APPLIED MACHINE LEARNING IN LOAD BALANCING

Junaidi[1)], Prasetyo Wibowo[2)], Dini Yuniasri[3)], Putri Damayanti[4)], Ary Mazharuddin Shiddiqi[5)], and Baskoro Adi Pratomo[6)]
[1, 2, 3, 4, 5)] Department of Informatics, Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60117
[6)] School of Computer Science and Informatics, Cardiff University
Cardiff, UK CF10 3AT
e-mail: neutrofoton@gmail.com[1)], pras.wibowo96@gmail.com[2)], dini.yuniasri@gmail.com[3)], putridamay200895@gmail.com[4)], ary.shiddiqi@if.its.ac.id[5)], pratomoba@cardiff.ac.id[6)]

ABSTRACT

A common way to maintain the quality of service on systems that are growing rapidly is by increasing server specifications or by adding servers. The utility of servers can be balanced with the presence of a load balancer to manage server loads. In this paper, we propose a machine learning algorithm that utilizes server resources CPU and memory to forecast the future of resources server loads. We identify the timespan of forecasting should be long enough to avoid dispatcher's lack of information server distribution at runtime. Additionally, server profile pulling, forecasting server resources, and dispatching should be asynchronous with the request listener of the load balancer to minimize response delay. For production use, we recommend that the load balancer should have friendly user interface to make it easier to be configured, such as adding resources of servers as parameter criteria. We also recommended from beginning to start to save the log data server resources because the more data to process, the more accurate prediction of server load will be.

***Keywords**: Load balancing, machine learning, server load.*

PENERAPAN PEMBELAJARAN MESIN DALAM *LOAD BALANCING*

Junaidi[1)], Prasetyo Wibowo[2)], Dini Yuniasri[3)], Putri Damayanti[4)], Ary Mazharuddin Shiddiqi[5)], dan Baskoro Adi Pratomo[6)]
[1, 2, 3, 4, 5)] Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60117
[6)] School of Computer Science and Informatics, Cardiff University
Cardiff, UK CF10 3AT
e-mail: neutrofoton@gmail.com[1)], pras.wibowo96@gmail.com[2)], dini.yuniasri@gmail.com[3)], putridamay200895@gmail.com[4)], ary.shiddiqi@if.its.ac.id[5)], pratomoba@cardiff.ac.id[6)]

ABSTRAK

Cara umum untuk menjaga kualitas layanan pada sistem yang berkembang pesat adalah dengan meningkatkan spesifikasi server atau dengan menambahkan server. Penambahan jumlah server idealnya harus diimbangi dengan ketersediaan load balancer untuk menyeimbangkan beban server. Dalam makalah ini kami mengusulkan algoritma machine learning yang memanfaatkan sumber daya server CPU dan memori untuk memperkirakan dan menentukan beban tiap sever di waktu mendatang. Kami mengidentifikasi rentang waktu prediksi harus cukup lama untuk meningkatkan akurasi distribusi beban sever pada saat runtime. Selain itu, pengumpulan informasi profil resource server, prediksi beban server, dan pendistribusian beban seharusnya dilakukan secara asynchronous dengan komponent penerima request dari load balancer guna meminimalkan keterlambatan respons. Untuk penggunaan dalam produksi, kami merekomendasikan bahwa load balancer harus memiliki antarmuka yang dapat mempermudah proses konfigurasi, seperti penambahan sumber daya server sebagai kriteria parameter beban server. Kita juga merekomendasikan pada saat awal untuk memulai menyimpan data log sumber daya server karena semakin banyak data yang di proses, maka semakin akurat prediksi yang akan di hasilkan.

***Kata kunci**: Beban server, load balancing, machine learning.*

I. INTRODUCTION

One of the technologies that emerges recently is the cloud computing. Cloud computing is used to store data in the cloud. A cloud in this context is known as the internet [1], [2]. A server can be placed anywhere on the internet and can be accessed from anywhere [3]. A user accesses data by using an application and data are retrieved from the server through the application [4]. In other words, users can access data everywhere without knowing where and which server that data is obtained from.

Cloud computing is a quite reliable technology but still has some problems with its application. An overloaded server due to excessive access traffic can cause downtime [5]. The usage of a load balancer in cloud computing

 DOI: http://dx.doi.org/10.12962/j24068535.v18i2.a940

server farm is to divide the traffic and avoid downtime [6]. A load balancer calculates servers' and decides which server to serve a recent request based on the suitability and availability of the server. A server with high suitability and low load will be the strongest candidate to be selected.

Load balancing refers to efficiently distributing incoming network traffic across a group of backend servers [7], also known as a server farm or server pool. Modern high traffic websites must serve hundreds of thousands, if not millions [5], [6], of concurrent requests from users or clients and return the correct text, images, video, or application data, all in a fast and reliable manner. Modern computing generally requires adding more servers. A load balancer acts as the "traffic cop" sitting in front of your servers and routing client requests across all servers [8]. The load balancer maximizes the speed and the capacity utilization and ensures that no server is overworked, which could degrade its performance. Higher accuracy of server load forecasting results is critical for a load balancer to spread out the workload to the server.

To improve the server load forecasting accuracy, researchers have proposed many approaches, for example PPLB algorithms [9]. The algorithms classify similar requests into one class, based on their CPU demands. The members of PPLB family of algorithms use utility function, RBFNN and ANFIS for correction procedure that results to PPLB-U, PPLB-RBF and PPLB-ANFIS algorithms, respectively. A load balancer (called IQRD) was applied to web servers [10]. In this paper, we propose a new load balancer method by implementing a machine learning algorithm. We use the machine learning algorithm to predict the condition of servers. By acquiring sufficient information about the condition of the servers, the load balancer can optimally balance the server loads.

II. Literature Review

There are several studies to optimize the use of load balancing. Ahmad et.al [11] developed a distributed load balancing system using a demultiplexer (and/or multiplexers) network. This allows the system to carry out large calculations that leads to gradual growth of data center capacity. Another study conducted by Golchi et al. [12] that implemented cloud computing in load balancing systems. The study implemented a hybrid of firefly and IPSO algorithms on a load balancer and compared the algorithms with four scheduling methods, i.e. Round Robin (RR), First Come First Service (FCFS), Short jobs First (SJF), and Genetic Algorithms (GA). The algorithm they proposed showed better performance than similar methods and flexible behavior in minimizing average load through a multi-purpose optimization. While Rathore et al. [13] proposed a hybrid load balancing for the Grid and compared it with existing methods, I.e. Least Clients, Round Robin, Least Load, and Fastest First. One of the the algorithms was redeveloped to improve its performance. The algorithms were compared to determine their load balancing efficiency for selecting servers from a pool of servers. Each algorithm has its own advantages and disadvantages, so there is no best algorithm.

While in the development of machine learning, Shafiq et al. [14] compared four machine learning methods, namely, Support Vector Machine, C4.5 decision tree, Naive Bays and Bayes Net, to classify network traffic. The research indicated that the C4.5 classifier method gives the highest accuracy compared to the other machine learning classifiers. Research on forecasting by implementing machine learning was conducted by Chen et al. [15]. The research proposed a new EMD-Mixed-ELM short-term electrical load forecasting method based on empirical mode decomposition (EMD) and extreme learning machine (ELM). The study compared EMD-Mixed-ELM with several other methods, such as RBF-ELM, UKF-ELM, and Mixed-ELM, MFES, ESPLSSVM and some combined methods. The results of the research indicated that the EMD-Mixed-ELM are proved to be better than all the others. Research that implemented machine learning on load balancers has been conducted in [16] and [17]. A separated algorithm to be used in machine learning [16] by implementing several machine learning algorithms to create a more efficient scheduling mechanisms in load balancing. A scheduler capable of assigning jobs to CPUs and GPUs in a load-balanced manner was developed in [17] by considering account job processing requirements, device suitability, and predicted performance on a particular computing device. In addition, researchers also utilized machine learning in load balancing to decide on a job's suitability for a particular multi-core processor. Machine learning on load balancing is trained using past experience and then machine learning models are trained to show adaptive behaviors (appropriate mapping) for different computing platforms and applications. Next, the research compared their scheduler systems with schedulers based only on CPUs or GPUs. The result of this research indicated that the proposed method has better performance than the scheduler which is only based on CPUs or GPUs.

III. Basic Theory / Background

Load balancing is a technique to distribute traffic load from two or more connections to optimize resource consumption. Besides that, load balancing can also avoid overloading processes from a connection. An effective load balancing technique ensures that every machine works with a similar amount of process. Therefore, a load

balancer needs to maximize the throughput of traffic load so that the time required for a process will be minimized [18]. Six parameters are used to evaluate the load balancing method:

- throughput: the amount of process to be executed,
- response time: time needed to load a process,
- fault-tolerance: the ability of a method to tolerate faults,
- resource utilization: the ability to use on various resources,
- scalability: flexibility to increase or decrease the number of nodes,
- performance: the measurement of all working parameters by considering the accuracy,

Figure 1 shows the architecture of load balancing. Load balancing techniques are divided into two algorithms: static algorithms and dynamic algorithms. The static algorithm is used for a stable environment; thus, this algorithm is not flexible to environment changes [19]. Static algorithms divide network traffic equally for every server. This division is based on the resource system and processor performance. Trafic allocation does not need current status, so the division becomes much fair to every server [18]. Dynamic algorithm work based on the current status of a server; it checks a server with low utilization and allocates the traffic to the server. The dynamic algorithm always checks every server before allocating traffic, so this algorithm is more flexible to the changes of the environment than the static algorithm. The advantage of the dynamic algorithm is that it can help improves system performance by considering the changes in the environment [18].

Machine learning is used to uncover the unknown pattern of training datasets [20]. Machine learning has four types of learning:

- Supervised Learning
  Supervised learning is a learning process where a machine learning algorithm creates a model from training data that are already labeled with classes.
- Semi-supervised Learning
  Semi-supervised learning is a learning process where a machine learning algorithm creates a model from training data that has a small number of labeled data, while the rest of the data are unlabeled. The machine learning algorithm learns from labeled and unlabeled training data.
- Unsupervised Learning
  Unsupervised learning is a learning process where a machine learning algorithm creates a model from training data from unlabeled data.
- Reinforcement Learning
  Reinforcement learning is a learning process where a machine learning algorithm creates a model from training data that consider the environment when creating the model. External factors and environment are considered as additional knowledge in the learning process.

Every model created by the machine learning algorithm will be used to classify unknown testing data.

IV. PROBLEM SPECIFICATION

In this study, balancing server load becomes challenging. This is due to the efficiency of managing workload management becomes more complicated, along with the unpatterned increasing number of user requests. To overcome this problem, we need an efficient way to optimize system resources and forecast next system resources.

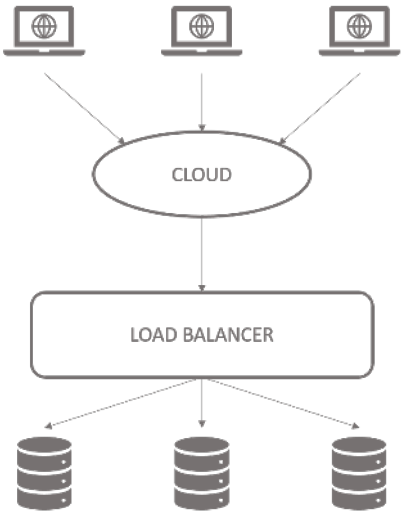


Fig. 1. Load balancing architecture.

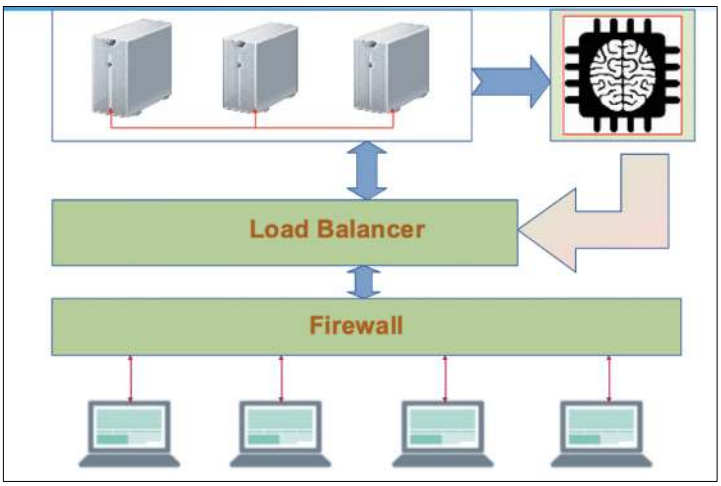


Fig. 2. Basic load balancing architecture with machine learning.

- In this paper, the focus of the problem is to identify the workload of each virtual machine by using load balancing techniques as the main limitation set and making predictions based on system resources. The limits are classified as follows: Lots of literature [21] focuses on the CPU utilization ratio or RAM utilization ratio but does not use both..
- In making predictions, the Log File becomes very important because it contains information that we need to predict. there is some literature like [22] that uses log files to predict and we use them to predict system resources.
- In this research, we will try to predict the log file system resources usage we gets optimal results.

There are several approaches that have been taken in clustering log files, but there are still some limitations. In [21], the authors propose an architecture using K-means as the main method of grouping VMs. The main attribute that is considered is RAM usage on the machine. In [16], load balancing is performed based on current CPU and RAM usage. A study in [22] used log files as a reference in grouping VMs based on the CPU and RAM utilization and as well as user job classifications (based on workload). This approach makes the workload more stable and opens up opportunities in elaborating machine learning to load balancing.

V. DESIGN AND ARCHITECTURE

We propose a machine-learning algorithm to perform load balancing on a server cluster. The machine learning is used to predict a server load based on the historical server load profiles such as CPU and memory usage. These aspects are chosen as they possess dominant roles in processing tasks.

Figure 2 shows the basic architecture of load balancing to receive client requests and dispatch them to a server that currently has the lowest load. The load balancer must have a rapid response to decide which server a client request should be assigned to while receiving other client requests. If the load balancer requires more time to decide, it costs a significant contribution delay to the request-response time. Thus, leading to poor quality of performance.

A load balancer should be able to predict future server load to avoid a significant delay. So, the load balancer does not have to load resource information (such as CPU, memory) of each server whenever the load balancer receives client requests. To do so, there should be a stateful or a daemon service that asynchronously pulls server load and predicts future resource information using a machine learning algorithm. The architecture of the load balancing algorithm is shown in Figure 3. Based this figure, at every cycle, prediction results are stored in the memory to be accessed by load balancing dispatcher submodule. This way, the dispatcher does not have to re-predict the resources.

Ideally, the load balancer should have an administration module that can be accessed by an administrator to make a change to the configuration of the load balancer. It will be useful to configure any dynamic parameters of computation needed by load balancing, such as CPU or memory usage. The other parameters that can be dynamically changed is forecasting timespan, i.e. how long timespan ahead of the server should be predicted and store in memory of load balancer.

VI. EXPERIMENT AND ANALYTICS

We used the dataset from http://gwa.ewi.tudelft.nl/datasets/gwa-t-13-materna, a raw data simulation of servers profile pulled periodically and asynchronously from a set of servers (347, 432, 511) by stateful service (Figure 2). The servers were chosen because the data in those servers has more reliable data, where the range of the data on

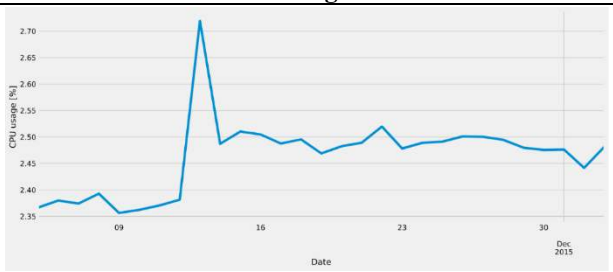
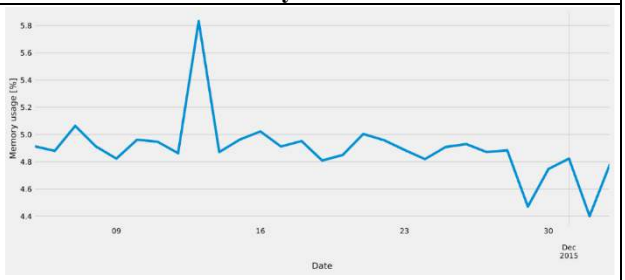
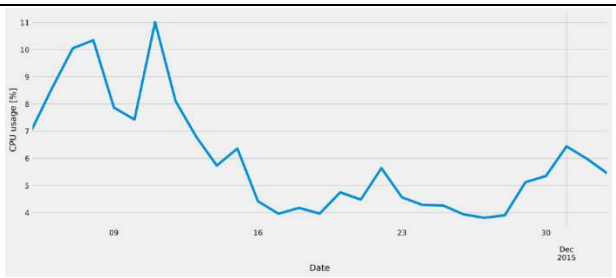
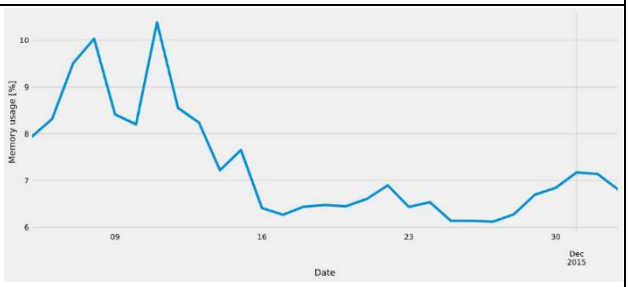
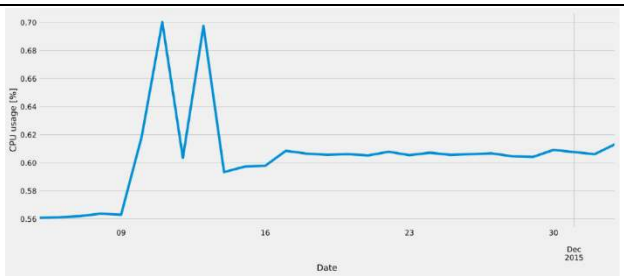
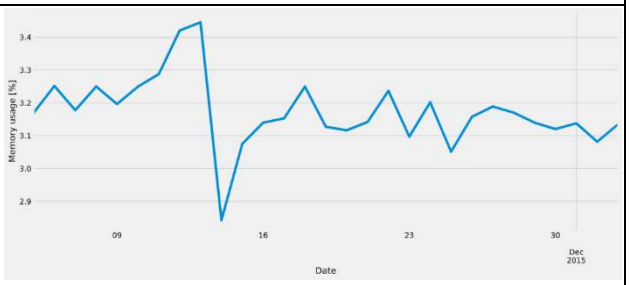
these servers is more stagnant than other. Data specifications of the servers are shown in Table I. The specifications (CPU and memory) of the servers are listed in Table II.

Table II shows that the CPU and memory usage of the servers are different depending on the tasks processed by the servers. The profile information dataset can be used to predict profiles of each server in the future with specified timespan. In this experiment, we use SARIMA (Seasonal Autoregressive Integrated Moving Average). SARIMA is a development of ARIMA (Autoregressive Integrated Moving Average) which has a seasonal time pattern data. In general, SARIMA is denoted by (p, d, q) (P, D, Q)s. The p, d, and q notations sequentially state the order of auto-regression, integration (differencing), and moving average. Whereas the s notation is denoted as the seasonal periods of the data.

TABLE I
SET OF SERVER SPECIFICATION.

	Server-347	**Server-432**	**Server-511**
minCPU	29 MHz	20 MHz	24 MHz
maxCPU	4608 MHz	3604 MHz	1144 MHz
avgCPU	265 MHz	86 MHz	41 MHz
Provisioned CPU Cores	**4 GHz**	**4 GHz**	**4 GHz**
minRAM	11 MB	81 MB	5 MB
maxRAM	7847 MB	7426 MB	7479 MB
avgRAM	507 MB	596 MB	136 MB
Provisioned RAM	**8 GB**	**8 GB**	**8 GB**

TABLE II
CPU AND MEMORY USAGE PROFILES OF SERVERS.

Server	**CPU Usage %**	**Memory Profile %**
347		
432		
511		

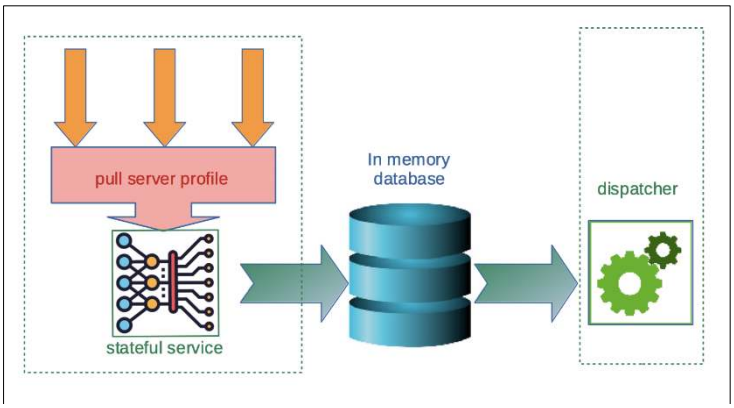


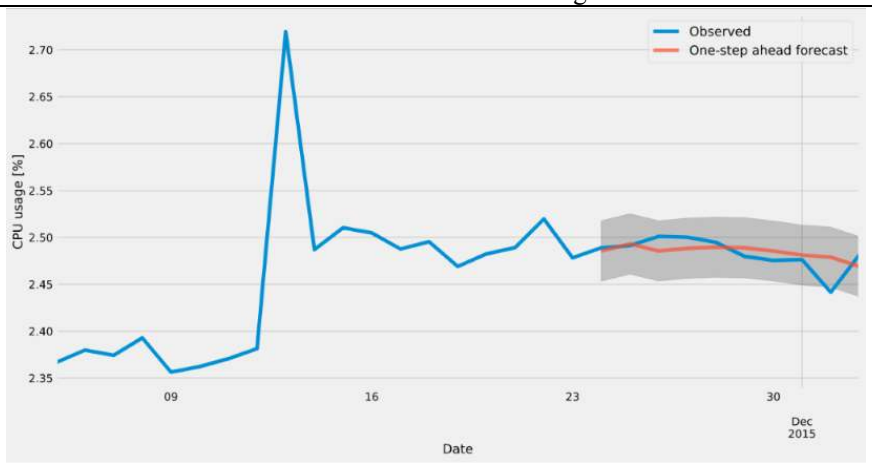
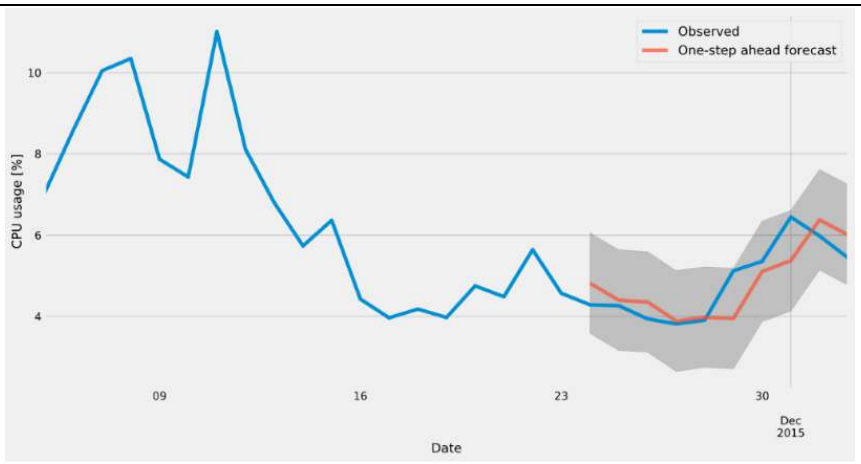
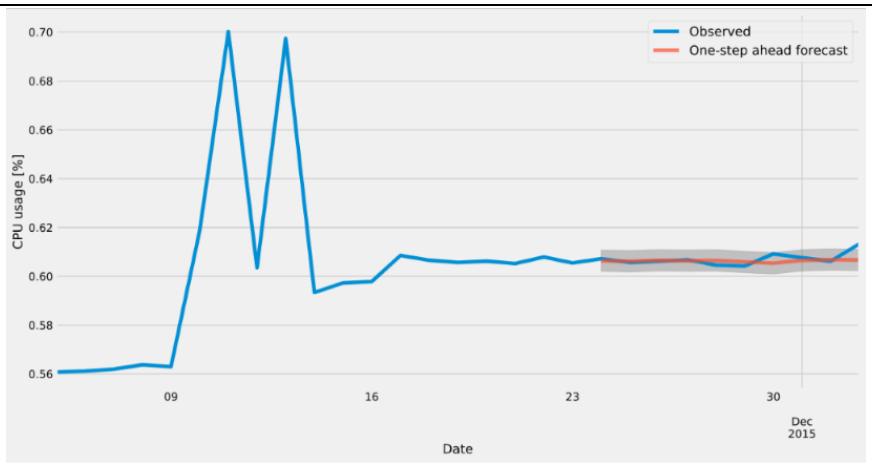
Fig. 3. Components of load balancing service state in memory database and dispatcher.

```
Statespace Model Results
==========================================================================================
Dep. Variable:                      CPU usage [%]   No. Observations:                   29
Model:             SARIMAX(1, 0, 1)x(1, 0, 0, 12)   Log Likelihood                42.672
Date:                            Wed, 04 Dec 2019   AIC                           -77.344
Time:                                    01:16:39   BIC                           -74.254
Sample:                                11-05-2015   HQIC                          -77.186
                                     - 12-03-2015
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.9994      0.001   1817.144      0.000       0.998       1.000
ma.L1         -0.7660      0.362     -2.114      0.034      -1.476      -0.056
ar.S.L12       0.0242      0.269      0.090      0.928      -0.503       0.552
sigma2         0.0003   8.46e-05      3.160      0.002       0.000       0.000
===================================================================================
Ljung-Box (Q):                      nan   Jarque-Bera (JB):                 0.53
Prob(Q):                            nan   Prob(JB):                         0.77
Heteroskedasticity (H):            0.89   Skew:                            -0.08
Prob(H) (two-sided):               0.90   Kurtosis:                         3.87
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).


The Mean Absolute Error of our forecasts is 0.0111467697
The Mean Squared Error of our forecasts is 0.0002181081
The Root Mean Squared Error of our forecasts is 0.0147684834
```

Fig. 4. Result of SARIMA methods in server 347.

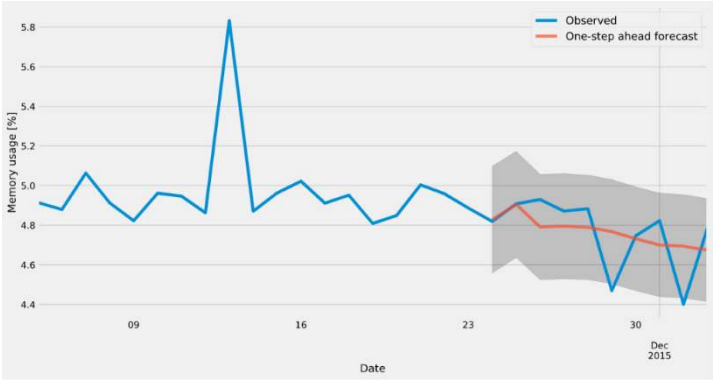
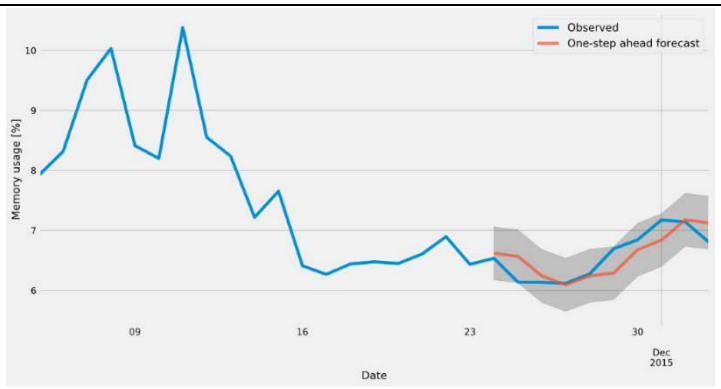
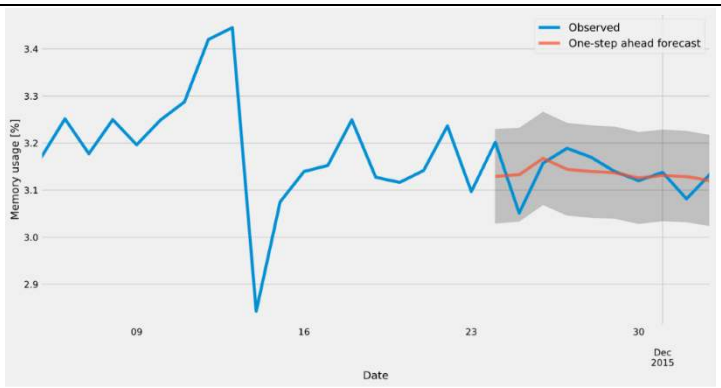
Figure 4 shows the 347 server prediction results using the values p, d, q (1, 0, 1) and the seasonal values P, D, Q, m (1, 0, 0, 12). Based on the results of the statespace model, it shows how robust the modeling used is by looking at the log-likelihood value, which shows the value of 42,672, which means the number of matches between the data in the model, so the higher the value obtained, the better. Moreover, we can see through the value of Akaike's Information Criterion (AIC), Bayesian Information Criterion (BIC), and Hannan-Quinn Information Criterion (HQIC), which show the same value by seeing the smaller the value obtained. For model evaluation, we can see through the value of Mean Absolute Error (MAE) 0.01114, Mean Squared Error (MSE) 0.0002, and Root Mean Squared Error (RMSE) 0.0147, which it shows that the value is close to 0 which means that the modeling results obtained are close to the best fit value. Whereas the forecasting results and the actual data based on the CPU is shown in Table III. The forecaseting result and the actual data in memory server is shown in Table IV.

TABLE III
CPU ACTUAL AND FORECASTING.

Server	CPU Actual and Forecasting
347	
432	
511	

Forecasting activity is key to choose the server a task should be assigned to accurately. The forecasting does not require a server profile for each incoming task to be dispatched. This mechanism simplifies the procedure and reduces delays (both form the server or the dispatcher) that they might cause significantly. The other concern regarding this research is the timespan of forecasting. It will be good if machine learning can predict long enough timespan to anticipate a high volume of transaction requests from clients. This is important for the dispatcher to anticipate memory allocations regarding the expected profile of each server each time they need to dispatch tasks. Finally, we can know the priority of servers to receive dispatches tasks using the prediction profiles we have.

TABLE IV
MEMORY ACTUAL AND FORECASTING.

Server	Memory actual and forecasting
347	
432	
511	

VII. CONCLUSION

We proposed a method to predict the resources of servers based on the profile of the servers. The server profiles are used as input parameters of a load balancer to predict future server loads. The more parameters used, the more accurate the prediction to the actual condition. The timespan of profile collection should be long enough to maximize the prediction accuracy during server runtime. We suggest that the procedures for server profile pulling, data forecasting, and dispatching should be performed in asynchronous mode to the request listener of the load balancer to minimize response delay. Also, each of the procedures should be created in loosely coupled modules.

REFERENCES

[1] A. Armbrust, "Above the clouds: A Berkeley view of cloud computing," *Univ. California, Berkeley, Tech. Rep. UCB*, pp. 07–013, 2009.
[2] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities," in *Proc. IEEE International Conference on High Performance Computing and Communications,* 2008, pp. 5–13.
[3] A. Darwish, A. E. Hassanien, M. Elhoseny, A. K. Sangaiah, and K. Muhammad, "The impact of the hybrid platform of internet of things and cloud computing on healthcare systems: opportunities, challenges, and open problems," *J. Ambient Intell. Humaniz. Comput.*, vol. 10, no. 10, pp. 4151–4166, 2019.
[4] W. Zhu, C. Luo, J. Wang, and S. Li, "Multimedia cloud computing," *IEEE Signal Process. Mag.*, vol. 28, no. 3, pp. 59–69, 2011.
[5] K. Ramana and M. Ponnavaikko, "AWSQ: An approximated web server queuing algorithm for heterogeneous web server cluster," *Int. J. Electr. Comput. Eng.*, vol. 9, no. 3, pp. 2083–2093, 2019.
[6] K. Ramana, "NDLB: Nearest Dispatcher Load Balancing approach for Web Server Cluster," *Helix*, vol. 8, no. 1, pp. 3023–3030, 2017.

[7] P. Patel *et al.*, "Ananta: Cloud scale load balancing," in *Proc. ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 2013, pp. 207–218.
[8] D. E. Eisenbud *et al.*, "Maglev: A Fast and Reliable Software Network Load Balancer Daniel," in *Proc. USENIX Symposium on Networked Systems Design and Implementation*, 2016, pp. 523–535.
[9] S. Sharifian, S. A. Motamedi, and M. K. Akbari, "A predictive and probabilistic load-balancing algorithm for cluster-based web servers," *Appl. Soft Comput. J.*, vol. 11, no. 1, pp. 970–981, 2011.
[10] S. Sharifian, S. A. Motamedi, and M. K. Akbari, "A content-based load balancing algorithm with admission control for cluster web servers," *Futur. Gener. Comput. Syst.*, vol. 24, no. 8, pp. 775–787, 2008.
[11] A. Najam *et al.*, U.S. Patent Application No 12/189,438, 2010.
[12] M. M. Golchi, S. Saraeian, and M. Heydari, "A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments: Performance evaluation," *Comput. Networks*, vol. 162, p. 106860, 2019.
[13] N. K. Rathore, U. Rawat, and S. C. Kulhari, "Efficient Hybrid Load Balancing Algorithm," *Natl. Acad. Sci. Lett.*, 2019.
[14] M. Shafiq, X. Yu, A. A. Laghari, L. Yao, N. K. Karn, and F. Abdessamia, "Network Traffic Classification techniques and comparative analysis using Machine Learning algorithms," in *Proc. IEEE Int. Conf. Comput. Commun.*, pp. 2451–2455, 2017.
[15] Y. Chen, M. Kloft, Y. Yang, C. Li, and L. Li, "Mixed kernel based extreme learning machine for electric load forecasting," *Neurocomputing*, vol. 312, pp. 90–106, 2018.
[16] B. Panchal and S. Parida, "An Efficient Dynamic Load Balancing Algorithm Using Machine Learning Technique in Cloud Environment," *International Journal of Scientific Research in Science, Engineering and Technology*, vol. 4, no. 4, pp. 1184–1186, 2018.
[17] Y. N. Khalid, M. Aleem, U. Ahmed, M. A. Islam, and M. A. Iqbal, "Troodon: A machine-learning based load-balancing application scheduler for CPU–GPU system," *J. Parallel Distrib. Comput.*, vol. 132, pp. 79–94, 2019.
[18] S. Sankara Narayanan and M. Ramakrishnan, "A Comprehensive Study on Load Balancing Algorithms in Cloud Computing Environments," *Res. J. Appl. Sci. Eng. Technol.*, vol. 13, no. 10, pp. 794–799, 2016.
[19] R. S. Sajjan, "Load Balancing and its Algorithms in Cloud Computing : A Survey," *International Journal of Computer Sciences and Engineering*, January, 2017.
[20] R. Boutaba *et al.*, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *J. Internet Serv. Appl.*, vol. 9, no. 1, 2018.
[21] V. Chavan and P. R. Kaveri, "Clustered virtual machines for higher availability of resources with improved scalability in cloud computing," in *Proc. International Conference on Networks and Soft Computing,* 2014, pp. 221–225.
[22] M. Elrotub and A. Gherbi, "Virtual Machine Classification-based Approach to Enhanced Workload Balancing for Cloud Computing Applications," *Procedia Comput. Sci.*, vol. 130, pp. 683–688, 2018.