Energy Saving and Maximize Utilization Cloud Resources Allocation via Online Multi-Dimensional Vector Bin Packing

Liang Guo*, Pu Du*, Abdul Razaque*, Muder Almiani+, and Amer Al Rahayfeh+
*Department of Computer Science New York Institute of Technology, Nanjing, Campus,
Email: {China lguo06,Pdu01,arazaque}@nyit.edu
+Computer Information Systems, College of Information Technology Al-Hussein Bin Talal University, Ma'an, Jordan
Email: {malmiani, aalrahay}@my.bridgeport.edu

***Abstract*—In data center, which use virtualization technology and provide virtual machines (VMs) for their customers, they often receive the VMs renting request with determined multiple resource demanded which tends to arrive at an arbitrary time. Resource allocation strategies become one of considerable problem as they can affect the energy efficiency and resource utilization significantly. In this paper, we explore the resource allocation strategies which can minimize the number of used servers, in order to save the energy and maximize the resources utilization. We model this problem as the Vector Bin Packing Problem (VBPP) which is a variant of classic bin packing, proposed Multi-Dimensional Cloud Resource Dynamic Allocation Model(MDCRA) based on VBPP. Since most of existed algorithms aim to solve offline VBPP. We generalized those algorithms to online and proposed Single Weight and Double Weight family online algorithms based on the offline work. Through both rigorous theoretical analyses and extensive simulations, we demonstrate that the proposed allocation strategies achieve energy saving and utilization maximization.**

I. INTRODUCTION

Recent years, cloud computing becomes a significant topic. The clear advantages such as lower cost, rapid deployment, attract more and more users to build enterprise or individual application service based on cloud server rather than maintain servers themselves [1]. Many companies have built data center and provided cloud computing service, Microsoft, Alibaba, Google and Amazon etc. Generally, those data centers tend to use virtual machine technology to divide one server into many virtual machines and let many users share a same server and bandwidth.

Most cloud service provider use IaaS (Infrastructure-as-a-Service) [2] as the basic cloud-service model, in which they offer server or virtual machines as a service to their customers. Many VMs are allocated on a single server where dedicated servers were required before. At the same time, the request of renting VMs arrival dynamically with multiple resource demanded, also some of VMs existed in the servers will leave at an arbitrary time.

The data centers often face the problem of dispatching a stream of VMs renting request for many cloud servers with free resource. Each requested VM has determined multiple resource (e.g. CPU and RAM) demanded. The resources of cloud server is fixed and the arrival time of each request is unknown.

Resource allocation strategies are crucial to data center. Because, different allocation strategies mean different resource utilization and energy consumption, which closely associated with the cost and profit of data center. In particular for a data center with thousands of servers, a little growth of utilization ratio will bring huge profit. Previous research has shown that energy consumption affects the cost in data centers significant [3]. It is no double that turn a total idle server to sleep can save huge energy, and that a server can be turned to sleep or waked up in several seconds [4], [5].

A lot of research contributions [1], [6-10] have focused on developing resource allocation strategies to achieve energy saving or utilization maximization, using bin packing technology. However most of them suppose the request are static, while in fact, the data center receive the request continuously. Otherwise, the major of them consider each resource demand of request are single-dimension.

In this paper, we mainly explore resource allocation strategies. We defined the whole problem as Multi-Dimensional Cloud Resource Dynamic Allocation Model (MDCRA). In MD- CRA, data center receive jobs or VMs renting request dynamically, in other word, the request can arrive at arbitrary time in the future rather than same time. Observe that each server possesses different type of resource, such as CPU, GUP, RAM, Bandwidth etc. The amount of each type of resource may also different, some server are assembled high- performance CPU with more than 20 cores and 2.5 GHZ main frequency and some server may be assembled CPU with less 10 cores. Accordingly, the number of resource type and amount of each resource may also different in terms of different request. The objective of this study is minimizing the number of server which are allocated

978-1-5386-5900-7/18/$31.00 ©2018 IEEE

jobs or VMs. In this way, the resource utilization is improved, and the serves who are totally free can enter in energy saving mode to save energy.

, We modeled this kind of resource allocation problem as Vector Bin Packing Problem (VBPP), and proposed some new algorithms to solve the VBPP, based the classic algorithms for VBPP. VBPP is a variant of classic bin packing. Binpacking problem has been researched for many years, the basic objective of this problem is using least number of bins to pack series item. Coffman [11] introduced and concluded solution in his work. Many variants of bin packing were proposed, those years. Vector Bin Packing is one of common variants proposed in Garey [12]. In the VBP problem, the size of the items are described by a d-dimensional vector, the objective is also minimizing the number of bins ever used. The recent work [1], [13] introduced some offline heuristics algorithm for VBPP. Offline means, algorithm know the detail of all items, and can sort them in specific order.

In this paper, we proposed Multi-Dimensional Cloud Resource Dynamic Allocation Model (MDCRA). In the MDCRA, we model the VM or job allocation problem as Vector Bin Packing Problem. Each server is considered as the bin and the VM request or job request is considered ad item. The cloud server possess fixed resources, the number of type, the amount of each type. This model was designed based VBP technology. We make the following contributions.

-We solve the dynamic multi-dimension VM placement or job dispatching in data centers which have heterogeneous servers cluster via online Vector Bin Packing Problem.
-We proposed Single Weight and Double Weight family algorithms for online Vector Bin Packing problem.
-We conduct extensive simulation and experiments. The results demonstrate good performance of our algorithms.

The remainder of this paper is organized as Section II introduce the related work about cloud resource allocation and VBPP. Section III presents the system model and formulates the problem. Section IV introduces preliminaries and proposes algorithms. Section V carries out an evaluation for the algorithms through simulations. Section VI, discuss the limit of this paper and future work. Finally, a conclusion is made in section VII.

II. Related Work

In this section, the salient features of existing approach are discussed and analyzed. Number of studies based on Bin Packing technology in cloud computing environment have extensively discussed. [1], [9] also employed the algorithms for Vector Bin Packing Problem to solve cloud resources allocation problem. However, their research in the offline condition, in other word, all the requests have been arrival, the algorithms can assign the request with the knowledge of those requests. When the request arrives dynamically, those algorithms are not available. Jitpukdeebodintra [6] uses classic bin packing algorithms, it also considers offline situation.

Stolyar [7] and Furlong [10] formulated the cloud resources allocation problem as the online bin packing problem, however they assume each request merely need one

The classical bin packing, problem aims to use least number of bins to load a series items. It is a famous, NP-hard problem. This problem and its variants have been studied extensively for or two types resource. Li [8] aimed to minimize the total using span with MinUsageTime DBP problem, it has different optimizing objectives for many years. Generally, bin packing problem have two versions online and offline [11]. In terms of online bin packing problem, it packs each items as soon as it is inspected, without any knowledge of the items not yet encountered. Many classical algorithms were proposed such as Next-Fit [14], Worst-Fit [15] etc. On the contrary, the offline problem has all items available for preprocessing, reordering, grouping, etc. before packing.

If the size of the bin and the item is more than one dimension, we consider this kind of bin packing problem as multi- dimensional bin packing problem. The classic bin packing problem, packs items with only one dimension [11], while both it and its variants have multi-dimensional extension.

Vector Bin Packing Problem (VBPP) is one of variants of bin packing problem proposed in Garey [12]. In the VBP problem, the size of the items and bins are described by a d-dimensional vector, the objective is also minimizing the number of bins ever used. VBPP is a NP-hard Problem, even limit the dimension to 1. When the dimension is more than 2, VBPP was proved that no asymptotic PTAS [16]. Many algorithms were researched to solve VBPP. Spieksma [17], gave heuristics for 2-dimensional VBPP and also proposed lower bounding. Caprara et. al [18] also studied the 2-dimensional version, they gave improved heuristics and algorithms. Stillwell [9] used the variants of First Fit Decreasing(FFD) to solve VBPP. In the recent work, Panigrahy [1] studied FFD algorithm, and proposed FFD family heuristics for multi-dimensional VBPP. Gabay [13] proposed some heuristic algorithm based on classic Best Fit algorithms for the Vector Bin Packing with heterogeneous bins. The VBPP, most of them researched is offline, almost no work related to online Vector Bin Packing.

III. System Model And Problem Formulation

A. Multi-Dimensional Cloud Resource Dynamic Allocation Model(MCRDA)

We demonstrated VMs or jobs dynamically dispatching as the online multi-dimensional Vector Bin Packing Problem where each cloud serve is considered as bin and each VM is the item to be packed in the could server. The VM to be dispatched will described by a finite set V . $V = (v_1, v_2, ..., v_k)$. Each VM $v_k \in V$ refers to a vector $(a_k, l_k, x_k^1, x_k^2, ..., x_k^j)$,$x_k^j$ is VM v_k's resource demand for resource type j, j is resource type and k is the index of VM.a_k and l_k are the arrival time and leaving time of request v_k

The number of servers resourced types should be consistent with or more than the request. Otherwise, the amount of each servers resources is identical and fixed. The the set of servers which the data center possesses, denotes as S, and the i *th* server is s_i, k is the index of server which is determined by the order of starting server. $S = (s_1, s_2, s_3, ..., s_i)$,$|S|$ denotes the number of servers which the data center possesses. The resource of server s_i denote as a vector R_i. The resource type j of server s_i denote as r_i^j, $R = (r_i^1, r_i^2, r_i^3, , r_i^j)$.

When the request v_k arrives, the algorithms need to assign this request to server immediately. Notes that the request can arrive at arbitrary time. On the same way, the jobs or VMs

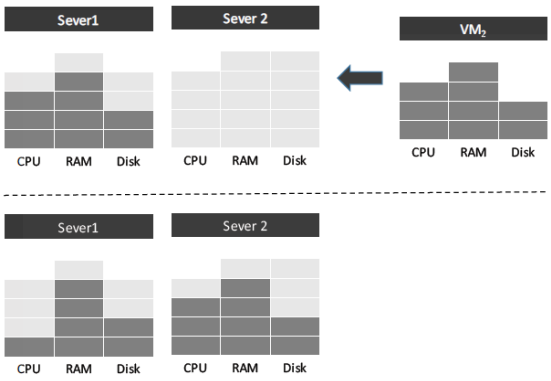

Fig. 1. MCRDA Model: Assigning a VM

who have been assigned can also leave at it declarative time l_k, or ahead or after.

Generally, this paper considers two kind of allocation strategies. In first strategies, algorithms merely consider the load of server, choose the server according to its load, maximum load or minimum load. we defined this kind of strategies as

Single Weight Algorithms. In those strategies, we proposed weight functions to measure the total size (weight) of requests and remaining capacity (also total capacity) of server for multi-dimensional case, which will be introduced in subsection IV-2.b Another kind of strategies consider both the load of server and the demand of request. Dot Production and Norm-based Greedy functions were proposed to measure the conjunct weight of server and request. This algorithms for this kind of strategies were called **Double Weight Algorithms.**

A System resource monitor is meaningful to declare here, it monitor the state of each server, if they are totally idle, it will take them to energy saving mode. This algorithm will be invoked every ten seconds.

The figure 1, illustrates how MCRDA Model assign a VM request to specific server. The light Grey represents the free resources of server and dark Grey stands for occupied resources. Upon the request arrived, algorithm assign it to a sever. As shown in figure 1, sever 1 cannot load the request VM_2, so the request is assigned to a new server.

B. Problem Formulation

The objective is to minimize the number of servers ever used, Namely N in equation 1. The current state server si denotes as C_i. C_i is a binary number, if server si is busy, C_i is equal to 1, otherwise C_i is equal to 0. Notes that each server can not overload. In other word, the sum of each type of resource which be allocated to VMs in each server should less than the corresponding resource capacity of this server. This constraint is shown in the equation 2. Notation d is the total number of server's resources type.

Minimize:

$$N = \sum_{i=0}^{|S|} C_i$$

$$C_i = \begin{cases} 0, & \text{Server is empty} \\ 1, & \text{otherwise S Server} \end{cases} \quad (1)$$

Subject to:

$$\prod_{j=0}^{d} r_j \geq \prod_{j=0}^{d} \sum_{v_{k \in s_i}, s_i \in S} X_{k,j} \quad (2)$$

IV. ALGORITHM

1) Best Fit Algorithm: Best Fit algorithm packs arrival item into an open bin of largest content in which it fits. If the there this no open bin which the arrival item fit, Best Fit algorithm open a new bin and pack the this item in it. Best Fit algorithm is a classic algorithm introduced in [19]. We employ a similar packing method in Single Weight Algorithms.

2) Weight Function: when we assume all the type of resource demand is identical, namely the demand is single dimensional, it is easy to compare the size of server load or the size of request resource. But when we consider multi dimension, we need to define a measure. In this paper, we consider it as Weight Function. In this paper, we tested two options. We give the Weight Function for server here, the Weight Function of request is similar.

$$\varphi(s_i) = \sum \alpha_j r_i^j \quad (3)$$

$$\varphi(s_i) = \prod_{j \leq D} \alpha_j r_i^j \quad (4)$$

α_j is a scale vector, the amount and significance of each resource is different. we use scale vector, emphasize the importance. Otherwise, αj helps quantify the amount of each resource. The αj in equation 3, 4, 5, 6 have same meaning.

3) Dot Production:

$$\phi(i,k) = \sum_{j \in D} \alpha_j r_i^j x_k^j \quad (5)$$

Notes that, we defined s_i^j as the resource type j of server s_i before, it represents the amount of resource type j the server s_i hold. It was changed dynamically, after a VM or Job were assigned to it.

4) Norm-based Greedy:

$$\phi(i,k) = \sum_{j \in D} \alpha_j (x_k^j - r_i^j)^2 \quad (6)$$

A. Single Weight Algorithms

$\varphi(i)$ refer to weight function. Since we have two functions, this Single Weight Algorithm(SWA) also have two version. Otherwise, there are also two ways to choose server, the maximum load priority and the minimum load priority. To avoid wasting space, we only focus on showing the maximum load priority.

In algorithm 1, from line 1 to 5 the algorithm calculates the load weight of each server s_i, and sort them according this value in descending order. In this way, the server whose index is 1 in the Set S has largest load. From line 6 to 12, SW tries to assign arrived request v to the server which it fits. If no started server can accept v, SWA will start a new server to put this request.

Algorithm 1 SWA:Maximum Load Priority

Input: Server set S, VM request v, Current Packing P
Output: new P

```
1: //Calculate the load weight of each server S_i
2: for i to |S| do
3:     L[i] ← φ(i)
4: end for
5: Sort set S in descending order of load weight L[i]
6: for i to |S| do
7:     if v fits into s_i then
8:         Pack v into s_i
9:         update P
10:        return P
11:    end if
12: end for
13: if v fits into new server then
14:    Pack v into a new server
15: else
16:    Reject request v
17: end if
```

B. Double Weight Algorithms

Double Weight Algorithms(DWA) consider both the load of server and the demand of request. Bear similarity to SWA, $\phi(i, k)$ also has two options, the dot production and Norm-based Greedy. So DWA has two version. We introduce the DWA with dot production in Algorithm 2.

Algorithm 2 DWA:Dot Production

Input: Server set S, VM request v, Current Packing P
Output: new P

```
1: for i to |S| do
2:     Calculate φ(i, k)
3: end for
4: Sort set S in descending order of Dot Production φ(i, k)
5: for i to |S| do
6:     if v fits into s_i then
7:         Pack v into s_i
8:         update P
9:         return P
10:    end if
11: end for
12: if v fits into new server then
13:    Pack v into a new server
14: else
15:    Reject request v
16: end if
```

Upon the request v arrived, DWA with dot production dispatches the request to the server who has the largest conjunct weight with v. In algorithm 2, from line to 4, it calculates conjunct weight and sorts set S. From line 6 to 12, it implements the process of dispatching.

TABLE I
CONFIGURATION OF EXPERIMENT MACHINE

	System	CPU	RAM
parameter	Ubuntu 14.04.4 LTS	Xeon CPU E5-2420	16 GB

C. Other Algorithms

- **SWA:minimum Load Priority:** It is similar to SWA:Maximum Load Priority. The deference is sorting the set S in ascending order.
- **SWA:minimum Load Priority-sum:** Observe that, both SWA:Maximum Load Priority and SWA:minimum Load Priority have two version, using weight function 3 or 4. The SWA:minimum Load Priority use weight function 3, denoted as SWA:minimum Load Priority-sum. Others are similar. Use 'Pro' represent equation 4.
- **DWA:Norm-based Greedy:** It is similar to DWA:Dot Production. However it chooses the server who makes $\phi(i, k)$ smallest.

V. PERFORMANCE EVALUATION

We have conducted thorough simulations to investigate the performance of proposed algorithms. For convenience, we use the naming rule of SWA-Max/Min-S/P to represent the Single Weight Algorithms in the Maximum/Minimum Load Priority using Sum/Production weight function. Use naming rule of DWA-DP/NG to represent the double Weight Algorithms using dot production/Norm-based Greedy. To investigate the effect-tiveness of our , we also implement one baseline algorithms with random select the server to assign the request: Ran-S

The request will arrive at arbitrary time in 7 days, in other word, distribute in any time in (0, 7x24). Observed that, the request arrives dynamically and also leave dynamically. The leaving time of each request actually after its arrival time. The requests' average duration of stay is set as 2, 3 or 4 days. The resource demands of each request are randomly selected from table III. Table II shows the real server hardware configuration we use in the simulation. This server's configure roughly equal to Dell PowerEdge(TM) R740-4 [20]. In the simulation, we default use total 2000 requests to test. There is no limit to the number of server.

Figure 6 shows the relation between the system load and energy consumption. Even the server is total idle, the server still consumes the energy, and the power usage of idle server is about 40% ∼ 50%. The power consumption will increase with the increase of system load.

All the simulations were run on an Ubuntu 14.04.4 LTS machine with Intel Xeon CPU E5-2420 and 16 GB memory, which is shown in table I. Each measurement is averaged over 100 instances.

A. The Number of Server

In this section, we use server configuration 1, to do the simulation, then we use server configuration 3 do a same experiment. Notes that, in configuration 1, the number of resources type is 4, while, in configuration 3, the number is

TABLE II
CONFIGURATION OF SERVER

Configure	CPU(core)	RAM(GB)	Storage(GB)	Bandwidth(Mbps)
1	64	128	8000	1000
2	54	64	1000	1000
3	64	128	4000	1000

TABLE III
TYPE OF REQUEST

Type	CPU(core)	RAM(GB)	Storage(GB)	Bandwidth(Mbps)
1	5	5	150	100
2	8	12	200	150
3	15	20	300	200
4	16	24	500	400
5	5	5	150	0
6	8	12	200	0

3. Both the two experiments, use the same requests average duration time, it is 3 days.

The requests arrive and leave at arbitrary time in 7 days. Figure 2, 3, 4, shows the mean number of staring server in each day produced from different algorithms. Figure 2 show the data from experiment which uses configure 1 as all the server's capacity. Figure 3, 4 has similar meaning. According to figure 2, 3, 4, SWA-Max-S and SWA-Max-P share almost same performance in the objective of minimizing the number of used server. On the contrary, SWA-Min-S and SWA-Min-P shows the lower performance, which can generally illustrate that preferentially choosing the server which has more load than others, is beneficial to minimizing the number of used server. DWA family algorithms have higher performance in single server hardware configure setting, while in mixed setting, such as configure 1 and 2, they reveal worse performance which is shown in the bar of figure 2, 3, 4.

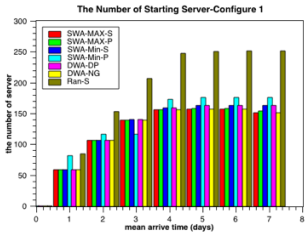

Fig. 2. Number of Server in C1

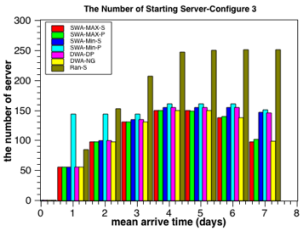

Fig. 3. Number of Server in C3

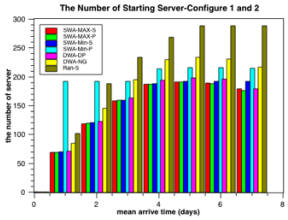

Fig. 4. Number of Server in C1 & 2

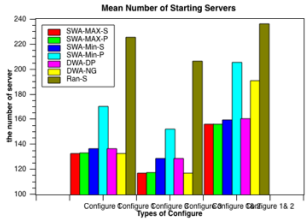

Fig. 5. Mean Number of Server

Figure 5 shows the average number of servers each algorithm used in 7 days, three different of server hardware configure setting are shown to compare and contrast. It is clearly that all the algorithms perform well in configure 3 setting, in other word, algorithms have better property in 3 dimension condition. Through both rigorous theoretical analyses and extensive simulations, we demonstrate that SWA-Max-S/P and DWA-NG/DP have excellent performance to achieve minimizing the number of used servers, implement energy saving and utilization maximization.

B. Energy Consumption

In this section, we researched the energy consumption. We used both configuration 1 and configuration 1 & 2 as the physical server setting. The requests' average duration time distribute in 2,3 days. To obtain the data of energy consumption, the model of energy consumption evaluation according to server load need to be proposed. In this paper, for convenience, we calculate the energy consumption on the basis of the consumption of two main component of server CPU and RAM. Figure 6 shows the RAM's power in idle and busy state. Clearly, larger memory consumes more energy. Figure 7 shows the power of Intel CPU in different load.

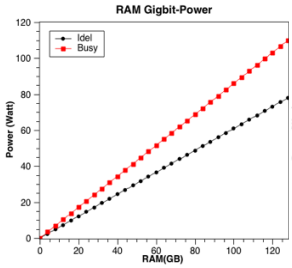

Fig. 6. The Power Relation of RAM

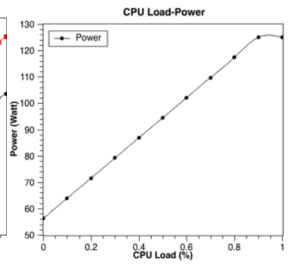

Fig. 7. The Power Relation of CPU

The data of figure 8, 9 obtain from same server setting, configure 1, however, the requests' average staying time is 2 days for figure 8, 3 days for figure 9. Figure 10, 11 have same meaning. Similar with the result in section V-A, SWA- Max-P/S and DWA-DP/NG algorithms achieve better performance. Both DWA and SWA family algorithms have better algorithms than baseline algorithms. Proposed algorithms are stable in various server setting and request staying time. The energy consumption arrives peak value in day 5, at this time point, the server cluster holds the largest number of request, most of them were leave after day 5.

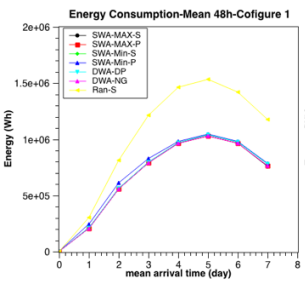

Fig. 8. Energy Consumption 48h-C1

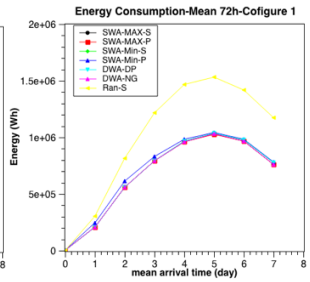

Fig. 9.Energy Consumption 72h-C1

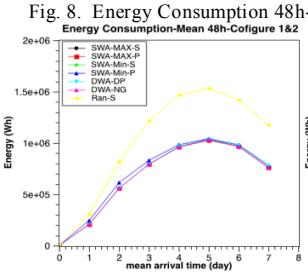

Fig. 10. Energy Consumption 48h-1&2

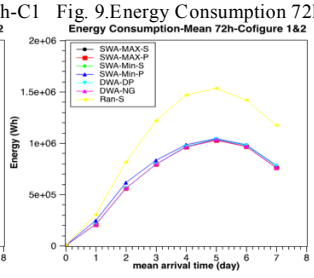

Fig.11. Energy Consumption 72h-1&2

C. Resources Utilization

We employ configure 1 &2 as the server setting and user 4 days as the default request staying time. Figure 12, shows SWA-Max-S/P and DWA-NG/DP can dispatch the requests to make the RAM load higher than 50% after the first day, when the requests increase. SWA-Min-S/P algorithms' performance are close to the baseline algorithm which can be explained as they cannot allocate most of requests to make some servers total idle, in order to turn them to sleep mode.

Figure 13 shows the load of CPU. SWA-Max-S and DWA- NG achieve the highest performance, since they keep the servers load higher than 80%. On the contrary, SWA-Min-S has the worst performance.

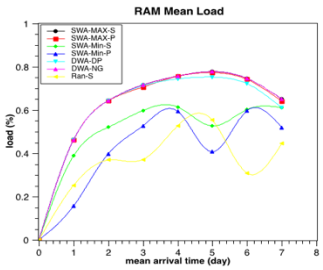

Fig.12.RAM Load

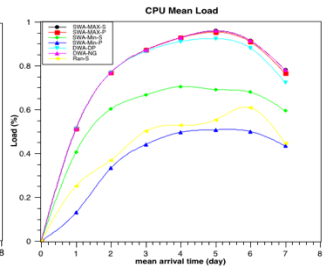

Fig.13.CPU Load

VI. Conclusion

In this paper, we explore the cloud resource allocation strategies which can minimize the number of used servers, in order to save the energy and maximize the resources utilization, in the condition that requests arrive and leave dynamically and the demand are multi dimensions. We modeled this problem as the Online Vector Bin Packing, and proposed SWA and DWA family algorithms. Through both rigorous theoretical analyses and extensive simulations, we demonstrate that SWA-Max-S/P and DWA-NG/DP can achieve minimizing the number of used servers, implement the energy saving and utilization maximization. The number of server used by SWA-Max-S/P is about 40.67% less than Ran-S, is 36.25% by DWA-NG/DP. Using SWA-Max-S/P algorithm can save about 33.10% than Ran-S, corresponding DWA-NG/DP has 30.06% saving.

REFERENCE

[1] Razaque, Abdul, Nikhileshwara Reddy Vennapusa, Nisargkumar Soni, and Guna Sree Janapati. "Task scheduling in cloud computing." In Systems, Applications and Technology Conference (LISAT), 2016 IEEE Long Island, pp. 1-5. IEEE, 2016.

[2] Gartner, "infrastructure-as-a-service-iaas," https://www.gartner.com/it-glossary/infrastructure-as-a-service-iaas/, accessed November 20, 2017.

[3] P. T. A. V. J.S. Chase, D.C. Anderson and R. Doyle, "Managing energy and server resources in hosting cen- ters," *Proc. 18th ACM Symp. Operating System Principles (SOSP01)*, vol. 33, no. 4, p. 103116, 2001.

[4] Razaque, Abdul, Saty Siva Varma Nadimpalli, Suharsha Vommina, Dinesh Kumar Atukuri, Dammannagari Nayani Reddy, Poojitha Anne, Divya Vegi, and Vamsee Sai Malllapu. "Secure data sharing in multi-clouds." In Electrical, Electronics, and Optimization Techniques (ICEEOT), International Conference on, pp. 1909-1913. IEEE, 2016.

[5] Y. Agarwal, S. Savage, and R. Gupta, "Sleepserver: a software-only approach for reducing the energy consumption of pcs within enterprise environments," in *Usenix Conference on Usenix Technical Conference*, 2010, pp. 22–22.

[6] R. Jitpukdeebodintra and S. Witosurapot, "Efficient cloud gaming resource provision via multi-dimensional bin-packing," 2016.

[7] A. L. Stolyar and Y. Zhong, "An infinite server system with general packing constraints: Asymptotic optimality of a greedy randomized algorithm," *2013 51st Annual Allerton Conference on Communication, Control, and Computing, Allerton 2013*, no. February, pp. 575–582, 2013.

[8] Y. Li, X. Tang, and W. Cai, "Dynamic Bin Packing for On-Demand Cloud Resource Allocation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 1, pp. 157–170, 2016.

[9] M. Stillwell, D. Schanzenbach, F. Vivien, and H. Casanova, "Resource allocation algorithms for virtualized service hosting platforms," *Journal of Parallel and Distributed Computing*, vol. 70, no. 9, pp. 962–974, 2010.

[10] J. Furlong, L. Shi, and R. Wang, "Empirical evaluation of vector bin packing algorithms for energy efficient data centers - Vinicius (consolidac¸ a˜o)," 2013 IEEE Symposium on Computers and Communications (ISCC), no.July 2013 [Online].

[11] E. G. Coffman, J. Csirik, G. Galambos, S. Martello, and D. Vigo, "Bin Packing Approximation Algorithms: Survey and Classification," in *Handbook of Combinatorial Optimization*, 2013, pp. 455–531. [Online]. Available: http://link.springer.com/10.1007/978-1-4419-7997-1

[12] Akbari, Elham, Francis Cung, Hardik Patel, Abdul Razaque, and Hemin Nilesh Dalal. "Incorporation of weighted linear prediction technique and M/M/1 Queuing Theory for improving energy efficiency of Cloud computing datacenters." In Systems, Applications and Technology Conference (LISAT), 2016 IEEE Long Island, pp. 1-5. IEEE, 2016

[13] M.GabayandS.Zaourar,"Vectorbinpackingwithheterogeneousbins: application to the machine reassignment problem," *Annals of Operations Research*, vol. 242, no. 1, pp. 161–194, 2016.

[14] D.S.Johnson,A.Demers,J.D.Ullman,M.R.Garey,andR.L.Graham, "Worst-case performance bounds for simple one-dimensional packing algorithms," vol. 3, no. 4, pp. 299–325, 2008.

[15] D.S.Johnson,"Fastalgorithmsforbinpacking*,"*JournalofComputer & System Sciences*, vol. 8, no. 3, pp. 272–314, 1974.

[16] G. J. Woeginger, There is no asymptotic PTAS for two-dimensional vector packing. Elsevier North-Holland, Inc., 1997.

[17] F. C. R. Spieksma, "A branch-and-bound algorithm for the two-dimensional vector packing problem," *Computers & Operations Research*, vol. 21, no. 1, pp. 19–25, 1994

[18] A. Caprara and P. Toth, "Lower bounds and algorithms for the 2-dimensional vector packing problem," *Discrete Applied Mathematics*, vol. 111, no. 3, pp. 231–262, 2001.

[19] E. G. C. Jr, J. Csirik, G. Galambos, S. Martello, and D. Vigo, *Bin Packing Approximation Algorithms: Survey and Classification*. Springer New York, 2013.