JOIN (Jurnal Online Informatika)
p-ISSN: 2528-1682, e-ISSN: 2527-9165
Volume 11 Number 2 | December 2026: 279-288
DOI: 10.15575/join.v11i2.1432

Integrating Portable Systems of Edge Computing Pocket Data Centers for Modernising MSME Services

Norma Ningsih[1*], Prima Kristalina[2], Bintang Desta Ramadhani[3]
[1,2,3]Department of Electrical Engineering, Politeknik Elektronika Negeri Surabaya, Indonesia

Article Info

Article history:

Received August 1, 2024
Revised February 06, 2025
Accepted March 26, 2025
Published September 18, 2026

Keywords:

Clustering
Data Center
Edge Computing
MSME

ABSTRACT

Micro, Small, and Medium Enterprises (MSMEs) play a crucial role in local economic growth by generating employment opportunities. However, their digital transformation is often hindered by high server infrastructure costs and limited access to efficient computing resources. To address these challenges, this study develops an edge computing-based pocket data center system designed to process transactions efficiently, affordably, and portably. The system integrates virtualization technologies such as virtual machines, Docker, Kubernetes, Ubuntu, and the LAMP stack to support MSME applications. The HAProxy load balancer employs a round-robin algorithm to optimize workload distribution. Experimental testing over five minutes with 9,000 total requests demonstrated the system's capability to handle all user requests with a latency of 367 ms, a throughput of 29.4 requests per second, and a processing rate of 30 requests per second. The system also exhibited efficient power consumption at 9.01 watts while achieving a 100% success rate in request completion. Compared to conventional server solutions, this approach significantly enhances accessibility, cost-efficiency, and scalability for MSMEs. These findings highlight the potential of low-cost, portable data centers to modernize MSME operations, ensuring reliability and high performance in the digital economy.

Corresponding Author:

Norma Ningsih,
Department of Electrical Engineering, Politeknik Elektronika Negeri Surabaya
Jl. Raya ITS, Keputih, Kec. Sukolilo, Surabaya, Jawa Timur 60111
Email: norma@pens.ac.id

1. INTRODUCTION

Based on their characteristics, MSMEs have differences compared to large industries today. The following are the differences between MSMEs and large companies, namely limited capital and number of employees, focusing on local and regional markets, fast product innovation and flexibility, and the level of ownership in individuals or families. MSMEs are one of the supporting factors to increase national economic growth. MSMEs are the largest business unit in Indonesia, amounting to 99.99 percent of the 64.20 million business units in Indonesia [1]. Based on [2], the role of MSMEs is divided into 4 categories, namely reducing poverty and social inequality, economic empowerment, economic growth and increasing opinions and consumption. On the other hand, MSMEs have various obstacles to developing businesses and entering the global market business including financing, training, technology

p-ISSN: 2528-1682
e-ISSN: 2527-9165

development and market access [3]. one of the contributions of digital transformation can be seen from the aspect of increasing access to information, knowledge and data which has an impact on increasing innovation, productivity, efficiency and community welfare [4]. However, high infrastructure costs and reliance on cloud services pose significant barriers to digital adoption among MSMEs.

Edge computing is a new paradigm that provides computing resources and data storage needed by reducing delays to users or devices [5]. The main idea of this concept is to make the computing process closer to the data source so that it is expected that latency and user response time can be reduced and efficiency is increased [6][7][8]. Edge computing is one of the next-generation technologies in communication networks in addition to the Internet of Things and artificial intelligence [9]. Edge computing infrastructure has centralized distributed data to perform realtime data processing, visualization and analysis. With this capability, edge computing can reduce data flow and costs when compared to using cloud services [10]. Edge Computing consists of 3 types, namely mobile edge computing, cloudlets and fog computing [6], [11], [12]. Discuss in detail the differences between these 3 technologies from the aspects of principles, system architecture, standards and applications. Edge computing technology has been implemented in many fields including the Internet of Things technology [13], [14].

A data centre is a centralised physical or virtual repository that is used for storage, computing, data and information distribution. based on the type of data centre can be divided into 5 namely colocation, enterprise, cloud, edge and micro data centre. In data center operations, both physical and virtual, resource management efficiency remains a major challenge. Kubernetes emerges as a solution for managing and optimizing workloads by orchestrating containers within the data center infrastructure. With its capabilities in automated scheduling, monitoring, and recovery of problematic containers, Kubernetes enhances the reliability and efficiency of data centers, whether on-premises or cloud-based [15]. Kubernetes enhances clustering by enabling automated workload distribution, failover handling, and resource scaling across multiple nodes. Clustering can handle task shifting and load equalisation from one node to another [16].

Existing studies have explored various cloud and edge computing frameworks for optimizing resource allocation. Chunhua Lin et al. [17] develop dynamic system allocation and application of cloud computing virtual resources based on system architecture. The objective of this research is to investigate the design and implementation of a dynamic virtual resource allocation system within cloud computing. This system's architecture facilitates load balancing among virtual resource pools and minimizes resource wastage. By employing a cluster network topology, the resource usage within the dynamic system cluster can be monitored in real time, enabling the automatic determination of the total cluster load based on the monitoring data. They conducted experiments based on performance testing and scenario testing. The test results indicate that the system's running time is approximately 22–27 milliseconds, with CPU utilization ranging between 90–95% and RAM usage around 3.5 milliseconds. These findings demonstrate that cloud technology can enhance resource scheduling for large tasks and optimize resource load balancing.

Yan guo et al. [18] create user allocation-aware edge cloud placement in mobile edge computing. They formulate this as a multi-objective optimization problem with the goals of balancing the workload between edge clouds and minimizing the service communication delay for mobile users. To achieve this, we propose an approximate method that employs K-means clustering and mixed-integer quadratic programming. Additionally, they conduct experiments using the base station dataset from Shanghai Telecom and compare our approach with other representative methods. The results demonstrate that their approach performs better to some extent in terms of workload balance and communication delay, thereby validating the proposed method.

Leni Fitriani et al. [19], [20] conducted related research about information technology strategy for micro, small, and medium enterprises in the era of Industry 4.0. This article seeks to assess information technology solutions to empower Micro, Small, and Medium Enterprises, aiming to identify potential opportunities and challenges for future research. MSMEs can take advantage of the opportunities offered by digitization in simple ways, such as by actively using social media platforms like Twitter, Facebook, and Instagram as business accounts to promote their advertising initiatives. Another option is to utilize online marketplaces and develop a blog to support their business online, for example, by

using a free blog. Furthermore, business owners can enhance their operations by incorporating digital platforms into their company strategies.

However, these studies primarily address general cloud and edge computing applications without specifically tackling the affordability and accessibility constraints faced by MSMEs. To bridge this gap, this study proposes a *portable pocket data center* designed to enhance MSME services using edge computing and Kubernetes-based clustering. Unlike conventional cloud-based solutions, our approach leverages local resources—networks, user devices, and hardware—to enable cost-effective and scalable data processing. The proposed system integrates virtualization technologies such as virtual machines, Docker, Kubernetes, Ubuntu, and the LAMP stack to facilitate MSME application deployment. HAProxy is employed for load balancing, ensuring stable and efficient service delivery.

This research introduces an innovative, low-cost, and energy-efficient infrastructure for MSMEs, providing a scalable and flexible alternative to traditional data centers. By leveraging edge computing and clustering, the system reduces dependency on expensive cloud services while maintaining high performance and reliability. This novel approach addresses MSME-specific challenges, supporting digital transformation and enhancing operational efficiency in a rapidly evolving economic landscape. This innovative solution addresses the specific needs of MSMEs, offering a scalable and flexible infrastructure that can be easily adapted to various business contexts

2. METHOD

This research follows a structured methodology to develop a pocket data center that serves as a server for the MSME information system. The pocket data center is designed to be compact, portable, and cost-effective while maintaining high performance and reliability. The methodology consists of several key stages, including system requirement identification, hardware and software design, system implementation, and performance evaluation.

2.1 Identification of System Requirement

To understand the technological needs of MSMEs, this study conducted observations and interviews with MSME actors. These discussions focused on business process workflows, technological limitations, and infrastructure challenges faced by MSMEs. The findings from these observations were used to define the functional and non-functional requirements of the system, ensuring that the developed pocket data center directly addresses the challenges of cost, performance, and ease of use. Additionally, a literature study was conducted to explore existing data center models, clustering methods, and load balancing techniques to select the most appropriate approach for MSME applications. The results guided the selection of edge computing, Kubernetes, and REST API as core technologies in this research

2.2 Hardware and Software Design of Pocket Data Center

In developing the pocket data centre system, some software is needed so that the system can run according to its function, namely first installing the Proxmox Virtual Environment, which can manage containers, virtual machines, storage, virtual networks, high availability clusters via a web interface or using the command line. By using Proxmox, we can create many instances to simulate the pocket data centre system. The instance will later have applications - the applications needed in this research.

Figure 1 shows the system block diagram that is used in this research. In developing the pocket data centre system, several applications are needed, such as virtual machines, Docker, Kubernetes, Ubuntu, and also the LAMP Stack to deploy MSME applications. Second, install Kubernetes, which is used as an orchestrator to deploy the required applications. In addition, Kubernetes also functions as a load balancer and automatically scales up and scales down according to the workload. 1 virtual machine as a Kubernetes controller and 2 virtual machines as Kubernetes nodes. The pocket data centre system also uses edge computing as a place to do fast data computing on local devices, so it does not depend on internet availability and increases system security. In making clustering on Kubernetes using the round-robin algorithm method, which is a scheduling algorithm that is generally applied to operating systems and network management where the principle is to divide processing time evenly among all existing nodes.

p-ISSN: 2528-1682
e-ISSN: 2527-9165

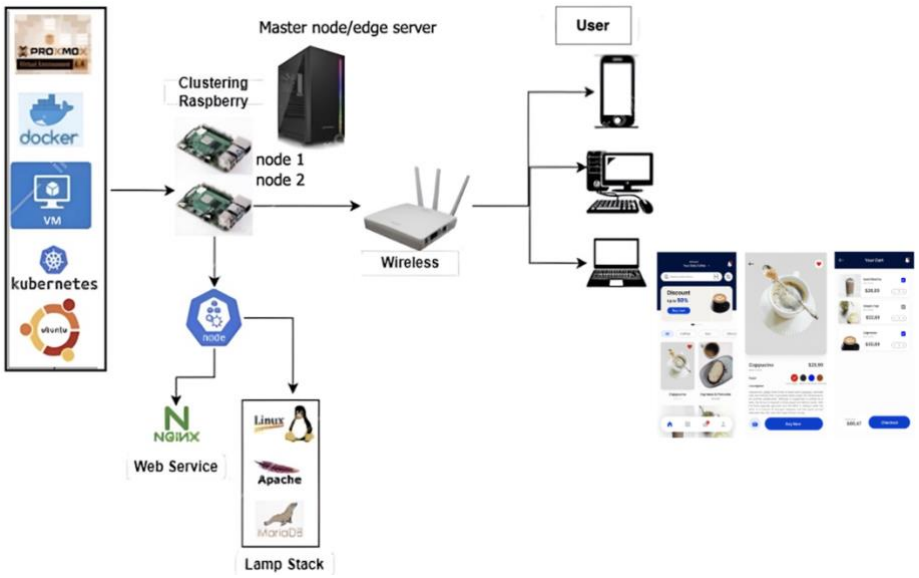

Figure 1. System Block Diagram

2.3 System Implementation and Integration

In this study, there are 2 nodes, where each node is configured on a Raspberry Pi. The master node in this study is configured on a virtual machine whose function is to manage and process the entire cluster in Kubernetes through the components in it, such as API Server, Etcd, Controller Manager, and Scheduler. The clustering configuration consisting of 2 nodes and 1 master node is developed using the concept of edge computing, where users get services from MSME applications directly, namely in the same network as the MSME system without relying on internet availability. In developing MSME applications, it starts with analysing and designing information systems using the object-oriented paradigm and hexagonal architecture. The following is an overall description of the MSME application.

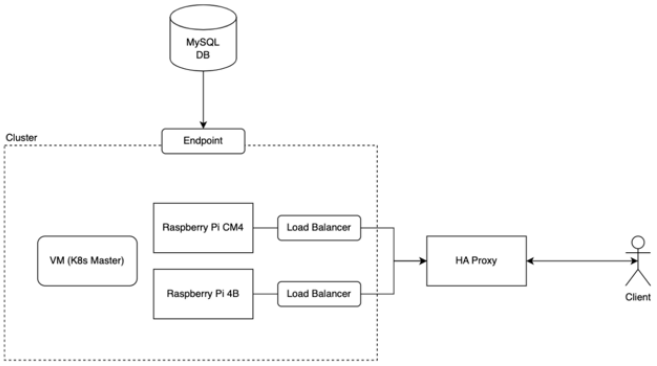

Figure 2. Cluster Diagram

Figure 2 explains the clustering architecture that has been made. Transaction data that occurs in MSME is stored in the MySQL database. When there is a request from a customer or client to the server, it will be received by a proxy, in this case using HAProxy (High Availability Proxy). HAProxy is load-balancing software and is an open-source proxy to distribute data traffic to multiple nodes or backend servers. HAProxy is useful for maximising the performance and availability of an application by minimising bottleneck servers. Through HAProxy, the load balancer will then forward traffic to the

nodes according to the algorithm used, namely round robin, where this algorithm works alternately by distributing the workload on network traffic evenly and sequentially on each node or server.

The round robin algorithm distributes the load in a rotating and sequential manner from one server to another. Each process is allocated a fixed time slice for execution. Since Round Robin operates with a static time quantum, once a process exhausts its allocated time, it is moved to the end of the ready queue for subsequent execution [21]. Scheduling formula in round robin :

$$User - j = mod((k-1), N) + 1 \text{..... (1)}$$

j refers to the index of the user or process scheduled at a specific time. **k** represents the Transmission Time Interval (TTI) number or the k-th cycle in the scheduling process. **N** denotes the total number of users or processes being scheduled within the system.

Table 1. REST API Endpoint of MSME Application

No.	Method	Request
1	GET	/api/small/stocks
2	POST	/api/small/cart
3	POST	/api/small/pay
4	GET	/api/big/delivery-price
5	POST	/api/big/pickup
6	POST	/api/big/(stuff_name)/stocks

User interface of the MSME Application: customers can see what menus or items are sold and then can place orders or transactions at the store. In developing the backend application using a hexagonal architecture, the code from the interface related to the client, business logic, and infrastructure (database) is made into separate classes. This aims to increase flexibility in replacing and modifying application components if there are changes in the development process without having to change the core of the system. If the customer wants to buy, they can make payments according to the items selected in the basket. In this study, communication between systems uses rest api (Representational State Transfer Application Programming Interface). The protocol used in rest api is HTTP, with various methods including POST and GET. Endpoints that have been made in this MSME application can be seen in Table 1.

2.4 Testing and Performance Evaluation

The system is tested to evaluate its performance efficiency and reliability under simulated MSME workloads. The evaluation focuses on key performance metrics, including latency, which measures the average response time of the system, and throughput, which assesses the number of transactions processed within a given time frame. Additionally, the success rate is analyzed to determine the percentage of successfully completed requests, while power consumption is measured to evaluate the system's energy efficiency for sustainable operations. The experiment is conducted by running a total of 9,000 requests over a duration of five minutes, comparing the performance of the clustering-based pocket data center against two alternative configurations: a single-server PC deployment and a single Raspberry Pi server. This comparative analysis serves to validate the effectiveness of edge computing and clustering in enhancing the reliability and efficiency of MSME services

3. RESULT AND DISCUSSION

This section describes the implementation and testing of the system to obtain data from the research that has been done. The following descriptions outline several tests that have been conducted as part of this process.

3.1 System Implementation

The results and discussion chapter will describe the system development that has been carried out along with the results of testing with several scenarios. Figure 3 shows the system prototype that has been made. The system prototype consists of 2 Raspberry Pis, each of which functions as a node that

p-ISSN: 2528-1682
e-ISSN: 2527-9165

represents two clusters in this system. The master node is deployed on a virtual machine where all nodes are configured using Kubernetes to be able to provide MSME application access to customers more quickly and efficiently. The development of this pocket data centre system applies the concept and architecture of edge computing.

Figure 3. Prototype and System Topology of Pocket Data Centre

Table 2 shows the hardware specifications used in developing the pocket data centre by implementing edge computing principles. The hardware is utilised to create a cluster consisting of 2 nodes configured on 2 Raspberry Pis and 1 master node configured on a virtual machine. All nodes are configured using the Kubernetes orchestrator. In addition, hardware and software specifications for single-server systems on Raspberry Pi and a single server on PC computers are also included.

Table 2. Pocket Data Centre Device Specifications

	Specifications	
Cluster (Cluster Specification)	**Single Server Raspberry Pi**	**Single Server PC**
Orchestration	Micro8s (kubernetes v1.27.5)	CPU 8 core@4.6 GHz 64 GB RAM 256 GB SSD Ubuntu Server 20.04.1
Virtual Machine (Control Plane/Master)	CPU 2 core@3.4 GHz 2 GB RAM Ubuntu Server 20.04.1	
Raspberry Pi CM4 (Worker)	CPU 4 core@1.5 GHz 4 GB RAM eMMC: 16 GB Ubuntu Server 20.04.5	
Raspberry Pi 4B (Worker)	CPU 4 core@1.5 GHz 8 GB RAM sd card : 64 GB eMMC: 16 GB Ubuntu Server 20.04.3	
Customer's computer to send benchmark request	Macbook Pro 2021 M1 Pro	

Figure 4 shows two objects for two services, umkm- app-api-lb-1 and umkm-api-lb-2. The object type is a service that is used to expose the application as a service in the kubernetes cluster. The port used is 9900 with the name HTTP using the TCP protocol where the target Port is 8080 is the destination port selected by the service. Both objects have the same configuration for load distribution on the backend service accessed through LoadBalancer in the Kubernetes environment.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: umkm-app-api-lb-1
  labels:
    app.kubernetes.io/name: umkm-app-api
    tier: backend
spec:
  type: LoadBalancer
  selector:
    app.kubernetes.io/name: umkm-app-api
    tier: backend
  ports:
  - name: http
    port: 9900
    protocol: TCP
    targetPort: 8080
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: umkm-app-api-lb-2
  labels:
    app.kubernetes.io/name: umkm-app-api
    tier: backend
spec:
  type: LoadBalancer
  selector:
    app.kubernetes.io/name: umkm-app-api
    tier: backend
  ports:
  - name: http
    port: 9900
    protocol: TCP
    targetPort: 8080
```

Figure 4. Load balancer configuration file on kubernetes

3.2 System Testing Result

This test using the Vegeta tool, which is an http load testing tool using the Golang programming language and aims to test http load testing. Testing is carried out with a duration of 5 minutes on the pocket data centre clustering system, a single server using a raspberry pi and a single server using a PC. This aims to compare and see the efficiency of using a pocket data centre on edge computing compared to a single server deployed on a raspberry pi and PC. Computer used to send benchmark requests: MacBook Pro 2021 M1 Pro. Connection to the server using WiFi.

Figure 5 shows the results of testing clustering pocket data centre on edge computing. The content of this part (object) is the result of testing the request to the "Add item to cart" endpoint. The endpoint address is /api/small/cart. Each request on this endpoint test is given an ID starting from 1 and increasing each time the request occurs. There is a condition where the client who is testing will make an additional request to the "Make a payment/purchase" endpoint. The endpoint address is /api/small/pay. The condition is that if the request ID is modulus 7, then an additional request is made. That way, the number of requests to the "Make a payment/purchase" endpoint is about 14%. The calculation details are 9000 divided by 7 is 1285 requests. The percentage is obtained from 1285 divided by 9000 multiplied by 100 per cent resulting in ~14.28%.

The conclusion of testing on this endpoint occurs for 4 minutes 59.8 seconds, a total of 9000 requests and all requests successfully get an OK response or no errors at all. Furthermore, the contents of this section (object) are the results of testing requests to the "Display stock items" endpoint. The endpoint address is /api/small/stocks. The conclusion of the test on this endpoint occurred for 4 minutes 59.6 seconds, a total of 30,000 requests, there was one error response from the API and the remaining 29,999 requests managed to get an OK response.


```
{
  "AddToCart": {
    "Latencies": {
      "Mean": "367.642669ms",
      "P50": "110.404202ms",
      "P90": "168.150709ms",
      "P95": "201.691834ms",
      "P99": "10.130173408s",
      "Max": "20.14276714s",
      "Min": "46.902315ms"
    },
    "BytesIn": {
      "Total": 833831,
      "Mean": 92.64788888888889
    },
    "Duration": "4m 59.966093789s",
    "Requests": 9000,
    "Throughput": 29.395791885530045,
    "Rate": 30.003391004353695,
    "Success": 9000,
    "Errors": 0
  },
```

```
'ListOfStocks": {
  "Latencies": {
    "Mean": "19.818618ms",
    "P50": "9.763075ms",
    "P90": "23.860835ms",
    "P95": "30.882111ms",
    "P99": "53.427392ms",
    "Max": "20.000353069s",
    "Min": "3.762329ms"
  },
  "BytesIn": {
    "Total": 5729917,
    "Mean": 190.99723333333333
  },
  "Duration": "4m 59.989890583s",
  "Requests": 30000,
  "Throughput": 97.99618797591052,
  "Rate": 100.0033699192264,
  "Success": 29999,
  "Errors": 1
}
```


Figure 5. Clustering edge computing test results

The same parameters and test scenarios were also carried out on single server raspberry pi and single server pc. The table is the result of a comparison trial of 3 servers under different conditions. The

p-ISSN: 2528-1682
e-ISSN: 2527-9165

results obtained from the experiments were rigorously analyzed to understand the performance dynamics of the proposed pocket data center system. The success rate, latency, and throughput were evaluated under different configurations to benchmark the system's efficiency. The single server Raspberry Pi's lower success rate prompted a detailed evaluation of its hardware and software constraints, as well as network performance under load. The clustering system's results were validated through repeated trials and cross-verified with independent benchmarks to ensure reliability and accuracy of the findings. This thorough analysis provides a robust foundation for the proposed system's applicability in real-world MSME scenarios.

Table 3. Trial Results

No.	Server	Latency (mean)	Throughput	Rate	BytesIn	Success
1	Clustering Server based Edge Computing	367.64 ms	29.4	30	833 831	100%
2	Single Server PC	17 ms	30	30	860 877	100%
3	Single Server Raspberry pi	1.79 s	24.6	25	800 133	64%

Table 3 presents the test results for different server configurations: Clustering Server Edge Computing, Single Server PC, and Single Server Raspberry Pi. The Raspberry Pi's lower success rate (64%) compared to the other servers (100%) is due to hardware limitations, including lower processing power, limited memory, and a constrained network interface, which contribute to higher latency (1.79s) and reduced throughput. Additionally, the lack of a cooling system may cause thermal throttling, further degrading its performance under heavy workloads.

The test was conducted for 5 minutes with 9,000 requests via WiFi, using a MacBook Pro 2021 to send requests to the /api/small/cart endpoint. The Single Server PC achieved the lowest latency (17 ms), followed by the Clustering Server (367.64 ms), and the Raspberry Pi (1.79s). This confirms that Kubernetes-based clustering improves response time and workload management compared to single-server configurations. Throughput testing shows the Single Server PC achieved the highest value (30 requests per second), followed by the Clustering Server (29.4), and the Raspberry Pi (25). While both the Clustering Server and Single Server PC maintained a 100% success rate, the Raspberry Pi experienced a 36% failure rate, highlighting its limitations under high traffic.

To improve performance, clustering two Raspberry Pi nodes is proposed. This approach distributes workloads more efficiently, reduces latency, improves reliability, and ensures scalability, making the pocket data center a viable solution for MSMEs.

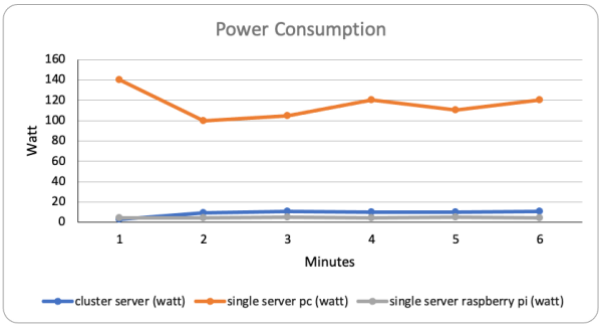

Figure 6. Power Consumption Results

Figure 6 shows a comparison of the power usage of 3 servers with different conditions and specifications but with the same workload during the test duration of 5 minutes. An important aspect of system efficiency is power consumption, particularly for resource-constrained MSMEs. While the single PC server consumed 115.83 watts, the clustering system maintained a low average of 9.01 watts, making it a more sustainable and cost-effective solution. Although the single Raspberry Pi had the lowest power usage (4.51 watts), its high latency and failure rate render it unsuitable for real-world MSME

applications. These results underscore the need for a balance between performance and energy efficiency, where clustering offers a scalable, low-cost alternative without significantly compromising responsiveness.

The test results demonstrate that the pocket data center, through Kubernetes-based clustering and edge computing, provides a scalable and efficient computing solution suitable for MSME operations. While the transactions tested in this study were simulated, they were designed to reflect real-world MSME workflows, including adding items to a cart and processing payments. The system successfully handled 9,000 requests with a 100% success rate in the clustering configuration, ensuring reliability for MSME digital transactions. This capability is crucial for MSMEs that rely on online transactions and e-commerce platforms, where system failures or slow response times can negatively impact sales and customer trust. The efficient power consumption (9.01 watts) further supports MSMEs in reducing operational costs while maintaining high availability. Compared to traditional cloud solutions, this approach allows MSMEs to operate with lower infrastructure investment and reduced reliance on external cloud providers, making digital transformation more accessible and sustainable for small businesses.

4. CONCLUSION

This research successfully developed an edge computing-based pocket data center designed to enhance MSME services by providing a cost-effective, portable, and efficient computing solution. The system utilizes Kubernetes clustering to manage workloads dynamically, ensuring optimized resource distribution and reduced server bottlenecks. Experimental results demonstrate that, within a 5-minute test duration and 9,000 total requests, the pocket data center responded to 100% of requests with a latency of 367 ms, throughput of 29.4 requests per second, and a processing rate of 30 requests per second.

Compared to a single PC server, the pocket data center exhibited slightly lower performance metrics in terms of latency and throughput. However, its energy efficiency was significantly higher, consuming an average of only 9.01 watts, making it a more sustainable and economical choice for MSMEs with limited operational budgets. This trade-off between performance and energy efficiency highlights the suitability of the proposed system for scenarios where cost and power consumption are critical factors, especially for MSMEs operating in areas with unstable power infrastructure.

The implementation of edge computing in the system reduces dependence on cloud services, lowering recurring costs associated with cloud subscriptions while enabling local data processing for improved response times. Additionally, Kubernetes-based clustering ensures scalability, allowing MSMEs to expand their computing capacity as business needs grow without the high costs of traditional data centers. Future research should explore higher-performance edge computing hardware, advanced load balancing techniques, and enhanced security measures to further optimize the system and support broader real-world MSME adoption.

ACKNOWLEDGEMENTS

The authors would like to thank the Center for research and community service Electronic Engineering Polytechnic Institute of Surabaya. This research is local research funded by EEPIS (Electronic Engineering Polytechnic Institute of Surabaya).

REFERENCES

[1] Dahiri, "Analisis Penguatan Umkm Dan Dampaknya Bagi Perekonomian Nasional Sebagai Upaya Mengatasi Dampak COVID-19," *Jurnal Budget*, vol. 5, no. 1, 2020.

[2] B. Surya, F. Menne, H. Sabhan, S. Suriani, H. Abubakar, and M. Idris, "Economic Growth, Increasing Productivity of SMEs, and Open Innovation," *Journal of Open Innovation: Technology, Market, and Complexity. MDPI.* , vol. 5, no. 1, 2020.

[3] L. Zahra Firdausya, D. Perwira Ompusunggu, and K. kunci, "Usaha Mikro Kecil Dan Menengah (Umkm) Di Era Digital Abad 21 Micro, Small And Medium Enterprises (MSME) The Digital Age Of The 21 St Century," *Talijagad*, vol. 2023, no. 3, pp. 14–18, 2023, [Online]. Available: https://journal.unusida.ac.id/index.php/tali-jagad/index|e

[4] G. Evangeulista, A. Agustin, G. P. E. Putra, and H. Madiistriyatno, "STRATEGI UMKM DALAM MENGHADAPI DIGITALISASI," *Jurnal Oikos-Nomos*, vol. 16, no. 1, 2023.

[5] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An Overview on Edge Computing Research," 2020, *Institute of Electrical and Electronics Engineers Inc.* doi: 10.1109/ACCESS.2020.2991734.

[6] M. Satyanarayanan, "The Emergence of Edge Computing," in *IEEE Computer Society*, IEEE Computer Society, 2017.
[7] I. Sittón-Candanedo and J. M. Corchado, "An Edge Computing Tutorial," *Oriental journal of computer science and technology*, vol. 12, no. 2, pp. 34–38, Jun. 2019, doi: 10.13005/ojcst12.02.02.
[8] M. S. Elbamby *et al.*, "Wireless Edge Computing With Latency and Reliability Guarantees," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1717–1737, Aug. 2019, doi: 10.1109/JPROC.2019.2917084.
[9] Y. Jararweh, A. Doulat, O. AlQudah, E. Ahmed, M. Al-Ayyoub, and E. Benkhelifa, "The future of mobile cloud computing: Integrating cloudlets and mobile edge computing," in *International Conference on Telecommunications (ICT)*, IEEE, May 2016.
[10] W. Yu *et al.*, "A Survey on the Edge Computing for the Internet of Things," Nov. 28, 2017, *Institute of Electrical and Electronics Engineers Inc.* doi: 10.1109/ACCESS.2017.2778504.
[11] Y. Ai, M. Peng, and K. Zhang, "Edge computing technologies for Internet of Things: a primer," *Digital Communications and Networks*, vol. 4, no. 2, pp. 77–86, Apr. 2018, doi: 10.1016/j.dcan.2017.07.001.
[12] G. I. Klas, "Fog Computing and Mobile Edge Cloud Gain Momentum Open Fog Consortium, ETSI MEC and Cloudlets," *Computer Science, Engineering, Environmental Science*, 2015, [Online]. Available: www.yucianga.info
[13] A. A. Pratama, Y. Yohanie, F. Panduman, D. K. Basuki, and S. Sukaridhoto, "Edge Computing Implementation for Action Recognition Systems," *Scientific Journal of Informatics*, vol. 7, no. 2, pp. 2407–7658, 2020, [Online]. Available: http://journal.unnes.ac.id/nju/index.php/sji
[14] O. Debauche, S. Mahmoudi, and A. Guttadauria, "A New Edge Computing Architecture for IoT and Multimedia Data Management," *Information (Switzerland)*, vol. 13, no. 2, Feb. 2022, doi: 10.3390/info13020089.
[15] M. A. Nugroho and C. Subiyantoro, "ANALISIS CLUSTER CONTAINER PADA KUBERNETES DENGAN INFRASTRUKTUR GOOGLE CLOUD PLATFORM," *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, vol. 3, no. 2, pp. 84–93, 2018.
[16] R. H. Putra and W. Sugeng, "Implementasi Cluster Server pada Raspberry Pi dengan Menggunakan Metode Load Balancing," *Jurnal Edukasi dan Penelitian Informatika (JEPIN)*, vol. 2, no. 1, 2016.
[17] C. Lin, L. Li, and Y. Chen, "Dynamic system allocation and application of cloud computing virtual resources based on system architecture," *Open Computer Science*, vol. 13, no. 1, Jan. 2023, doi: 10.1515/comp-2022-0259.
[18] Y. Guo, S. Wang, A. Zhou, J. Xu, J. Yuan, and C. H. Hsu, "User allocation-aware edge cloud placement in mobile edge computing," in *Software - Practice and Experience*, John Wiley and Sons Ltd, May 2020, pp. 489–502. doi: 10.1002/spe.2685.
[19] Y. Kuleh, M. A. Kadafi, and Z. Ilmi, "Prospects of Digitalization of MSMEs Business Expansion in Sepakat Village," *International Journal of Social Science and Business*, vol. 7, no. 3, pp. 769–782, Oct. 2023, doi: 10.23887/ijssb.v7i3.53268.
[20] L. Fitriani and H. Nugroho, "Information Technology Strategy for Micro, Small, and Medium Enterprises in the Era of Industry 4.0," *IJAIT (International Journal of Applied Information Technology)*, p. 111, Jul. 2023, doi: 10.25124/ijait.v6i02.5975.
[21] Sakshi *et al.*, "A new median-average round Robin scheduling algorithm: An optimal approach for reducing turnaround and waiting time," *Alexandria Engineering Journal*, vol. 61, no. 12, pp. 10527–10538, Dec. 2022, doi: 10.1016/j.aej.2022.04.006.