BFD-NN: Best Fit Decreasing-Neural Network for Online Energy-Aware Virtual Machine Allocation Problems

Thanh Nguyen-Duc[2], Nguyen Quang-Hung[1], Nam Thoai[1]
Faculty of Computer Science and Engineering
HCMC University of Technology, VNU-HCM
Ho Chi Minh City, Vietnam
[1]{hungnq2, nam}@cse.hcmut.edu.vn
[2]{51203402}@hcmut.edu.vn

ABSTRACT

High performance computing (HPC) clouds consume a lot of energy (kWh); therefore reducing energy consumption is a high priority for any cloud provider. This paper studies the applications of vector bin packing heuristic and Neural Network (NN) to allocate virtual machines (VMs) onto physical machines (PMs) that minimizes total energy consumption of the physical machines. In our scenario, a list of virtual machines from request queue needs to assign to system for every interval time (T) and minimize total energy consumption. We proposed Best Fit Decreasing Neural Network (BFD-NN), which contains an evaluation function (f) that finds the most efficient physical machine for each VM from requests in system. We also proposed a process to optimize weight of coefficients in f for every request which users submit to the system based on information of users' requests in the past by using Parallel Genetic Algorithm (PGA) and Neural Network. Our method is a new approach because it not only uses knowledge from requests of users but also considers time dimension of virtual machines. Two job parallel workload models in Parallel Workloads Archive are used to evaluate our approach. The simulation results illustrate that BFD-NN could reduce up to 15% total energy consumption compared with state-of-the-art heuristics (such as Best Fit and First Fit Decreasing) in online allocation virtual machines.

CCS Concepts

•**Theory of computation** → **Scheduling algorithms;**
•**Software and its engineering** → **Virtual machines;**
•**Computing methodologies** → *Neural networks;*

Keywords

Cloud Computing; Resource Management; Energy efficiency; Virtual Machine Allocation; Vector Bin Packing; Neural Network Application; Parallel Genetic Algorithm; BFD-NN

SoICT '16, December 08-09, 2016, Ho Chi Minh City, Viet Nam

DOI: http://dx.doi.org/10.1145/3011077.3011116

1. INTRODUCTION

The development of technology leads to demanding computing resources of many fields in economy, science, and engineering. Large companies such as Google, Amazon, and Microsoft provide cloud computing services which enable Infrastructure-as-a-Service (IaaS) that users can run their applications in virtual machines instead of a real physical machines [3, 4, 8]. Cloud computing is a solution for management resources, energy and also support for large scale system with hundreds or thousands nodes [3–5, 8, 13, 23]. In the big systems, energy consumption is a major problem that we have to concern because it requires multiple megawatts electricity and costs more than fifteen million U.S. dollars per year of each data center [7, 13]. The consumed electricity of these data centers is also rapidly increasing in recent year. Therefore, advanced scheduling techniques for reducing the energy consumption is highly interested in cloud providers. Green computing is always a hot research topic due to increasing energy cost in cloud systems and greenhouse effect.

Allocation VMs in cloud computing is currently no optimal solution for all cases in polynomial time [8, 13, 23] because many previous works [7, 15] showed that the virtual machine placement problem can reduce to vector bin-packing problem with all VMs have both same starting time. Moreover, multidimensional vector bin packing problem is NP-Hard for every d and APX-Hard for $d \geq 2$ [15]. Therefore, it is not surprising that algorithms for VBP have been implemented in systems that place VMs onto PMs. In our problem, set of items are users' required virtual machines which each VM has *5* dimensions (CPU Utilization, RAM, Disk, Network Bandwidth, and Time) and set of bins are physical machines with *5* dimensions as VMs. A valid packing is a mapping between VM and PM that each dimension of VM does not exceed the capacity of host (PM). The efficient assignment of VM is to find a valid packing while reducing total electricity consumption. Much methods is presented in [4,5,15] using vector bin packing problem heuristics such as First-Fit Decreasing [15], Best-Fit Decreasing [5] and their solutions aim to minimize number of running physical machines in system. The idle machines are turned to sleep/hibernate mode as much as possible. In this scenario, no-guarantee that minimize number running machines is a good solution.

In this paper we consider energy-aware virtual machine problems where users request VM with fixed starting time and duration time (running time) as in previous works [18,

20]. However, this paper proposes a new solution Best Fit Decreasing Neural Network that combines neural network, Best-Fit Decreasing and genetic algorithm. We analyze and use the requirements of users in the past and combining machine learning and genetic algorithm apply on this data set to adjust the weight of coefficients in evaluation function which help system evaluate the position of physical machine mapped by virtual machine. This process consists of three steps: finding coefficients weight of evaluation function from data log file, training neural network, and real scheduling, respectively. The finding coefficients of evaluation function is done through parallel genetic algorithms to generate expected coefficients weight set of evaluation function from data log file. Next step is using machine learning, namely perceptron neural network, to mining the knowledge in data set from PGA output. Finally, the learned neural network predicts the future coefficients weight in evaluation function when the real requests are processed. Thus, this process can adapt any specific system because of optimizing coefficients step from data log file of its system and neural network. We simulated our solution in MATLAB with assigning virtual machines from user requests and modeling energy consumption of computers in system. In addition, our result which is mentioned later is compared with other heuristics (Best Fit Decreasing, First Fit Decreasing following CPU utilization, RAM, Duration Time) that usually be used in the real system. The archived results are very positive in reducing energy consumption and algorithm complexity is nearly Best Fit Decreasing heuristics.

The rest of this paper is structured as follows. Section 2 discusses related works. Section 3 proposes our approach in energy-aware virtual machine allocation with online requests of users. In section 3, we also describe characteristics of cloud system and a process to adapt our BFD-NN to individual system from data log file in the past. Section 4 discusses our performance evaluation using simulations. Section 5 concludes this paper and introduces future works.

2. RELATED WORK

There are lot of works that have proposed to address energy-aware virtual machine allocation problems in cloud data centers. Several works [2, 24] suggested scheduling algorithms to flexibly change processor speed that reduces power consumption of processors when users execute their applications. Some works aim to minimize energy consumption by minimizing number of physical machines algorithms [4–6, 10, 21] which assign consolidate VMs onto a small set of physical servers in datacenters.

Many works have illustrated the VM placement problem as a bin packing problem. They have used bin packing heuristics to place VMs onto minimum number of physical machines to minimize energy consumption [4, 5]. Microsoft research group [15] has studied first fit decreasing (FFD) based on vector bin packing problem to minimize number PMs in VM allocation problem. In the VM allocation problems, minimizing the number of used physical machines does not equal to minimizing the total energy consumption of all physical machines. Another works also used meta-heuristics algorithms to minimize power consumption such as hill-climbing [9], genetic algorithm [19] but it takes a lot of running time to find result that may infeasible in online scheduling.

Beloglazov et al. [4,5] have proposed VM allocation problem as bin-packing problem and presented a power-aware best-fit decreasing (denoted as PABFD) heuristic. PABFD sorts all VMs in a decreasing order of CPU utilization and tends to allocate a VM to an active physical server that would take the minimum increase of power consumption. However, choosing a minimum increasing power consumption of host does not guarantee minimizing total energy consumption in energy-aware VM allocation problems. The PABFD also does not consider the starting time and duration time of VM so that it is unsuitable for power-aware VM allocation.

HPC applications (or jobs) is considered in HPC clouds [8, 13]. Garg et al. [4] proposed a meta-scheduling problem to distribute HPC applications to cloud systems with distributed N data centers. The objective of scheduling is minimizing CO_2 emission and maximizing the revenue of cloud providers. Le et al. [13] distribute VMs across distributed cloud virtualized data centers whose electricity prices are different in order to reduce the total electricity cost.

Kovalyov et al. [12] has described characteristics of a fix interval time, fixed processing time and jobs only are processed in fixed duration time on an available machine. The scheduling problem can be applied in other domains. In this paper, we also research in fixed duration time to place VMs onto PMs in cloud system that reduce total energy consumption.

3. PROBLEM DETAIL AND SOLUTION

3.1 Energy Consumption Model

The energy consumption of the host j (E_j) in the period of time [t_1, t_2] is formulated as follow:

$$E_j = \int_{t_1}^{t_2} P_j(U_j(t))dt \quad (1)$$

where:

- $P_j(t)$: Power consumption of a single physical machine j at a time point t.
- $U_j(t)$: CPU utilization of physical machine j at time t and $0 \leq U_j(t) \leq 1$.

The power consumption model of a physical machine is nearly close to linear function [4,7] (our linear function P_i = a*100*load + b with a,b $\in \mathbb{R}$). Thus, we propose multiple linear functions in every load segment for more accuracy. For example, first function (P_0) is generated to calculate energy consumption from 0% to 10% computer load, second function (P_1) is also created to compute between 10% and 20% load and so on. If physical machine j has 17% load as energy model of IBM server in Table 1Power model of IBM server x3550table.1, for example, P_1 is used (Power consumption mode P(U) = 1.1 * 100* 0.17 + 87 (Watt)). Therefore, power model is presented by 10 linear functions.

3.2 System Description and Constraints

Users send requests to require resources (virtual machine) from system with CPU utilization (core), RAM (Mb), Disk (GB), Network Bandwidth (Mbs), Start Time and Duration Time (second) attributes as Figure 1Attributes of virtual machine request and physical machine in systemfigure.1-(a). Start Time is the time remaining until it is served by system and Duration Time is the running time of virtual machine in

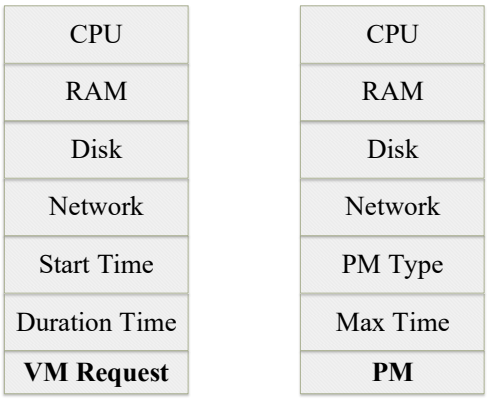

Figure 1: Attributes of virtual machine request and physical machine in system.

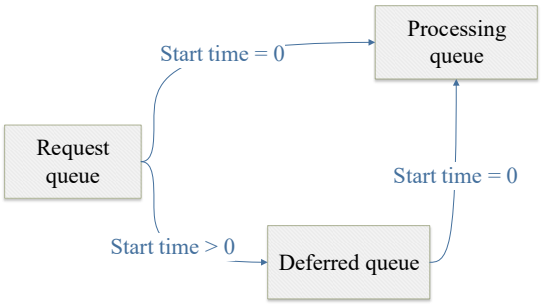

Figure 2: Three request queues in system.

system. Moreover, physical machines in cloud system have *6* attributes CPU utilization, RAM, Disk, Network Bandwidth, PM Type, Max Time as Figure 1Attributes of virtual machine request and physical machine in systemfigure.1-(b). PM Type is stored by integer number to denote kind of PM because real system may have different kinds of PM. Max Time is a longest duration time of VM in its PM.

Cloud computing system has a Request Queue, which contains all users' requirements. Every interval time (T), requests from Request Queue and Deferred Queue with *Start Time = 0* are pushed into Processing Queue. Otherwise, Deferred Queue receives other requests in Figure 2Three request queues in systemfigure.2. Hence, a list requested VMs are generated every T that need to assign to PMs in system.

Vector bin packing models well static resource allocation problems so the scheduling problem has the following hard constraints:

- Each VM is run by a physical server (host) at any time.
- VMs do not request any resource larger than corresponding dimensional resource of their hosts.
- The sum of total demand resources of these allocated VMs is less than or equal to total capacity of the system.

3.3 Evaluation Function (f)

Best-Fit Decreasing (BFD) following time algorithm sorts all VMs in the decreasing order of duration time and allocate each VM to a host. Thus, we propose an evaluation function (f) to find where is the best physical machine (maximum value of f) in system that a requested VM is assigned. In our problem, the dimensions have different measurement unit. CPU utilization, RAM, Network bandwidth and Duration time, for example, are measured by number of processor(s), Mb, Mbs, second(s), respectively. Therefore, f converts all dimensions into the same unit (percentage).

$$f = \sum_{j \in J} \alpha_j \left(\frac{(R_j)_{VM}}{(R_j)_{PMfree}}\right)^2 + \beta\left(1 - \frac{|VM_{DTime} - PM_{MTime}|}{max(VM_{DTime}, PM_{MTime})}\right)^2 \quad (2)$$

where:

- R_j: are dimensional resource j, J={CPU utilization, RAM, Disk, Network Bandwidth}.
- $(R_j)_{VM}$: are resource dimension j that a VM is required.
- $(R_j)_{PMfree}$: are remaining resource dimension j that can assign requested VM.
- α_j, β: are weight of dimensions, $0 \leq \alpha_j, \beta \leq 1$.
- VM_{DTime}: is duration time of VM.
- PM_{MTime}: is a longest duration time (MaxTime) of VM in its PM.

The value of f denotes how a VM fits in PM with its dimensions. The ratio between $(R_j)_{VM}$ and $(R_j)_{PMfree}$ reaches the maximum value (equal 1) when j-dimension of VM ($(R_j)_{VM}$) completely fits to remaining PM j-dimension ($(R_j)_{PMfree}$). In time dimension, virtual machines have similar duration time, which are assigned to one physical machine because all jobs of its physical machine can stop nearly the same time. To compare among physical machines in system, virtual machine is mapped with PM that has the maximum value of f.

The coefficients α_i and β adjust the weight of dimensions in f but it also changes in different systems and states of its system. The log file of system contains all requests in the past which can used for predicting coefficients weight of f in the future that can adapt to specific system. In this paper, we introduce a process that combines parallel genetic algorithm and neural network as Figure 3Using data in log file for scheduling in cloud systemfigure.3 to predict coefficients weight for each real request of users.

3.4 Finding Coefficients Weight of Evaluation Function by PGA

To prepare data for training neural network, we use PGA [11, 14, 22] to find coefficients weight of f from log file (as good as possible) for each list requested VMs then PGA's output is input (training set) of neural network. Every list of VMs of data in log file is a input into PGA, which find coefficients weight of f for each VM in its list VMs as in Figure 4The process of PGA to generate training setfigure.4. The genetic algorithm begins by creating a random initial population which is a vector with five times number VMs of

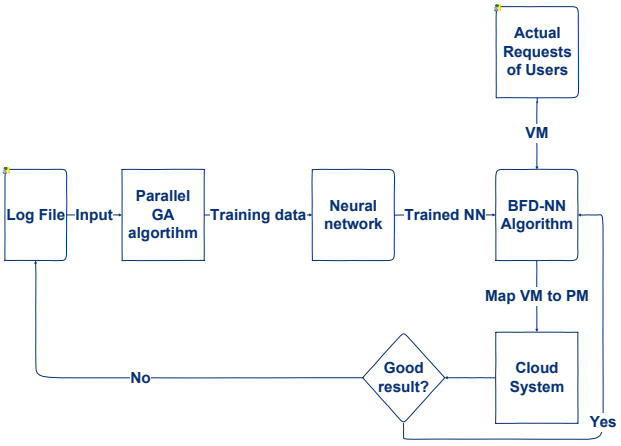

Figure 3: Using data in log file for scheduling in cloud system.

list length in range [0, 1] because f has five coefficients. The algorithm then creates a sequence of new populations. At each step, the algorithm uses the individuals in the current generation to create the next population. To create the new population, the algorithm performs the following step:

- Computes fitness value for each member of the current population.
- Scales the raw fitness value to convert into a more usable range values.
- Selects member, called parents, based on scaled fitness value.
- Some of the individuals in the current population that have lower fitness are chosen as elite. These elite individuals are passed to the next population.
- Produces children from the parents. Children are produced either by making random changes to a single parent-mutation-or by combining the vector entries of a pair of parents-crossover.
- Replaces the current population with the children to form the next generation.

The algorithm stops when the number of generations reaches the value of generations. Fitness function is a crucial function which PGA aims to maximum and also selects individuals in population for next generation. We propose a fitness function which evaluate running time in system as Figure 5Calculating fitness function of PGA in systemfigure.5. For example, 1^{st}-PM contains 3 VMs with 10, 5, 3 hours running time, respectively and 2^{nd}-PM has 4 VMs with 12, 7, 8, 2 hours running time so value of $fitnessfunction = \frac{(10+5+3)+(12+7+8+2)}{10*3+12*4}$. The fitness function reaches maximum value when the system has an expected state that almost VMs of system have the same duration time in its PM. Therefore, PM can easily switch to sleeping mode because all its VMs terminate at the same time that could reduce total running time of the system.

In Figure 6An example of reducing total running timefigure.6, for example, the vertical axis is capacity of resources PM and

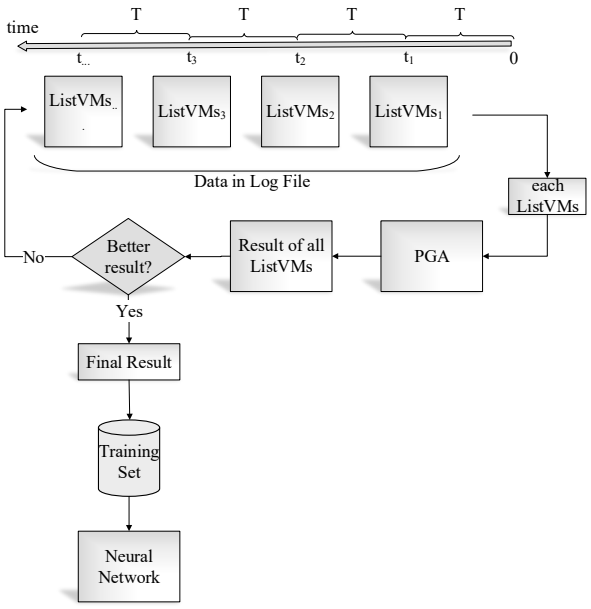

Figure 4: The process of PGA to generate training set.

Pseudo code: Calculate fitsness function of PGA

```
1:  MaxRunTime = 0; RealRunTime = 0;
2:  for PM in RunningPMs
3:      for VM in PM
4:          RealRunTime = RealRunTime + DurationTime(VM)
3:      endfor
4:      MaxRunTime = MaxRunTime +NumberVMinPM(PM)
                     * MaxDurationTimeVM(PM)
5:  endfor
6:  return RealRunTime/MaxRunTime
```

Figure 5: Calculating fitness function of PGA in system.

the horizontal axis is duration time of VMs in PM. The total running of system is 54 hours (16 + 18 + 20) in Figure 6An example of reducing total running timefigure.6-(a) but when VMs are assigned to PMs have nearly the same duration time, the total running time of system is only 39 hours in Figure 6An example of reducing total running timefigure.6-(b) while resources capacity is not change.

After all lists is run by PGA, result of all lists is compared to the best previous result by amount of energy consumption because PGA minimize running time for each list request that deals with local optimization. If result is better than old, final result stores it. Otherwise, loop continues finding result until terminate condition. The final result is solution that give minimum energy consumption for all lists and then we have a data set that contains data for training neural network as process in Figure 4The process of PGA to generate training setfigure.4.

3.5 Training Neural Network

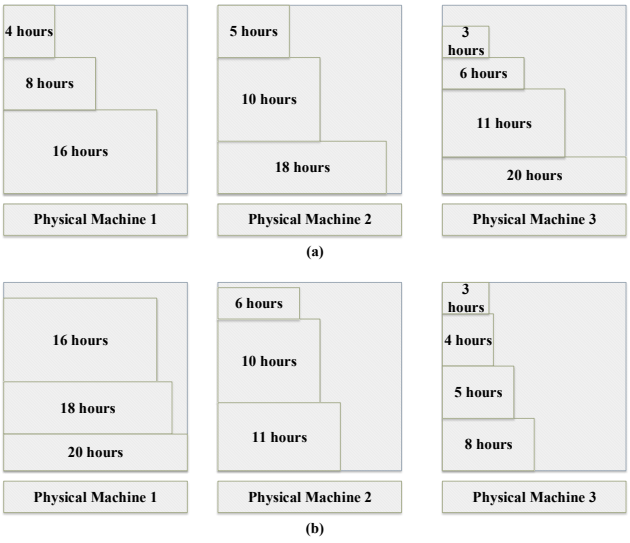

Figure 6: An example of reducing total running time.

The training set which is created from finding coefficients weight of evaluation function step by PGA is input for training neural network. The principal reason of this step is mining knowledge from output data of finding coefficients weight of f step that help predict coefficients weight in the future. The input features for neural network are the characteristics of list VMs of every T such as min, max, mean, standard deviation of each dimension (CPU utilization, RAM, Disk, Network bandwidth, Duration time) and characteristics of VM in list (CPU utilization, RAM, Disk, Network bandwidth, Duration time). The saved system states for every interval time (T) in PGA process are also included in input for training. We store the remaining percentage of resources (CPU, RAM, Disk, Network bandwidth) of system before every request in system is assigned in previous step for training process.

$$\%R_j = \sum_{k=1}^{m} \frac{freeR_{jk}}{totalR_{jk}} \tag{3}$$

where:

- $j \in$ {CPU utilization, RAM, Disk, Network Bandwidth}.
- $\%R_j$ are remaining percentage of system resources.
- m is number of physical machines in system.
- $freeR_{jk}$ are unused resource j of one physical machine k in system.
- $totalR_{jk}$ are total resource j of one physical machine k in system.

Then every rows in data (X) are re-scaled by formula 4Training Neural Networkequation.3.4 to generate training set. The min, max, and mean values is stored for using in real scheduling step. Neural network is trained by the normalized training set that is generated by PGA.

$$scale(x) = \frac{x - mean(x)}{max(x) - min(x)} \tag{4}$$

Pseudo code: Finding efficient PM to assign VM by f

```
1:  f_max = −1; chosedPM = null;
2:  for PM in CanAssign & Running PMs
3:      value = f(VM, PM)
4:      if value > f_max
5:          f_max = value
6:          chosedPM = PM
7:      endif
8:  endfor
9:  if chosedPM == null && findSleepPM() == null
10:     return "Can not assign VM to system"
11: elseif findSleepPM() != null
12:     chosedPM = findSleepPM()
13: endif
14: return chosedPM
```

Figure 7: Finding efficient PM to assign processing VM in system.

3.6 BFD-NN Scheduling Algorithm

BFD-NN combines the trained neural network and f to allocate each VM from real users onto PMs in system as in Figure 3Using data in log file for scheduling in cloud systemfigure.3. The role of trained neural network is to predict weight of coefficients of f and then f with predicted coefficients evaluates where processing request VM is assigned to system as pseudo code in Figure 7Finding efficient PM to assign processing VM in systemfigure.7. Every interval time T, the system serves list of virtual machine in Processing Queue. These requests are sorted in decreasing order duration time because the longest duration machines may consume more energy so it has higher priority. To assign a processing VM to system, the characteristics of VM, list VMs, and current state of system are given to neural network that predict coefficients weight in f. The function (f) evaluates every running physical machine(s) which can be assigned in system. Physical machine that has maximum value of f is mapped with the processing VM as algorithm in Figure 7Finding efficient PM to assign processing VM in systemfigure.7. Otherwise, when no running machine cannot contain the VM, a new PM is opened to serve this request. We also observe performance of system to adjust algorithm in system when behavior of requests changes or poor result shows in reducing energy consumption.

3.7 Algorithm Complexity

VMs in list from Processing Queue are sorted in decreasing order of duration time and trained neural network generates coefficients weight of f for each request. The system assigns all VMs to PMs based on value of f which find the most efficient physical machine. Therefore, our algorithm performs as well as Best Fit Decreasing because trained neural network computes fast enough. For more general, the formula 5Algorithm Complexityequation.3.5 is the complexity of one list VMs for every interval time T.

$$O(n * log(n) + n * m + n * (k/2)) \tag{5}$$

where:

- n: number of VMs in list

Table 1: Power model of IBM server x3550

Function	Load percentage(%)	a	b
P_0	00-10	3.96	58.4
P_1	10-20	1.1	87
P_2	20-30	0.9	91
P_3	30-40	1.0	88
P_4	40-50	1.2	80
P_5	50-60	1.3	75
P_6	60-70	1.7	51
P_7	70-80	1.9	37
P_8	80-90	1.6	61
P_9	90-100	1.7	52

Table 2: Types of virtual machine in first data set

Type	CPU (Core)	RAM (Mb)	DISK (Gb)	NET (Mbs)
1	1	1024	30	100
2	2	2048	60	100
3	3	4096	100	150
4	4	6144	150	200
5	5	8192	200	250
6	6	10240	250	300
7	7	12288	300	400
8	8	14336	500	500

- m: the complexity of neural network for every prediction
- k: number of physical machines in system
- $n * log(n)$: the cost of sorting VMs in decreasing order of duration time
- $n * m$: the cost of prediction for all VMs in list
- $n * (k/2)$: the average cost of finding efficient PM for all VMs in list.

4. SIMULATION AND EXPERIENCE

We simulated the system with queues, data structures, and behavior as description in section 3 (problem detail and solution). It has 1000 IBM servers x3550 [1] (2x Xeon X5675 3067 MHz 16 cores, 16 GB RAM, 1000 GB storage, 1000 Mbs network bandwidth) and the energy model is illustrated by 10 linear functions in Table 1Power model of IBM server x3550table.1. Simulation process is run in MATLAB R2013a with Windows 10 operating system.

To test our algorithm, two data workload models are Feitelson's Parallel Workloads Archive [17] for first data set and Downey's Parallel Workloads Archive [16] for second data set that was used for simulation. Each data set is divided to 2 parts (60% input for BFD-NN process (e.g: finding coefficients weight, training neural network), 40% for users' real requests) and also have 8 types of VM with dimensions CPU utilization(core), RAM(MB), DISK(GB), and Network bandwidth (Mbs), but different attributes as Table 2Types of virtual machine in first data settable.2 and 3Types of virtual machine in second data settable.3.

In finding coefficients weight of evaluation function, the several options of PGA are used for data sets. Two options in Table 4The options of Parallel Genetic Algorithmtable.4

Table 3: Types of virtual machine in second data set

Type	CPU (Core)	RAM (Mb)	DISK (Gb)	NET (Mbs)
1	1	1875	20	100
2	2	4800	55	100
3	3	5500	120	200
4	4	6700	210	200
5	5	8200	270	300
6	6	9900	325	300
7	7	12288	410	400
8	8	15336	520	400

Table 4: The options of Parallel Genetic Algorithm

PGA options	Data set 1	Data set 2
Population size	20	20
Initial range	[0;1]	[0;1]
Fitness scaling	Top	Top
Selection	Uniform	Roulette
Elite Count	2	2
CrossoverFraction	0.8	0.8
Mutation	Adapt Feasible	Constraint Dependent
Crossover	Scattered	Two point
Hybrid function	Pattern search	None
Generation	100	100

give the best result of each data set which has minimum energy consumption that is input training set to next training step. For training neural network step, we separate training set into 3 parts (60% for training, 20% for validation, 20% for testing); then trained neural network is used for predicting coefficients weight of f.

Cloud system is tested with users' real requests and collected all information. The results of energy consumption and running time of algorithms are showed in Table 5Total energy consumption results of first data settable.5 and 6Total energy consumption results of second data settable.6. In two data sets, BFD-NN gives the lowest energy consumption when it compares to other heuristics such as: First Fit Decreasing and Best Fit Decreasing following CPU, RAM, Duration Time (FFD-CPU, FFD-RAM, FFD-TIME, BFD-CPU, BFD-RAM, BFD-TIME, respectively) as in Figure 8The comparison among energy consumption of algorithmsfigure.8.

The testing results of our algorithm (BFD-NN) show the positive result up to 15% reducing energy consumption. The second efficient algorithm in first data set is BFD-TIME that provides more over 150 kWh electricity usage than BFD-NN. In the other hand, results in second data set illustrate that FFD-CPU is ranked at second position about reducing energy consumption. Hence, the results of two data sets also show that First Fit Decreasing and Best Fit Decreasing cannot adapt to system when characteristics of requests are changed. The performance of BFD-NN is very well at peak over 15% in second data set and over 5% in first data set because different kind of machines and characteristics of requests lead to different results of other algorithms between two data sets. Generally, BFD-NN always shows the result better and more stable than other algorithms in our data sets that depend on characteristics of data. It can adapt with individual cloud system because PGA and neural net-

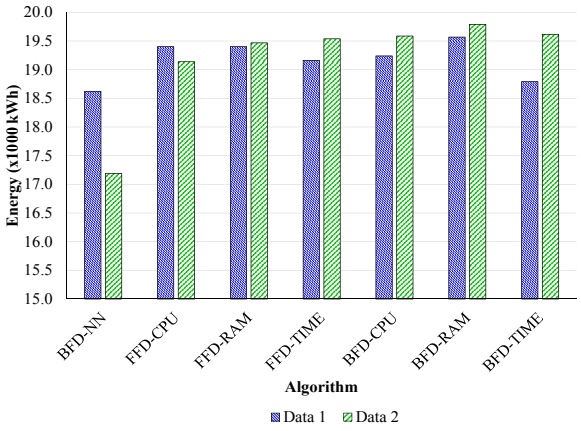

Figure 8: The comparison among energy consumption of algorithms.

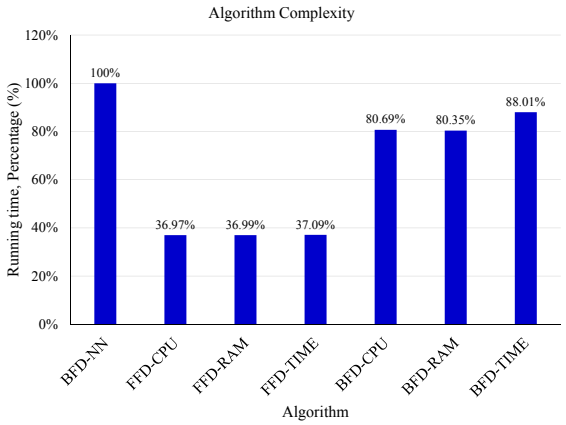

Figure 9: The comparison among running time of algorithms.

Table 5: Total energy consumption results of first data set

Algorithm	Energy (Kwh)	Running Time (second)
BFD-NN	18623.30	18.291
FFD-CPU	19402.15	6.719
FFD-RAM	19402.15	6.724
FFD-TIME	19161.47	6.751
BFD-CPU	19238.27	14.568
BFD-RAM	19567.54	14.535
BFD-TIME	18791.44	15.955

Table 6: Total energy consumption results of second data set

Algorithm	Energy (Kwh)	Running Time (second)
BFD-NN	17192.09	18.064
FFD-CPU	19138.03	6.721
FFD-RAM	19467.60	6.722
FFD-TIME	19538.03	6.732
BFD-CPU	19586.40	14.765
BFD-RAM	19789.14	14.673
BFD-TIME	19616.68	16.038

work get knowledge from data log file in the past.

We also compare algorithm complexity when the system serves requests. The average running time of two data is showed in Figure 9The comparison among running time of algorithmsfigure.9 that BFD-NN performs nearly to Best Fit Decreasing and First Fit Decreasing always point out the most efficient.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed BFD-NN for energy-aware VM allocation problem. In our algorithm, we suggested the evaluation function (f) to find the efficient PM in system that VM can be assigned and using PGA, neural network to predict coefficients weight of f for each real request from users. Therefore, BFD-NN can be adapted to specific system. Our approach aims to adjust coefficients weight of f for reducing energy consumption when system serves requests based on knowledge from data log file in the past and evaluation function (f) can compare among physical machines in system to find efficient machine.

In the future, we concern methodology to reduce computational time of the algorithm process and online machine learning algorithms to learn newest information when algorithm is applied in system.

6. ACKNOWLEDGEMENTS

This research was conducted within the "Developing a Virtual Lab based on Cloud" funded by HCMUT (under grant number T-KHMT-2013-74).

7. REFERENCES

[1] Standard Performance Evaluation Corporation. SPECpower ssj2008. http://www.spec.org/power_ssj2008/results/res2011q2/power_ssj2008-20110406-00368.html. Accessed: 19-July-2016.

[2] S. Albers. Energy-efficient algorithms. *Communications of the ACM*, 53(5):86–96, 2010.

[3] L. A. Barroso, J. Clidaras, and U. Hölzle. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis lectures on computer architecture*, 8(3):1–154, 2013.

[4] 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*, 28(5):755–768, 2012.

[5] A. Beloglazov and R. Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience*, 24(13):1397–1420, 2012.

[6] L. Chen and H. Shen. Consolidating complementary vms with spatial/temporal-awareness in cloud datacenters. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 1033–1041. IEEE, 2014.

[7] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *ACM SIGARCH Computer Architecture News*, volume 35, pages 13–23. ACM, 2007.

[8] S. K. Garg, C. S. Yeo, A. Anandasivam, and R. Buyya. Energy-Efficient Scheduling of HPC Applications in Cloud Computing Environments. *arXiv preprint arXiv:0909.1146*, abs/0909.1(September 2009), 2009.

[9] I. Goiri, F. Julia, R. Nou, J. L. Berral, J. Guitart, and J. Torres. Energy-aware scheduling in virtualized datacenters. In *2010 IEEE International Conference on Cluster Computing*, pages 58–67. IEEE, 2010.

[10] T. Knauth and C. Fetzer. Energy-aware scheduling for infrastructure clouds. In *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, pages 58–65. IEEE, 2012.

[11] J. Kolodziej, S. U. Khan, and F. Xhafa. Genetic algorithms for energy-aware scheduling in computational grids. In *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2011 International Conference on*, pages 17–24. IEEE, 2011.

[12] M. Y. Kovalyov, C. Ng, and T. E. Cheng. Fixed interval scheduling: Models, applications, computational complexity and algorithms. *European Journal of Operational Research*, 178(2):331–342, 2007.

[13] K. Le, R. Bianchini, J. Zhang, Y. Jaluria, J. Meng, and T. D. Nguyen. Reducing electricity cost through virtual machine placement in high performance computing clouds. *2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 1–12, 2011.

[14] H. Mühlenbein, M. Schomisch, and J. Born. The parallel genetic algorithm as function optimizer. *Parallel computing*, 17(6-7):619–632, 1991.

[15] R. Panigrahy, K. Talwar, L. Uyeda, and U. Wieder. Heuristics for Vector Bin Packing. *Research.Microsoft.Com*, 2011.

[16] Parallel Workload Models. Downey Model. http://www.cs.huji.ac.il/labs/parallel/workload/models.html#downey97. Accessed: 19-July-2016.

[17] Parallel Workload Models. Feitelson Model. http://www.cs.huji.ac.il/labs/parallel/workload/models.html#feitelson96. Accessed: 19-July-2016.

[18] N. Quang-Hung, D.-K. Le, N. Thoai, and N. T. Son. Heuristics for energy-aware vm allocation in hpc clouds. In T. K. Dang, R. Wagner, E. Neuhold, M. Takizawa, J. Küng, and N. Thoai, editors, *Future Data and Security Engineering: First International Conference, FDSE 2014, Ho Chi Minh City, Vietnam, November 19-21, 2014, Proceedings*, pages 248–261. Springer International Publishing, 2014.

[19] N. Quang-Hung, P. D. Nien, N. H. Nam, N. Huynh Tuong, and N. Thoai. A genetic algorithm for power-aware virtual machine allocation in private cloud. In K. Mustofa, E. J. Neuhold, A. M. Tjoa, E. Weippl, and I. You, editors, *Information and Communication Technology: International Conference, ICT-EurAsia 2013, Yogyakarta, Indonesia, March 25-29, 2013. Proceedings*, pages 183–191, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[20] N. Quang-Hung and N. Thoai. Minimizing Total Busy Time for Energy-Aware Virtual Machine Allocation Problems. *Proceedings of the Sixth International Symposium on Information and Communication Technology - SoICT 2015*, pages 179–186, 2015.

[21] I. Takouna, W. Dawoud, and C. Meinel. Energy efficient scheduling of hpc-jobs on virtualize clusters using host and vm dynamic configuration. *ACM SIGOPS Operating Systems Review*, 46(2):19–27, 2012.

[22] R. Tanese. Parallel genetic algorithm for a hypercube. In *Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA*. Hillsdale, NJ: L. Erlhaum Associates, 1987., 1987.

[23] H. Viswanathan, E. K. Lee, I. Rodero, D. Pompili, M. Parashar, and M. Gamell. Energy-aware application-centric VM allocation for HPC workloads. *IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*, pages 890–897, 2011.

[24] G. von Laszewski, L. Wang, A. J. Younge, and X. He. Power-aware scheduling of virtual machines in dvfs-enabled clusters. In *2009 IEEE International Conference on Cluster Computing and Workshops*, pages 1–10. IEEE, 2009.