2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops

Energy Saving Virtual Machine Allocation in Cloud Computing

Ruitao Xie*, Xiaohua Jia†, Kan Yang*, Bo Zhang*
Department of Computer Science
City University of Hong Kong
*{ruitao.xie, kan.yang, Bo.Zhang}@my.cityu.edu.hk
†csjia@cityu.edu.hk

***Abstract*—In the data center, a server can work in either active state or power-saving state. The power consumption in the power-saving state is almost 0, thus it is always desirable to allocate as many VMs as possible to some active servers and leave the rest to power-saving state in order to reduce the energy consumption of the data center. In this paper, we study such a VM allocation problem. Given a set VMs and a set of servers in a data center, each VM has a resource demand (CPU, memory, storage) and a starting time and a finishing time, and each server has resource capacity. There is an additional energy cost for a server to switch from power-saving state to active state. The servers are non-homogeneous. The problem of our concern is to allocate the VMs onto servers, such that the VMs resource demands can be met and the total energy consumption of servers is minimized. The problem is formulated as a boolean integer linear programming problem. A heuristic algorithm is proposed to solve the problem. Extensive simulations have been conducted to demonstrate our proposed method can significantly save the energy consumption in data centers.**

Keywords: Virtual Machine Allocation, Energy Saving, Cloud Computing, Data Center

I. Introduction

Virtual Machines (VMs) are a type of infrastructure services provided by data centers. Cloud users request a VM from a data center by specifying the resource demands (in terms of CPU, memory, storage), platform, and duration of the VM. The data center, upon receiving this user request, will allocate the VM to a server and reserve all necessary resources on the server. A server can host multiple VMs, depending on the capacity of the server and the resource demands of the VMs. As energy consumption has become a major cost factor and an environmental issue of data centers, it is important to allocate VMs to the servers properly such that the energy consumption can be reduced [1]–[6].

In this paper we study the issue of assigning user VM requests to servers aiming for minimizing energy consumption in the data centers. A server can work in either active state or power-saving state. The power consumption in the power-saving state is almost 0. Therefore, it is always advantageous to allocate as many VMs as possible to some active servers and leave the rest to power-saving state, such that the energy consumption of the data center can be reduced. However, there is an energy cost for switching the server between the active and the power-saving states. It is no good to switch on or off a server too often. The problem of our concern can be defined as follows. We are given m VMs and n servers. Each VM is associated with a set of resource demands (CPU, memory, storage, etc.) and a starting time and a finishing time. To get more benefits from the short idle time of servers, we consider the time unit on the minute or more fine-grained scale. Each server has resource capacity. Since servers usually share large disk space by using high speed optical systems in data centers, as for resource demand of VMs and capacity of servers, we only focus on CPU and memory. Our problem is to allocate the VMs onto the servers, such that the resource demands of VMs can be met and the total energy consumption of the servers is minimized.

Our problem distinguishes from the existing works in several aspects: 1) Our problem allocates a VM on a server throughout its time duration. It is different from the VM allocation problem in a single time unit formulated as bin-packing [7], [8]. 2) In our problem, servers are non-homogeneous. Each server has its own resource capacity, power consumption and transition cost. Thus, the VMs can not be assigned across the servers uniformly. While in the existing works [2]–[5], the loads are assigned uniformly across the homogeneous servers. 3) The objective of our problem is to minimize the energy consumption of servers. It is different from the works on fixed interval job scheduling aiming for minimizing busy time [9], [10], or minimizing the cost of running jobs [11]–[13].

The problem is formulated as a boolean integer linear programming problem. We propose a heuristic algorithm to allocate VMs in the increasing order of their starting time. For each VM, there is a subset of servers having sufficient spare resources throughout its time duration. From the subset, a server is selected such that the incremental energy cost is minimum. Extensive simulations have been conducted to demonstrate our proposed method can significantly save the energy consumption in data centers.

The rest of this paper is organized as follows. Section II presents system model and problem formulation. Section III presents the heuristic algorithm. The simulations and performance evaluations are presented in Section IV. Section V introduces the previous works related to the virtual machine allocation in data centers. Section VII concludes the paper.

978-0-7695-5023-7/13 $25.00 © 2013 IEEE
DOI 10.1109/ICDCSW.2013.37

II. System Model and Problem Formulation

The system consists of m VMs and n servers. The VM v_j has time duration $[t_j^{\mathrm{s}}, t_j^{\mathrm{e}}]$, where t_j^{s} is the starting time and t_j^{e} is the finishing time. Let R_{jt}^{CPU} and R_{jt}^{MEM} denote the CPU and memory demands respectively of VM v_j at the time unit t. Each server, say s_i, has CPU and memory capacity, denoted by C_i^{CPU} and C_i^{MEM}, respectively. We consider the VM allocation problem in an entire period of $[1, T]$, where T is the longest time of system planning.

The energy cost of a server in data center has three components: 1) the operation cost of a server to keep itself in active state, 2) the operation cost of a server to run VMs, and 3) the transition cost α incurred in switching a server back-and-forth between the active state and the power-saving state. The operation cost of a server can be modeled by an affine power function as following [4], [14],

$$P(u) = P_{idle} + (P_{peak} - P_{idle})u, \tag{1}$$

where P_{idle} is the power when the server is idle, P_{peak} is the power consumed by the server under peak load, and the load $0 \le u \le 1$ is the percentage of CPU capacity used by the VMs running on it.

Since we consider non-homogeneous servers in the system, servers have different energy cost for state transition and power function parameters. Let α_i denote the transition cost, and $P_{idle,i}$ and $P_{peak,i}$ denote the power function parameters of the server s_i. Let P_i^1 denote the power of the server s_i consumed by a VM with one unit of CPU demand. It is

$$P_i^1 = \frac{(P_{peak,i} - P_{idle,i})}{C_i^{\mathrm{CPU}}}. \tag{2}$$

Let W_{ij} denote the cost of the server s_i consumed by running the VM v_j, which can be represented as:

$$W_{ij} = P_i^1 \sum_{t=t_j^{\mathrm{s}}}^{t_j^{\mathrm{e}}} R_{jt}^{\mathrm{CPU}}. \tag{3}$$

Let the boolean variable x_{ij} denote whether the VM v_j is allocated on the server s_i. Then, the operation cost of the server s_i to run VMs throughout time span $[1, T]$ is:

$$\sum_{j=1}^{m} W_{ij}\, x_{ij}. \tag{4}$$

Let the boolean variable y_{it} denote whether the server s_i is active or not during time unit t. The operation cost to keep the server s_i in active state throughout time span $[1, T]$ is:

$$\sum_{t=1}^{T} P_{idle,i}\, y_{it}. \tag{5}$$

We assume before and after the entire time period $[1, T]$ all servers are in the power-saving state, that is $y_{i0} = 0$ and $y_{i,T+1} = 0$. The energy cost of the third component, that is the transition cost, of the server s_i can be represented as:

$$\sum_{t=1}^{T} \alpha_i\, (y_{it} - y_{i,t-1})^+, \tag{6}$$

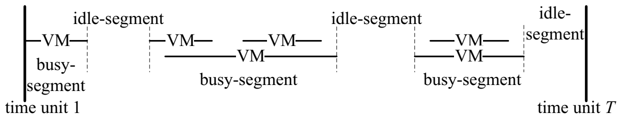


Fig. 1. A server experiences a sequence of time-segments alternating in running VMs and running no VM.

where $f(x)^+$ denotes $\max\{0, f(x)\}$. Thus, the total energy cost of the system is

$$\sum_{i=1}^{n}\sum_{j=1}^{m} W_{ij}\, x_{ij} + \sum_{i=1}^{n}\sum_{t=1}^{T}(P_{idle,i}\, y_{it} + \alpha_i\,(y_{it} - y_{i,t-1})^+) \tag{7}$$

The problem is formulated as following,

$$\min \sum_{i=1}^{n}\sum_{j=1}^{m} W_{ij}\, x_{ij} + \sum_{i=1}^{n}\sum_{t=1}^{T}(P_{idle,i}\, y_{it} + \alpha_i\,(y_{it} - y_{i,t-1})^+) \tag{8}$$

$$s.t. \sum_{j=1}^{m} R_{jt}^{\mathrm{CPU}}\, x_{ij} \le C_i^{\mathrm{CPU}}\, y_{it}, \qquad \forall i, \forall t \tag{9}$$

$$\sum_{j=1}^{m} R_{jt}^{\mathrm{MEM}}\, x_{ij} \le C_i^{\mathrm{MEM}}\, y_{it}, \qquad \forall i, \forall t \tag{10}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j \tag{11}$$

$$x_{ij} \le y_{it} \qquad \forall i, \forall j, t_j^{\mathrm{s}} \le t \le t_j^{\mathrm{e}} \tag{12}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, \forall j \tag{13}$$

$$y_{it} \in \{0, 1\} \qquad \forall i, \forall t \tag{14}$$

Constraints (9) and (10) are capacity constraints, and ensure that in each time unit, the resource volume of a server used by the VMs on it does not exceed its capacity. Constraints (11) ensure that each VM is allocated on exact one server. Constraints (12) ensure that each VM is allocated only on the servers which are active throughout its time duration. In our model all variables are boolean.

III. Heuristic Algorithm

The VM allocation problem to minimize the energy cost is formulated as an integer linear programming, which is NP-hard. In this section, we present a heuristic algorithm to solve it. Our algorithm allocates VMs in the increasing order of their starting time. For each VM, there is a subset of servers having sufficient spare resources throughout its time duration. From the subset, a server is selected such that after the allocation the incremental energy cost is minimum.

Firstly we explain how to get the energy cost in our allocation algorithm. Considering a server s_i runs a set of VMs V_i throughout the entire period $[1, T]$, it experiences a sequence of time-segments alternating in running VMs (called busy-segment) and running no VM (called idle-segment), as shown in Fig. 1. For the busy-segment, the server must be in active state, the energy cost consists of two components:

1) the cost to run VMs and 2) the cost to keep the server in active. Let BS_i denote the set of busy-segments of server s_i. Throughout all these time-segments, the server s_i consumes the cost as following,

$$\sum_{v_j \in V_i} W_{ij} + \sum_{[t,\tau] \in BS_i} P_{idle,i}\,(\tau - t + 1). \tag{15}$$

For the idle-segment $[t, \tau]$, the server can switch off to save energy if the transition cost is less than the cost to keep the server active; otherwise, the server keeps in active state during the idle-segment. Let IS_i denote the set of idle-segments of server s_i. Throughout all these time-segments, the server s_i consumes the cost as following,

$$\sum_{[t,\tau] \in IS_i} \min\{P_{idle,i}\,(\tau - t + 1),\ \alpha_i\}. \tag{16}$$

Thus, throughout the entire period $[1, T]$, the energy cost consumed by the server s_i is

$$\begin{aligned} Cost_i = &\sum_{v_j \in V_i} W_{ij} + \sum_{[t,\tau] \in BS_i} P_{idle,i}(\tau - t + 1) \\ &+ \sum_{[t,\tau] \in IS_i} \min\{P_{idle,i}(\tau - t + 1),\ \alpha_i\}. \end{aligned} \tag{17}$$

Our algorithm allocates the VMs in the increasing order of their starting time. All servers are kept in power-saving state initially. The energy cost of each server is initialized as 0. At each iteration the algorithm selects a server to allocate the VM v_j. Firstly, there is a subset S_j of servers having sufficient spare CPU and memory for the VM v_j throughout its time duration. Secondly, for each server in the subset S_j, the energy cost is evaluated from Eq. (17) supposing the VM v_j is allocated on. Thirdly, a server is selected from the subset S_j such that the incremental energy cost is minimum. The energy cost of the selected server is updated. Finally, the above steps are repeated until all VMs are allocated.

By allocating each VM on a server with the least incremental energy cost, our algorithm can save energy due to the following reasons. Firstly, our algorithm tends to use the energy efficient servers. Secondly, our algorithm leads to high resource utilization. The servers with small resource capacity usually consume lower power than those with large resource capacity. Our algorithm consolidates VMs on servers with small resource capacity, so that the server resources are adequately used. In particular, as the system load is light, our algorithm can lead to high resource utilization. Thirdly, our algorithm tends to use the server with low transition cost. For example, suppose all servers are in the power-saving state, a VM would be allocated on a server with less transition cost.

IV. Simulations and Analysis

A. Baseline method and evaluation metrics

We use First Fit Power Saving (FFPS) method as a baseline to evaluate our algorithm. In this baseline method, VMs are allocated in the increasing order of their starting time, and servers are randomly sorted. Each VM is allocated on the first searched server which can provide sufficient resources to the VM throughout its time duration. After all VMs are allocated, each server's state throughout entire period can be determined. That is, each server switches off during the idle-segment if the transition cost is less than the idle power to keep itself active; while it keeps active during other time-segments. The energy cost of each server can be calculated from Eq. (17). We define *energy reduction ratio* as the reduced cost divided by the cost of FFPS method.

TABLE I
The types of resource demands of VMs

type	CPU(compute unit)	memory(GBytes)
standard type 1	1	1.7
standard type 2	2	3.75
standard type 3	4	7.5
standard type 4	8	15
memory-intensive type 1	6.5	17.1
memory-intensive type 2	13	34.2
memory-intensive type 3	26	68.4
CPU-intensive type 1	5	1.7
CPU-intensive type 2	20	7

B. Simulation settings

1) VM settings: In our simulations, VM requests arrive according to the Poisson process. The mean inter-arrival time varies from 0.5 to 4 time units. The length of VMs follows the exponential distribution. The mean length is set to 2, 5 or 8 time units. The starting time and the finishing time of VMs are integer. The resource demands of each VM is stable and randomly set to a type in Table I. The parameters in Table I refer to Amazon Elastic Compute Cloud [15]. Both CPU-intensive VMs and memory-intensive VMs are simulated to evaluate our algorithm.

2) Server settings: It is not easy to quantify the CPU of servers using the compute unit defined by Amazon Elastic Compute Cloud in practice. Thus, several hypothetical servers are considered in our simulations, as shown in Table II. Every type of server corresponds to a pair of CPU and memory capacity and a pair of power consumption parameters. It is also difficult to quantify the server power consumption by CPU and memory. Since a server's power consumption consists of the power consumption of all system components: CPU, memory, disk drivers, network switches, fans etc. [14]. We set the power parameters as the following simple rules: 1) the server with CPU of 64 compute units and memory of 80 GBytes is roughly equivalent to the blade server HP ProLiant BL460c G6 with 80 GBytes memory [16]; 2) the power usage in the idle state is about 40%-50%, which is common for the servers in the data center [14]; 3) server power consumption increases as resource capacity increases.

3) Transition cost settings: During the whole time when the server switches on, power is consumed at peak rate [17]. Thus, the server's transition cost is P_{peak} times of transition time. It takes 30s to setup a desktop from hibernate state [17]. In our simulations, the transition time of servers ranges from 30s to 3min. The time unit in our model is 1 minute.

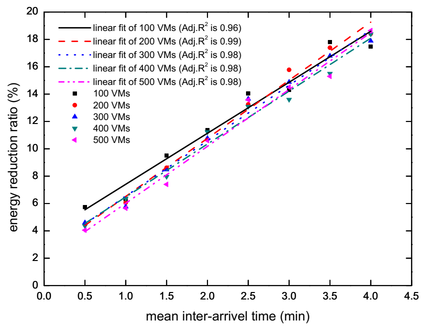


Fig. 2. The energy reduction ratio of the allocation of all types of VMs on all types of servers.

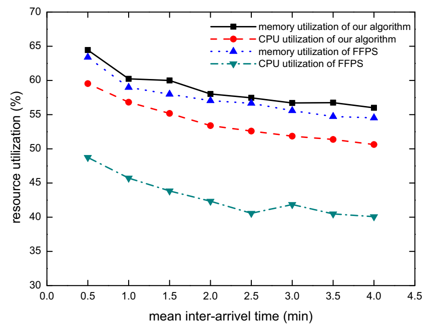


Fig. 3. The average CPU utilization and memory utilization of servers with 100 VMs allocated on.

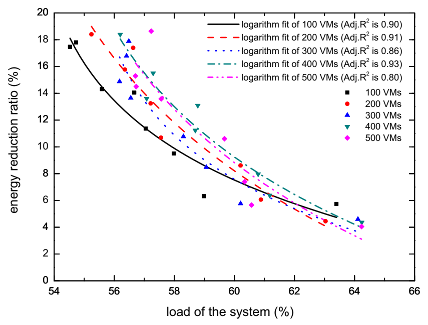


Fig. 4. The energy reduction ratio vs. the memory load of the system.

TABLE II
THE TYPES OF RESOURCE CAPACITIES AND POWER CONSUMPTION PARAMETERS OF SERVERS

type	CPU (compute unit)	memory (GBytes)	P_{idle} (W)	P_{peak} (W)	$\frac{P_{idle}}{P_{peak}}$
type 1	8	16	25	50	50%
type 2	16	32	41.25	90	45.83%
type 3	32	64	80	175	45.71%
type 4	32	32	70	162.5	43.08%
type 5	64	80	145	325	44.62%

C. Reduction of energy cost

In this group of simulations, we consider all types of VMs and all types of servers. The transition time of all servers is set to 1 min. The mean length of VM time durations is set to 5 min. The number of VMs varies from 100 to 500, and the number of servers is set to half the VMs'. The other parameters are set as introduced in IV-B. Each simulation result is averaged over 50 random runs.

Fig. 2 shows the energy reduction ratio compared to the FFPS method, where the lines are linear fitting curves of energy reduction ratio. The adjusted r-square (Adj. R^2 for short) value measures the goodness of fit. The closer the fit is to the data points, the closer it will be to the value of 1. As shown in Fig. 2, the energy reduction ratio approximately increases linearly with the mean inter-arrival time. Our algorithm can save about 18% energy cost compared to the FFPS method when the mean inter-arrival time is 4 min. As inter-arrival time is short, VM requests arrive at a high rate, and this incurs more VMs at each time unit. The allocation by the FFPS method can lead to a high resource utilization. As the inter-arrival time is long, VM requests arrive at a low rate, and there are fewer VMs at each time unit. The allocation by the FFPS method can easily lead to a low resource utilization. While our algorithm allocates each VM on the server with the least incremental cost and leads to a high resource utilization. Thus, more energy cost is reduced.

Fig. 3 shows the average CPU utilization and average memory utilization of servers with 100 VMs allocated on, by using the FFPS method and our algorithm. The CPU utilization of a server at time t is the percentage of CPU capacity used by the VMs running at that time. The average CPU utilization is calculated by averaging nonzero utilization values, measuring the CPU usage when the server is active. The average memory utilization is calculated similarly. As shown in Fig. 3, the CPU utilization in the FFPS method is low. It incurs large unevenness between two resource utilizations. While our algorithm can significantly improve the CPU utilization and leads to two resource utilizations more even. It is also observed that resource utilization decreases as the mean inter-arrival time increases. That is because the longer the inter-arrival time is, the fewer VMs there are at each time unit.

Fig. 2 demonstrates that similar results are obtained with the varying number of VMs. With the fixed mean inter-arrival time and mean length of VM time durations, the more VMs are generated, the longer the entire period of optimization is. Although a VM time duration overlaps with others, the effect of other VMs' allocation long time away is negligible. Thus, the energy reduction ratio is similar for 100-500 VMs. This result demonstrates that our algorithm is scalable.

We quantify the CPU load and the memory load of the system respectively by the average CPU utilization and average memory utilization of servers calculated by the FFPS method. Fig. 4 shows the energy reduction ratio vs. the memory load, where the lines are logarithm fitting curves of energy reduction ratio. It demonstrates that as the load increases, the energy reduction ratio decreases and the decrease rate becomes slower.

D. The impact of transition time of servers

In this section, we evaluate the impact of transition time of servers on the performance of our algorithm. In this simulation, we set the mean length of VMs to 5 min. We allocate 100 VMs on 50 servers. All types of VMs and all types of servers are used. Fig. 5 shows the energy reduction ratio with varying transition time settings, where lines are fitting curves. The shorter the transition time is, the less the transition cost becomes, and the more energy a server can save by switching off in the idle-segment. Thus, as the transition time becomes shorter, more energy can be saved by our algorithm, as shown in Fig. 5.

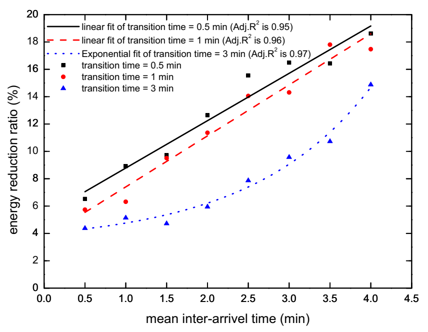


Fig. 5. The energy reduction ratio with varying transition time settings.

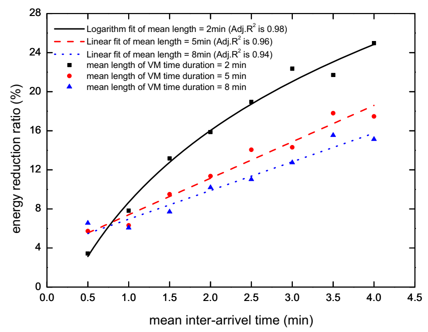


Fig. 6. The energy reduction ratio with varying mean length of VMs.

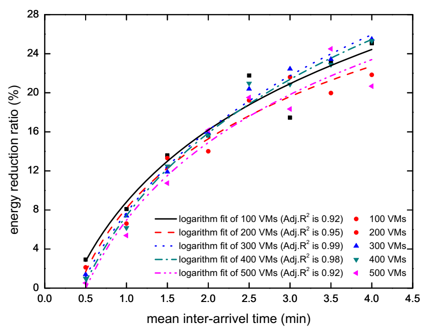
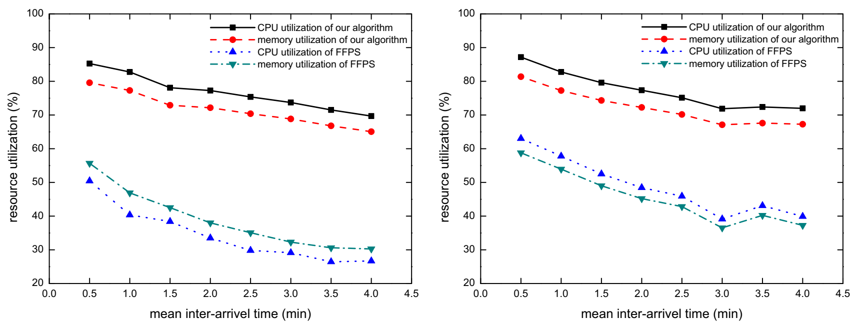


Fig. 7. The energy reduction ratio of the allocation of standard types of VMs on types 1-3 of servers.



(a) all types of servers used

(b) types 1-3 of servers used

Fig. 8. The average CPU utilization and memory utilization of servers with 100 standard types of VMs allocated on.

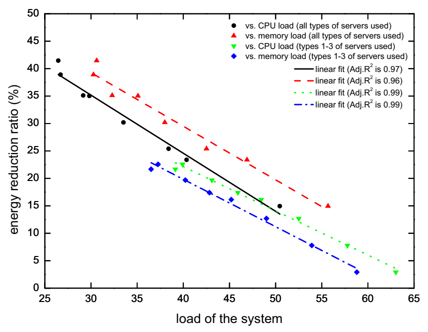


Fig. 9. The energy reduction ratio vs. the load of the system.

E. The impact of mean length of VMs

In this section, we evaluate the impact of mean length of VM time durations on the performance of our algorithm. In this simulation, we allocate 100 VMs on 50 servers. All types of VMs and all types of servers are used. The mean length of VMs varies among 2, 5 and 8 min. Fig. 6 shows the energy reduction ratio with varying mean length of VMs, where lines are fitting curves. It is shown that the shorter mean length is, the better our algorithm is compared to the FFPS method. As the mean length is shorter, the load becomes lighter and more dynamic. In this case, the FFPS method can easily lead to low resource utilization. While as the mean length is long, the load is heavy, thus the FFPS method can get high resource utilization.

F. Simulations of allocation of standard types of VMs

In this section, we evaluate our algorithm to allocate standard types of VMs in Table I. Fig. 7 shows the energy reduction ratio of the allocation of standard types of VMs on types 1-3 of servers, where the lines are the logarithm fitting curves. Our algorithm can save up to 24% energy compared to the FFPS method. In addition, as the mean inter-arrival time is long, the load becomes light, thus less energy is saved compared to the FFPS method as shown in Fig. 7.

Fig. 8(a) and Fig. 8(b) show that the average CPU utilization and memory utilization of servers. It demonstrates that our algorithm can significantly improve both utilization above 70%. In comparison, when all types of servers are used, the utilization by using the FFPS method is low to 30%.

As discussed in Section IV-C, we quantify the CPU load and the memory load of the system respectively by the average CPU utilization and average memory utilization of servers calculated by the FFPS method. We evaluate the allocation of standard types of VMs on both types 1-3 of servers and all types of servers. Fig. 9 shows the energy reduction ratio vs. the load of the system, where the lines are linear fitting curves. It demonstrates that the energy reduction ratio decreases closely linearly as the load increases. In addition, the energy reduction ratio when all types of servers are used is higher than the case when only the types 1-3 of servers are used. It is because in the former case, the FFPS method incurs the lower utilization by allocating a VM on a server with large capacity; while our algorithm has the same high utilization in both cases, as shown in Fig. 8.

V. Related Work

Power consumptions drastically increase in the data centers, and research works on energy efficient resource allocation are emerging in the cloud computing [1], [3]–[6], [18]. [3] researched to save the energy costs of data center servers in content delivery networks by turning off servers during periods of low load. It considered the tradeoffs between three objectives: to maximize energy savings, to satisfy the availability of customer requests, and to minimize server transition times between turning on and off. [4] researched to reduce the data center cost by adjusting the number of active servers according to the dynamic load. [5] researched to determine the system configurations according to the workload, which are the set of VMs, the physical machines VMs hosted, and the capacity of VMs, to optimize both the performance and the power saving of configurations and adaptations from previous configuration to the new configuration. In the existing works [2]–[5], servers are homogeneous and the loads can be assigned uniformly across the servers. Our problem considers non-homogeneous servers, thus the VMs can not be assigned across the servers uniformly. [6] and [18] researched to save energy consumption in data centers by dynamic migration of VMs according to the current resource utilization. In comparison, our problem focuses on saving energy consumption by VM allocation instead of VM migration.

There are also some works about VM scheduling with other objectives. [19] researched scheduling VM configurations and load balancing among servers by a stochastic model, where VM requests arrive according to a stochastic process. It optimized the maximum rates at which jobs can be processed in the system. In addition, the VM scheduling is closely relative to the fixed interval scheduling [9]–[13], [20]. The existing works on the fixed interval scheduling problem aim for minimizing the number of machines to accommodate all jobs, minimizing the job completion time, minimizing the cost of scheduling all jobs, maximizing the profit of a selected subsets of jobs and so on [11]. There are also some works about efficient architectures of parallel processors [21], [22].

VI. Acknowledgement

This work was supported by grant from Research Grants Council of Hong Kong [Project No. CityU 114112].

VII. Conclusion

In this paper, we study such a VM allocation problem. Given a set VMs and a set of servers in a data center, each VM has a resource demand (CPU, memory, storage) and a starting time and a finishing time, and each server has resource capacity. There is an additional energy cost for a server to switch back-and-forth between power-saving state and active state. The servers are non-homogeneous. The problem of our concern is to allocate the VMs onto servers, such that the VMs resource demands can be met and the total energy consumption of servers is minimized. A heuristic algorithm is proposed to solve it. The simulation results demonstrate that our algorithm can save energy significantly compared to the first fit power saving method. The impact of various parameters on the performance of our algorithm is evaluated, such as the mean inter-arrival time, the mean length of VMs and the transition cost.

References

[1] S. Albers, "Energy-efficient algorithms," *Commun. ACM*, vol. 53, no. 5, pp. 86–96, May 2010.

[2] A. Gandhi, V. Gupta, M. Harchol-Balter, and M. A. Kozuch, "Optimality analysis of energy-performance trade-off for server farm management," *Perform. Eval.*, vol. 67, no. 11, pp. 1155–1171, Nov. 2010.

[3] V. Mathew, R. Sitaraman, and P. Shenoy, "Energy-aware load balancing in content delivery networks," in *INFOCOM, 2012*, March 2012, pp. 954–962.

[4] M. Lin, A. Wierman, L. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," in *INFOCOM, 2011*, April 2011, pp. 1098–1106.

[5] G. Jung, M. Hiltunen, K. Joshi, R. Schlichting, and C. Pu, "Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures," in *ICDCS, 2010*, June 2010, pp. 62–73.

[6] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[7] D. Breitgand and A. Epstein, "Sla-aware placement of multi-virtual machine elastic services in compute clouds," in *Integrated Network Management (IM), 2011*, May 2011, pp. 161–168.

[8] J. Xu and J. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *2010 IEEE/ACM Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, Dec. 2010, pp. 179–188.

[9] M. Flammini, G. Monaco, L. Moscardelli, H. Shachnai, M. Shalom, T. Tamir, and S. Zaks, "Minimizing total busy time in parallel scheduling with application to optical networks," in *IPDPS 2009*, May 2009, pp. 1–12.

[10] R. Khandekar, B. Schieber, H. Shachnai, and T. Tamir, "Minimizing busy time in multiple machine real-time scheduling," in *FSTTCS'10*, 2010, pp. 169–180.

[11] M. Y. Kovalyov, C. Ng, and T. E. Cheng, "Fixed interval scheduling: Models, applications, computational complexity and algorithms," *European Journal of Operational Research*, vol. 178, no. 2, pp. 331–342, 2007.

[12] G. Calinescu, A. Chakrabarti, H. Karloff, and Y. Rabani, "An improved approximation algorithm for resource allocation," *ACM Trans. Algorithms*, vol. 7, no. 4, pp. 48:1–48:7, Sep. 2011.

[13] A. Darmann, U. Pferschy, and J. Schauer, "Resource allocation with time intervals," *Theoretical Computer Science*, vol. 411, no. 49, pp. 4217–4234, 2010.

[14] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, Dec. 2007.

[15] Amazon elastic compute cloud. http://aws.amazon.com/ec2/.

[16] J. Beckett, R. Bradfield, and the Dell Server Performance Analysis Team, "Power efficiency comparison of enterprise-class blade servers and enclosures," *A Dell Technical White Paper*, 2011. [Online]. Available: http://www.dell.com/downloads/global/products/pedge/en/BladePowerStudyWhitePaper_08112010_final.pdf

[17] A. Gandhi, M. Harchol-Balter, and M. A. Kozuch, "Are sleep states effective in data centers?" in *IGCC*. IEEE Computer Society, 2012, pp. 1–10.

[18] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, no. PrePrints, 2012.

[19] S. Maguluri, R. Srikant, and L. Ying, "Stochastic models of load balancing and scheduling in cloud computing clusters," in *INFOCOM, 2012*, March 2012, pp. 702–710.

[20] M. Shalom, A. Voloshin, P. Wong, F. Yung, and S. Zaks, "Online optimization of busy time on parallel machines," in *Theory and Applications of Models of Computation*, 2012, vol. 7287, pp. 448–460.

[21] J. Fan, X. Jia, and X. Lin, "Optimal embeddings of paths with various lengths in twisted cubes," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 18, no. 4, pp. 511–521, 2007.

[22] J. Fan and X. Jia, "Edge-pancyclicity and path-embeddability of bijective connection graphs," *Information Science*, vol. 178, no. 2, pp. 340–351, Jan. 2008.