hyper ledger fabric image from the official repository into the NFS folder so it can be used by all connected nodes in the network. There are three folders we will get after downloading the file from the official repository: a bin, and a configuration folder. The bin folder will consist of all the binaries that we will use to create and operate the network, and the config folder will consist of all the configurations that we'll need.

Hyperledger Configuration

After all of the environments have been established, we can proceed to the

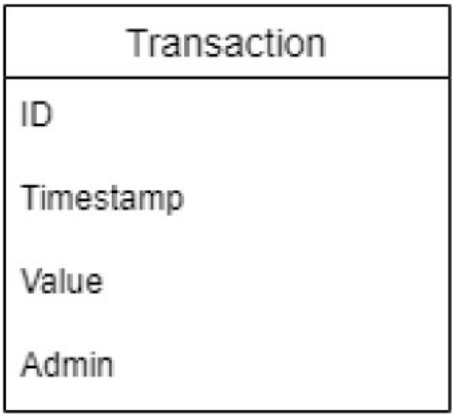


Figure 2. Transaction Data Model.

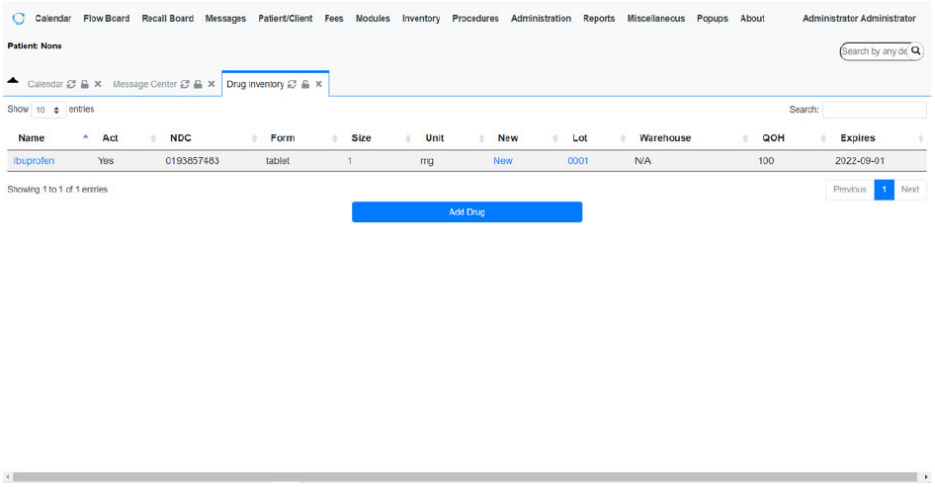


Figure 3. OpenEMR drug inventory.

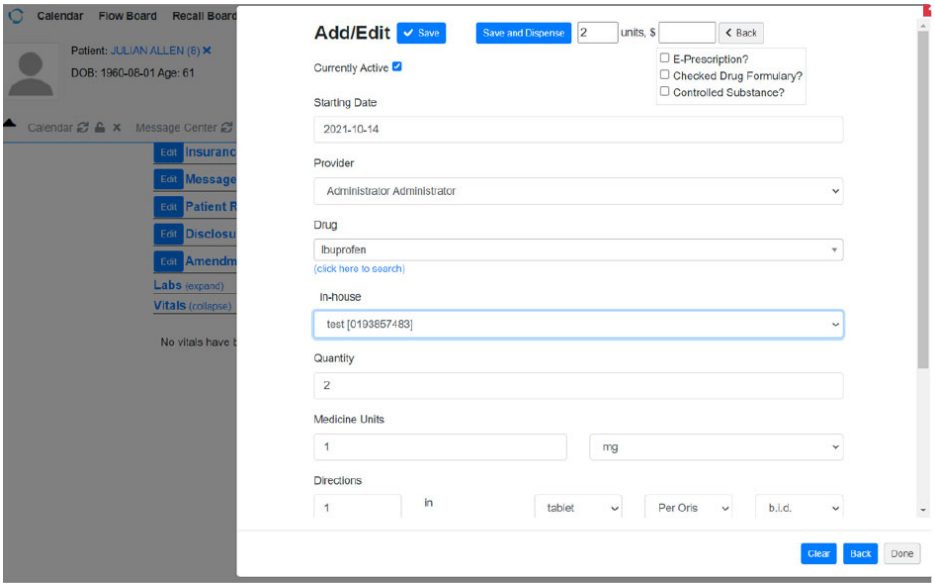


Figure 4. OpenEMR drug dispense.

next step, which is the deployment and configuration of the hyper ledger fabric network. To deploy and configure the network, there are a few steps that must be completed. Each node will have a certificate authority (CA) that will be used to register every service that is permitted to operate within the network; an orderer service that will accept transaction invocations, pack all of the transactions into a block, and distribute them to the peer service and another orderer in the network; and last but not least, two peers who will validate and write the block into the blockchain. For security reasons, every service will communicate over the TLS layer. Now, in each node, we create a Docker Compose file that meets the requirements of the requirement.

Network Configuration

To build the network, first, we deploy CA in every organization that will be connected to the network and TLS CA on one of the nodes. After that, we generate an identity for every service that is running in the node. In our case, we will create identities for one orderer, two peers, and one admin for every node. Next, we bring up the rest of the services in every node. Now, we generate the first block that consists of the network configuration and channel name. Afterward, we connect every service to the channel except the CA. When this process is completed, we are ready to use the network.

Chaincode Development

Chaincode is a term that has been given by Hyperledger Fabric for smart contracts. A smart contract is a group of functions that are embedded in the blockchain network that will be executed to create a transaction. Before creating the function, we first define the data model as seen in figure 2.

For this particular case, every transaction will consist of an ID, a timestamp, a value, and an admin. The ID represents the identification of the admin that changed the data from openEMR. This will also be the asset identification. The timestamp represents the time when the transaction happened. The value represents the new data that was inserted into the blockchain. Admin represents the admin username in the openEMR. After that, we start working on the function. These are the following functions that can be called.

InitLedger, this function will be called the first time after the chaincode lifecycle is done. It contains the first transaction for this chaincode.

CreateAsset, this function is used to initiate the creation of an asset. In the blockchain, any new admin who has no prior experience of inputting data must use this method in order to create a new record.

ReadAsset, this function retrieves a particular asset from the blockchain.

UpdateAsset, this function is used to do asset updates. If the admin who enters the data already has a record, the asset will be modified rather than created. In this manner, no asset creation is needed for each activity, and each action can be tracked via transaction history.

DeleteAsset, this function is used to remove an asset from the blockchain in a soft manner. Due to the fact that this is a soft delete, the data will remain on the blockchain but may be disregarded.

AssetExists, this function will check if there is an asset with a specific identification. This function will be called when the createAsset function is triggered to make sure that every asset has no redundancy.

GetAllAssets,this function will get all asset on its latest state.

Chaincode Lifecycle

The chain code lifecycle is a process that enables several nodes to agree on the operation of a chain code before it is deployed on a channel. The chain code will be packed and distributed to every node, but since we use an NFS server, the distribution process can be skipped. After that, chain code will be installed on every peer that participates in the network. Next, one of the peers in every node approves the installed chain code to be used. After there are enough nodes that approve, the chain code is committed to the network. The last step is to trigger the initLedger function as a chain code initial transaction.

Blockchain API

BlockchainAPI is a REST API designed to store data on the blockchain using HTTP. To initiate a transaction, we must execute a bash script that invokes the appropriate function. To initiate a transaction, the script will invoke a function with the sender's identity that will sign the transaction and transmit it to all nodes in the network. If the transaction is verified by a majority of nodes, it is added to the blockchain. Given the need to run a bash script, an API is required. Nodejs is used as a middleware, which enables the bash script to be called through normal HTTP.

Record Data Changes

Getting the most recent data from the database requires the use of Kafka and debezium connect. These services will extract new data from the database, which means that anytime new data is produced or changed, these services will store it as topics. To begin with, Kafka will be installed on Kubernetes. Strimzi is used to make Kafka deployment easier. Strimzi is a Kube-native Kafka management system that enables the administration of brokers, topics, and users. Strimzi provides container images and operators for running Kafka on Kubernetes. The operators provided with Strimzi are purpose-built with specialist operational knowledge to effectively manage Kafka.[12] A data hub between the database and Kafka is required to monitor the database. Here is when debezium connect comes in handy. It will monitor database updates and write them to Kafka. From here, consumer service is used to get data from Kafka and call the blockchain API to insert data into the blockchain.

RESULT

In this paper, all the systems are integrated and tested using table 1 below. The system has integrated and worked as expected where we can work normally using OpenEMR as seen in figure 3 and figure 4 and every change of the data stored in the blockchain seamlessly as seen in figure 5. One of the things that we are most concerned about is data transfer. The data transfer has to be small enough so there

Table 1. **Server testing specification.**

Node	Spec
Node 1	CPU intel core i7 RAM 2x16GB SSD 512GB
Node 2	CPU intel core i7 RAM 2x16GB SSD 512GB
Node 3	CPU intel core i7 RAM 2x16GB SSD 512GB

Table 2. **Test Result Comparison.**

Operation	Our System	MediTrans
Read	27 ms	0.98 s
Write	45 ms	2.85 s

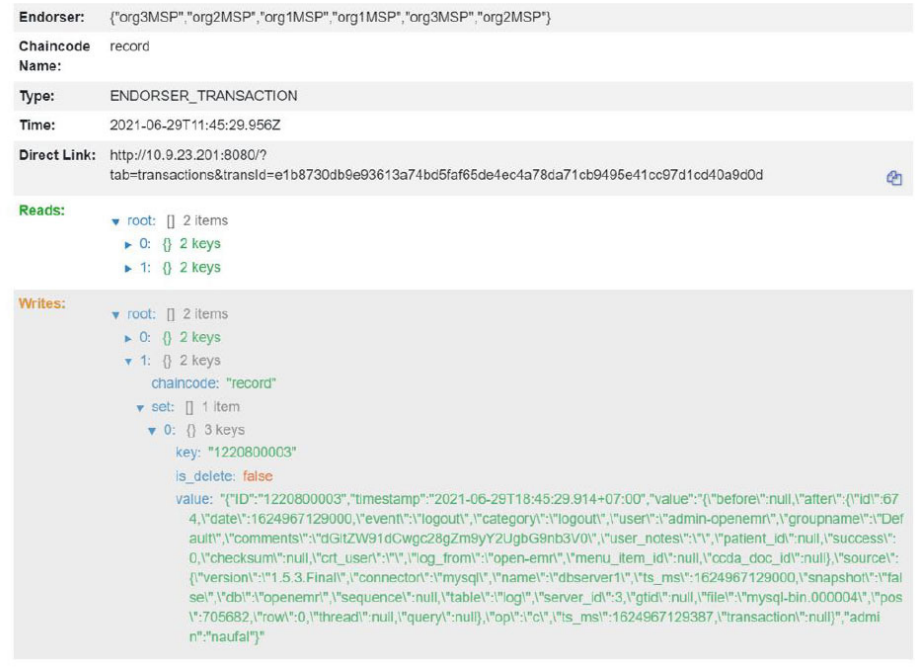


Figure 5. OpenEMR data record in blockchain.

is no data loss during the transmission process. There will be two testing procedures that will be conducted. The first is blockchain performance, and the second is integrated system data transfer.

Based on this paper[13], there are four aspects that depict blockchain performance. Read latency, read throughput, transaction latency, and transaction throughput. Read latency is the time interval between the time the read request is sent and the time the response is received. Read throughput is a metric that indicates the number of reading operations done during a certain time frame, represented in reads per second (RPS). Transaction latency is the time required for a transaction to become valid throughout a network. The time period covered by the measurement is from the point at which the request is filed to the point at which the result is generally accessible on the network.

In the read latency test, we obtained a value of 23 ms for 50 consecutive reads. As the number of reads increased, the elapsed time also increased and ultimately stabilized at roughly 27 ms.

Based on the latency test, we can estimate that the read throughput will be 45 reads per second for 50 consecutive reads and that it will stabilize at roughly 36 reads per second as the number of reads increases.

We received a result of 39ms for 50 consecutive transactions in the transaction latency test. The elapsed time increased as the number of transactions increased and eventually steadied at around 45ms.

The latency test indicates that the transaction throughput will be 25 transactions per second for 50 consecutive reads and will stabilize at around 22 transactions per second as the number of transactions grows.

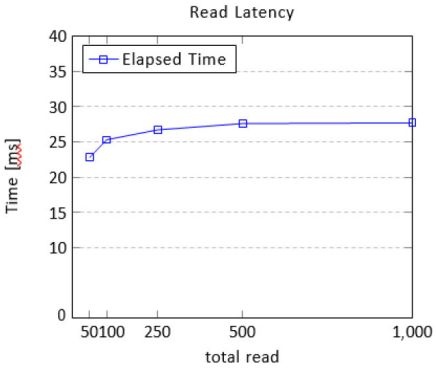
From this test, we compare our result with this mediTrans[14] which has an average of 2.85 s when writing data to the blockchain and an average of 0.98 s when reading data from the blockchain. The comparison can be seen in Table 2.

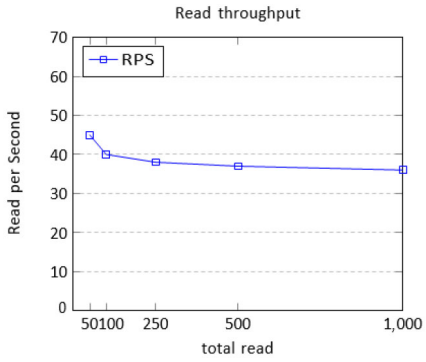
In the integrated system data transfer test, we took 10 data samples as a sample with the scenario where we change the data in the Hospital Management System (openEMR). When we do update or change the data in openEMR, the data is transferred into the blockchain, and we make changes there too. The time from when we change the data in openEMR until the blockchain makes the changes are recorded and displayed in the graph below.

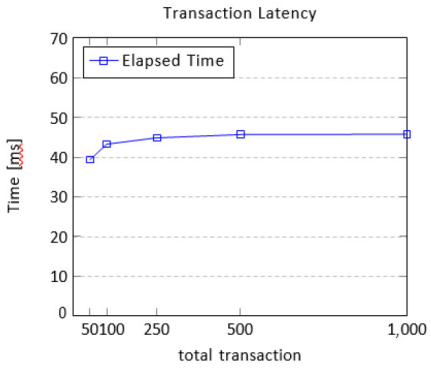
As we can see from the graph, the average data transfer time is 111.36 ms, with the highest time being 151.57 ms and the lowest time being 97.88 ms. All the data that has been submitted is successfully recorded on the blockchain we build. We also tested manipulating data through the MySQL client. For testing purposes, we deploy phpMyAdmin, one of the most popular MySQL clients. As we manipulate the data, all the changes are managed to be inserted into the blockchain.

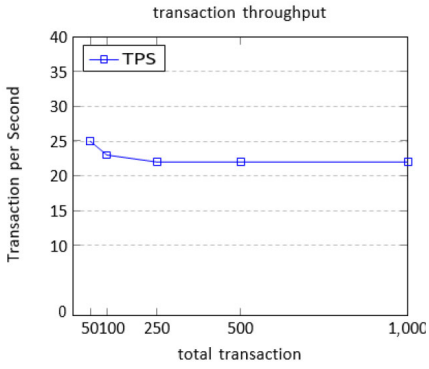
DISCUSSION

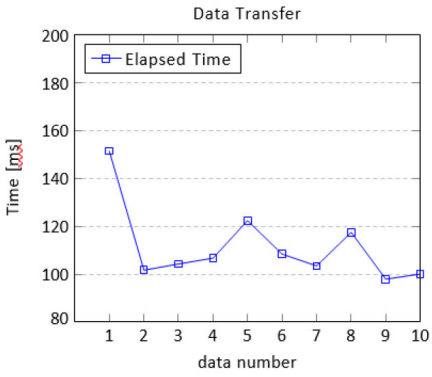
A blockchain could be a conveyed exchange log that empowers each hub of P2P (peer-to-peer) arrange to claim a comparative duplicate of a specific record. The record is confirmed and synchronized with the creation of a modern square through an agreement convention. The agreement convention presented by a conveyed P2P blockchain organizes dispenses with the perquisite a centralized hub or a dependable substance, such as a government office. This sophisticated technology is becoming a revolution in











every aspect, including in healthcare facilities. Blockchain technology is a feasible technique in a real-time environment, providing confidentiality, and security on the patient's data.[15]

Blockchain is one of the modalities that can be done to handle the medical records of patients so it will be secure, accessible, confidential, and decentralized. Based on this study, all services were successfully deployed to Kubernetes with no errors. Kubernetes is an open-source platform, a container orchestration tool, that can be used to manage applications and is very useful for data management and discovery. In addition, this study achieved an average read latency of 27ms, an average read throughput of 36 reads per second, an average transaction latency of 45ms, an average transaction throughput of 22 transactions per second, and an average integrated system data transfer of 111.36ms based on our test. According to previous studies, Kubernetes also has an effective CPU usage pod when scalability has been settled. This is due to the container that has been distributed to each worker in Kubernetes.[16]

Installation and configuration are carried out on 1 (one) server that functions as a master node for controllers on the deployed computer, 2 (two) servers that function as worker nodes to execute processes that have been run by the master node, 1 (one) client that is enabled to use which has been specified. Another previous study found that the Kubernetes system was able to provide requests from the client when all the nodes are in active condition or one node is inactive, including when the container condition is down. This can be achieved as a result of the availability function provided by Kubernetes. If one host is down, the service will be replaced with an active host. When one of the nodes is down, the container on that node performs self-healing by creating a new container that automatically appears on one of the active server nodes. In addition, when the containers on all server nodes are down, the system will automatically perform self-healing by creating new containers. This is a deploy technique that can prevent a single point of failure and when one or all of the nodes forming the deployment are disrupted (inactive), the system will automatically self-healing to form new containers.[17]

CONCLUSION

All services were successfully deployed to Kubernetes with no errors. All service work is as it should be. One service can integrate with another through the internal network of the Kubernetes and from outside the cluster using ingress. OpenEMR can be used normally as indicated in the official documentation and the data change is stored in the blockchain. We achieved an average read latency of 27ms, an average read throughput of 36 reads per second, an average transaction latency of 45ms, an average transaction throughput of 22 transactions per second, and an average integrated system data transfer of 111.36ms based on our test.

DISCLOSURE

Author contribution

All authors contributed equally in the preparation of the manuscript, seen, revised, and approved the final version herewith submitted for publication.

Funding

This work was supported by research funding from the Indonesian Ministry of Research and Technology/National Foundation in Research and Innovation (Kemenristek/BRIN) (Grant number: PRJ-11/LPDP/2020).

Conflict of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. The researchers also confirm their independence from funders and sponsors.

Ethical Consideration

The Ethical Clearance was obtained from the Ministry of Health of Republic Indonesia (Ethical Clearance (EC) no. LB.02.01/2/KE.351/2020) and the Food and Drug Advisory Agency (approval no. R-RG.01.06.1.3.05.20.156).

ACKNOWLEDGMENTS

We sincerely thank to Politeknik Elektronika Negeri Surabaya for the support so that this paper can be completed. This script would hopefully give a positive contribution to the educational development, public health, or those who are willing to conduct further research.

REFERENCES

1. Kementrian Kesehatan RI, Peraturan Menteri Kesehatan RI Nomor 82 tentang Sistem Informasi Manajemen Rumah Sakit.
2. Oyeyemi, P. Scott, Interoperability in health and social care: organisational issues are the biggest challenge, BMJ Health & Care Informatics 25 (3) (2018) 196–197. doi:10.14236/jhi.v25i3.1024.URL https://informatics.bmj.com/lookup/doi/10.14236/jhi. v25i3.1024
3. S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, Decentralized Business Review (2008) 21260.
4. S. Aggarwal, N. Kumar, Chapter sixteen hyperledger working model., in: S. Aggarwal, N. Kumar, P. Raj (Eds.), The Blockchain Technology for Secure and Smart Applications across Industry Verticals, Vol. 121 of Advances in Computers, Elsevier, 2021, pp. 323–343. doi:https://doi.org/10.1016/bs.adcom.2020.08.016. URL https://www.sciencedirect.com/science/article/pii/S0065245820300711
5. X. Larrucea, I. Santamaria, R. Colomo-Palacios, C. Ebert, Microservices, IEEE Software 35 (3) (2018) 96–100. doi:10.1109/MS.2018.2141030.
6. A. Roehrs, C. A. da Costa, R. da Rosa Righi, Omniphr: A distributed architecture model to integrate personal health records, Journal of Biomedical Informatics 71 (2017) 70–81. doi:https://doi.org/10.1016/j.jbi.2017.05.012. URL https://www.sciencedirect.com/science/article/pii/ S1532046417301089
7. H. Lycklama `a Nijeholt, J. Oudejans, Z. Erkin, Decreg: A framework for preventing double-financing using blockchain technology, in: Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts, BCC '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 29–34. doi:10.1145/3055518.3055529. URL https://doi.org/10.1145/3055518.3055529
8. A. Fujihara, Powap: Proof of work at proximity for a crowdsensing system for collaborative traffic information gathering, Internet of Things 10 (2020) 100046, special Issue of the Elsevier IoT Journal on Blockchain Applications in IoT Environments. doi:https://doi.org/10.1016/j.iot.2019.02.006. URL https://www.sciencedirect.com/science/article/pii/S254266051830177X
9. Z. Wei-guo, M. Xi-lin, Z. Jin-zhong, Research on kubernetes' resource scheduling scheme, ICCNS 2018, Association for Computing Machinery, New York, NY, USA, 2018. doi:10.1145/3290480.3290507. URL https://doi.org/10.1145/3290480.3290507
10. Y. Huang, K. cai, R. Zong, Y. Mao, Design and implementation of an edge computing platform architecture using docker and kubernetes for machine learning, HP3C '19, Association for Computing Machinery, New York, NY, USA, 2019. doi:10.1145/3318265.3318288. URL https://doi.org/10.1145/3318265.3318288
11. L. A. Vayghan, M. A. Saied, M. Toeroe, F. Khendek, Kubernetes as an availability manager for microservice applications, arXiv preprint arXiv:1901.04946.
12. Strimzi, Using Strimzi (0.26.0). URL https://strimzi.io/docs/operators/latest/using.html
13. Hyperledger Performance and Scale Working Group, Hyperledger Blockchain Performance Metrics.
14. M. George, A. Mary Chacko, MediTrans—Patient-centric interoperability through blockchain, International Journal of Network Management doi:10.1002/nem.2187. URLhttps://onlinelibrary.wiley.com/doi/10.1002/nem.218718
15. Mondal S, Shafi M, Gupta S, Gupta SK. Blockchain Based Secure Architecture for Electronic Healthcare Record Management. GMSARN Int J. 2022;16(4):413–26.
16. Dewi LP, Noertjahyana A, Palit HN, Yedutun K. Server Scalability Using Kubernetes. 2019 4th Technol Innov Manag Eng Sci Int Conf. 2019;1–4.
17. Widyawati L, Santoso H, Budika H. ANALISA PENERAPAN SERVER DEPLOYMENT MENGGUNAKAN KUBERNETES UNTUK MENGHINDARI SINGLE OF FAILURE. 2021;3(1):267–71.