ICIC Express Letters
Volume **13**, Number **11**, November **2019**

 ISSN 1881-803X
pp. **1021–1029**

SCALABLE REVERSE OFFLOADING FOR DECENTRALIZED PROOF OF LOCATION IN WEB APPLICATION

Kemal Anshari Elmizan and Suharjito

Computer Science Department, BINUS Graduate Program – Master in Computer Science
Bina Nusantara University
Jl. K. H. Syahdan No. 9, Kemanggisan, Palmerah, Jakarta 11480, Indonesia
kemal.elmizan@binus.ac.id; suharjito@binus.edu

Received April 2019; accepted July 2019

Abstract. *Scalability for web application has been a problem for system architect over the years. Many system changes are proposed from within the application structure, data structure and computing structure to adjust with fluctuation of mobile network connectivity, diversity between a plethora of modern devices, and the development of application programming paradigm. There have been many mobile and cloud offloading schemas that have been proposed on the last couple of years, each with their own objective in mind. When addressing proof of location in web application, the current approach of using server-client architecture has problems in scalability. Server can only handle limited computations concurrently, and when addressing the problem of scalability, a good approach is to use decentralized system. This paper is proposing a decentralized approach using reverse offloading architecture to address proof of location as an alternative to global positioning system in web application. The development method for our decentralized web-app is using WebAssembly, and our scalability performance is evaluated using docker swarm simulation test, and our proposed reverse offloading architecture shows better performance in network throughput and computing resources.*
Keywords: Decentralization, Reverse offloading, Proof of location, Scalability, Web application

1. **Introduction.** The demand to improve performance of web applications is substantial. Online economic activity is growing rapidly in developed and developing countries [1]. The modern world is always active and hyper-connected, which means that user expectations in terms of application experience and performance are higher than in previous times. If a website does not immediately send a response after being requested, or if the web application always runs with a long delay, the user will immediately switch to another website.

Modern web application development is one of the application development processes that focus on the performance of a web application. There are several modern methods of web application development that will be discussed here, namely the mobile offloading system method for web-centric devices [4] which proposes the offloading method from client to server in a device-independent manner using web technology. To be able to offload computing, Park et al. developed the offloading system as follows: offloading is carried out on a new client-server for the mobile environment where the client's core workload is carried out, run remotely from the server side. This system also controls the offloading function based on annotations from the program side to simplify implementation and minimize the process of parsing overhead from web resources.

Offloading is a mechanism to move computing from one computer to another. Usually offloading is done to divide computing from client to server. Yang, researcher from Intel. proposes Manageable Application Code Offloading (MACO) techniques that maintain the

DOI: 10.24507/icicel.13.11.1021

responsive side of applications and granularity of codes so that they are easily managed [5]. This research focused on code and user interface to analyze performance. Focusing on modern offloading methods moves the trend to a term called reverse offloading [6], which directs the understanding of the offloading process to how data processing carried out in a cloud data center can be transferred to an edge or client device, to limit large data transfers through public networks and avoid major latency due to these transfers.

The problem that will be discussed as a form of implementation of reverse-offloading is a proof of location problem. Proof of location is a marker of the location of a device. Traditionally, GPS is used for this process, where the position of a device is marked by the latitude and longitude position of the GPS tracking machine that is on the device. However, this approach has several problems, namely the amount of data sent to the server, which then resulted in the server-side service proof of location not being scalable. Another problem is that the use of GPS is very easy to mock and spoof, and hence, a lot of cheat occurs.

From Figure 1, we can see proof of location using current approach, where device sends GPS location every certain unit of time, (e.g., every 30 seconds). With this approach, server can easily be overwhelmed with requests; thus, making this approach is not scalable.

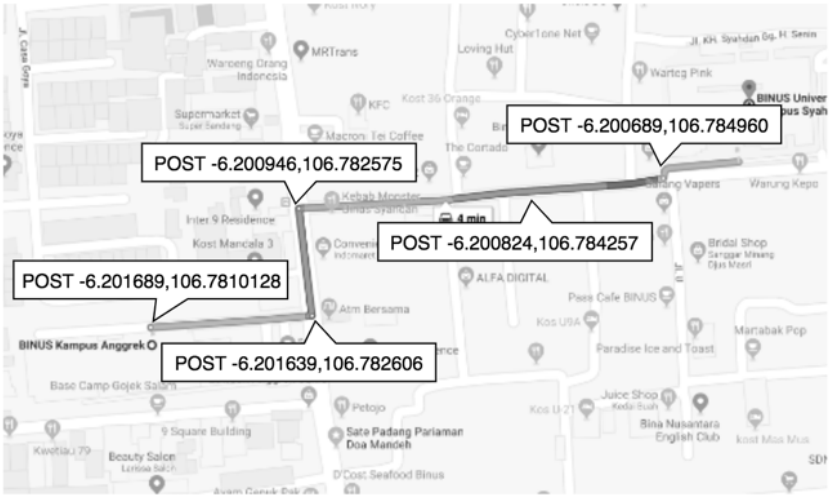

FIGURE 1. Proof of location with centralized approach

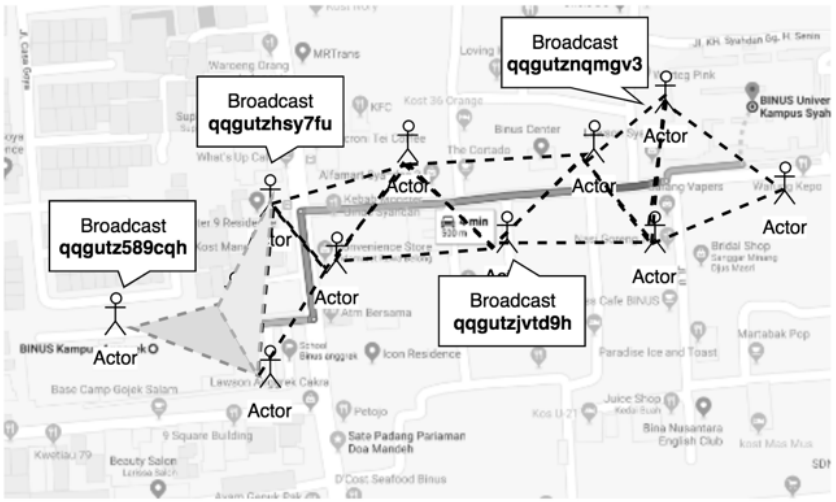

FIGURE 2. Proof of location with decentralized approach

This paper will propose a decentralized method of defining proof of location. In Figure 2, each node acts as a reviewer of their peer's geohash proof of location. Each actor who wants to validate their proof of location needs to have at least 3 reviewers to validate it. This will improve positioning accuracy of the actor and makes it easy to validate which actor spoofs their location, as well as increasing server's scalability.

This paper consists of 5 sections. In the first section we are elaborating background of the problem. Next section will consist of literature review on related works about reverse offloading mechanism and proof of location. The third section of the paper will discuss our proposed method and architecture of reverse offloading. The fourth section will elaborate the simulation scenario to benchmark our method with state-of-the-art client-server scenario, as well as analyzing and comparing simulation statistics data between them. Last section will contain our conclusion and future works that might be derived based on our paper.

2. **Related Works.** On proof of location, some researches like [7,8] are experimenting on using blockchain technology to alleviate the proof of actor's claim of location. Amoretti et al. [7] propose a scheme that prioritizes on privacy and trustworthiness of the location claims. This can be achieved by implementing a decentralized peer-to-peer and blockchain-based scheme. Amoretti et al. also tackle some fraud scenario on location-based services on their robustness analysis, e.g., cheating on actor's or peer's location, replaying proof of location and colluding with other peers.

Yang et al. [9] propose implementation of position-based cryptography with location privacy, applied on top of fog computing. Some other researches also include fog and edge computing as part of the implementation scenario, mainly as an IoT system. Tang et al. [10] implement their proposed system architecture to reverse offload computation into mobile edge servers, as part of the decentralized peer-to-peer network to improve connectivity on mobile devices.

Zbierski and Makosiej [11] propose the use of web workers as a method of cloud and mobile optimization. Zbierski and Makosiej present a system for offloading web workers on HTML5 from a mobile application to a remote server. The proposed solution is fully transparent to developers, i.e., the existing application code does not need to be changed in any way to benefit from this offloading method.

This research is a quantitative research that will discuss the design of a reverse-offloading web application system using WebAssembly and compare the results of the scalability test of the web application with the results of scalability testing of web applications with client-server architecture which is state of the art. This study measures the ability of the application system performance if the user accesses this application concurrently in large numbers and within a certain period. Application performance is measured based on load or load that occurs on the server and network load used by both methods. Further performance measurement processes will be explained in the sub-section of the evaluation phase.

3. **Methodology.** This research is experimental research, which will compare the measurement results of the scalability capability of a web application architecture that can be seen in Figure 3. with a web application architecture using the reverse-offloading method using WebAssembly, which can be seen in Figure 4.

In Figure 3, we can see the flow data and application process in the usual client-server architecture in a web application. The browser on the client requests from the UI logic controller, using JavaScript. HTTP requests are made on the endpoint API from the server side to get results from computation or data processing carried out on the server side; in other words, the actual computing process is done on the server side, and the client is only responsible for formatting data generated from the server side.

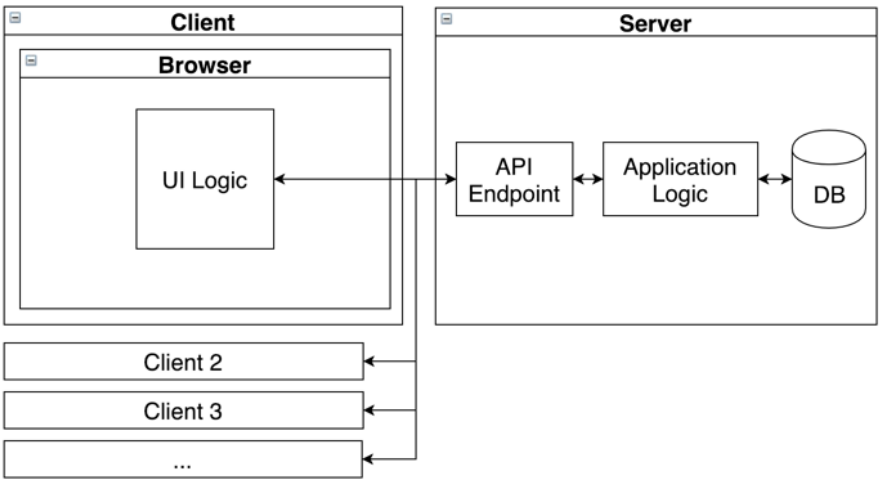

FIGURE 3. Client-server application architecture

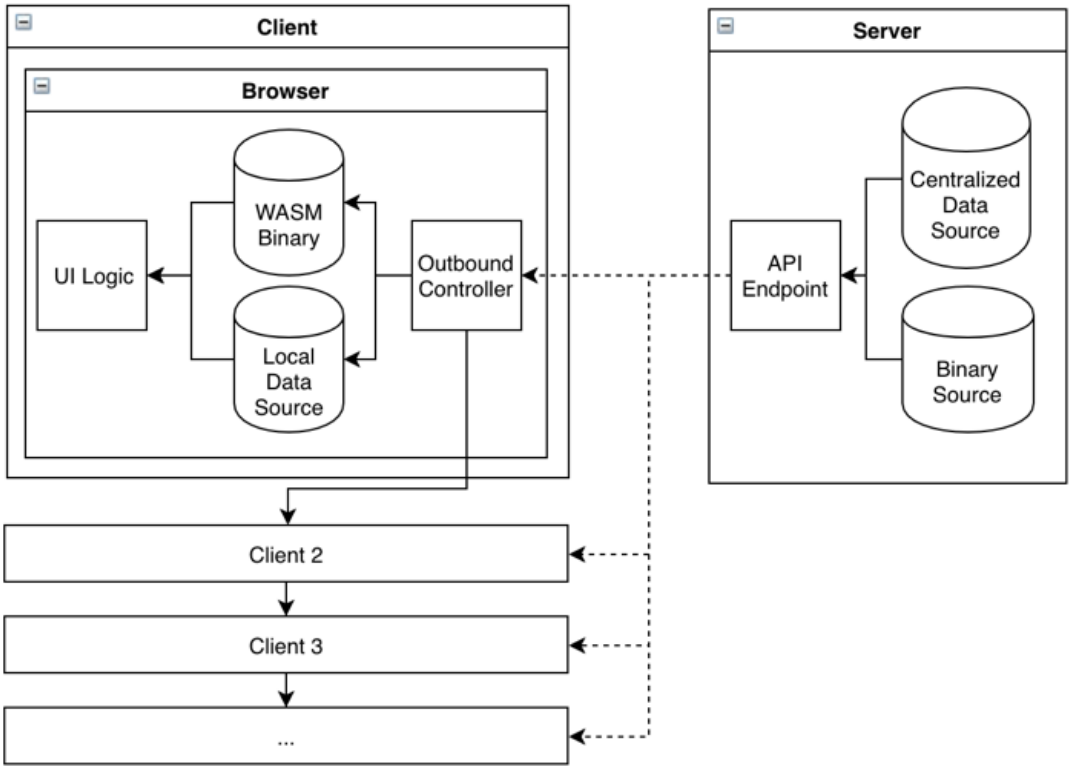

FIGURE 4. Proposed reverse-offloading system design

This research would like to propose a reverse offloading application architecture, where the computational logic that previously existed on the server, was offloaded, so that the computation was done on the client side. The assumption is, for a small user scale, like one or two clients, this kind of architecture will not feel the benefits of the performance, instead it will make the application processing on the client side become heavier, because the main application is run not on the server side, but on the client side. This design refers to Figure 4.

This reverse offloading system focuses on transferring the computation and storage of data that is shared (offloaded) from the server to the client. In Figure 4, it can be seen that in the client architecture there are WASM Binary and Local Data Source which are the main logic of this application web. WASM Binary is taken from the server when the

client first requests the server and is taken from the binary source. Local Data Source is also replicated from centralized data sources on the server and is taken when the client first connects to the server. This client architecture scheme is then replicated for subsequent clients, who have their own WASM Binary and Local Data Source.

4. **Results and Discussion.**

4.1. **Simulation scenario.** The simulation will run in this following machine with the following specification: MacBook Pro 2016, intel core i7 2.7GHz, and 16GB RAM. Simulation will run on a docker container to simulate scalability, similar to deployment mechanism done by [12]. This simulation will simulate both client-server and reverse-offloading application.

4.2. **Data analysis.** After doing the simulation with the steps above for each scenario, we obtained simulation statistics that can be seen in Figures 5-7. Figure 5 shows the comparison of CPU resource usage, Figure 6 displays the results of comparison of resource memory usage, and Figure 7 displays a comparison of Net I/O resource usage. Figures 8 and 9 show a comparison of these three metrics in one diagram.

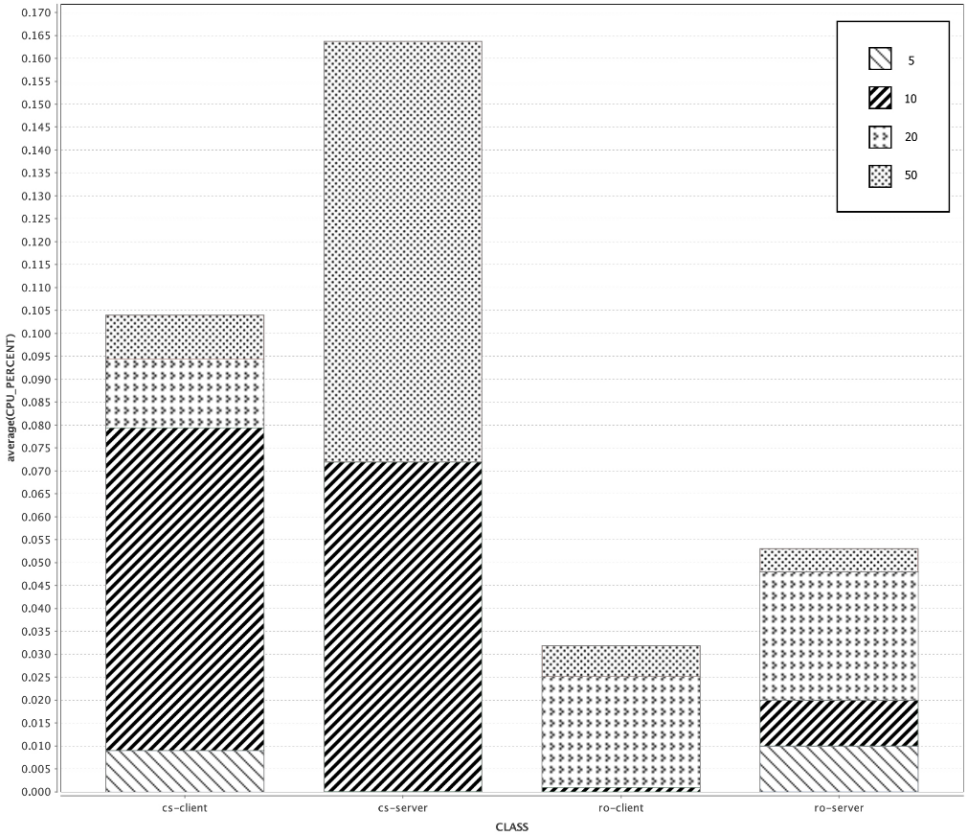

FIGURE 5. CPU usage (in percent) comparison

Results shown in Figures 5 to 7 show that in the reverse-offloading simulation, CPU usage and Net I/O are lower than the client-server architecture while maintaining similar memory usage as the client-server architecture; even on a 50x scale. This shows that the reverse-offloading architecture has the advantage of increasing scalability compared to the client-server architecture. To discuss the details of the results of further scalability statistics, we can see in Figures 8 and 9 for the details of this simulation classification scenario.

On Figure 8, we classify 3 metrics (network I/O throughput with label NET_IO, memory usage percentage with label MEM_USAGE, and CPU usage percentage with label

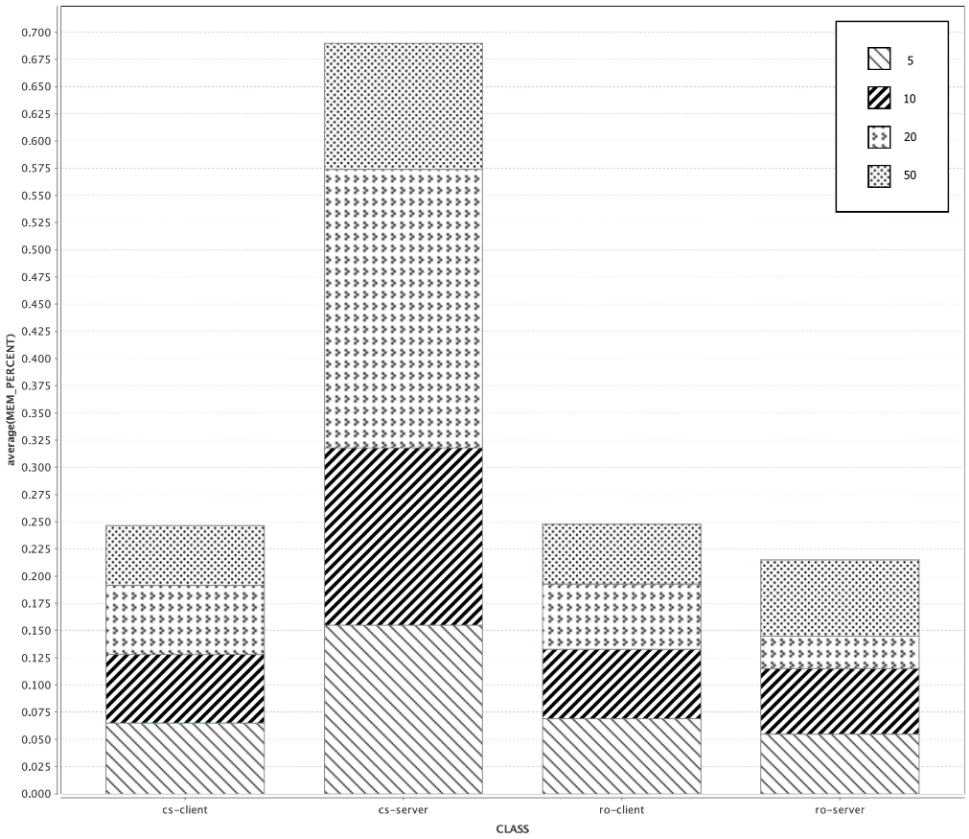

FIGURE 6. Memory usage (in percent) comparison

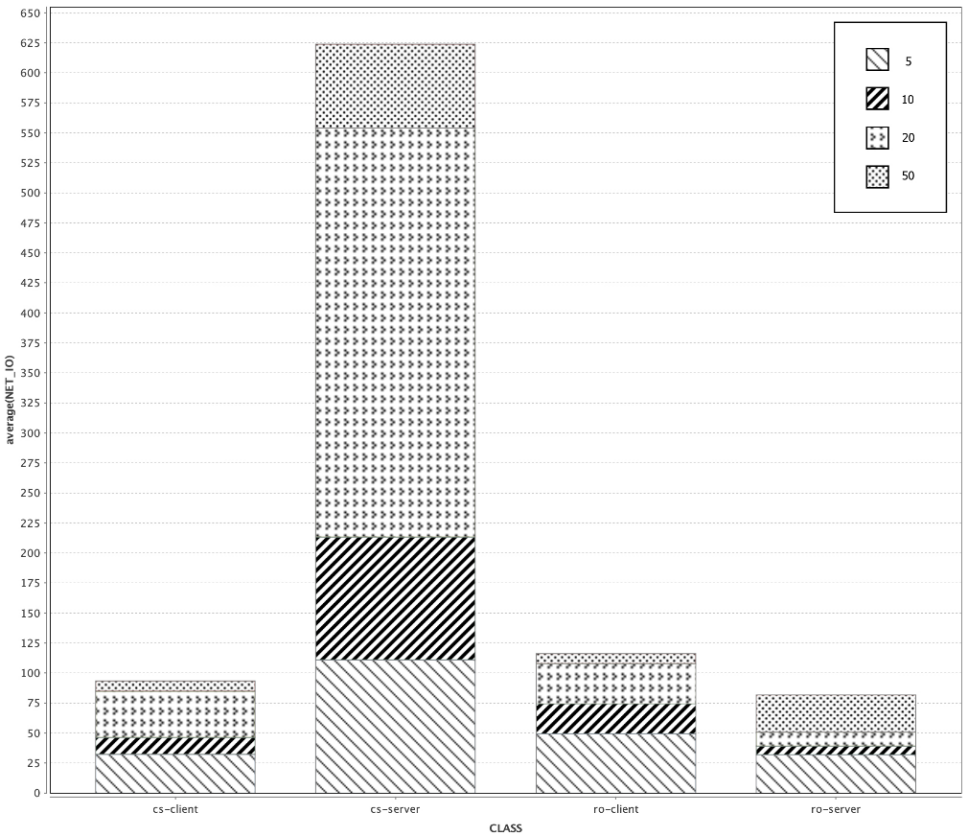

FIGURE 7. Net I/O usage (in kbps) comparison

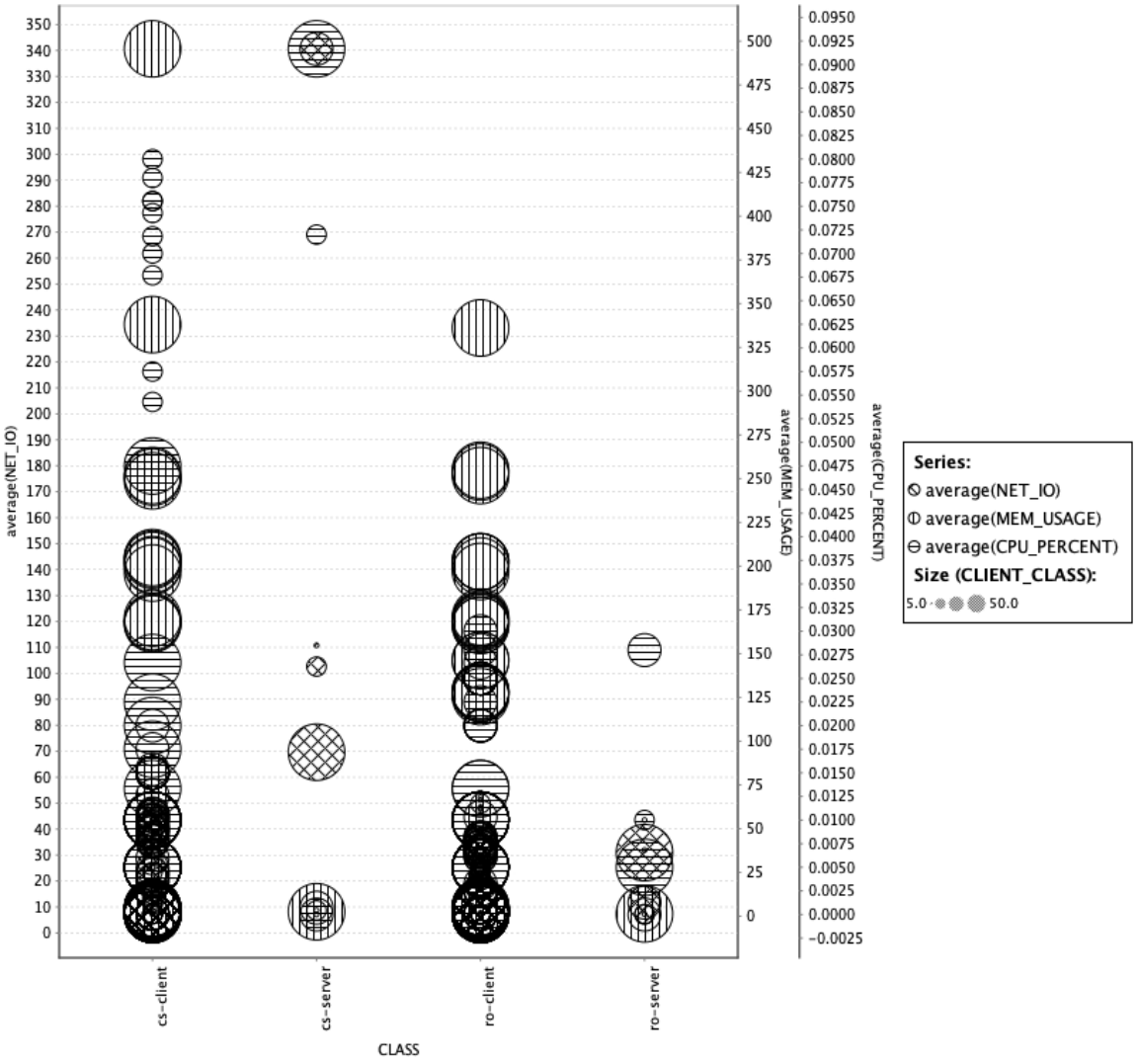

FIGURE 8. Evaluation result comparing client-server and reverse-offloading class on network I/O, memory usage, and CPU percentage

CPU_PERCENT) as a metered comparison between each class. In this picture we can also see 4 different classes scattered among 5 to 50 simulation scenarios. Those classes are cs-client, cs-server which represent client-server architecture, and ro-client, ro-server which represent reverse-offloading architecture.

The result shown in Figure 8 shows that reverse-offloading simulation has lower CPU and Net I/O compared to client-server architecture, while maintaining similar memory usage, even on 50x scale. This demonstrates reverse-offloading scalability performance advantage over the client-server architecture. To further break down these statistics into scalability results, we can refer to Figure 9 for the simulation scenario classification.

On Figure 9, we can see the scalability breakdown of reverse-offloading performance improvement compared to client-server on CPU and Net I/O attributes. Classes ro-client and ro-server show stable figure on CPU and Net I/O, while having similar values on memory usage compared to cs-client and cs-server classes.

5. **Conclusion and Future Works.** In this paper, we analyze two separate methods in web application development: the current state-of-the-art client-server architecture and our proposed reverse-offloading architecture. We proposed a decentralized system architecture and compared its scalability performance, by running a docker swarm scalability testing and gathered the CPU usage, memory usage, and Net I/O of both architectures.

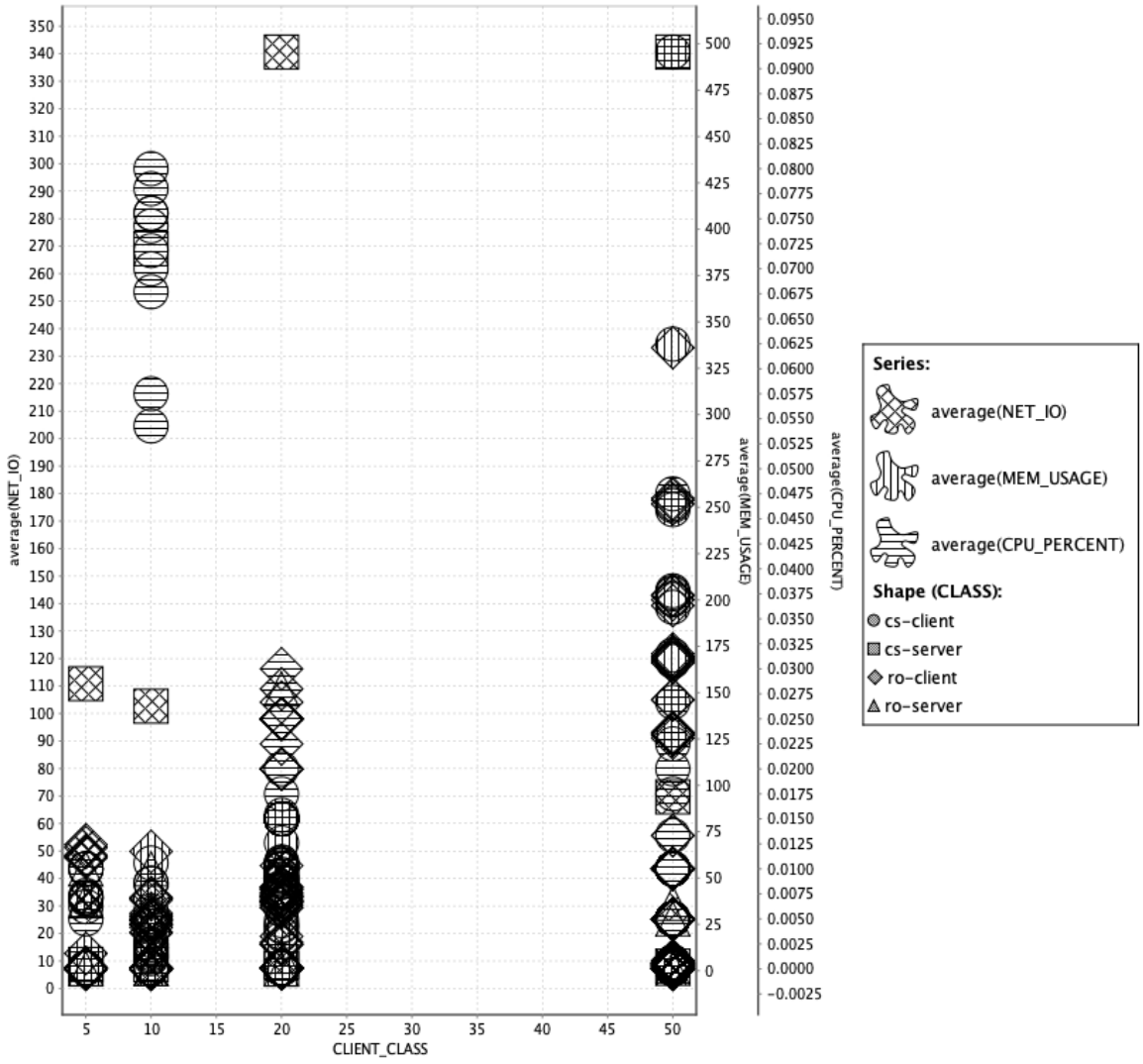

FIGURE 9. Evaluation result of docker swarm scalability simulation from 5 to 50 concurrent clients

After running simulation, we can conclude that scalability improved when we switched to our proposed reverse-offloading architecture, especially on CPU and Net I/O.

Based on the results of our simulation, a decentralized solution can be applied to solving a scalability problem and improving performance of proof-of-location systems. These promising results are based on our simulation experiment; therefore, future research can include the following topics: hardware architecture system of decentralized connectivity, e.g., by using low-energy Bluetooth and Wi-Fi to implement this idea and simulation of reverse-offloading architecture to tackle different problems other than proof-of-location.

Acknowledgment. The authors wish to thank Bina Nusantara University Computer Science Department, BINUS Graduate Program.

REFERENCES

[1] Global e-retail sales share by region 2016 | Statistic, *Statista*, https://www.statista.com/statistics/239300/number-of-online-buyers-in-selected-countries/, [Accessed: 25-Sep-2018].

[2] F. Smith, 10 tips to improve application performance, *NGINX*, https://www.nginx.com/blog/10-tips-for-10x-application-performance/, [Accessed: 25-Sep-2018].

[3] J. Jackson, How bad performance impacts ecommerce sales (Part I), *Load Impact*, http://blog.loadimpact.com/blog/how-bad-performance-impacts-ecommerce-sales-part-i/, [Accessed: 25-Sep-2018].

[4] S. Park, Y. Choi, Q. Chen and H. Y. Yeom, SOME: Selective offloading for a mobile computing environment, *IEEE International Conference on Cluster Computing*, 2012.

[5] S. Yang, Manageable granularity in mobile application code offloading for energy savings, *IEEE International Conference on Green Computing and Communications*, 2012.
[6] N. Antonopaulos and L. Gillam, *Cloud Computing*, Springer, 2010.
[7] M. Amoretti, G. Brambilla, F. Medioli and F. Zanichelli, Blockchain-based proof of location, *IEEE Int. Conf. Softw. Qual. Reliab. Secur. Companion*, pp.146-153, 2018.
[8] J. Alcalde-Unzu and M. Vorsatz, Strategy-proof location of public facilities, *Games Econ. Behav.*, vol.112, pp.21-48, 2018.
[9] R. Yang, Q. Xu, M. H. Au, Z. Yu, H. Wang and L. Zhou, Position based cryptography with location privacy: A step for fog computing, *Futur. Gener. Comput. Syst.*, 2017.
[10] W. Tang, X. Zhao, W. Rafique, L. Qi, W. Dou and Q. Ni, An offloading method using decentralized P2P-enabled mobile edge servers in edge computing, *J. Syst. Archit.*, 2019.
[11] M. Zbierski and P. Makosiej, Bring the cloud to your mobile: Transparent offloading of HTML5 web workers, *Proc. of Int. Conf. Cloud Comput. Technol. Sci. CloudCom*, pp.198-203, 2015.
[12] X. Wan, X. Guan, T. Wang, G. Bai and B. Y. Choi, Application deployment using Microservice and Docker containers: Framework and optimization, *J. Netw. Comput. Appl.*, vol.119, pp.97-109, 2018.