J Inf Process Syst, Vol.18, No.6, pp.741~754, December 2022
https://doi.org/10.3745/JIPS.03.0179

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

PBFT Blockchain-Based OpenStack Identity Service

Youngjong Kim[1,*], Sungil Jang[1], Myung Ho Kim[1], and Jinho Park[2]

Abstract

Openstack is widely used as a representative open-source infrastructure of the service (IaaS) platform. The Openstack Identity Service is a centralized approach component based on the token including the Memcached for cache, which is the in-memory key-value store. Token validation requests are concentrated on the centralized server as the number of differently encrypted tokens increases. This paper proposes the practical Byzantine fault tolerance (PBFT) blockchain-based Openstack Identity Service, which can improve the performance efficiency and reduce security vulnerabilities through a PBFT blockchain framework-based decentralized approach. The experiment conducted by using the Apache JMeter demonstrated that latency was improved by more than 33.99% and 72.57% in the PBFT blockchain-based Openstack Identity Service, compared to the Openstack Identity Service, for 500 and 1,000 differently encrypted tokens, respectively.

Keywords

Blockchain, Cloud, Decentralization, Openstack Identity Service, PBFT

1. Introduction

Openstack, which is a representative open-source infrastructure of the service (IaaS) platform, is composed of nodes such as a controller and a computer-based on each role [1]. Each node is composed of Keystone, Nova, Cinder, and Neutron, etc., which are components of Openstack [2], and these constituent components control validation and authorization, etc., with integration through Keystone [3], which is the component that provides the Identity Service.

The Openstack Identity Service, Keystone, is a centralized approach component [4] that offers Identity Service based on a token [5] in order to deal with validation and authorization, etc., upon requests between Openstack components. It saves token by using a database such as MySQL as its backend [6] and provides the Identity Service for API requests between Openstack services by using Memcached [7], which is a memory object caching system for the token cache.

Keystone, which is a centralized approach component that uses the database as the backend, suffers from the problem of performance degradation as tokens are saved in the centralized server, and because the token validation requests for each service of Openstack are concentrated on the centralized server even though it is composed of Memcached. Memcached is the in-memory key-value store [8] for cache, and data are lost upon termination. The blockchain, as a decentralized approach, can be used to provide

Manuscript received October 8, 2021; first revision December 8, 2021; accepted December 19, 2021.

* **Corresponding Author:** Youngjong Kim (youngjong@ssu.ac.kr)
[1] School of Software, Soongsil University, Seoul, South Korea (youngjong@ssu.ac.kr, sungil@soongsil.ac.kr, kmh@ssu.ac.kr)
[2] Dept. of Multimedia Engineering, Dongguk University, Seoul, South Korea (gomalove@dgu.edu)

the Identity Service as a data duplication approach [9] for decentralizing requests [10].

There have been several studies [11] on authentication based on blockchain in the Internet of Things (IoT) cloud platform [12] and the blockchain-based digital identity that protects personal information [13]. However, there have not yet been any studies on the blockchain-based decentralization approach in the cache and data duplication, performance, and security for the Openstack Identity Service, which is the IaaS platform.

The blockchain can be operated by members of the public, a consortium/community, or private blockchains [14]. The PBFT [15] algorithm ensures safety, which means that a consensus is reached among the nodes, all of which have the same value. This algorithm [16] is designed as an asynchronous one [17] to enable efficient transactions, thus sacrificing a degree of liveness, which means that a consensus must be reached among all nodes if there is no problem. Public blockchains face problems with efficient transactions due to the overload of the consensus algorithm [18].

This study proposes the PBFT blockchain-based Openstack Identity Service, which is composed of private blockchains for efficient transactions, and uses the PBFT algorithm which is a consensus algorithm designed to enable efficient transactions. The PBFT blockchain-based OpenStack Identity Service proposed in this paper is the PBFT blockchain-based cache (PBc) implemented on the basis of the PBFT blockchain framework [19]. The proposed method can improve the performance (of Openstack Identity Service) by decentralizing the token validation requests to all PBc Peers on the PBc Peer Network and reducing the security vulnerability through the token database stored on all PBc Peers.

The paper is organized as follows: Section 2 describes, compares, and matches methods; Section 3 presents the experimental settings and results, and analysis and discussion of the results; and Section 4 presents the conclusion.

2. Materials and Methods

2.1 Openstack Identity Service

Openstack components send the encrypted token to the keystone along with identity requests, and the keystone returns the decrypted token value to the Openstack components. At this moment, the decrypted token value is cached on to the Memcached of Keystone and, subsequently, it brings the decrypted token from the Memcached of Keystone without going through the decryption process.

Fig. 1 illustrates the operation process of the Openstack Identity Service regarding the OpenStack API Node with Nova, which is one of the core components of Openstack.

The working procedure of the Keystone, which is the Openstack Identity Service procedure, is described in ⓐ-ⓓ [20].

ⓐ nova-api of the Openstack API Node receives the API request message from the API Gateway.

ⓑ nova-api of the Openstack API Node sets the value, which is the X-Auth-Token of the delivered request message header, and converted into SHA-256 as the key value, and checks the corresponding key value in the Memcached of Keystone. If there were a key value, the token would be validated by using the value saved in the Memcached of Keystone; whereas if there were no key value, it would go on to Phase ⓒ.

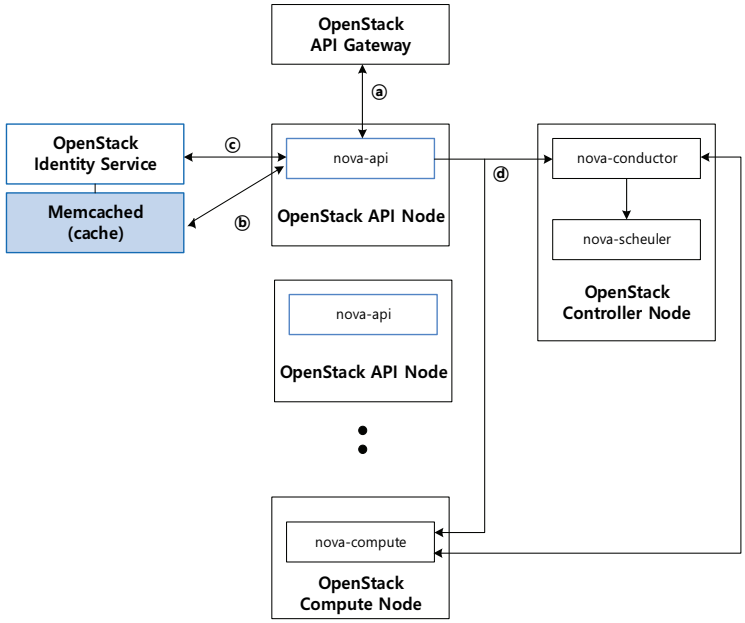

Fig. 1. Openstack Identity Service.

ⓒ nova-api of the Openstack API Node queries the value of X-Auth-Token to the Keystone that is the OpenStack Identity Service, and the Keystone delivers the token for the value of X-Auth-Token. Then, nova-api validates the token using the received token of X-Auth-Token, stores the hashed value in the process ⓑ as the key, and stores the received token of X-Auth-Token as the value in the Memcached of Keystone.

ⓓ Upon completion of the token validation for the request message, the request message is delivered to the nova-conductor of the Openstack Controller Node and the nova-compute of the Compute node.

2.2 PBFT Blockchain-based Openstack Identity Service

2.2.1 PBFT blockchain framework

The PBFT blockchain framework proposed in this paper upgrades RocksDB [21], which is the backend to version 5.17.2, by forking the PBFT-based Hyperledger Fabric 0.6, and realizes the PBFT blockchain framework that Go [22] which is a Chaincode programming language [23] is upgraded to the version 1.11.1.

The peers of the PBFT blockchains framework consist of Non-validating Peers, Validating Peers, and a Leader elected from among the Validating Peers. The Validating Network of PBFT blockchains framework comprises the Validating Peers and Leader that participate in the consensus process, but excludes Non-validating Peers that do not participate in the consensus process. The PBFT algorithm-based Block Creation procedures[24] are described in ⓐ-ⓓ.

ⓐ Once the Validating Network receives requests from the Client, it collects the corresponding requests and makes them into a block (request).

ⓑ The leader delivers the block to the Validating Peers (pre-prepare).

ⓒ The peers inform the other peers that they have received the block (prepare).

ⓓ In the event that more than two-thirds of the peers receive the block, the corresponding block is validated, and then the result is disseminated to the peers (commit).

It can be confirmed that the process from ⓐ to ⓓ is identical to the process of creating a block based on the existing PBFT algorithm.

2.2.2 PBFT blockchain-based Openstack Identity Service

Fig. 2 shows the architecture of the PBFT blockchain-based Openstack Identity Service regarding the PBc Peer of the API Node implemented based on the PBFT blockchain framework, which has been modified for the Openstack Identity Service with an example of Nova, one of the core components of Openstack.

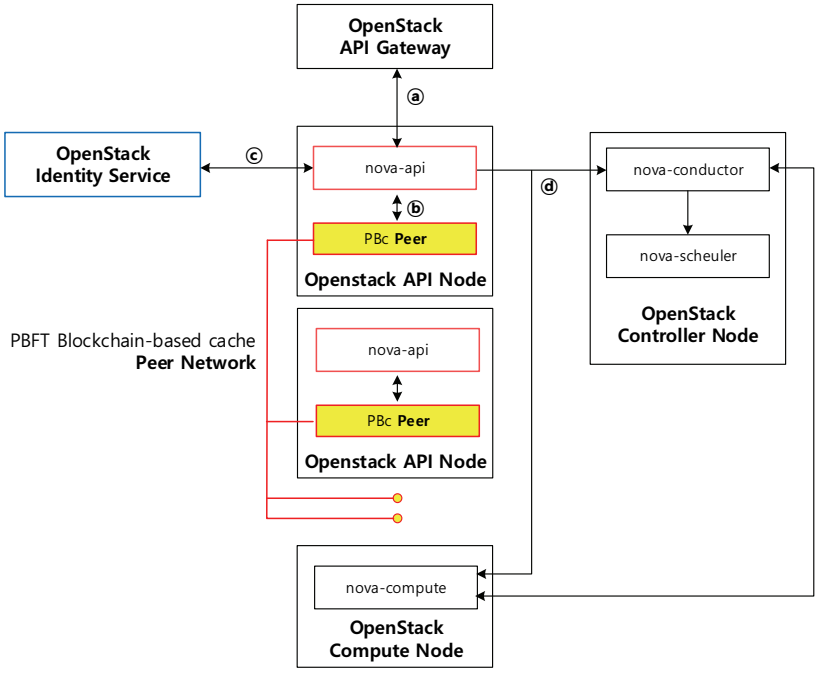

Fig. 2. PBFT blockchain-based Openstack Identity Service.

The way of the decrypted token value is delivered by transferring the encrypted token in Keystone that is Openstack Identify Service is identical, but the delivered token value is saved in the token database of the PBc Peer of the API Node, rather than in the Memcached of Keystone, and each of the PBc Peers in the Peer Network are synchronized with one another.

The token database is replicated amongst all the PBc Peers of the Peer Network, making it possible to reduce the security risk of the token database. The working procedures of the PBFT blockchain-based Openstack Identity Service are described in ⓐ-ⓓ.

ⓐ nova-api receives the API request message from the API Gateway.

ⓑ nova-api hashes the X-Auth-Token of the delivered request message header by SHA-256 and uses

it as a key value, and queries the corresponding key value in the token database of the PBc Peer on the API Node. When there was a value, the token would be validated by using the token value saved in its own token database; whereas if there was no key value, it would go on to Phase ⓒ.

ⓒ nova-api of the Openstack API Node queries the value of X-Auth-Token to the Keystone, which is the OpenStack Identity Service, and the Keystone delivers the token for the value of the X-Auth-Token. nova-api validates the token using the received token of the value of X-Auth-Token and saves the hashed value in the process ⓑ as the key and the received token of X-Auth-Token as the value to the token database of the PBc Peer on the API Node, rather than saving it to the Memcached of Keystone.

ⓓ Upon completion of the token validation for the request message, the request message is delivered to the nova-conductor of the Openstack Controller Node and the nova-compute of the Compute node.

Like the Openstack Identity Service, since validation is made by using the value hashed from the X-Auth-Token value as a key, and the value delivered from the Keystone as a value in the same way as the Openstack Identity Service, all requests validate the token in the same way as the existing Openstack Identity Service.

3. Results and Discussions

3.1 Experimental Setting

The experiment set is composed of the Controller Node and the Compute Node, as shown in Fig. 3, in the same way as the actual environment, the API Node and Load Generator dealing with the API requests. HAProxy [25] is set up in the Controller Node, which serves as the API Gateway, while 4 API Nodes are set up. The requests created from the experiment are disseminated and processed in the 4 API Nodes [26].

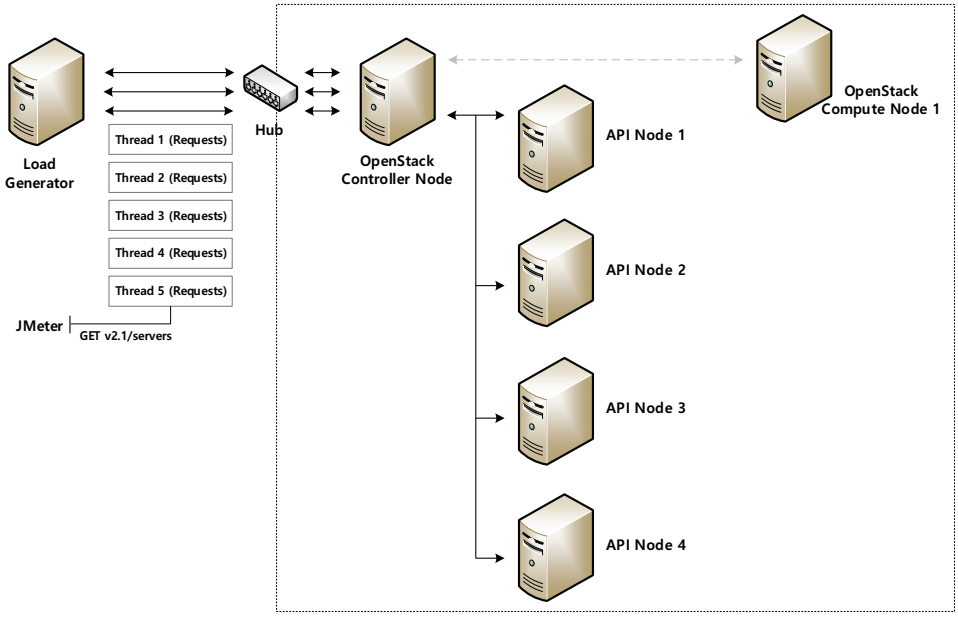

Fig. 3. Schematic diagram of experiment setting.

The composition of hardware and software in Fig. 3 is as shown in Table 1. Like the environment in which actual requests are generated, the experiment was conducted by sampling the token operation procedure actually created in Nova, which is a core component, and by using the differently encrypted token and generating different requests. For the experiment, the Request Query Example using Nova, one of the Openstack core components, is shown in Fig. 4.

Table 1. Hardware and software characteristics

Criteria	Hardware		EA	OS
	CPU (GHz)	RAM (GB)		
Controller Node	Intel 1.7	32	1	
Compute Node	Intel 2.4	8	1	Ubuntu 18.04
	Intel 2.4	8	1	(Openstack Rocky)
API Node	Intel 3.2	8	1	
	Intel 2.9	8	2	
Load Generator	Intel 3.4	16	1	Windows 10

```
Connection: keep-alive
Content-Type: application/json
X-Auth-Token:
gAAAAABdJwqAZjMcLuTwjWZYkPW6tQYZ0j1irMBwhR0kx6jcYu-
brSqG6AH4zobxt1u54zembPw9CwXBwwMDk-
vNL3DOxQ5B5glSZ4vNZv3ZWRTXUR5nySJP1uhlOuX6UdiVU-
bUms5JJfh2t3fJxhFeUq50gbhXSLd85YYLNb8VjFPolBFNBFs
Host: controllr_Node:8774
User-Agent: Apache-HttpClient/4.5.7 (Java/1.8.0_201)
```

Fig. 4. Request Query example.

The Request makes the request query, as shown in Fig. 4, by using version 5.1.1 of Apache JMeter [27] at the load generator, and the HTTP GET Request "GET http://controller_Node:8774/v2.1/servers" is sent to the API Gateway. The Port of 8774 of the HTTP GET Request is used by nova-api. The experimental scenario is described in ⓐ-ⓔ.

ⓐ Requests generated with differently encrypted tokens are sent to the Controller Node, which serves as the API Gateway, at intervals of 1 second.

ⓑ The delivered requests are disseminated to the 4 API Nodes via HAProxy of the Controller Node.

ⓒ Each API Node checks the encrypted Token of the delivered request.

ⓓ If it were a previously decrypted token, it would be used immediately, whereas if it were not a decrypted token, a token decryption request would be sent to the Keystone which is the Openstack Identity Service.

ⓔ If the decrypted token received through the Keystone was validated as a right token, the request would be processed and then a response would be made.

The experimental values according to the number of tokens used for the experiment are as shown in Table 2.

Table 2. Variables by number of tokens used in the experiment

Number oft	Number of threads	Ramp-up period (s)	Number of requests
1	5	10	100,000
10	5	10	100,000
100	5	10	100,000
500	5	10	100,000
1,000	5	10	100,000

3.2 Results of Experiment

When the number of differently encrypted tokens is 1, 10, 100, 500, and 1,000, as shown in Table 2, for the Requests of Fig. 4 combined with Table 1, the results of the experiment on 100,000 requests according to the scenario are as shown in Fig. 5. The black line indicates the Openstack Identity Service, while the red line indicates the PBFT Blockchain-based Openstack Identity Service.

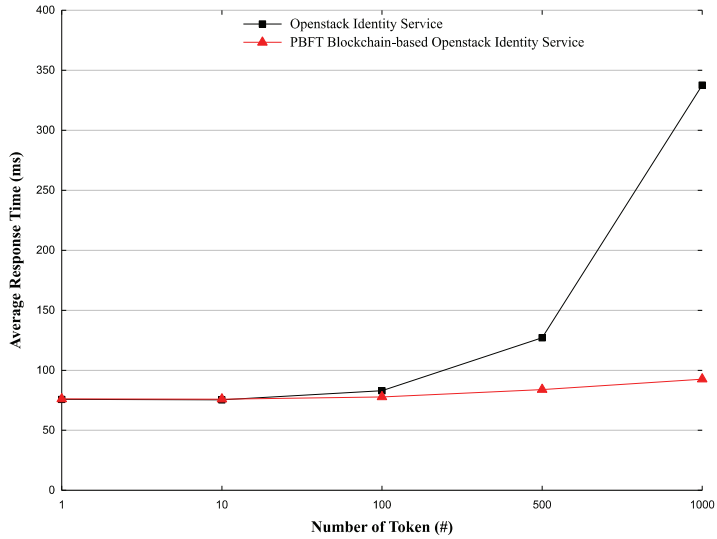

Fig. 5. Comparison of average latency.

As shown in Fig. 5, the results of the experiment confirm that there was no difference in latency between the Openstack Identity Service and the PBFT Blockchain-based Openstack Identity Service when the number of the differently encrypted tokens ranged from one to ten. However, the PBFT Blockchain-based Openstack Identity Service was reduced by approximately 6% in terms of average latency, compared to the Openstack Identity Service, when the number of tokens was 100.

The PBFT Blockchain-based Openstack Identity Service was reduced by approximately 33.99% in terms of average latency, compared to the Openstack Identity Service, when the number of tokens was 500, while the PBFT Blockchain-based Identity Service was reduced by approximately 72.57% in terms of average latency, compared to the Openstack Identity Service, when the number of tokens was 1,000.

Furthermore, it could be confirmed that the PBFT Blockchain-based Openstack Identity Service maintained constant latency and offered a stable service regardless of the number of differently decrypted tokens. Table 3 summarizes the experimental results of Fig. 5.

Table 3. Comparison of average latency (unit: ms)

Number of tokens	Openstack Identity Service	PBFT blockchain-based Identity Service
1	75.75	76.16
10	75.40	75.98
100	82.97	77.85
500	127.15	83.92
1,000	337.66	92.61

The distribution of latencies of the Openstack Identity Service for 100,000 requests when the number of T\tokens is 1,000 is shown in Fig. 6. The distribution of latencies of the PBFT Blockchain-based Openstack Identity Service for 100,000 requests when the number of tokens is 1,000 is shown in Fig. 7.

It could be confirmed that the existing Openstack Identity Service showed the distribution of latency to be approximately 337.66 ms on average for 100,000 requests when the number of differently encrypted tokens was 1,000, as shown in Fig. 6.

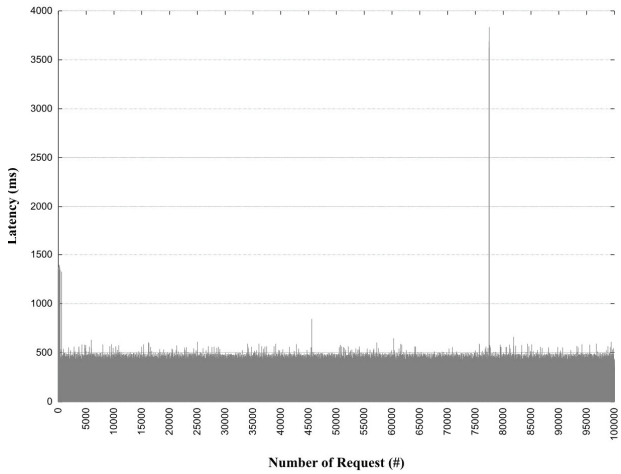

Fig. 6. Distribution of latencies of the Openstack Identity Service for 100,000 requests.

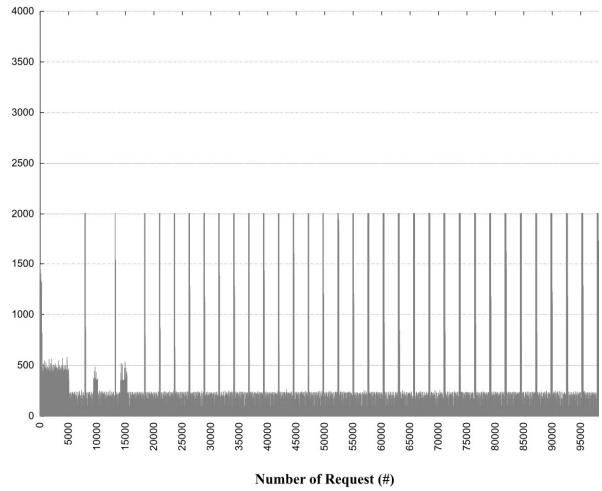

Fig. 7. Distribution of latencies of the blockchain-based Openstack Identity Service for 100,000 requests.

As shown in Fig. 7, the PBFT Blockchain-based Identity Service showed the distribution of latency to be similar to that for the Openstack Identity Service up to approximately 5,000 requests due to the process of token decryption, but after that, what was validated from the token database of each PBc Peer showed the distribution of latency to be approximately 92.61 ms on average for 100,000 requests when the number of the differently encrypted tokens was 1,000.

It was confirmed that the PBFT blockchain-based Identity Service was reduced by approximately 72.57%, compared to the Openstack Identity Service, and showed a stable distribution of latency.

3.3 Analysis of the Results

With the Openstack Identity Service, the CPU usage at the user level of API Nodes 1, 2, 3, and 4 of the Openstack Identity Service when 100,000 requests were sent respectively according to the number of differently encrypted tokens is shown in Fig. 8.

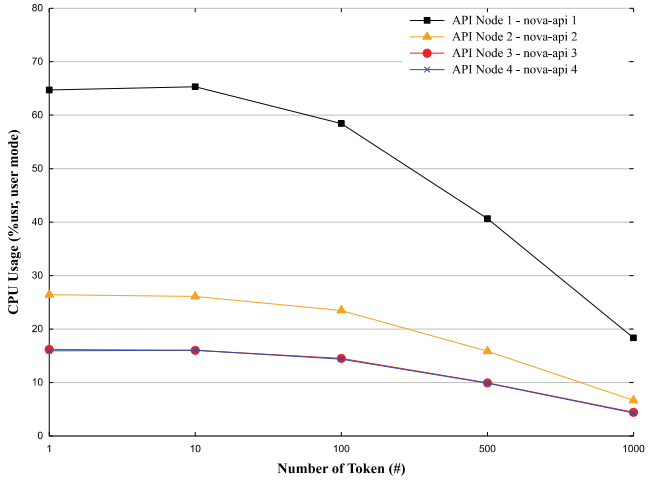

Fig. 8. Comparison of CPU Usage (%usr, user mode) Openstack Identity Service.

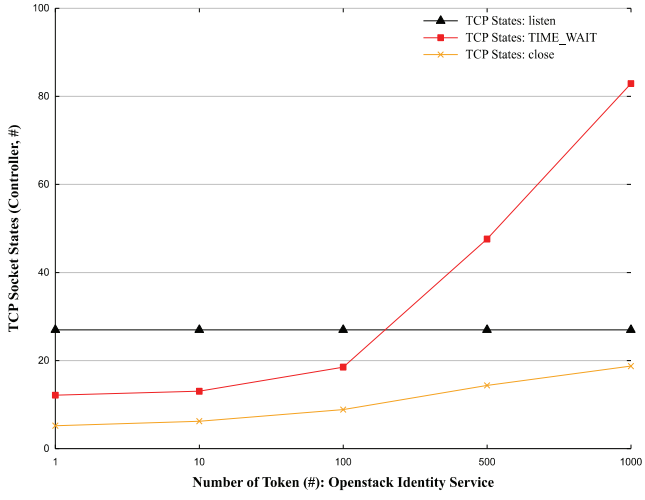

Fig. 9. Comparison of TCP states of controller, Openstack Identity Service.

In the case of API Node 1, which is a machine with a low specification, the CPU usage was greater than for API Nodes 2, 3 and 4, but when the number of tokens was 500, it started reducing, thus confirming that the CPU usage decreased remarkably when the number of tokens was 1,000. The analysis of the TCP socket states of the Controller Node is shown in Fig. 9.

It could be confirmed that the TCP TIME_WAIT [28] state count started increasing when the number of tokens was 100; that it increased by approximately 35 or more on average when the number of tokens was 500; and that it increased by approximately 70 or more on average when the number of tokens was 1,000.

For 100,000 requests, the increase in the TIME_WAIT state count as the number of differently encrypted tokens increased was caused by the elongated response time, because the token validation request was concentrated on the Controller Node, which is the API Gateway of the Openstack Identity Service. As a result, it could be confirmed that latency increased sharply as the number of tokens increased, as represented by the black line shown in Fig. 5.

With the PBFT blockchain-based Openstack Identity Service, the CPU usage of the PBc Peer API Nodes 1, 2, 3, and 4 of the PBFT blockchain-based Openstack Identity Service when 100,000 requests were sent respectively according to the number of the differently encrypted tokens in Table 2 is as shown in Fig. 10.

In the case of API Node 1, which is a machine with a low specification, it could be confirmed that its CPU usage was higher than that of API Nodes 2, 3, and 4, but Nodes 1, 2, 3, and 4, regardless of the number of tokens, all used the CPU constantly. The analysis of the TCP socket states of the Controller Node is shown in Fig. 11.

The TCP TIME_WAIT state count started increasing when the number of tokens was 500, but this was a process for dealing with numerous requests (i.e., as many as 100,000 were generated in this experiment), and it could be confirmed that it increased by about 8 on average when the number of tokens was 500, and by about 14 on average when the number of tokens was 1,000.

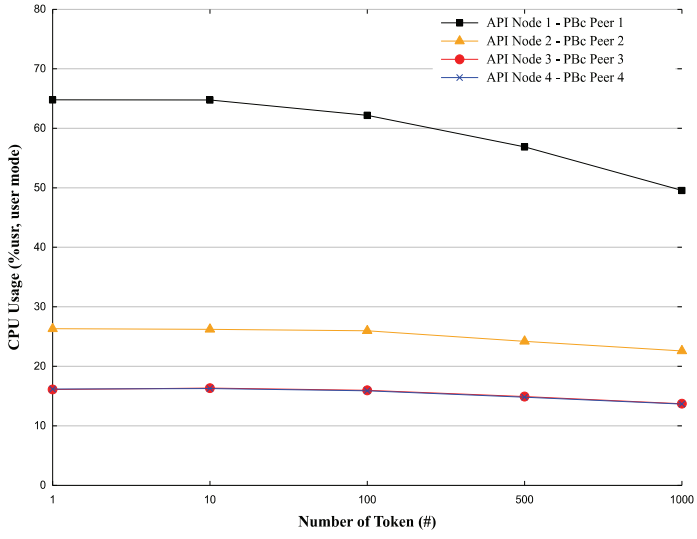

Fig. 10. Comparison of CPU Usage (%usr, user mode) PBFT blockchain-based Openstack Identity Service.

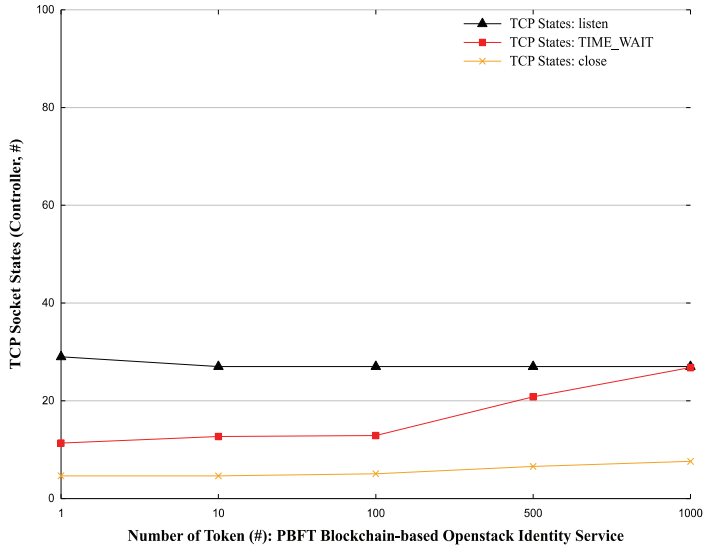

Fig. 11. Comparison of TCP states of controller, PBFT blockchain-based Openstack Identity Service.

For the 100,000 requests according to the number of differently encrypted tokens, the TIME_WAIT state count increased evenly for the requests even though the number of tokens increased for the PBFT blockchain-based Identity Service. This was because each PBc Peer dealt with the token validation request, rather than the Controller Node. As a result, it could be confirmed that it maintained constant latency regardless of the number of tokens, as indicated by the red line in Fig. 5, and dealt with it stably.

The comparison of the TIME_WAIT state count for 100,000 requests according to the number of each token of the PBFT Blockchain-based Openstack Identity Service with the Openstack Identity Service is as shown in Fig. 12.

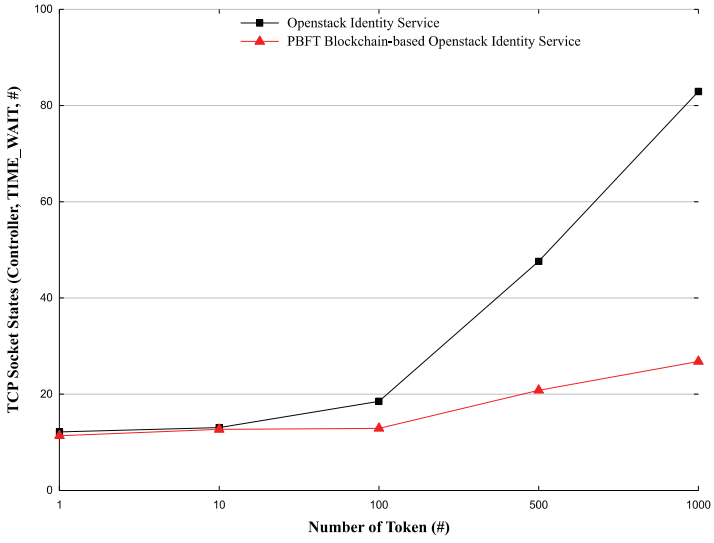

Fig. 12. Comparison of TCP states of TIME_WAIT of controller between the Openstack Identity Service and the PBFT blockchain-based Openstack Identity Service.

The average TCP states of TIME_WAIT of the Openstack Identity Service and the PBFT blockchain-based Openstack Identity Service in Fig. 12 are summarized in Table 4.

Compared to the Openstack Identity Service, it could be confirmed that since the PBFT blockchain-based Openstack Identity Service was the token database replicated among all the PBc Peers through the blockchain-based decentralized data store approach and validated token at each PBc Peer of API Node, the API Nodes showed a uniformed distribution of the TIME_WAIT state count for requests according to the number of each token, regardless of the number of tokens, and dealt with them stably.

Table 4. Comparison of TCP states of average number of TIME_WAIT

Number of tokens	Openstack Identity Service	PBFT blockchain-based Identity Service
1	12.15	11.36
10	13.05	12.68
100	18.53	12.89
500	47.61	20.81
1,000	82.92	26.80

4. Conclusion

The Openstack Identity Service, i.e., the Keystone, is a centralized approach component based on the token, including the Memcached for cache, which is the in-memory key-value store. As the number of differently encrypted tokens increases, requests for token validation are concentrated on a central server, and Memcached is the in-memory key-value store for cache, and the data are lost upon termination.

In the proposed PBFT blockchain-based Openstack Identity Service, the token database is stored at each PBc Peer of the Peer Network through the blockchain-based decentralized approach, and is synchronized amongst all the PBc Peers of the Peer Network. Token validation requests are validated at each PBc Peer, except for the initial decryption request, and it can improve the performance (of Openstack Identity Service) by decentralizing the token validation requests to all PBc Peers, while reducing the security vulnerability through the token database stored on all PBc Peers.

The results of the experiments carried out using version 5.1.1 of Apache JMeter confirmed that latency was reduced by 33.99% when the number of tokens was 500 and by 72.57% when the number of tokens was 1,000. Furthermore, since the token database is replicated on all PBc Peers which are blockchain peers, the security risk of the token database could be reduced. Future studies based on this research could seek to improve performance of the PBFT blockchain framework.

Acknowledgement

This research was supported by the MSIT (Ministry of Science and ICT) of South Korea under the Innovative Human Resource Development for the Local Intellectualization Support Program (No. IITP-2022-RS-2022-00156360) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation).

References

[1] W. K. Sze, A. Srivastava, and R. Sekar, "Hardening openstack cloud platforms against compute node compromises," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, Xi'an, China, 2016, pp. 341-352.

[2] T. Rosado and J. Bernardino, "An overview of openstack architecture," in *Proceedings of the 18th International Database Engineering & Applications Symposium*, Porto, Portugal, 2014, pp. 366-367.

[3] K. Jackson, *Openstack Cloud Computing Cookbook*, 2nd ed. Birmingham, AL: Packet Publishing Ltd., 2013.

[4] A. Beloglazov, S. F. Piraghaj, M. Alrokayan, and R. Buyya, "Deploying OpenStack on CentOS using the KVM hypervisor and GlusterFS distributed file system," Cloud Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report CLOUDS-TR-2012-3, 2012.

[5] K. Hogan, H. Maleki, R. Rahaeimehr, R. Canetti, M. van Dijk, J. Hennessey, M. Varia, and H. Zhang, "On the universally composable security of Openstack," 2018 [Online]. Available: https://eprint.iacr.org/2018/602.pdf.

[6] OpenStack, "Setting up keystone," [Online]. https://docs.Openstack.org/keystone/pike/contributor/set-up-keystone.html.

[7] J. Jose, H. Subramoni, M. Luo, M. Zhang, J. Huang, M. Wasi-ur-Rahman, et al, "Memcached design on high performance RDMA capable interconnects," in *Proceedings of 2011 International Conference on Parallel Processing*, Taipei, Taiwan, 2011, pp. 743-752.

[8] "Memcached: overview," 2020 [Online]. Available: https://github.com/memcached/memcached/wiki/Overview (accessed on 10 Sept 2020).

[9] A. K. Gupta and K. Ostner, "Database backup system using data and user-defined routines replicators for maintaining a copy of database on a secondary server," U.S. Patent 7383293, Jun 3, 2008.

[10] P. De Filippi, "The interplay between decentralization and privacy: the case of blockchain technologies," *Journal of Peer Production*, 2016 [Online]. https://ssrn.com/abstract=2852689.

[11] N. Tapas, G. Merlino, and F. Longo, "Blockchain-based IoT-cloud authorization and delegation," in *Proceedings of 2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, Taormina, Italy, 2018, pp. 411-416.

[12] P. Ganguly, "Selecting the right IoT cloud platform," in *Proceedings of 2016 International Conference on Internet of Things and Applications (IOTA)*, Pune, India, 2016, pp. 316-320.

[13] F. Buccafurri, G. Lax, A. Russo, and G. Zunino, "Integrating digital identity and blockchain," in *On the Move to Meaningful Internet Systems (OTM 2018 Conferences)*. Cham, Switzerland: Springer, 2018, pp. 568-585.

[14] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, "A taxonomy of blockchain-based systems for architecture design," in *Proceedings of 2017 IEEE international conference on software architecture (ICSA)*, Gothenburg, Sweden, 2017, pp. 243-252.

[15] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: architecture, consensus, and future trends," in *Proceedings of 2017 IEEE International Congress on Big Data (BigData Congress)*, Honolulu, HI, 2017, pp. 557-564.

[16] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, New Orleans, LA, 1999, pp. 173-186.

[17] M. Castro and B. Liskov, "Practical Byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems*, vol. 20, no. 4, pp. 398-461, 2002.

[18] S. Duan, M. K. Reiter, and H. Zhang, "BEAT: asynchronous BFT made practical," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, Toronto, Canada, 2018, pp. 2028-2041.

[19] Y. Kim and J. Park, "Hybrid decentralized PBFT blockchain framework for OpenStack message queue," *Human-centric Computing and Information Sciences*, vol. 10, article no. 31, 2020. https://doi.org/10.1186/s13673-020-00238-6

[20] OpenStack, "OpenStack API documentation v3," 2021 [Online]. Available: https://developer.Openstack.org/api-ref/identity/v3/#authentication-and-Token-management.
[21] F. Yang, K. Dou, S. Chen, M. Hou, J. U. Kang, and S. Cho, "Optimizing NoSQL DB on flash: a case study of RocksDB," *in Proceedings of 2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, Beijing, China, 2015, pp. 1062-1069.
[22] The Go programming language [Online]. Available: https://golang.org.
[23] C. Cachin, "Architecture of the Hyperledger blockchain fabric," 2016 [Online]. https://www.zurich.ibm.com/dccl/papers/cachin_dccl.pdf.
[24] H. Sukhwani, J. M. Martínez, X. Chang, K. S. Trivedi, and A. Rindos, "Performance modeling of PBFT consensus process for permissioned blockchain network (Hyperledger Fabric)," in *Proceedings of 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, Hong Kong, China, 2017, pp. 253-255.
[25] HAProxy: a reliable, high performance TCP/HTTP load balancer [Online]. Available: http://www.haproxy.org
[26] OpenStack, "Configuring stateless services," 2021 [Online]. Available: https://docs.Openstack.org/ha-guide/control-plane-stateless.html#api-services.
[27] Y. Jing, Z. Lan, W. Hongyuan, S. Yuqiang, and C. Guizhen, "JMeter-based aging simulation of computing system," in *Proceedings of 2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering*, Changchun, China, 2010, pp. 282-285.
[28] T. Faber, J. Touch, and W. Yue, "The TIME-WAIT state in TCP and its effect on busy servers," in *Proceedings of IEEE Conference on Computer Communications (INFORCOM) and the 18th Annual Joint Conference of the IEEE Computer and Communications Societies: The Future is Now*, New York, NY, 1999, pp. 1573-1583.

Youngjong Kim **https://orcid.org/0000-0003-0811-0215**

He received M.S. and Ph.D. degrees from Soongsil University. He is currently employed as a professor at the School of Software, Soongsil University, Korea. Dr. Kim's research interests include cloud computing, network computing, blockchain and security, etc.

Sungil Jang **https://orcid.org/0000-0002-3618-8637**

He received M.S. degrees from the School of Software of Soongsil University. Since March 2019, he has been a Ph.D. candidate at the School of Software of Soongsil University. His current research interests include blockchain and network computing.

Myung Ho Kim **https://orcid.org/0000-0002-1933-7987**

He is currently serving as a professor in the Department of Software Convergence of Soongsil University, Seoul, South Korea. He obtained his Ph.D. from Pohang University of Science and Technology, Pohang, South Korea, in 1995. His research interests are information security and system software.

Jinho Park **https://orcid.org/0000-0003-1961-6983**

He earned a bachelor's degree from the Department of Software Engineering, Soongsil University, Master and Doctorate Majoring in Computer Science and Software Engineering, Soongsil University. He is currently serving as a professor at the Faculty of Multimedia Engineering, Dongguk University.