Server Scalability Using Kubernetes

Lily Puspa Dewi, Agustinus Noertjahyana, Henry Novianus Palit, Kezia Yedutun
Informatics Department, Faculty of Industrial Technology Petra Christian University
Surabaya, Indonesia
lily@petra.ac.id

***Abstract*— An enterprise that has implemented virtualization can consolidate multiple servers into fewer host servers and get the benefits of reduced space, power, and administrative requirements. Sharing their hosts' operating system resources, containerization significantly reduces workloads, and is known as a lightweight virtualization. Kubernetes is commonly used to automatically deploy and scale application containers. The scalability of these application containers can be applied to Kubernetes with several supporting parameters. It is expected that the exploitation of scalability will improve performance and server response time to users without reducing server utility capabilities. This research focuses on applying the scalability in Kubernetes and evaluating its performance on overcoming the increasing number of concurrent users accessing academic data. This research employed 3 computers: one computer as the master node and two others as worker nodes. Simulations are performed by an application that generates multiple user behaviors accessing various microservice URLs. Two scenarios were designed to evaluate the CPU load on single and multiple servers.**

On multiple servers, the server scalability was enabled to serve the user requests. Implementation of scalability to the containers (on multiple servers) reduces the CPU usage pod due to the distribution of loads to containers that are scattered in many workers. Besides CPU load, this research also measured the server's response time in responding user requests. Response time on multiple servers takes longer time than that on single server due to the overhead delay of scaling containers.

Keywords—server scalability; kubernetes; microservice

I. INTRODUCTION

In recent years, information technology plays a strategic role to excel the business processes. This role attains the enterprise's competitive edge in the market. Information technology is used in many aspects of business to support a well-groomed enterprise [1]. Complex resource requirements of enterprise services and needs to serve for peak demands are reasons for using high-capacity physical servers. Therefore, multicore processors are found implemented in many physical servers. As server processing power and capacity increase, bare-metal applications are not able to exploit the new abundance in resources. Multicore processors have enough abilities to do a variety kind of processes simultaneously, but only several applications at this time can exploit this capability. Therefore, server utilization is typically low, which incurs high investments and high operational cost. A previous study estimated that the utilization of servers in a typical company was around 15-20 percent [2]. Currently, server consolidation emerges as the way to eliminate unnecessary costs and increase the return on investment in the data center. Although, server consolidation yields an improved server resources usage, it needs complex configurations on data, application and network. Consolidation does not mean cramming as many applications into the server as one can find or afford. This effort will bring more problems than it solves. The aim of consolidation is to create a group of systems so that they can be managed and maintained more easily and efficiently. This condition causes the requirement of an expert user to deal with the issue. To reduce this complexity, enterprises use the virtualization technology to simplify the detailed server resources from users while optimizing resource sharing.

Virtualization enables one single server to function as multiple "virtual machines", with each virtual machine able to operate in a different environment. Virtualization is based on the computer architecture and provides functionality of a physical server. As a result, an enterprise that has implemented virtualization can consolidate multiple servers into fewer host servers and get the benefits of reduced space, power and administrative requirements. Virtualization can increase IT flexibility and scalability while creating significant cost savings. It can serve greater workload mobility and improve performance and availability of resources; those are the advantages that make IT administration easier to manage and operate [3].

As a virtual machine has its own operating system, it has a benefit that the application running on it has access to full resource which the operating system can provide. However, a virtual machine utilizes a lot of system resources, since the virtual machine starts not just a full copy of an operating system but a copy of the whole hardware that the operating system runs on. That's why virtual machine is called as heavyweight virtualization [4]. Another type of virtualization is host-level virtualization and known as container. This virtualization sits on top of a physical server with one operating system installed to support several independent systems. Containerization wraps the application code together with the related configuration files, libraries, and all the dependencies required for it to run. This wrap of application or container is abstracted away from the host operating system; thus, it becomes portable and standalone. The wrap runs like in the sandbox – able to run across any platform [5].

By sharing its host's operating system resources, this system significantly reduces workloads, and known as a lightweight virtualization [6]. The size of the container is usually measured in tens of megabytes and it only takes 1–2 seconds to provision one. When the application workload is increasing, new containers can be created and deployed fast. Workload of application that runs on a container can be monitored using a system called Kubernetes [7]. Kubernetes is

978-1-7281-3755-1/19/$31.00 ©2019 IEEE

an open-source platform to manage containerized applications, including managing workloads and services [8]. Kubernetes is designed to automate deploying, scaling, and operating containerized applications. With the scalability feature of Kubernetes technology, the container automation process can be implemented according to the number of concurrent users accessing it. The scalability process of this container can be applied to Kubernetes with several supporting parameters. It is expected that the exploitation of scalability will improve performance and server response time to users without reducing server utility capabilities.

This research focuses on applying the scalability in Kubernetes and evaluation its performance on overcoming the increasing number of concurrent users accessing academic data.

II. Research Method

In the previous study, Emiliano Casalicchio and Vanessa Perciballi [9] explained how containers in Kubernetes are used to manage scalability using relative and absolute metrics. Relative metrics comprises values which are based on data collected from virtual systems / groups using tools such as docker stats or cAdvisor. For example, in docker, the relative CPU utilization is used to measure the percentage usage of total capacity. Whereas absolute metrics are collected from the file system using standard monitoring tools such as mpstat or sar. Fig. 1 shows absolute and relative metric.

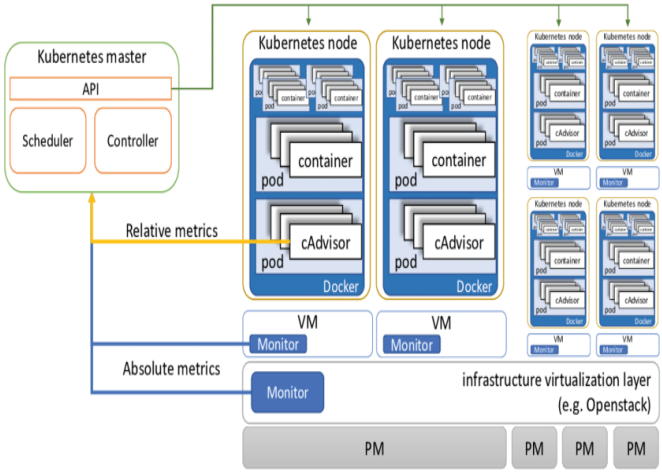

Fig. 1. Architecture of auto scaling in Kubernetes with absolute and relative metrics [9]

For example, Kubernetes makes it possible to create and use units called pods. The method used is called KHPA (Kubernetes Horizontal Pods Auto-scaling), implemented in Kubernetes based on a control loop with a period of $\tau = 30$ seconds. KHPA takes the target input as a percentage of the requested CPU, and the output is the pods target number (P). The algorithm collects the pod relative CPU utilization which is measured by cAdvisor every second and saves it into U vector. The KHPA algorithm also includes the possibility to determine the minimum and maximum number of pods, and allows to postpone the allocation / deallocation of resources to avoid the effects of instability. By comparing the KHPA usage, previous research proposed a new autoscaling algorithm based on absolute metric scaling decisions. Performance comparisons showed that the usage of absolute metrics makes it possible to properly control application response times and keep them below a specified threshold, e.g., in the range of 85–90% for high-powered servers with absolute CPU utilization.

This research applied academic services in the university as the case to test the scalability server. In the current services, the large number of simultaneous accesses may cause a slowdown to the single server resulting in connection timeouts. Implementing Kubernetes will provide an alternative route for users accessing the server. This research employed 3 computers: one computer as the master node and two others as worker nodes. Simulations are performed by an application that generates multiple user behaviors accessing various microservice URLs. The master node is set to manage microservice workers, each of which will have several containers later. Fig. 2 presents the system architecture.

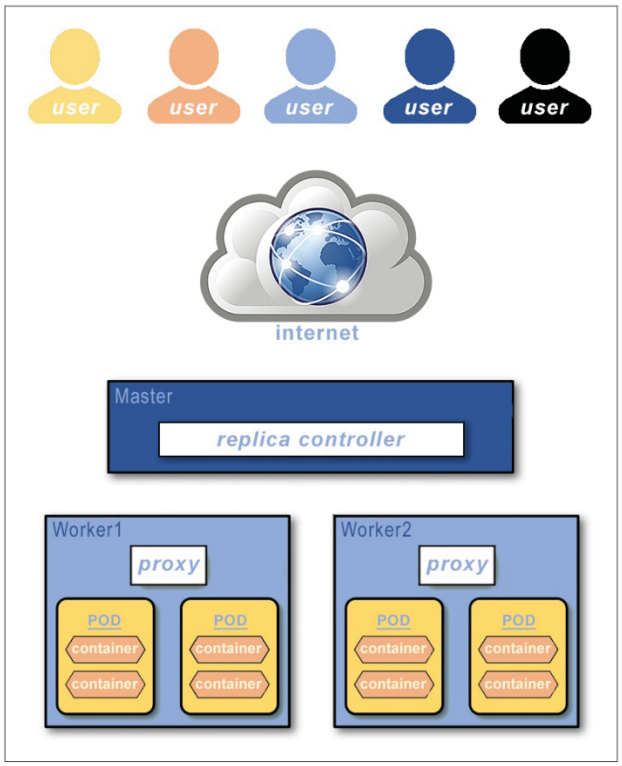

Fig. 2. System architecture

Kubernetes is the most popular container orchestration tool available and is maintained by one of the fastest-growing open-source communities. The Kubernetes project originated within Google, a long-time user of a massive number of containers. To manage these containers well, they need to develop a system for container orchestration. Kubernetes combines the lessons that Google learned from years of container usage into a single tool with an array of features that make container orchestration simple and adaptable to the wide variety of use cases in the technology industry. Since it became open source in July 2015, the capabilities of Kubernetes have continued to grow [10]. Activities required in this research include

Kubernetes initialization, script creation in microservices, scalability implementation in Kubernetes, and testing.

Kubernetes initialization process was carried out on masters and workers with containers inside. The Kubernetes masters (or controllers) are machines (virtual or physical) that run the API server, controller manager, and scheduler components of the Kubernetes cluster. The Kubernetes workers (or nodes) are machines (virtual or physical) that run the kubelet component of the Kubernetes cluster. The workers are the resources on which Kubernetes schedules containers (or pods).

In general, the software application has three main components which are user interface, application logic, and data access. In a monolithic model, all three components are merged in one large system. They are dependent on each other and deployed together. A monolithic application has one huge program code where all components (user interface, backend job, application logic) get attached one another. Although monolithic applications are preferred in certain circumstances, there are many problems with them. Microservice is a way to create independent applications. It divides the existing monolithic service into smaller parts (services) where they are mutually independent and have different functions [11]. Fig. 3 contrasts the monolithic and microservice architectures.

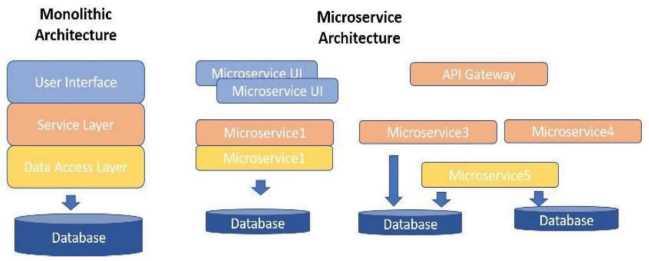

Fig. 3. Monolithic vs Microservice archicture [11]

In creating microservices, REST will be used as the API for every microservice. The academic process is divided into multiple services, realized into GET, POST, PUT, and DELETE operations on resources.

Next step is configuring scalability in Kubernetes. The horizontal pod auto-scaling settings contain the CPU usage percentage as well as the minimum and maximum numbers of pods to determine the replica container that will be created as needed. CPU load limit and memory load will be set for every microservice pod. When the number of user requests exceeds the CPU load limit, a pod replica will automatically be created. On the other hand, when the number of user requests is below the CPU load limit, then Kubernetes will automatically scale down the number of pods.

III. RESULT

The benefits of server virtualization are the result of a reduction in the total number of systems and associated recurring costs (rack space, cooling, power, peripherals, etc.) As virtualization consolidates the existing infrastructure, server capabilities are more strategic in view of the goal of integrating infrastructure [12].

The evaluation performance metrics of microservices include the number of concurrent users and the response time of single and multiple servers. The evaluated simulation was conducted in two scenarios. The first scenario was simulating the user access to microservices in single server and multiple servers (scalability). The testing was performed on three microservices which are 2 GET services and 1 POST service, associated with an academic process in the university. The second scenario was simulating the user access with a scripted behavior, which requires 2 GET services, 7 POST services, 1 DELETE service and 1 PUT process. User accesses were generated to make simultaneous and continuous requests until the CPU load exceeded the specified target (i.e., above 80%).

A. First Scenario

The scenario was conducted by simulating user accesses to single and multiple servers (i.e., implementing the scalability). Tests were carried out by running 8000 user requests and measuring CPU usage pod on a single server and multiple servers. Table 1 presents the test result.

TABLE I. COMPARING CPU USAGE POD IN SINGLE AND MULTIPLE SERVERS (SCENARIO I)

Task	Microservice	CPU Usage Pod (in millicores) on Single Server	CPU Usage Pod (in millicores) on Multiple Servers
1	GET-1	622.00	230.00
	GET-2	584.00	217.00
	POST	550.00	273.00
2	GET-1	609.00	267.00
	GET-2	570.00	247.00
	POST	609.00	311.00
3	GET-1	645.00	213.00
	GET-2	632.00	237.00
	POST	544.00	208.00
Average	**GET-1**	**625.33**	**236.66**
	GET-2	**595.33**	**233.66**
	POST	**567.66**	**264.00**

B. Second Scenario

The second scenario was using a scripted behavior to simulate user requests to the server. The script simulated 150 users accessing various microservices with a total of 150 × 11 = 1,650 requests. Table 2 shows the comparison of CPU usage pod in single and multiple servers (using scalability). The result shows that CPU usage pod on multiple servers is lower than that on single server.

TABLE II. COMPARING CPU USAGE POD IN SINGLE AND MULTIPLE SERVERS (SCENARIO II)

Task	CPU Usage Pod (in millicores) on Single Server	CPU Usage Pod (in millicores) on Multiple Server
1	576.00	209.00
2	526.00	369.00
3	498.00	333.00
Average	**533.33**	**303.66**

C. Evaluating the Response Time

In addition to measuring CPU usage pod on single server and multiple servers, evaluation on response time was also carried out. There was a difference in response time between deployments on a single server and on multiple servers. Due to CPU and memory limitation on the client's device, sometimes this limitation caused a long time delay. When the CPU load exceeded the specified target, the process of deploying containers also took time so that it affected the response time.

TABLE III. RESPONSE TIME ON SINGLE AND MULTIPLE SERVERS

Response Time (ms)	Single Server (no scalability)	Multiple Servers (with scalability)
Get data (first task)	00:00:44	00:01:07
Get data (second task)	00:00:40	00:01:06

IV. CONCLUSION

Virtualization using containers enables much flexibility for capacity management in a server. In this research, we had conducted two scenarios to evaluate the CPU load on single and multiple servers. On multiple servers, the server scalability was enabled to serve the user requests. Implementation of scalability to the containers (on multiple servers) reduces the CPU usage pod due to the distribution of loads to containers that are scattered in many workers.

Besides the CPU load, this research also measured the server's response time in responding user requests. Response time on multiple servers takes longer time than that on single server due to the overhead delay of scaling containers.

ACKNOWLEDGMENT

This work is supported by Directorate General of Higher Education Indonesia and Petra Christian University, Research Center.

REFERENCES

[1] S. Dutta and I. Mia, "The global information technology report 2009–2010," in World Economic Forum and INSEAD, SRO-Kundig Geneva, Switzerland, 2010.

[2] Soaring Eagle Database Consulting, "What is server consolidation and when is it necessary?." 2018. [Online]. Available: https://soaringeagle.biz/what-is-server-consolidation-and-when-is-it-necessary/.

[3] C. Chuan-Fu and C. Shiuann-Shuoh, "Implement server virtualization and consolidation using 2P-cloud architecture," Journal of Applied Science and Engineering, vol. 20, no. 1, pp. 121-130, 2017.

[4] S J. Vaughan-Nichols, "Containers vs. virtual machines: How to tell which is the right choice for your enterprise." 2016. [Online]. Available: https://www.networkworld.com/article/3068392/containers-vs-virtual-machines-how-to-tell-which-is-the-right-choice-for-your-enterprise.html

[5] D. Ageyev, O. Bondarenko, T. Radivilova, and W. Alfroukh, "Classification of existing virtualization methods used in telecommunication networks," in *IEEE 9th International Conference on Dependable Systems Services and Technologies (DESSERT)*, pp. 83-86, Ukraine, 2018. doi: 10.1109/DESSERT.2018.8409104.

[6] IBM Cloud Education, "Containerization." 2019. [Online]. Available: https://www.ibm.com/cloud/learn/containerization.

[7] Kubernetes, "Production-Grade Container Orchestration." 2019. [Online]. Available: https://kubernetes.io/.

[8] Kubernetes, "Concepts." 2018. [Online]. Available: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/.

[9] E. Casalicchio and V. Perciballi, "Auto-Scaling of Containers: The Impact of Relative and Absolute Metrics," in *IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS*W),* Tucson, AZ, USA, September 18-22, 2017.

[10] Elastisys, "Setting up highly available Kubernetes clusters." 2018. [Online] Available: https://elastisys.com/wp-content/uploads/2018/01/kubernetes-ha-setup.pdf.

[11] N. Rajagopalan and K. Gowda, "Journey from Monolith Application to Microservices," IEEE India Info, vol. 14, no. 2, pp. 120-124, Apr - Jun 2019.

[12] V.V. Rao and M.V. Rao, "A Survey on Performance Metrics in Server Virtualization with Cloud Environment," Journal of Cloud Computing, vol. 2015, 2015, Article ID 291109. doi: 10.5171/2015.29 1109.