Kubernetes-based Workload Allocation Optimizer for Minimizing Power Consumption of Computing System with Neural Network

Ryuki Douhara*, Ying-Feng Hsu†, Tomoki Yoshihisa†, Kazuhiro Matsuda†, Morito Matsuoka†
* Graduate School of Information Science and Technology, Osaka University, Osaka, Japan
Email: r-douhara@ist.osaka-u.ac.jp
† Cybermedia Center, Osaka University, Osaka, Japan
Email: {yf.hsu@, yoshihisa@, kazuhiro.matsuda@ane., matsuoka@}cmc.osaka-u.ac.jp

***Abstract*—Edge computing has been attracting attention due to the spread of the Internet of Things. For edge computing, containerized applications are deployed on multiple machines, and Kubernetes is an essential platform for container orchestration. In this paper, we introduce a Kubernetes based power consumption centric workload allocation optimizer (WAO), including scheduler and load balancer. By using WAO built with power consumption and response time models for actual edge computing system, 9.9% power consumption was reduced compared to original Kubernetes load balancer. This result indicates that the WAO developed in this study exhibits promising potential for task allocation modules as a micro service platform.**

Keywords—kubernetes, edge computing, neural network, osmotic computing

I. INTRODUCTION

Edge computing, which mainly uses the computing resources of machines at the edge of a computer network, has been attracting attention due to the spread of the Internet of Things (IoT) [1]. While edge computing potentially reduces communication time, large-scale computing resources will be required to process the information acquired from a vast number of user devices. As a result, computing resources have dramatically increased in number and their management has become more complex. Kubernetes (K8s), an open-source container orchestration tool, is a valuable tool for managing complex container-based applications [2]. With the advent of K8s, it has become easier to manage large amounts of computing resources, but the increase in the total power consumption of those computing resources remains an critical issue. Since communication traffic will continue to increase in the future, reducing power consumption in edge computing should be prioritized when allocating tasks among computing resources.

Kube-scheduler is the default task scheduler in K8s which uses Pod as the smallest deployable unit. It consists of single or multiple containers. When distributing Pods to Nodes, K8s does not provide an implementation of advanced network load-balancers. When allocating tasks to candidate Pods, even with MetalLB, K8s cluster only provides simple load-balancers with equal probability. Therefore, from the aspect of optimizing power consumption, these standard functions of K8s fall short and have room for improvement. Consequently many researches have been actively conducted on scheduling algorithms centered on load balancing as a task allocation method in the cloud and edge computing systems. In these algorithms, the scheduler distributes loads and reduces standby power consumption by sequentially stopping unnecessary computing resources [3]. Studies have shown an achievement of an excellent reduction of power consumption in the simulation environment.

However, in a real environment, repeatedly stopping and starting computing resources impairs the robustness of services because of the startup time and the deployment time of these services. Also, a non-proportional relationship between the power consumption of servers and CPU usage needs to be considered. In general, as both CPU usage and temperature around servers rise, the power consumption of servers increases, because the rotation speed of cooling fans in the server also increases. Therefore, for power saving in edge computing systems, it is necessary to allocate tasks by considering the relationship between power consumption and CPU usage. The scheduler should process a huge amount of variables (the types, placement, and usage of computing resources, and the number of resources required by tasks, among others), so various heuristics and new paradigms have been proposed. Osmotic Computing is one solution that optimally allocates tasks based on the principle of osmotic pressure in the cloud edge environment [4], [5]. However, it is difficult to optimize Pod and task allocation without considering processing performances. To resolve this issue, machine learning provides a solution to predict each task's resource usage and processing time in advance.

In this paper, we propose a proof-of-concept design of low power consumption policies for an open-source Kubernetes container orchestrator. We employ workload allocation optimizer (WAO) and extends it through the K8s container orchestrator platform to realize the power consumption reduction in an edge computing system [6], [7]. Based on the K8s platform, we implement both a WAO based scheduler (WAO-scheduler) and WAO based load balancer (WAO-LB) to

978-1-7281-7624-6/20/$31.00 ©2020 IEEE
DOI 10.1109/CSCI51800.2020.00238

minimize overall data center power consumption. Our proposed work provides optimal real-time task allocation strategies by accounting for both power consumption and client-server response time. In edge computing, a task's response time is one of the most important service operating factors. Our WAO-schedule architecture ranks those selected Nodes by using a neural-network–based power prediction model. A higher rating for a Node means that it is expected to have a lower increase in power consumption when using the computing resources. In WAO-LB, we define an evaluation formula based on the concept of osmotic pressure and add power consumption models (PC model) and response time models (RT model) created using neural networks. The optimizer is also set to perform specifically to the power consumption and response time requirement of each application. There are two major contributions in this paper.

1. We propose a WAO-scheduler, a computational resource allocation scheme for K8s. It is a customized Kube-scheduler that uses neural networks to reduce power consumption. We also introduce WAO-LB, which considers the power-saving and response time required for edge computing.
2. To show the feasibility of both WAO-scheduler and WAO-LB, we evaluated the performance of the proposed system through large-scale object detection tasks and showed the results of reduced power consumption.

The remainder of the paper is structured as follows. In Section II, we briefly review related works. In Section III, we present the details of both the proposed WAO-scheduler and WAO-LB. In Section IV, we use a real testbed edge data center to evaluate the performance of our proposed approach. We conclude the paper in Section V with brief future work.

II. Related Works

With the development of cloud-edge computing and container technology, K8s has been gaining a lot of attention. The Kube-scheduler system is used to allocate Pods to Nodes and has been one of the most active research community topics. The scheduler determines which Nodes are the proper placement for each Pod in the scheduling queue based on constraints and available resources. Chang et al. proposed a platform that dynamically manages the number of Pods deployed on a K8s cluster according to Node resource usage [8]. In their platform, Nodes are monitored using multiple monitoring tools, and the number of Pods is increased or decreased when the overall CPU usage is above or below a certain threshold. Townend et al. clarified the importance of considering the characteristics of hardware and software when scheduling Pods [9]. In their scheduler, Nodes are monitored and modeled using specialized machines. In an ideal environment, this scheduler facilitates a reduction in overall power consumption. However, there is no research focusing on the effectiveness of model creation. An appropriately designed model can provide flexibility over control of power consumption and response time.

The development of cloud-edge computing has also add importance to scheduling algorithms. However, finding the optimal solution for scheduling is a challenging task, due to the heterogeneous environment dealing with the diversity of computing resources, layout and configuration of data centers, and dynamic task requirements. As a result, various methods for introducing heuristic algorithms and new paradigms have been proposed. Nishant et al. achieve load balancing in cloud computing by using ant colony optimization (ACO), an algorithm inspired by how ants form a path to food by using pheromones [10].

In [11] and [12], a framework based on the concept of osmotic computing was proposed and evaluated. This framework has also been used in combination with various heuristic algorithms. Gamal et al. proposed a load balancing algorithm that combines the osmotic behavior with bio-inspired heuristics, ACO, and artificial bee colony (ABC) [3]. Efficient load balancing is achieved by combining ACO and ABC while also considering the power consumption on the basis of osmotic theory. Kaur et al. have proposed a framework based on osmotic computing for optimal task allocation using hyper-heuristics, where the two evaluation axes are energy consumption and latency [13].

The studies above have all achieved excellent load balancing in a simulated environment. However, this is not the optimal solution for overall reduction of power consumption. Since the regional edge computing environment varies, QoS-aware functionality scheduling methods provide a better solution. Our proposed system focuses on reducing power consumption while keeping a short response time. It also uses power consumption and response time models that keep an eye on both the requirements of tasks and the status of the computing resources.

III. Methodology

Since our proposed WAO-scheduler and WAO-LB were implemented based on K8s, in this section, we first review the foundation of K8s. We then elaborate on the schema of the proposed WAO-scheduler and WAO-LB from the aspects of architecture, PC models, RT models, and optimization equations. K8s is an open-source container orchestration platform that deploys, monitors, and scales containerized applications. In K8s, containerized applications are managed as Pod, which are the smallest deployable computing units and consist of one or more containers. The deployment plan defines the number of Pod replicas, port openings, resource requirements, and other definitions. When the deployment plan is defined, the Kube-scheduler is responsible for assigning Pods to Nodes, which can be either virtual or physical machines. Besides, K8s defines Pods as a logical set using an abstract object called Service. Service provides the same IP address for Pods with the same Pod labels; This allows clients to access the applications without considering any information changes in Pods caused by application destruction and regeneration.

A. WAO based Scheduler

1) Operation of default Kube-scheduler: Kube-scheduler is Kubernetes's default scheduler and its control plane for allocating Pods to Nodes. When a Pod deployment request arrives, the Kube-scheduler determines to which Node each Pod in the scheduling queue should be placed, based on the available resources and constraints. This scheduling process includes three orderly phases.In the filtering phase, the Kube-

scheduler lists all available Nodes by checking which Nodes have an available resource that meets the Pod's resource requirement. In the scoring phase, the Kube-scheduler ranks those selected Nodes based on their scores calculated with the default priority functions. And the Node with the highest score is selected. In short, the scoring stage is responsible for determining the best Node for Pod allocation. In the binding phase, via API server, it notifies the selection to the best Node in which a Pod is formed. The recently released version of Kube-scheduler made this customized control rules of each phase (filter, score, bind) possible. With the add-on component, named Scheduling Framework, we implemented the WAO-scheduler based on the scoring phase to provide the function of optimal workload allocation and power efficiency management.

2) Operation of WAO based scheduler: The architecture of the WAO-scheduler is shown in Fig. 1. Based on the Scheduling Framework, we design a power consumption efficient control component the Power Consumption based Scorer (PCS) in the scoring phase. The PCS first collects information on each candidate Node using the metrics server and intelligent platform management interface (IPMI) exporter. The TensorFlow Serving server uses gRPC to provide inference from PC models created by TensorFlow. The Metrics Server aggregates the Nodes' resource usage in the K8s cluster and each temperature around Node is obtained from the IPMI exporter. The filtering phase first lists all available Nodes by eliminating inappropriate Nodes. And then, the scoring phase runs and takes the appropriate Node list from the filtering phase, scores each Node, and selecting the Node with the highest score. In the WAO-scheduler, the scoring phase uses the resource usage from the Metrics Server and IPMI exporter and prediction from TensorFlow Serving. In other words, our Pod allocation strategy considers not only the increased computing resources due to Pods deployment but also the increase in power consumption each Node (in this study, a physical server). We have designed a neural network-based PC prediction model for the WAO-scheduler and have deployed it to the TensorFlow Serving server in K8s. TABLE I summarizes the structure and hyperparameters of this PC model.

TABLE I. DESCRIPTION OF NEURAL-NETWORK–BASED POWER CONSUMPTION PREDICTION MODEL

Input		CPU usage, temperature around Node
Neural network model	Layers	1 hidden layer with size of 5,000 nodes
	Optimizer	Adam (with hyperparameters of lr=0.0005, beta1=0.9, beta2=0.999, epsilon=1e-8, decay=0.0)
	Loss function	mean squared error (MSE)
	Batch size	256
Output		Power consumption of each Node

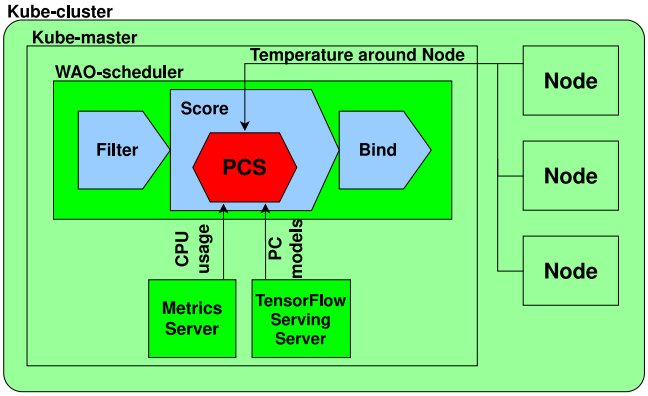

Fig. 1. Architecture of WAO-scheduler.

B. WAO based load balancer

After the WAO-scheduler has completed the Pod allocation, it is ready to process clients' requests. Currently, K8s does not provide an implementation of advanced network load-balancers. MetalLB, provides a basic implementation of a load-balancer for bare metal K8s clusters. The mechanism of load balancing is implemented by rewriting the iptables and using equal distribution rules for task allocation. For example, when there are three Pods, there is a 33% chance for each Pod to process a task. To the best of our knowledge, our WAO-LB is the first attempt to extend K8s load balancing by considering the minimization of power consumption. Besides, we implement the WAO-LB optimizer by using go language, which is the native language of Kubernetes, and having the benefits of using parallel processing to reduce the processing time for large-scale Nodes power consumption evaluation.

The architecture of optimizer WAO-LB is shown in the Fig. 2. Upon receiving service requests from clients, the optimizer gathers information about each processing device. Please note that we define the device as a physical server in this study. In general, the device can be any other computing device, such as a Raspberry Pi or Arduino. On each device, we install a container monitoring tool, cAdvisor, to collect data on CPU resource usage, memory, and bandwidth. Additionally, we use the IPMI exporter to collect data on the temperature around servers. The TensorFlow serving server's machine learning models will use the device status data obtained from cAdvisor and the IPMI exporter to predict the corresponding power consumption and response time.

We use the concept of Osmotic Computing and develop (1) to determine to which Pod client's tasks should be allocated while lowering increase of power consumption and response time. The Pod selection is based on the evaluation score; the smaller evaluation score, the higher the priority.

$$\text{Evaluation Score} = \alpha PC + \beta RT \quad (1)$$

where PC and RT are indexes and they indicate increase of power consumption and response time, and α and β are weights of each index (α+β=1). Applications running on edge computing platforms have different operating requirements. For example, for applications related to autonomous driving, low latency in

response time is critical. On the other hand, for non-real-time related applications (such as sensor Nodes that observe the weather for a long time), the response time is not as important and lowering power consumption can be prioritized. To meet the operating requirements of various applications, each index's weight must be adjusted according to the characteristics of the application itself. Thus, for osmotic pressure (π) at index X, the evaluation value of each device can be formulated, as in (2):

$$\pi = \alpha X \tag{2}$$

where α is a set weight of the index. In the case of a mixed solution, the osmotic pressure of the solution can be calculated by summing each solute's osmotic pressures. Therefore, when considering multiple indices, (3) can be used to represent the phenomena:

$$\pi = \alpha_1 X_1 + \alpha_2 X_2 \tag{3}$$

where X_1 and X_2 are indexes, and α_1 and α_2 are weights of each index. However, it is challenging to formulate X from the factor of the computational resources of real devices and behavior of applications. In this paper, we use power consumption and response time as two indices and create a prediction model for them. Based on our evaluation, we observe a correlation of -0.569 between increase of power consumption and response time. This inverse correlation implies that one variable increases as the other decreases and vice versa.

Please note that the cAdvisor API provides near real-time device status information, and there is several seconds delay of data transferring from IPMI exporter. Therefore, we set the IPMI exporter data collection as a background process with a fixed interval of one minute, and the optimizer (WAO-LB) will always refer to the latest data. Since the temperature around the device collected by the IPMI exporter does not change frequently, it does not affect the accuracy of power consumption prediction. WAO-LB uses both the PC model and the RT model to evaluate the priority of clients' tasks allocation to Nodes. TABLE I describes the detailed PC model, and TABLE II summarizes the implementation RT model.

TABLE II. DESCRIPTION OF NEURAL-NETWORK–BASED RESPONSE TIME PREDICTION MODEL

Input		CPU usage, memory info, network info, temperature around Node, date-time information
Neural network model	Layers	3 hidden layers with a size of 2,000 nodes per layer
	Optimizer	Adam (with hyperparameters of lr=0.0005, beta1=0.9, beta2=0.999, epsilon=1e-10, decay=0.00001)
	Loss function	MSE
	Batch size	256
Output		Respond time against user's request

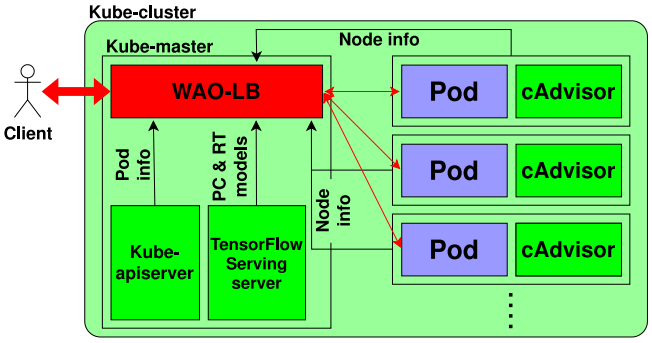

Fig. 2. Architecture of WAO-LB built in this study.

IV. EXPERIMENTAL RESULT

In this section, we evaluate our proposed WAO-scheduler and WAO-LB architectures in terms of the methodology discussed in the previous section. Our method provides the power consumption reduction solution for K8s by using optimal task allocation. To demonstrate the feasibility of our method, we use an object detection container orchestration service to test the level of power reduction. Our tests were conducted on a K8s cluster at a private testbed edge data center. We organized our experiment setting and evaluation as follows:

A. We first describe the experimental setting for our testbed edge data center, and it includes;

1) A description of the testbed edge data center
2) A choice of testing application
3) An observation of power consumption behavior in the testbed edge data center and
4) A finding of the optimal preset temperature of air conditioner for the power consumption reduction experiments.

B. Based on the above experimental setting, we provide a detail power-saving comparison between our proposed WAO-scheduler and K8s default scheduler (Kube-scheduler) and

C. Perform a power consumption comparison between proposed WAO-LB and K8s' metal load-balancer (MetalLB).

A. Experimental Setting

1) Environment of the testbed edge data center: Unlike simulated and virtual machine studies, this work is based on a real private testbed edge data center with about 200 servers located in Osaka, Japan. The servers are Fujitsu Primergy RX2530 M4 with two Intel Xeon Silver 4108 CPUs (16 cores total), 16 GB of memory, and a 1 TB HDD. As a client in this experiment, we used a desktop computer that was located roughly 10 km away from the data center.

2) Choice of the testing application: In this experiment, we prepared a service that performs object detection based on TensorFlow. There are many use cases of object detection in IoT, such as security cameras, self-driving cars, and smartphone applications (such as Google Lens). When this

container receives a compressed image from a client device, it annotates objects and organisms in the image, compresses it again, and sends it back to the client device.

3) Behavior of power consumption: Fig. 3. shows the dynamic of power consumption for CPU usage and temperature around the server. The power consumption value depends on both the CPU usage rate and the temperature around the server. We observe that the server's power consumption increased significantly when the CPU usage was between 10% and 30%, and then increased gradually after that.

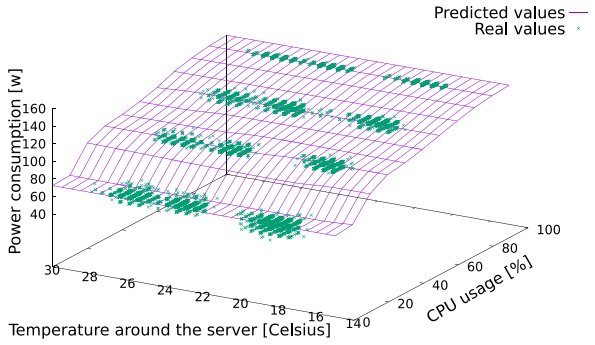

Fig. 3. Typical power consumption model for server.

4) Best preset temperature of air conditioner for power consumption reduction: Based on the power consumption behavior from Fig. 3, other than CPU usage, temperature around servers also has a significant influence on server power consumption. When the temperature is low, the fan will rotate at a low speed (lower power consumption). On the contrary, when the temperature is high, the fan needs to rotate at a higher speed and will incur greater power consumption.

When the preset temperature was 24°C, WAO-scheduler reduced more power consumption than other temperatures as Fig. 4 shows. This graph demonstrates the power consumption difference between the WAO-scheduler and K8s default Kube-scheduler at a various air conditioner preset temperature ranging from 20 to 27°C. The server fans, which are controlled by CPU usage and temperature around servers, started rotating at the temperature of 24°C. This resulted in the spike in the graph and indicates 24°C of preset temperature is the most suitable operating temperature for our testbed edge data center. Thus, we will fix the temperature parameter as 24°C for both the WAO-scheduler and WAO-LB for power consumption reduction.

B. Experimental Results of WAO based scheduler

In this experiment, we first deployed the Pods to Nodes using either the WAO-scheduler or the default Kube-scheduler. After the Pods are deployed, the request will be sent to K8s and observed the increase of power consumption for task processing. As mentioned earlier, we applied the optimal preset temperature from the air conditioner of 24°C that was obtained from our

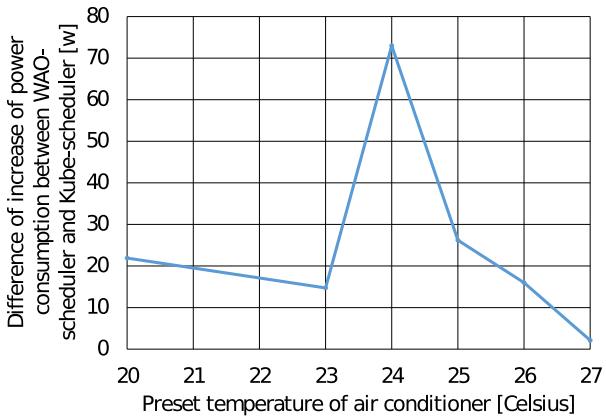

Fig. 4. Relationship between power consumption reduction and preset temperature of air conditioner.

previous experimental results. Both approaches used MetalLB for the task distribution. We will show another power-saving result of using WAO-LB in a later experiment. Within our expectation, the power consumption level is proportional to the CPU usage and number of Pods.

There are several observations that can be made from the results shown in Fig. 5. The horizontal axis is the sum of the CPU usage and the number of Pods of all servers, and the vertical axis is the total power consumption of those servers. First, power saving can be achieved in all cases with the same number of Pods. Additionally, when the number of Pods is 10 and 30, we observed that WAO-scheduler exhibited power consumption reduction compare to default Kube-scheduler at the total CPU usage ranging from 10 to 50%. However, as for 50 pods, power consumption converges at the total CPU usage of about 50%. This implies that when there are more available resources (such as idle CPU cores), WAO-scheduler will have more flexible resources to control for Pod allocation and consequently achieve a greater degree of energy saving. For a more detailed discussion, the result when using 10 Pods in Fig. 5 are shown in Fig. 6. We observe that power saving starts right when applying the WAO-scheduler. The degree of power-saving continues to increase until a maximum of 89 watts, which is about 8% less power consumption than the Kube-scheduler at the total CPU usage of 20%. In a typical data center, servers utilization is often 20-40% [14]. Therefore, the proposed WAO-scheduler can achieve the maximum power consumption reduction under optimal CPU usage when using 10 pods.

C. Experimental Results of WAO based load balancer

We use different weights for the PC model and RT model in WAO-LB to compare with MetalLB, as shown in Fig. 7. WAO-LB which gives the highest priority to power consumption (WAO-LB (PC:RT=10:0)) exhibited power consumption reduction compare to MetalLB when CPU usage is less than 50%. We can also observed WAO-LB which consider power consumption and response time equally (WAO-LB (PC:RT=5:5)) exhibited power consumption reduction when CPU usage is less than 40%. On the other hand, WAO-LB which gives the highest priority to response time (WAO-LB

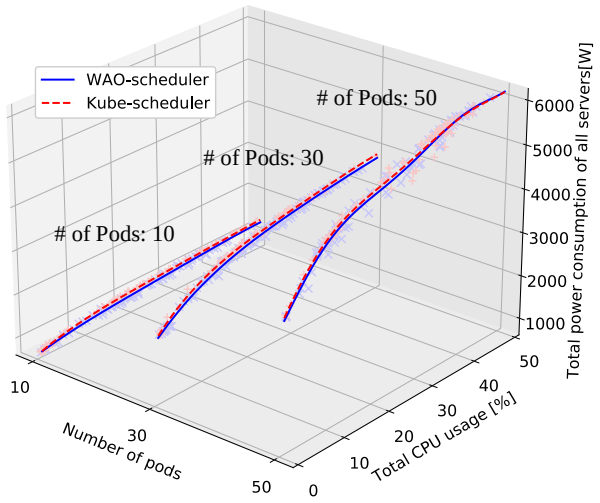

Fig. 5. Typical relationship among power consumption for WAO-scheduler and the default Kube-scheduler.

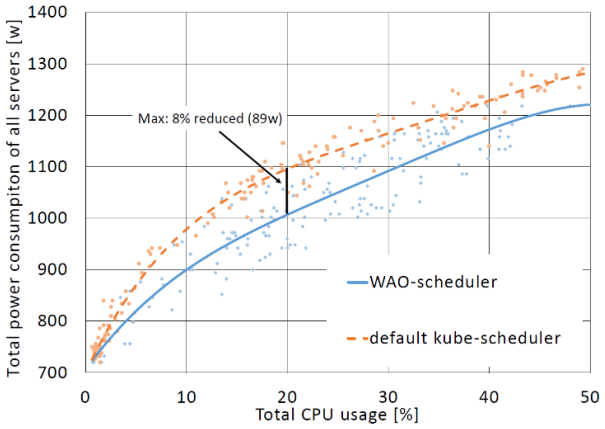

Fig. 6. Typical relationship among power consumption for WAO-scheduler and the default Kube-scheduler when using 10 pods.

(PC:RT=0:10)) consumed more power consumption than MetalLB. From these results, we can see that power consumption fluctuated in accordance with the different weight ratios. WAO-LB (PC:RT=10:0) achieved a maximum reduction of 470 watts, which is about 9.9% less power consumption than MetalLB at the total CPU usage of 20%. As mentioned earlier, in a typical data center, servers utilization is often 20-40%. Therefore, the WAO-LB also can demonstrate the maximum power consumption reduction under optimal CPU usage.

We also evaluated relationship between CPU usage and response time as shown in Fig. 8. The response time for each CPU usage rate (10%-50%) is shown as box plots. WAO-LB (PC:RT=0:10) and WAO-LB (PC:RT=5:5) achieved a lower average response time than the MetalLB in all cases. Additionally, in both WAO-LB (PC:RT=0:10) and WAO-LB (PC:RT=5:5), the response times almost never exceeded 40 seconds. Conversely, in WAO-LB (PC:RT=10:0), the average response time was longer than MetalLB. From these results, also in terms of response time, we can see that it's values fluctuated in accordance with the different weight ratios.

These results reflected the characteristics of servers. As mentioned earlier, there is an inverse correlation between the increase of power consumption and response time. This is due to the difference of distribution of tasks. Power consumption increases significantly when the CPU usage is low, and it increases gently when the CPU usage is 30% or more, as shown in Fig. 3. Therefore, considering power consumption, tasks are concentrated on some servers. As a result, response time increases because of the concentration of tasks. On the other hand, considering response time, tasks are allocated to servers with sufficient resources. Accordingly, servers with low CPU usage is selected, and overall power consumption increases. Most importantly, the flexible control of our proposed WAO-LB enables it to be used with applications with different response time requirements.

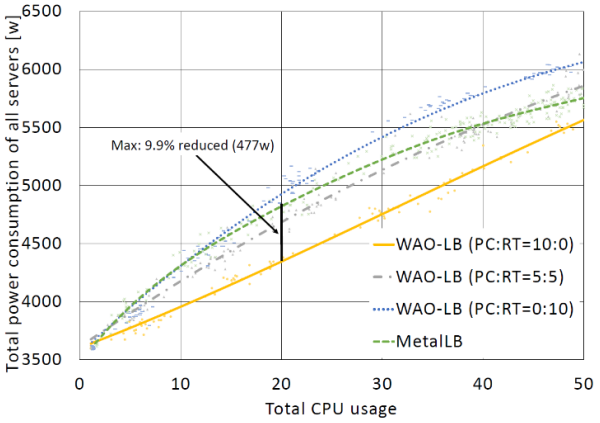

Fig. 7. Typical relationship between power consumption for WAO-LB and MetalLB.

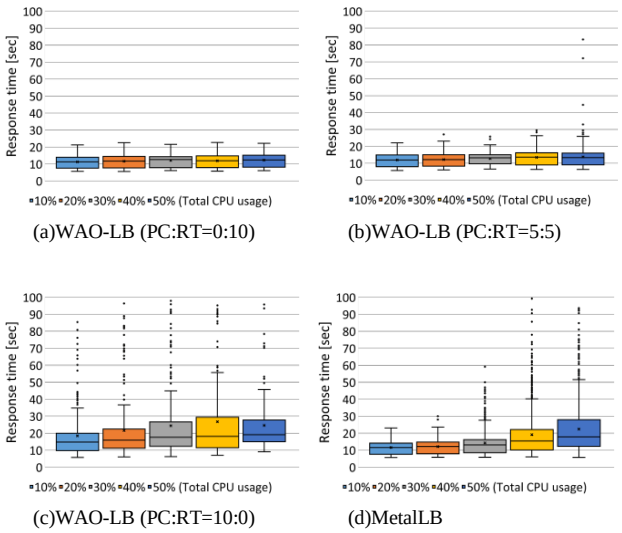

Fig. 8. Typical relationship between response time for WAO-LB and MetalLB.

V. CONCLUSION

In this paper, we proposed both WAO-scheduler and WAO-LB architecture, which optimize Pod allocation and task allocation, respectively, as an AI-based power consumption reduction function for K8s. By scoring each Node in Kube-scheduler through a neural network-based PC model, the WAO-scheduler can choose Pods with lower power consumption. Our experimental results show that the WAO-scheduler demonstrates around 8% power consumption reduction compared to conventional K8s default Kube-scheduler, and WAO-LB exhibit 9.9% power consumption reduction compared to conventional MetalLB. WAO-LB defines an evaluation formula also as an Osmotic Computing and allocates tasks using the predictions of the PC and RT models built for an actual computing system. WAO-LB provides a flexible mechanism that allows K8s to appropriately balance between reducing the power consumption reduction and lowering the response time. These results indicate that the WAO developed in this study exhibits promising potential for task allocation modulus as a micro service platform.

ACKNOWLEDGMENT

The edge data center is located in the Konohana Building, Osaka, Japan, of Nippon Telegraph and Telephone West (NTT West) Corporation. We would like to thank NTT West Corporation for their support.

REFERENCES

[1] P. G. L´opez, A. Montresor, D. H. J. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. P. Barcellos, P. Felber, and E. Rivi`ere, "Edge-centric computing: Vision and challenges," Computer Communication Review, vol. 45, pp. 37–42, Sep. 2015.

[2] D. Bernstein, "Containers and cloud: From LXC to docker to kubernetes," IEEE Cloud Computing, vol. 1, pp. 81–84, Sep. 2014.

[3] M. Gamal, R. Rizk, H. Mahdi, and B. E. Elnaghi, "Osmotic bio-inspired load balancing algorithm in cloud computing," IEEE Access, vol. 7, pp. 42 735–42 744, Apr. 2019.

[4] M. Villari, M. Fazio, S. Dustdar, O. Rana, and R. Ranjan, "Osmotic computing: A new paradigm for edge/cloud integration," IEEE Cloud Computing, vol. 3, pp. 76–83, Dec. 2016.

[5] M. Nardelli, S. Nastic, S. Dustdar, M. Villari, and R. Ranjan, "Osmotic flow: Osmotic computing + iot workflow," IEEE Cloud Computing, vol. 4, pp. 68–75, Mar. 2017.

[6] Y. Hsu, H. Kuwahara, K. Matsuda, and M. Matsuoka, "Toward a workload allocation optimizer for power saving in data centers," in Proc. IEEE International Conference on Cloud Engineering '19, Jun. 2019, pp. 56–66.

[7] H. Kuwahara, Y. Hsu, K. Matsuda, and M. Matsuoka, "Real-time workload allocation optimizer for computing systems by using deep learning," in Proc. IEEE 12th International Conference on Cloud Computing '19, Jul. 2019, pp. 190–192.

[8] C. Chang, S. Yang, E. Yeh, P. Lin, and J. Jeng, "A kubernetes-based monitoring platform for dynamic cloud resource provisioning," in Proc. IEEE Global Communications Conference '17, Dec. 2017, pp. 1–6.

[9] P. Townend, S. Clement, D. Burdett, R. Yang, J. Shaw, B. Slater, and J. Xu, "Invited paper: Improving data center efficiency through holistic scheduling in kubernetes," in Proc. IEEE International Conference on Service-Oriented System Engineering '19, Apr. 2019, pp. 156–15 610.

[10] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, Nitin, and R. Rastogi, "Load balancing of nodes in cloud using ant colony optimization," in Proc. UKSim 14th International Conference on Computer Modelling and Simulation '12, Mar. 2012, pp. 3–8.

[11] V. Sharma, I. You, R. Kumar, and P. Kim, "Computational offloading for efficient trust management in pervasive online social networks using osmotic computing," IEEE Access, vol. 5, pp. 5084–5103, Mar. 2017.

[12] V. Sharma, K. Srinivasan, D. N. K. Jayakody, O. F. Rana, and R. Kumar, "Managing service-heterogeneity using osmotic computing," ArXiv, vol. abs/1704.04213, Apr. 2017.

[13] K. Kaur, S. Garg, G. Kaddoum, S. H. Ahmed, and D. N. K. Jayakody, "En-osco: Energy-aware osmotic computing framework using hyperheuristics," in Proc. the ACM MobiHoc Workshop on Pervasive Systems in the IoT Era '19, Jul. 2019, pp. 19–24.

[14] L. A. Barroso and U. Hölzle, "The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines", 2009.