KUBERNETES RESOURCE PLANNING TOOL FOR WEB APPLICATIONS

Thanaroj CHAROENPHUTHIWAT[1] and Utharn BURANASAKSEE[1]
1 Faculty of Science and Technology, Rajamangala University of Technology Suvarnabhumi, Thailand; 164480322003-st@rmutsb.ac.th (T. C.); utharn.b@rmutsb.ac.th (U. B.)

ARTICLE HISTORY
Received: 19 April 2024 **Revised:** 3 May 2024 **Published:** 17 May 2024

ABSTRACT

This research proposes a suitable resource management plan for enterprise use on the Kubernetes platform to effectively reduce costs. By shifting investment in organization resources to cloud services. The main objective of this research is to reduce organizational costs and choose a simple and specific resource management plan that can help by selecting the most appropriate resources according to current workload for web application of the organization. This study measures the performance of the tool in comparison to individual experience to assess appropriate resources. The resource management tool will be helpful in deciding the most efficient resources within the scope of the budget available in the organization. Therefore, using this resource management plan will help the organization save costs and increase efficiency in web application.

Keywords: Cloud, Cost-effective, Kubernetes

CITATION INFORMATION: Charoenphuthiwat, T., & Buranasaksee, U. (2024). Kubernetes Resource Planning Tool for Web Applications. *Procedia of Multidisciplinary Research, 2*(5), 20.

INTRODUCTION

Nowadays, cloud systems have become a common choice for organizations, transitioning from investing in their own resources to utilizing rental services. Presently, there are numerous service providers available, such as AWS (Amazon Inc., 2023a), Google Cloud Platform (GCP) (Google Inc., 2023), and Digitalocean (DigitalOcean Inc., 2023). Renting resources aids organizations in reducing the costs associated with employing knowledgeable staff to manage servers. However, cloud expenses can be high if organizations over-provision resources. Thus, selecting appropriate and sufficient resources has become an interesting approach. In cloud systems, most expenses are attributed to Microservices and applications running on Kubernetes Pods. Each Pod comprises various Microservices interacting within a cluster, facilitating scalability. Nonetheless, these nodes often incur high costs (Beda et al., 2023; Richardson & Smith, 2023).

Various cost-saving strategies exist. For example, Amazon EC2 Spot instances (Amazon Inc., 2023b) can partially reduce costs by utilizing suitable amounts of resources. While useful for short-term cost management, Spot instances may not be suitable for all tasks, such as Database Management Systems or systems requiring continuous availability. Another approach involves selecting low-feature servers for initial installation and conducting load testing to match usage levels. Although this approach allows for future upgrades, it may lead to downtime during upgrades and resource shortages. Therefore, cost-effective resource allocation remains a challenge. To address this, a tool has been developed to model Kubernetes-based applications, assisting in selecting the most cost-effective resources tailored to application requirements, user loads, and system efficiency, all within a reasonable budget (TechTarget Contributor., 2023; Mullins, 2023; Amazon Inc., 2023b).

In response to these challenges, this research aims to developing a tool to create models on the Kubernetes platform to select the most cost-effective resources for web applications or software, aiming to reduce costs and enhance system efficiency. This tool addresses the issue of selecting server features suitable for the system, user load, and system efficiency, all within a reasonable budget.

LITERATURE REVIEWS

Kubernetes has emerged as a highly popular open-source platform and the most widely adopted standard for continuously available systems. It is extensively utilized across various workloads, such as big data, web services, and IoT, supporting flexible applications, environmental consistency, OS mobility, and application-centric resource management (Beda et al., 2023).

Previous research by Zhong and Buyya (2020) discussed diverse strategies for cost-effective container management by enhancing resource utilization and flexibly pricing servers with three main approaches. First, supporting different workload profiles to adjust the initial placement of containers in available resources by job packing. Second, resizing clusters to match changing workloads through automatic scaling algorithms. And third, introducing mechanisms for scheduling server shutdowns when underutilized to save costs and reallocating related jobs without compromising progress. However, this research focuses on refining Kubernetes' operational mechanisms to reduce server costs, but it acknowledges the challenge of selecting servers suitable for system usage, lacking guidance on cost-effective server selection based on price, features, and system efficiency.

Ding et al. (2022) proposed a new placement model to group the related pods into the same node. This results in reducing the shared dependency libraries among multiple microservice instance and memory usages. While our research focuses on finding appropriate physical resources that can serve the load requirement, this research addresses different problems by trying to improve the efficiently within the node itself.

Though Kubernetes allow resource usages to be adjusted by the current load, the repair and recovery actions that increase or decrease the number of nodes can cause downtime to the system especially those stateful microservices (Vayghan et al., 2021). Therefore, in this paper, we focus on finding the most suitable server that can fit the needs while having the lowest cost to take advantage of the cloud computing that we can adjust the server specification if the workload changes.

RESEARCH METHODOLOGY

The researchers developed a tool for building models on the Kubernetes platform to select the most cost-effective resources for web applications. The development process consisted of four steps:

Step 1: Researchers conducted a study on cloud service providers supporting server rental on the Kubernetes platform, installation methods, and related research.

Step 2: Researchers analyzed and designed a tool to explore the feasibility of building a model-building tool on the Kubernetes platform for selecting the most cost-effective resources for web applications.

Step 3: Researchers developed a model-building tool on the Kubernetes platform to select the most cost-effective resources for web applications. This involved using tools such as:

1) Utilizing PHP language for developing web applications to test loads.
2) Employing MySQL database for data storage.
3) Scripting for server configuration based on desired test criteria.
4) Utilizing load testing tools with Wrk (Wu, 2019; Will, 2023) to assess loads.
5) Employing cloud services from Digitalocean to utilize the Kubernetes platform.

Step 4: Quality assessment and testing of the developed system involved comparing its features with previous methodologies to evaluate advantages and disadvantages. Expert evaluation was conducted.

RESEARCH RESULTS & DISCUSSION

Upon investigating cloud service providers supporting server rentals on the Kubernetes platform, installation methods, and related research, it was found that Digitalocean is another interesting option with reasonably priced servers compared to others. To enable the system to serve as a model-building tool on the Kubernetes platform for selecting the most cost-effective resources for web applications, servers must have two essential features:

Creation of Kubernetes platform servers allowing selection of Datacenter region. For this research, the author chose the data center in Singapore, which is the closest to Thailand. The minimum node requirement is 2 Nodes, and server features and prices can be selected according to Table 2. The minimum feature will be Basic nodes with 1GB RAM/1 vCPUs at $12/month per node ($0.018/hour).

Table 1 List of available droplet instance

Machine type (Droplet)	vCPUs	RAM	Price/month	Price/hour
Basic nodes	1	1GB	$12	$0.018
Basic nodes	2	1GB	$18	$0.027
Basic nodes	2	2.5GB	$24	$0.036
Basic nodes	4	6GB	$48	$0.071
Basic nodes	8	13GB	$96	$0.143
Basic nodes(Premium Intel)	1	1GB	$14	$0.021
Basic nodes(Premium Intel)	2	1GB	$21	$0.031
Basic nodes(Premium Intel)	2	2.5GB	$28	$0.042
Basic nodes(Premium Intel)	4	6GB	$56	$0.083

Machine type (Droplet)	vCPUs	RAM	Price/month	Price/hour
Basic nodes(Premium Intel)	8	13GB	$112	$0.167
Basic nodes(Premium AMD)	1	1GB	$14	$0.021
Basic nodes(Premium AMD)	2	1GB	$21	$0.031
Basic nodes(Premium AMD)	2	2.5GB	$28	$0.042
Basic nodes(Premium AMD)	4	6GB	$56	$0.083
Basic nodes(Premium AMD)	8	13GB	$112	$0.167

Creation of servers for installing MySQL database version 5.7 by deploying a Droplet server. The server should have the following features: 8CPUs, 16GB RAM, 320 GB SSD Disk, Debian 11 OS. Both features enable the model-building tool on the Kubernetes platform to select the most cost-effective resources for web applications, making it a valuable resource recommendation tool for application developers.
This research focuses on the process of developing a model-building tool on the Kubernetes platform to select the most cost-effective resources for web applications. The analysis and design of the system depicted in Figure 1 when the system administrator has configured the testing environment. In Step 1, the system administrator installs the database, followed by Step 2, where the system administrator installs the load testing tool (WRK). Subsequently, in Step 3, the system administrator deploys the cluster and installs the web application system. Step 4 involves the system administrator sending load testing commands to the load testing tool (WRK). Upon receiving the testing commands, the load testing tool executes the load test and reports the results back to the system administrator. Step 5 entails the system administrator sending commands to delete the cluster and the web application system, along with reporting the deletion results. The steps keep repeated from step 3 to step 5 to test another configuration. Finally, the system administrator compares the test results to make informed decisions on selecting the most cost-effective resources for web applications.

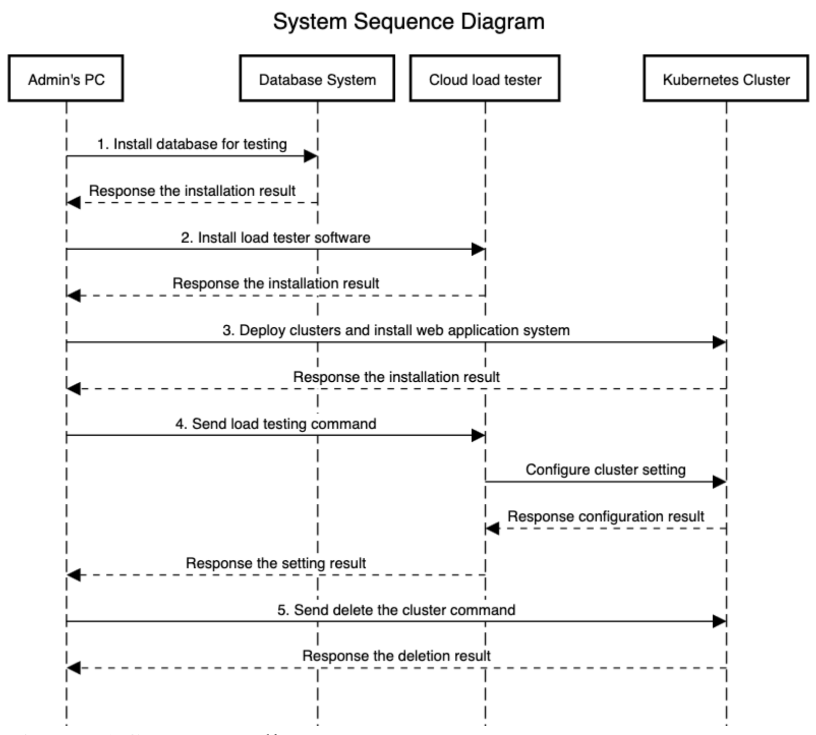
System Sequence Diagram
Admin's PC
Database System
Cloud load tester
Kubernetes Cluster
1. Install database for testing
Response the installation result
2. Install load tester software
Response the installation result
3. Deploy clusters and install web application system
Response the installation result
4. Send load testing command
Configure cluster setting
Response configuration result
Response the setting result
5. Send delete the cluster command
Response the deletion result

Figure 1 Sequence diagram

In developing the system, the researchers opted to utilize PHP language for constructing the web application to log employee data into a MySQL database for load testing purposes using the Wrk load testing tool. Moreover, it was designed to be deployable on both Docker and Cluster environments. The researchers opted to use cloud services from Digitalocean, a notable provider supporting the Kubernetes platform and offering competitive pricing compared to other providers. The architectural framework is depicted in Figure 2.
When the system administrator dispatches a set of commands to create a Cluster and installs the web application system according to predefined conditions, the subsequent step involves the system administrator sending a load testing command set to the load testing tool. The load testing tool receives the command and executes load testing based on the testing configurations set for the web application system through the Kubernetes cluster installed on Docker. The load testing tool simulates system usage and randomly generates sample data to log employee information into the web application system. The web application system then stores the data in the database and returns results to notify the load testing tool of the outcome.

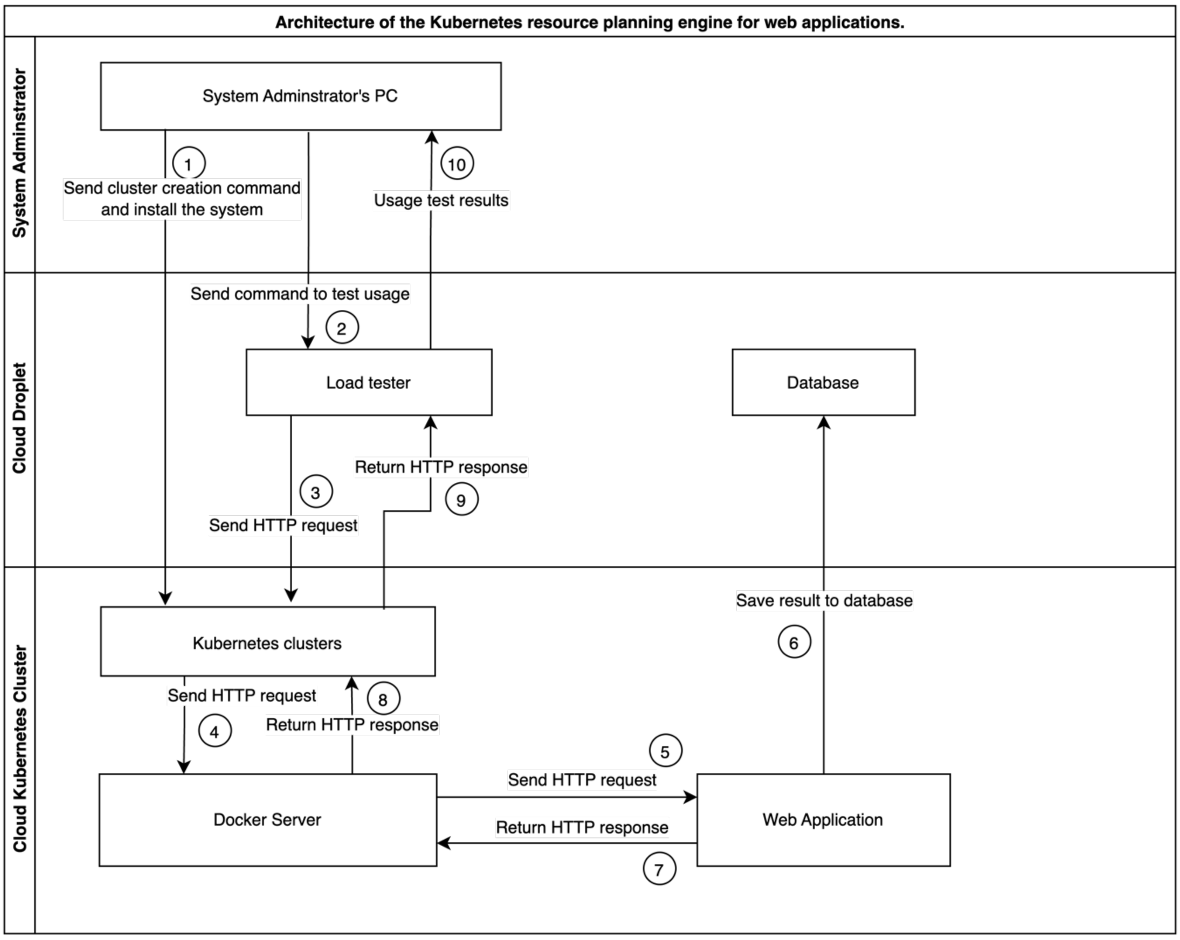

Figure 2 Architecture of the Kubernetes resource planning engine for web applications.

After the installation and testing of the system, the researchers compared the testing data as shown in Figure 3. It can be observed that the basic node of 1vCPU 2GB RAM has a latency average of 14.23ms at a cost of $12 per month. Following this, the basic node of 2vCPU 2GB RAM has a latency average of 17.64 ms at a cost of $18 per month. Subsequently, the basic node of 2vCPU 4GB RAM exhibits a latency average of 24.75ms with a monthly cost of $24 . Moreover, the basic node of 8vCPU 16GB RAM shows a latency average of 17.51ms with a monthly cost of $96 .Moving on to CPU-optimized instances, the instance of 2vCPU 4GB

RAM displays a latency average of 12.36ms at a cost of \$42 per month. Additionally, CPU-optimized instance of 4vCPU 8GB RAM presents a latency average of 12.16ms with a monthly cost of \$84 . Finally, CPU-optimized instance of 8vCPU 16GB RAM demonstrates a latency average of 6.27ms at a monthly cost of \$168.
The experiment shows that though the basic node has higher number of core count than that of the CPU-optimized node, the CPU-optimized instances outperform the basic node on the latency results which can be interpreted to more responsive result on the web application. Furthermore, the instance of CPU-optimized of 2vCPU 4GB RAM cost lesser than the instance of basic node of 8vCPU 16GB RAM.

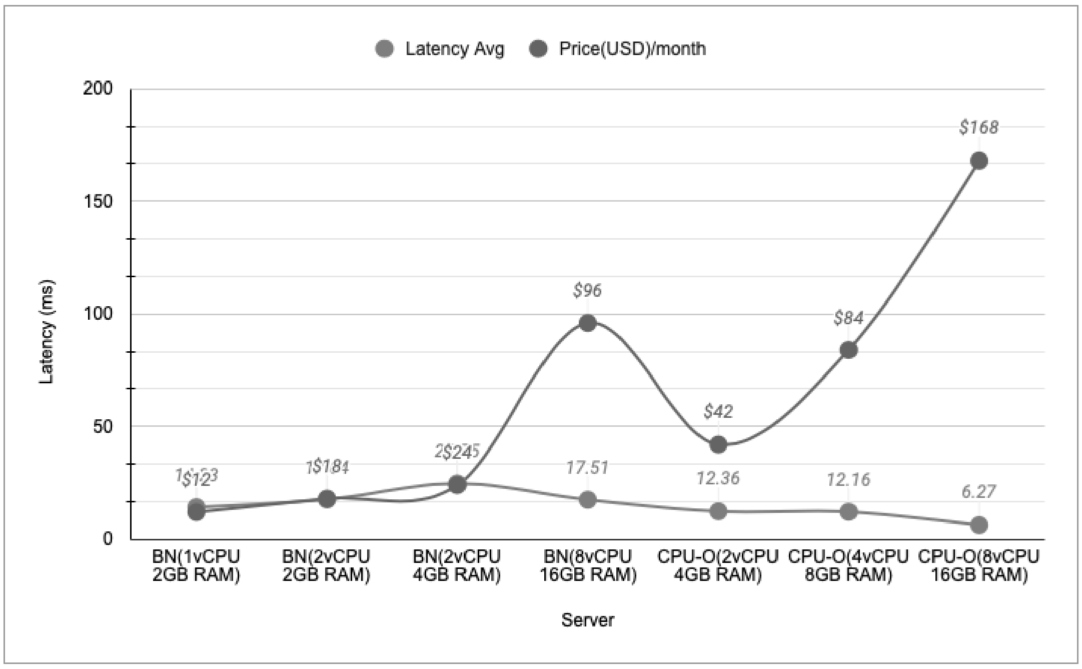

Figure 3 Graph compare average latency and price per droplet size

In addition, the system's quality was evaluated by comparing the advantages and disadvantages of the proposed methods, as illustrated in Table 2 . The method proposed in this research can effectively develop a tool for modeling on the Kubernetes platform to select the most cost-effective resources for web applications. The presented method proves to be cost-effective and efficient.

Table 2 Comparison of the Proposed Method with the Expert System

Feature	Proposed Method	Expert System
Server Selection	Server recommendation system available	Manual selection required
Installation	Automated installation commands	Manual installation
Performance	Can measure server performance	Use tools for performance measurement
Cost-effectiveness of Server Choice	Displays performance metrics and pricing	Not available

CONCLUSION

Given the current trend in investing in cloud service rentals and utilizing web applications on Kubernetes platforms to reduce system costs, this research focuses on selecting suitable and cost-effective server resources for web applications. The approach proposed in this study

involves developing a modeling tool on the Kubernetes platform to assist organizations in selecting the most suitable and efficient resources for web application operations.
The researchers analyzed and processed data to determine the maximum cost-effectiveness for resources, considering factors such as server resource usage volumes at different time intervals and individual experience data to assess the suitability of resources for web applications.
The outcomes of this research will enable organizations to select suitable and efficient resources within the budgetary constraints. Furthermore, the researchers plan to develop a system interface in the future to allow general users to access and select server resources independently to simulate selecting the most cost-effective resources.

REFERENCES

Amazon Inc. (2023a). *Amazon Web Services (AWS)*. Retrieved from https://aws.amazon.com

Amazon Inc. (2023b). *Amazon EC2 Spot*. Retrieved from https://aws.amazon.com/th/ec2/spot

Beda, J., Burns, B., & McLuckie, C. (2023). *Kubernetes*. Retrieved from https://kubernetes.io/blog/2018/07/20/the-history-of-kubernetes-the-community-behind-it.

Digitalocean Inc. (2023). Digitalocean. Retrieved from https://www.digitalocean.com.

Ding, Z., Wang, S., & Jiang, C. (2022). *Kubernetes-oriented microservice placement with dynamic resource allocation*. IEEE Transactions on Cloud Computing.

Google Inc. (2023). *Google Cloud Platform*. Retrieved from https://cloud.google.com/gcp

Mullins, C. S. (2023). *Database management system*. Retrieved from https://www.techtarget.com/searchdatamanagement/definition/database-management-system.

Richardson, C., & Smith, F. (2023). *Microservices: From Design to Deployment*. Retrieved from https://www.nginx.com/blog/microservices-from-design-to-deployment-ebook-nginx.

TechTarget Contributor. (2023). *Load testing*. Retrieved from https://www.techtarget.com/searchsoftwarequality/definition/load-testing.

Vayghan, L. A., Saied, M. A., Toeroe, M., & Khendek, F. (2021). A Kubernetes controller for managing the availability of elastic microservice based stateful applications. *Journal of Systems and Software, 175*, 110924.

Will, G. (2023). *Wg/wrk: Modern HTTP benchmarking tool*. Retrieved from https://github.com/wg/wrk.

Wu, W., Feng, X., Zhang, W., & Chen, M. (2019, November). MCC: a predictable and scalable massive client load generator. In *International Symposium on Benchmarking, Measuring and Optimization*. Denver, CO, USA. 319-331

Zhong, Z., & Buyya, R. (2020). A cost-efficient container orchestration strategy in kubernetes-based cloud computing infrastructures with heterogeneous resources. *ACM Transactions on Internet Technology (TOIT), 20*(2), 1-24.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

Conflicts of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.